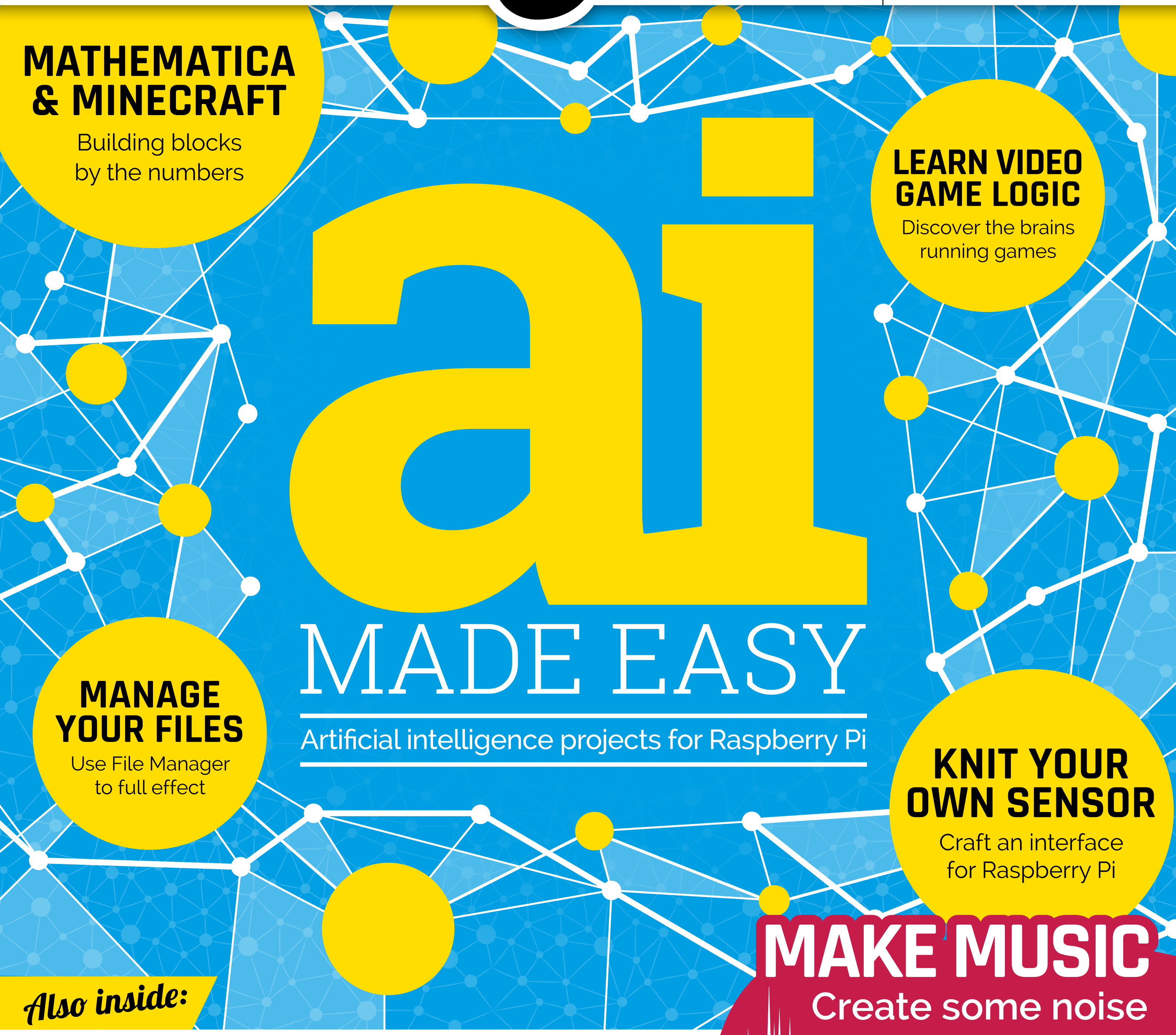
The MagPi

The official Raspberry Pi magazine

Issue 72 August 2018

raspberrypi.org/magpi



MATHEMATICA & MINECRAFT

Building blocks by the numbers

LEARN VIDEO GAME LOGIC

Discover the brains running games

ai MADE EASY

Artificial intelligence projects for Raspberry Pi

MANAGE YOUR FILES

Use File Manager to full effect

KNIT YOUR OWN SENSOR

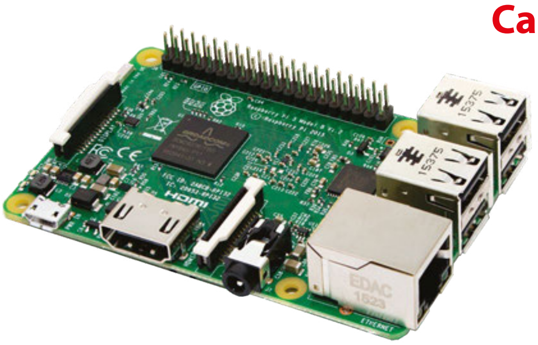
Craft an interface for Raspberry Pi

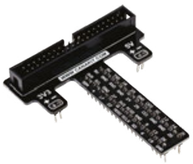
MAKE MUSIC

Create some noise with Raspberry Pi

Also inside:

HOW TO BUILD YOUR OWN MINI MAGIC MIRROR





WELCOME TO THE OFFICIAL MAGAZINE

A wise maker called Ben Driscoll once said: “If you ever code something that ‘feels like a hack but it works,’ just remember that a CPU is literally a rock that we tricked into thinking.”

This month, we’re putting that ‘hack that works’ concept into the trickiest of thinking concepts: artificial intelligence (AI).

AI works right at the coalface of computing. It’s exciting, complicated, baffling, and brilliant. But you don’t need a PhD in neural networks to get our AI services up and running. These projects are picked to help you understand AI with the minimum of fuss.

The Raspberry Pi is a perfect fit for AI. Because while you might need a fast CPUs and multiple GPUs to create AI models (the decision-making plans), you can run models on microcomputers like the Raspberry Pi and access powerful AI tools via cloud services.

And the Raspberry Pi acts as the interface to physical computing hardware, turning those mysterious AI models into practical real-world projects.

AI is one of the most misunderstood technologies in the modern world. A sprinkling of AI dust on a physical computing project can create a seemingly magical device; read on and you’ll discover how down-to-earth AI really is.

Lucy Hattersley
Editor

THIS MONTH:

FIND US ONLINE raspberrypi.org/magpi

GET IN TOUCH magpi@raspberrypi.org

The MagPi

Available on the App Store

GET IT ON Google Play

EDITORIAL
Editor: **Lucy Hattersley**
lucy@raspberrypi.org
Features Editor: **Rob Zwetsloot**
rob.zwetsloot@raspberrypi.org
Sub Editors: **Phil King and Jem Roberts**

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

DESIGN
Critical Media: **criticalmedia.co.uk**
Head of Design: **Dougal Matthews**
Designers: **Mike Kay and Lee Allen**
Illustrator: **Sam Alder**

SUBSCRIPTIONS
Raspberry Pi Press
Mann Enterprises, Unit E, Brocks
Business Centre, Haverhill, CB9 8QP
To subscribe:: **magpi.cc/subscribe**
To get help: **magpi.cc/subshelp**

PUBLISHING
For advertising & licensing:
Publishing Director: **Russell Barnes**
russell@raspberrypi.org | **+44 (0)7904 766523**
Director of Communications: **Liz Upton**
CEO: **Eben Upton**

CONTRIBUTORS
Wes Archer, Guenter Bartsch, Alex Bate, Brian Beuken, Mike Cook, David Crookes, PJ Evans, Nicola King, Joe McLoone, Martin O’Hanlon, KG Orphanides, David Pride, Jose Marcial Portilla, Nik Rawlinson, Matt Richardson, Richard Smedley, Mark Vanstone, Clive Webster, Robert Zakon

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

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

The MagPi magazine is published by Raspberry Pi (Trading) Ltd., 30 Station Road, Cambridge, CB1 2JH. The publisher, editor, and contributors accept no responsibility in respect of any omissions or errors relating to goods, products or services referred to or advertised in the magazine. ISSN: 2051-9982.

Raspberry Pi PRESS

Contents

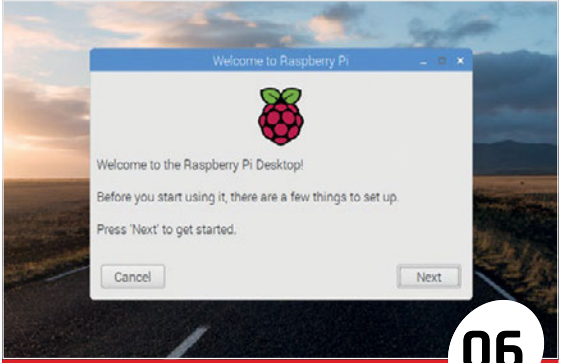
Issue 72 August 2018

raspberrypi.org/magpi

TUTORIALS

COVER FEATURE

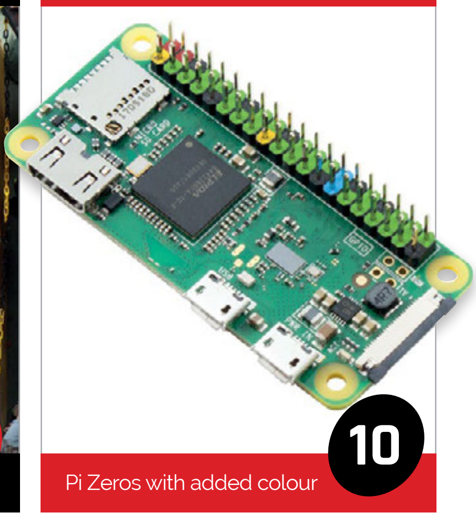
IN THE NEWS



Welcome to the Raspberry Pi Desktop!

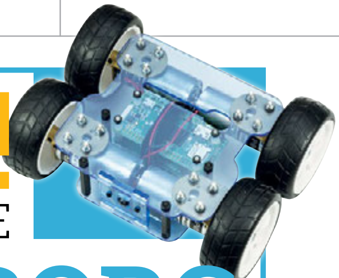
Before you start using it, there are a few things to set up.

Press 'Next' to get started.



Some of the tools and techniques shown in The MagPi magazine are dangerous unless used with skill, experience, and appropriate personal protection equipment. While we attempt to guide the reader, ultimately you are responsible for your own safety and understanding the limits of yourself and your equipment. Children should be supervised. Raspberry Pi (Trading) Ltd does not accept responsibility for any injuries, damage to equipment, or costs incurred from projects, tutorials or suggestions in The MagPi magazine. Laws and regulations covering many of the topics in The MagPi magazine are different between countries, and are always subject to change. You are responsible for understanding the requirements in your jurisdiction and ensuring that you comply with them. Some manufacturers place limits on the use of their hardware which some projects or suggestions in The MagPi magazine may go beyond. It is your responsibility to understand the manufacturer's limits.

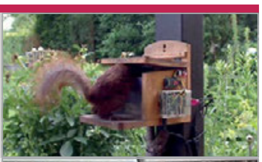
THE BIG FEATURE

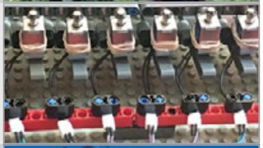


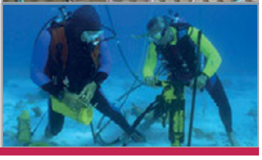
WIN! ONE OF FIVE YETIBORG RASPBERRY PI ROBOT KITS!

In association with MODMYPI

YOUR PROJECTS







REGULARS

COMMUNITY

REVIEWS

RASPBIAN UPDATED

Latest OS update gets setup wizard, app store and *The MagPi*

The Raspberry Pi Foundation has released a new version of Raspbian, including a setup wizard for the first time.

As Simon Long, Senior Principal Software Engineer, Raspberry Pi Foundation, explains about previous versions: "When a new user first boots up a new Pi … they might not have much of an idea what they ought to do next.

"With the new update," Simon continues, "a simple setup wizard runs automatically to walk you through the basic setup operations."

Simon tells us, "The essentials are getting the keyboard and language right (because otherwise input and output don't work properly), setting the WiFi country (as otherwise you can't use the network), changing the password (basic security), checking for updates (basic security and reliability), and establishing a network connection (required to check for updates)."

Personalisation options, such as setting the desktop background, weren't included in the setup wizard, as "the system will work perfectly fine without [them]."

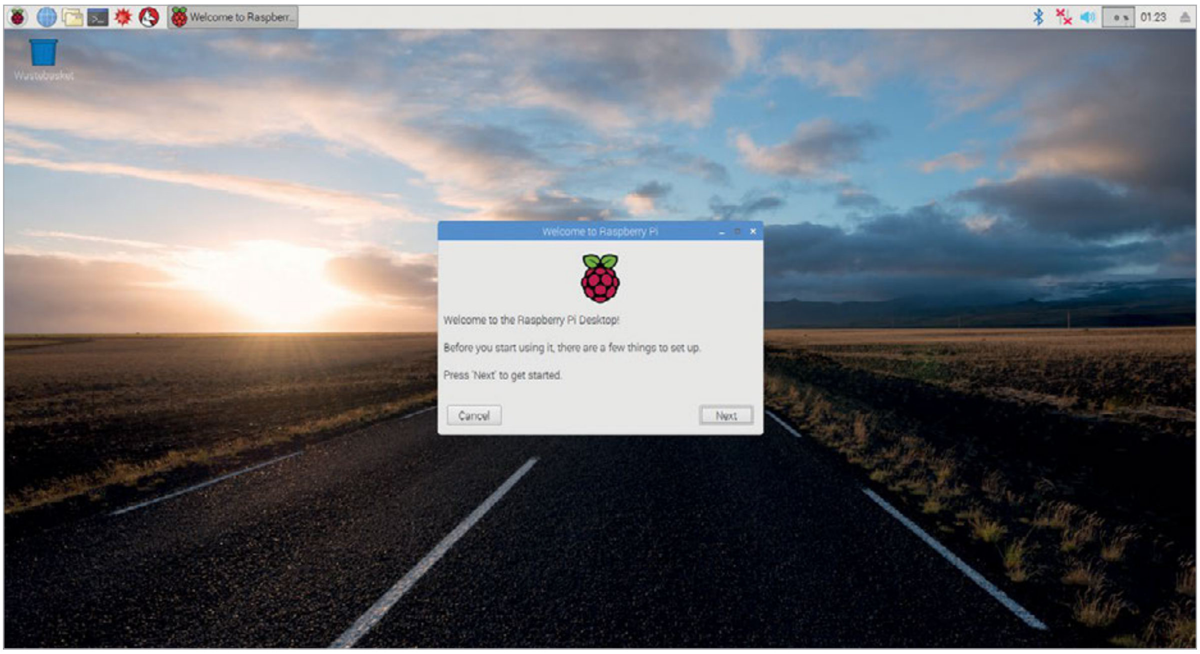
Setting the WiFi country correctly is important since the release of the Raspberry Pi 3B+, as Simon clarifies: "In order for 5G WiFi hardware to be certified for use, it must not radiate unless a country has been set."

The new version of Raspbian also includes a Recommended Software 'app store' for the first time – see the 'Raspbian App Store' box for more details.

A new PDF viewer

Raspbian has used Xpdf since its inception, but this venerable application has become dated in its look, capabilities, and performance.

The new version of Raspbian has replaced Xpdf with qpdfview because, Simon reveals, the default Raspbian PDF viewer "needed to render fast, preload

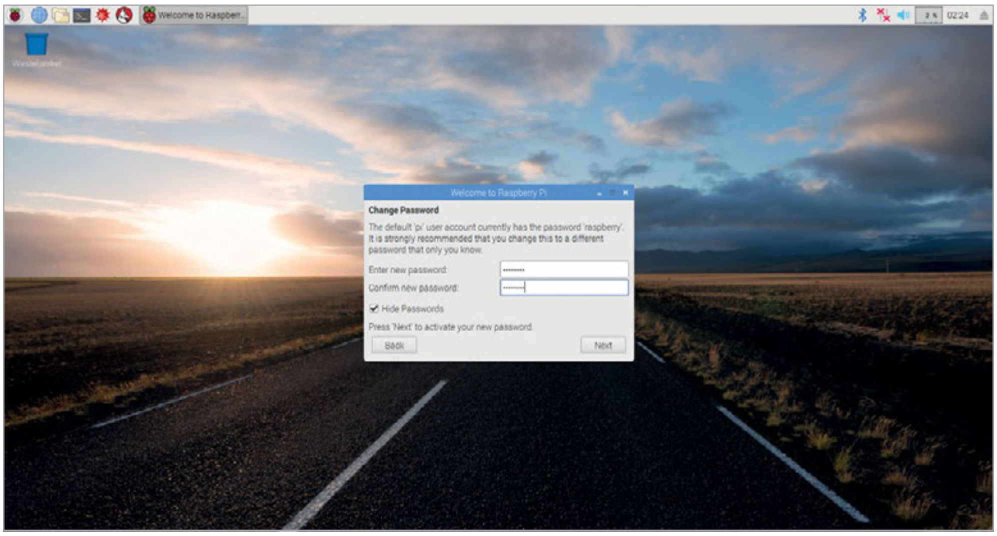


Welcome to the Raspberry Pi Desktop!

Before you start using it, there are a few things to set up.

Press 'Next' to get started.

Right The new setup wizard guides you through the essential setup process



Welcome to Raspberry Pi

Change Password

The default 'pi' user account currently has the password 'raspberry'. It is strongly recommended that you change this to a different password that only you know.

Enter new password: ••••••••

Confirm new password: ••••••••

☑ Hide Passwords

Press 'Next' to activate your new password.

Left Raspbian now prompts users to change the password when they first boot up the Raspberry Pi

pages, have a modern UI … and to fit our look and feel.”

The more well-known Okular wasn't chosen; Simon explains that “the fact that it's a 200MB install, including an older version of the Qt toolkit than we already ship” counted against it.

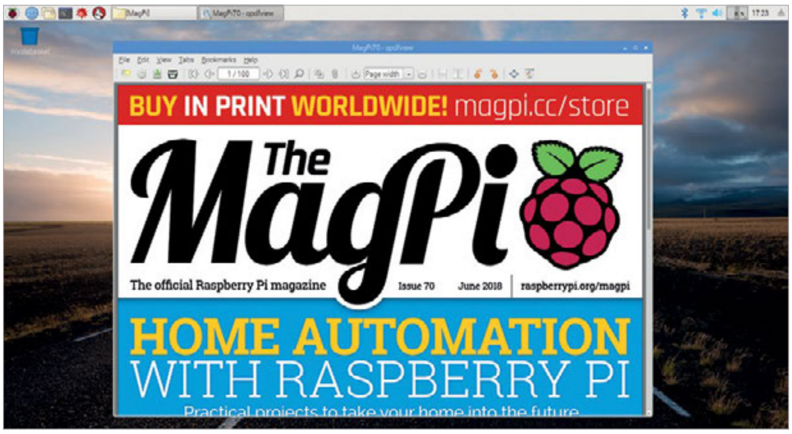
> Raspbian also includes a Recommended Software ‘app store’ for the first time

Simon continues, “qpdfview is Qt based, which is non-ideal – our standard GUI toolkit is GTK, but I was able to hack the Qt theming to closely resemble GTK.”

You might also be pleased to hear that “we are now including the latest issue of *The MagPi* as a PDF file,” Simon confirms. “Look in the ‘MagPi’ directory in your home directory ‘pi’.” You'll still be able to download the PDF version of this magazine from **magpi.cc/issues.**

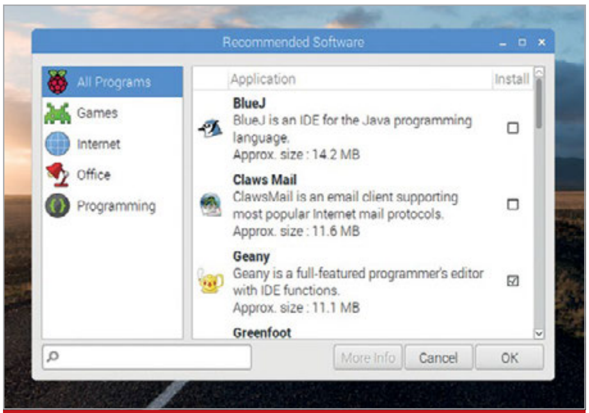
Upgrade Raspbian through the Terminal in the usual way (**`sudo apt-get update; sudo apt-get dist-upgrade`**). To view the startup wizard and Recommended Software, download the image from **magpi.cc/PVNGfh**.

The x86 desktop version of Raspbian has also been updated with “most of the changes”.



BUY IN PRINT WORLDWIDE! magpi.cc/store
The MagPi
The official Raspberry Pi magazine
Issue 70
June 2018
raspberrypi.org/magpi
HOME AUTOMATION
WITH RASPBERRY PI
Practical projects to take your home into the future

Above The new PDF viewer is a welcome update from Xpdf, and there's a digital copy of *The MagPi magazine* waiting in your ‘pi’ home directory to read



Recommended Software

Application	Install
BlueJ BlueJ is an IDE for the Java programming language. Approx. size : 14.2 MB	☐
Claws Mail ClawsMail is an email client supporting most popular Internet mail protocols. Approx. size : 11.6 MB	☐
Geany Geany is a full-featured programmer's editor with IDE functions. Approx. size : 11.1 MB	☑
Greenfoot	

RASPBIAN APP STORE

The new version of Raspbian also includes a new app store called Recommended Software. This is a separate application to the software manager, allowing Raspbian users to select which of the previously default software to install.

Not everyone needs BlueJ, Node-RED or VNC Viewer, so including these by default only leads to a larger download, clutter, and confusion. Installation is now optional.

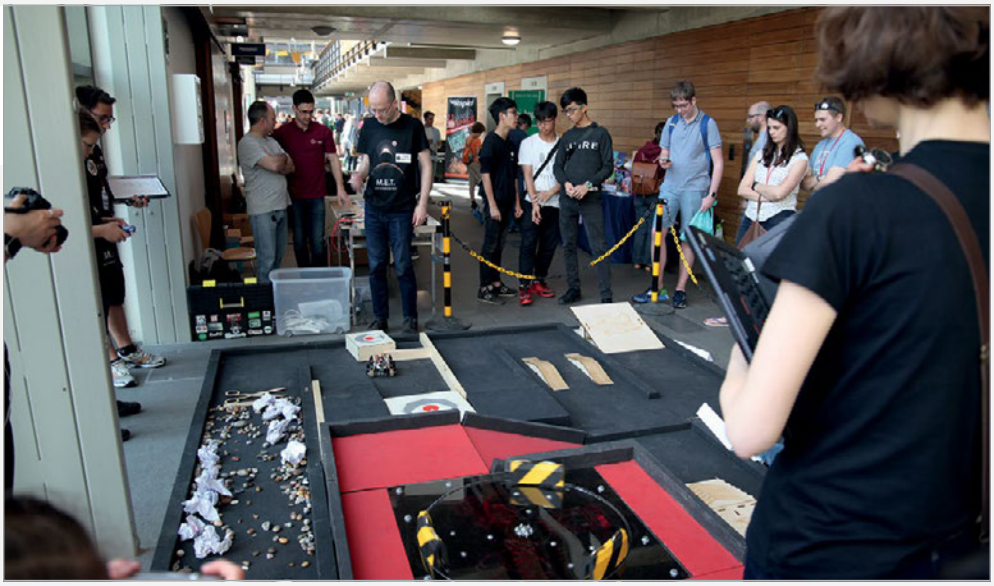
The Recommended Software app brings together all of the software that, as Simon notes, “several third-party companies have generously offered … in some cases giving free licences,” as well as software recommended for use on a Pi by the Raspberry Pi Foundation.

For example, a Mathematica licence would normally cost “several hundred pounds,” says Simon, while there's also a “free licence for RealVNC Viewer and Server.”

The Recommended Software application can be found in the Preferences menu of the new Raspbian release. To add the application to your Raspbian install, use the command `sudo apt-get install rp-prefapps`.

PI WARS 2019

Applications open, challenges announced



Above The Pi Wars 2018 obstacle course drew a crowd of roboteers and supporters

The application process for Pi Wars 2019 is now open, and this year sees the first Pi Wars theme: space exploration!

Pi Wars co-organiser Mike Horne confirms: "This is the first time we have themed Pi Wars, in honour of the 50th year since the first moon landing." The idea came from previous winner Dave Pride.

The theme means cosmetic changes to both the challenges and the venue, which remains as the William Gates Building of the Cambridge Computer Laboratory. "Lots of painting of the courses is going to occur over the summer," Mike tells us.

> This is the first time we have themed Pi Wars, in honour of the 50th year since the first moon landing

Challenger

The space theme introduces new challenges, but Mike says that "the rules are pretty similar to before, with a few tightened up and few loosened."

For example, the new Hubble Telescope Challenge is based on 2018's Over the Rainbow, where robots needed to identify coloured balls and drive to them in sequence. "This was the hardest course by a long way," Mike reveals. "We will be making the targets larger, but have not yet finalised that."

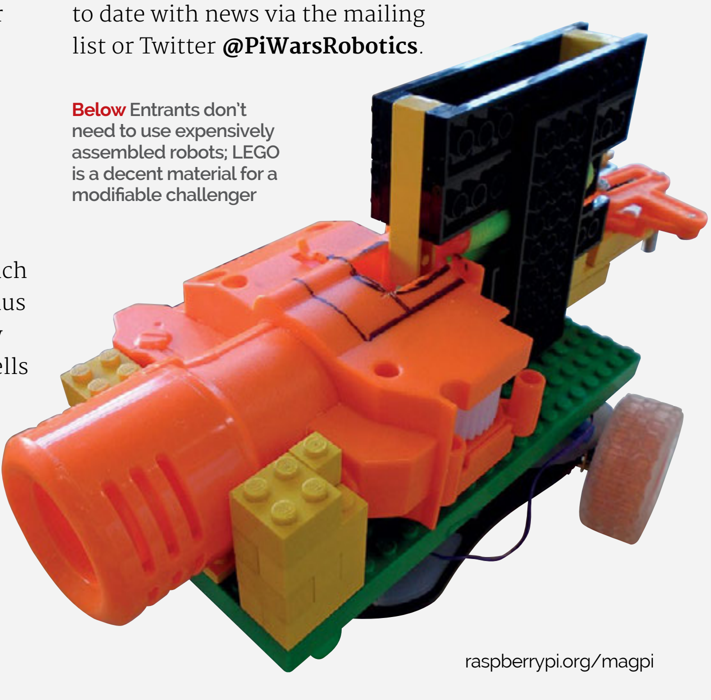
Space Invaders is based on the Duck Hunt challenge, "with the same skills required to score." The Spirit of Curiosity challenge involves driving your robot over an obstacle course, collecting a sample (which "will vary in size, but not by much") and driving back.

Re-entry

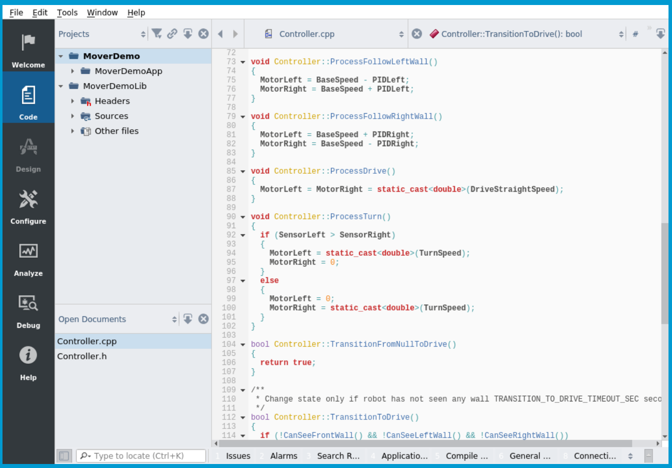
Pi Wars 2018 received "over 150 entrants," Mike confirms, "which had to be cut down to just 76, plus reserves." This involves a "very hard couple of days", as Mike tells us that "we cannot just choose previous entrants as we want to make space for noobs."

Mike has this advice for your application form: "Ensure the entrants show their enthusiasm [and] we like detail."

Pi Wars 2019 will take place on 30 and 31 March, so head over to **piwars.org** to read about rules, challenges, and to apply. Keep up to date with news via the mailing list or Twitter **@PiWarsRobotics**.



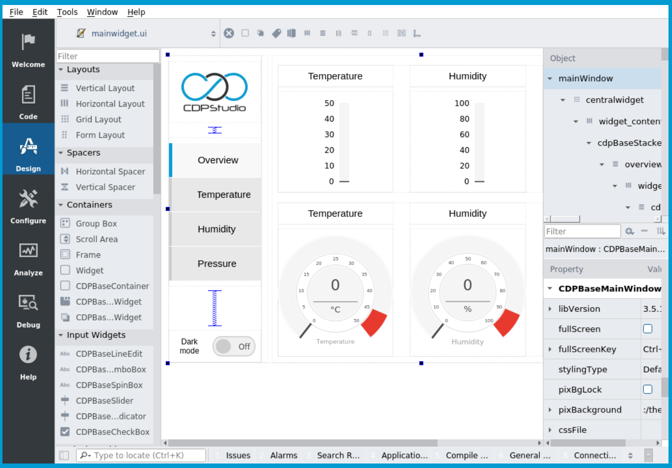
Below Entrants don't need to use expensively assembled robots; LEGO is a decent material for a modifiable challenger



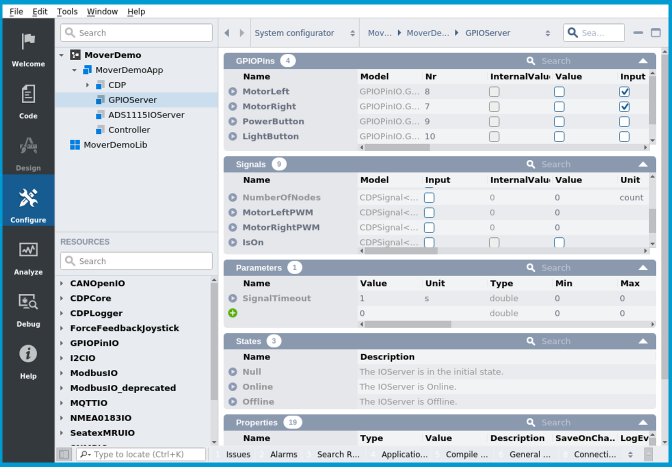
Controller.cpp

```
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



System configurator › Mov... › MoverDe... › GPIOServer

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

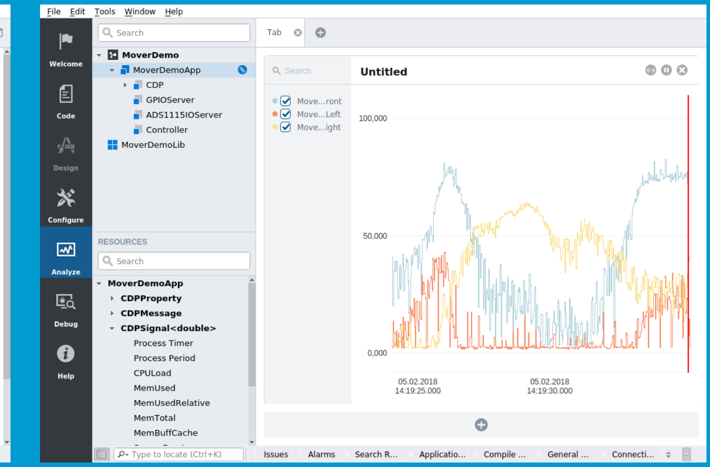
States 3

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.

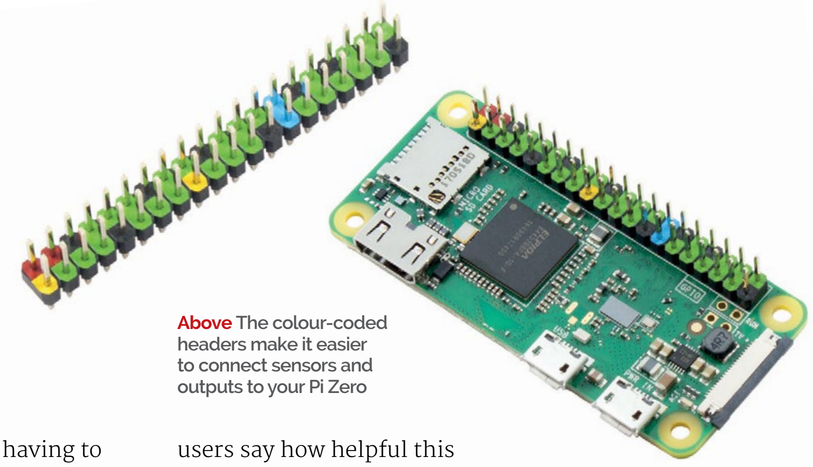
Free download on www.cdpstudio.com

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

COLOUR -CODED PI ZEROS



Above The colour-coded headers make it easier to connect sensors and outputs to your Pi Zero

Making making easier

Pi Supply has released a pair of Pi Zero boards with colour-coded GPIO pins, as well as a separate colour-coded header, to make your mini-builds a bit easier.

As John Whiting, director of marketing for Pi Supply, explains, "A lot of users have struggled in the past remembering the GPIO layout, so the coloured header is a low-cost way of not having to remember the layout or go off and Google it."

The colour-coding follows the usual convention, where red is +5V, yellow is +3.3V (also referred to as 3V3), black is ground, blue is DNC (or reversed I²C), and green shows the GPIO pins.

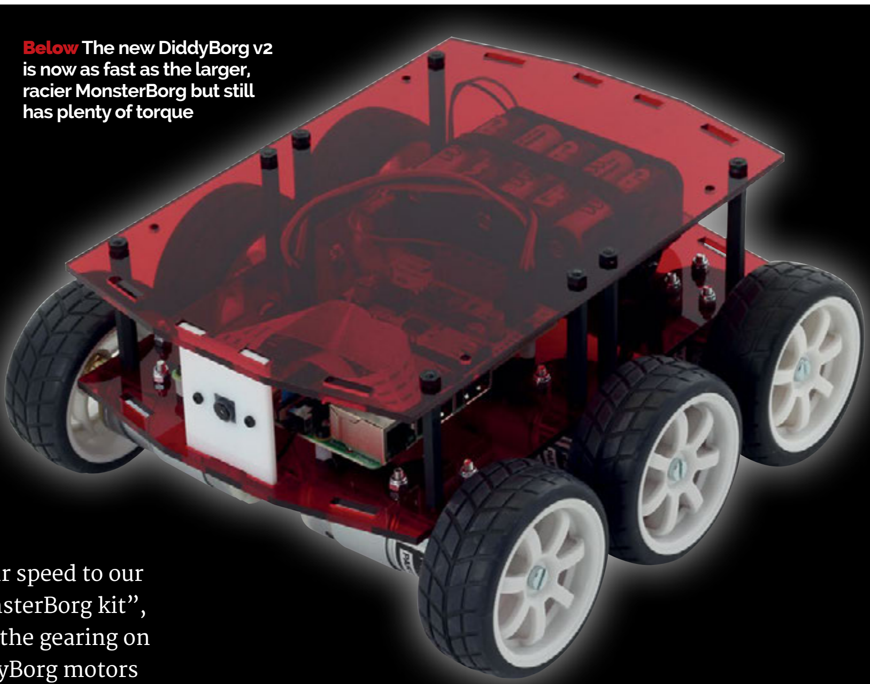
"The reaction has been great", John tells us. "We've had many users say how helpful this GPIO header is, particularly for beginners."

John adds that "we hope to see a colour-coded header on the main Raspberry Pi boards in the future!"

The colour-coded Pi Zero 1.3 costs £9.16, the Pi Zero W £12.50, and the header only £1.25, all from **uk.pi-supply.com**.

NEW DIDDYBORG ROBOT

Below The new DiddyBorg v2 is now as fast as the larger, racier MonsterBorg but still has plenty of torque



The v2 proves red ones are faster!

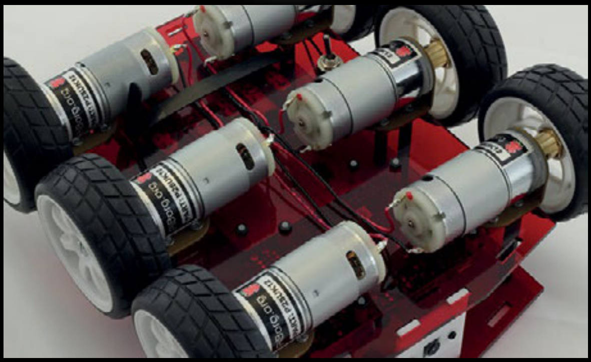
One of the most popular robot kits has received an update, with more powerful motors and a newly designed chassis that's still compatible with add-ons such as the touchscreen and camera mounts.

Timothy Freeburn, director of Freeburn Robotics and PiBorg, explains that "the DiddyBorg v2 Red is essentially a high-speed version of the DiddyBorg."

We reviewed the DiddyBorg V2 in *The MagPi* #70 (**magpi.cc/70**).

While Timothy and the design team "wanted the DiddyBorg v2 Red to be a similar speed to our very popular MonsterBorg kit", merely changing the gearing on the existing DiddyBorg motors would increase the speed, "but the torque would come right down, which would have made tank steering impossible."

The extra grunt of the DiddyBorg v2 comes from six 590 rpm, 28 mm motors. Timothy notes that "you can easily upgrade the motors" of a DiddyBorg v1 with six P28UK12 motors for £64.80 (total) from **piborg.org**. The new DiddyBorg v2 Red costs £220 from the same site.



ALPINE OS COMES TO THE PI

Light, secure, and fast

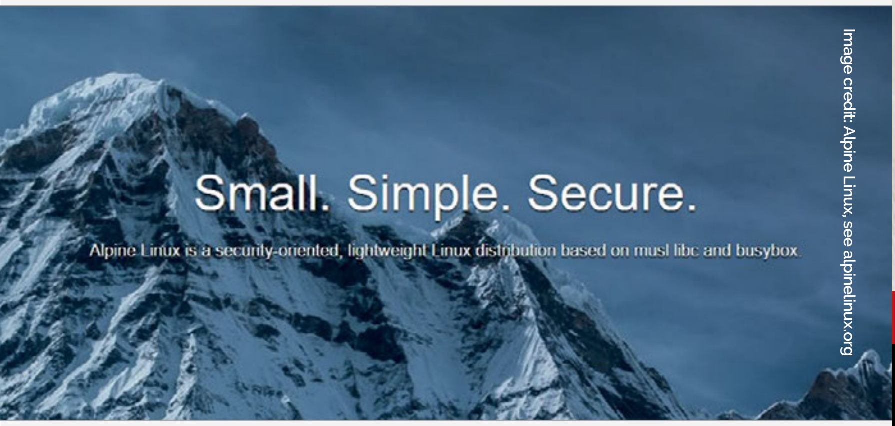
The latest version of Alpine Linux, an 'independent, non-commercial, general-purpose Linux distribution designed for power users', supports 64-bit Raspberry Pi models, including the Pi 3B+.

While version 3.2 of Alpine Linux supported the 32-bit Raspberry Pi 2, the new release uses an aarch64 image to support 64-bit Pi devices – you can download this image for free from **alpinelinux.org/downloads.**

Alpine Linux 'is built around musl libc and Busybox', giving the distro a claimed install size of 'around 130MB' while 'a container requires no more than 8MB'.

Alpine Linux is said to be 'designed with security in mind' by using an 'unofficial port of grsecurity/PaX' while 'all userland binaries are compiled as position-independent executables (PIE) with stack-smashing protection.' Which surely all makes Alpine Linux a sound choice for 'a home PVR, or an iSCSI storage controller, a wafer-thin mail server container, or a rock-solid embedded switch', according to the About page.

> "Alpine Linux is a sound choice for a home PVR, or an iSCSI storage controller"



Small. Simple. Secure.

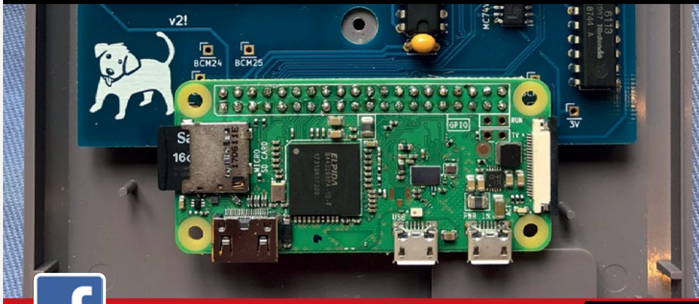
Alpine Linux is a security-oriented, lightweight Linux distribution based on musl libc and busybox

Image credit: Alpine Linux, see alpinelinux.org

Above Alpine Linux is a small, focused OS ideal for making Pi-based single-use devices

NOW TRENDING

The stories we shared that flew around the world



REVERSE-EMULATED SNES

magpi.cc/hNHkYT

To make a point about the nature of humour, Tom Murphy VII 'reverse-emulated' an unmodified NES to run SNES games via a Raspberry Pi embedded in the NES game cartridge.

TESLONDA

magpi.cc/plSXSC

What do you get if you cross a Tesla engine with a 1981 Honda Accord? Not much, unless you then integrate a Raspberry Pi to unleash the 536 hp and achieve 0–60 mph in 2.48 seconds!

THERMAL PAPER POLAROID

magpi.cc/gbPlac

Tim Jacobs revived the thrill of instant photography by incorporating a Pi Zero, a £2 webcam, and a tiny thermal printer in an old Polaroid camera body.

PI-MONITORED SOURDOUGH STARTER

Use a Pi for the perfect loaf

Justin Lam, a mechatronics engineer (in training), had a problem: how to make the perfect sourdough bread. He applied his knowledge of Linux, the Raspberry Pi, and image and statistical analysis to predict peak fermentation to achieve the perfect crumb.

Sourdough bread uses slightly unpredictable 'wild yeast' rather than the "stable and consistent" instant dry yeast used in standard bread, Justin explains. But, instant yeast "lacks the complex flavours that wild yeast [in] sourdough starter provides."

To bake a sourdough loaf, you need to take some starter (a mixture of flour and water, exposed to the air to 'catch' the wild yeast) and 'feed' it with some flour and water. You then have to wait for this mixture to achieve maximum fermentation before proceeding to make a loaf. If you start making that bread dough too early or too late in the fermentation process, the resultant loaf won't be good eating.

Even worse, Justin tells us that while "the time to max fermentation will be consistent" if you use the same volumes of starter and feed, "temperature and humidity [changes] throughout the season will alter" the time until maximum fermentation.

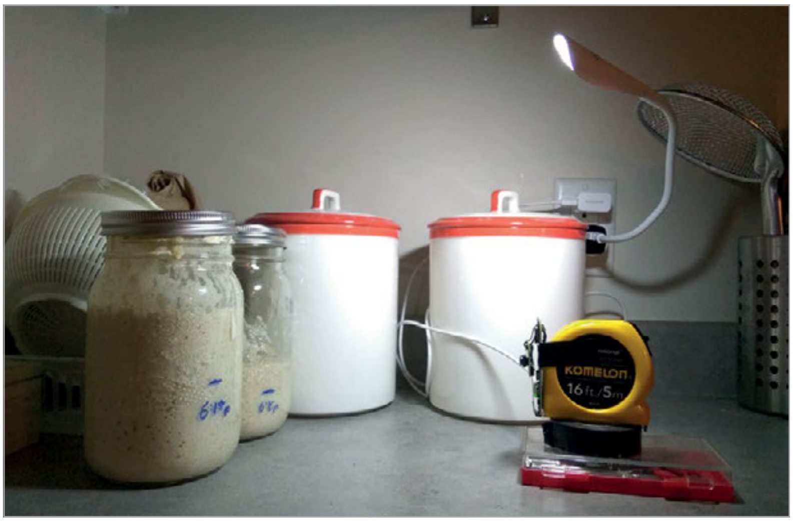
BreadCam

Justin's solution was to monitor the rise of the starter with a Pi Zero fitted with a Raspberry Pi camera. Justin says, "I was able to leverage my previous experience with image analysis" to build a system that could accurately monitor the fermentation process in a jar.

Justin used the Pi Zero "for speed of development" and because "the RPi community is a significant advantage over other platforms."

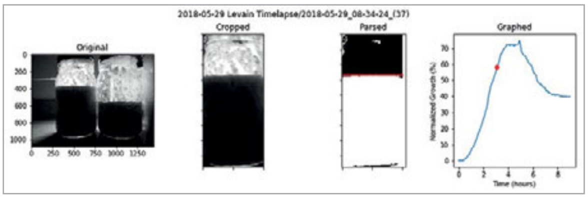
He then used a 'threshold algorithm' to measure the level of the starter in the jar, graphing the rise over time. His starter typically achieved peak fermentation after eight hours, with a window of around one hour to start forming a loaf-sized dough to bake.

Justin says the analyses he did were "extremely simple", using Scikit-learn. The details are well documented on Justin's blog – see **magpi.cc/cCGydQ**.



Far Left The Pi Zero with camera was strapped to a tape measure; the jars of sourdough starter were backlit in the final project

Below Justin says his analyses of the visual and statistical data were "extremely simple" – perhaps 'relatively' might be more accurate?



Above Use an ARM-powered device, such as the Raspberry Pi, to invent an autonomous robot

HACKSTER AND ARM AUTONOMOUS ROBOT CHALLENGE

Win some goodies with your autonomous vehicle design

Hackster has joined with processor designer ARM to launch the Autonomous Robot Challenge, 'to push the limits of what low-cost, open-source hardware and deep learning' can do for humanity.

The basics of the challenge are to design a machine based around an ARM device (such as a Raspberry Pi) that can do at least one of these two tasks: 'autonomously transport a package in an urban, rural, or underwater environment'; 'autonomously assist in a real-world scenario'.

> "Autonomously transport a package in an urban, rural, or underwater environment"

Adam Benzion, CEO of Hackster, explains, "We feel that the era of working drones, driving machines, diving systems has arrived. ... We want to see how society can benefit from these new capabilities and combine the trifecta of low-cost UAVs, AI, and [machine learning] into new inventions."

Rex St John, Senior Manager, IoT Ecosystem at ARM, adds: "We feel there is a good side to the topic of autonomous machines that we want to highlight and inspire the creativity of developers to think more broadly about how this low-cost AI technology can be used for positive reasons."

You can enter by "submitting code, schematics, and BOM [bill of materials]," reveals Adam, adding that "it has to be a functional project. Not perfect, but inventive and working."

The Autonomous Robot Challenge is offering some great prizes for various categories, such as an X PlusOne HD drone for the two projects making best use of AI, and a Geth-spider-like Robugtix T8X robot for the two most creative projects. See **magpi.cc/TzzoWh** for details.

Participation is open until 30 September 2018, 11:59pm Pacific Time. Winners will be announced by 15 October 2018.

Below The open-source DonkeyCar is a suggested device that might form a useful basis for your project. See **donkeycar.com** for more details

SUBSCRIBE TODAY FROM JUST £5

YOUR OFFICIAL RASPBERRY PI MAGAZINE
The MagPi
The official Raspberry Pi magazine
Issue 72
August 2018
raspberrypi.org/magpi
MATHEMATICA & MINECRAFT
Building blocks by the numbers
LEARN VIDEO GAME LOGIC
Discover the brains running games
ai MADE EASY
Artificial intelligence projects for Raspberry Pi
KNIT YOUR OWN SENSOR
Craft an interface for Raspberry Pi
MANAGE YOUR FILES
MINI

SAVE UP TO 35%

£5 FREE! MODMYPI VOUCHER FOR ALL SUBSCRIBERS

MODMYPI

Pricing

Rolling Subscription

- £5 a month
- Quick and easy to set up
- No long-term commitment

* Leave any time applies to Rolling Subscription only

Subscribe for a year:

£55 (UK)

£80 (EU)

£90 (USA)

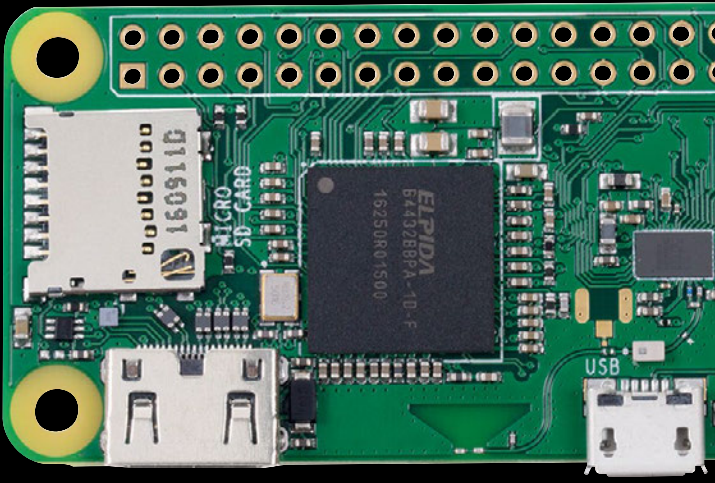
£95 (Rest of World)

Subscription benefits:

- FREE! Delivery to your door
- EXCLUSIVE! Raspberry Pi offers and discounts
- NO OBLIGATION! Leave any time*

magpi.cc/subscribe

JOIN FOR 12 MONTHS AND GET A FREE PI ZERO W STARTER KIT WITH YOUR SUBSCRIPTION



WORTH £20

Subscribe in print for 12 months today and you'll receive:

- Pi Zero W
- Pi Zero W case with three covers
- USB and HDMI converter cables
- Camera Module connector

SUBSCRIBE ON APP STORES

Available on the App Store

GET IT ON Google Play

FROM £2.29

BUY IN PRINT WORLDWIDE! magpi.cc/store
The MagPi
The official Raspberry Pi magazine
ai MADE EASY
MATHEMATICA & MINECRAFT
LEARN VIDEO GAME LOGIC
MANAGE YOUR FILES
KNIT YOUR OWN SENS
MAKE MUSI
HOW TO BUILD YOUR OWN MINI MAGIC MIRROR
RUN ANDROID ON RASPBERRY PI

ai MADE EASY

Give your Raspberry Pi an IQ boost with some AI projects

Last year we released a very special issue of *The MagPi* that included the Google AIY Projects Voice Kit, and it was a huge smash. People went on to send us loads of fun voice assistant projects made out of the kit, getting their introduction to the world of artificial intelligence in the process.

With the maturation of cloud computing, image recognition, and voice assistants, AI is quickly becoming something that the masses can have access to.

What exactly is AI and machine learning, though? And how can you use it with a Raspberry Pi?

Read on to find out....

WHAT IS AI?

JOSE MARCIAL PORTILLA

Jose is the Head of Data Science at Pierian Data Inc, and an AI expert. He has several AI learning courses you can find here: magpi.cc/rxcYLk

The term ‘artificial intelligence’ was coined in 1956 at a workshop in Dartmouth College by John McCarthy, a few years after Alan Turing had written his now famous paper ‘Computing Machinery and Intelligence’ in 1950.

A common misunderstanding is to conflate the terms ‘machine learning’ and ‘artificial intelligence’. Machine learning is the use of algorithms that have the ability to learn from a data source, allowing the algorithm to then create predictions or decisions based on that data. Examples include spam email classification, housing price prediction, and product recommendations in e-commerce.

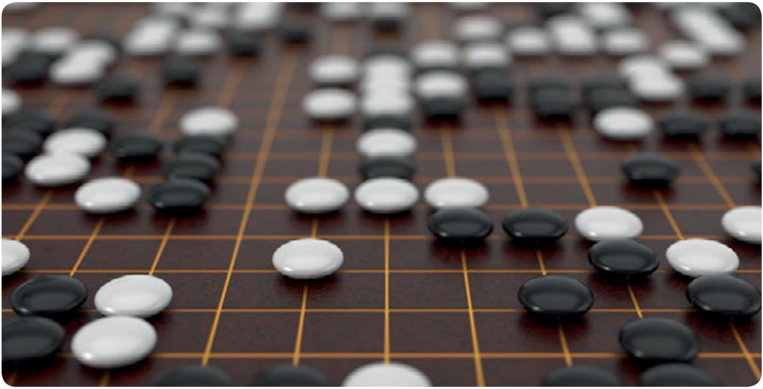
While machine learning is extremely useful, we typically don’t think of these single machine-learning algorithms as ‘intelligent’ in the same way we think of humans as ‘intelligent’. This is why machine learning is a subset of the larger goal of artificial intelligence.

There are also instances of what is commonly referred to as ‘narrow AI’ tasks; these include more complex applications of neural networks (a specific framework of machine-learning algorithms modelled after biological neurons). These narrow AI tasks can include things such as image classification, language translation, and face recognition. There have been huge advancements in this field over the past few years due to a large increase in availability in computational power, especially due to advancements in GPU performance.

We have recently seen developments from Google in these sort of narrow artificial intelligence tasks, including creating a computer that can defeat the world’s best Go players (**magpi.cc/FRLdsa**), developing self-driving cars with WayMo (**magpi.cc/inTtzd**), and creating a Google Assistant capable of calling and creating appointments with interactions (**magpi.cc/itbNbz**). These developments help pave the way towards the creation of ‘strong AI’, which are artificially intelligent systems that become indistinguishable from the intelligence of humans. This is a future where AI can begin to develop music, create works of art, and hold a normal conversation with a human.

While there have been many developments in these individual topics, we’re still far away from a truly artificially intelligent system, but working on these more narrow AI problems can help researchers solve the issues that may face them when pursuing general strong AI.

> “We’ve recently seen developments from Google in these sort of narrow artificial intelligence tasks”



AlphaGo Zero learns to play Go (a board game) without help. After 70 hours it becomes a superhuman player and after 40 days trains itself to become, arguably, the best Go player on Earth **magpi.cc/oSPVEz**.

LINE FOLLOWING WITH OPENCV

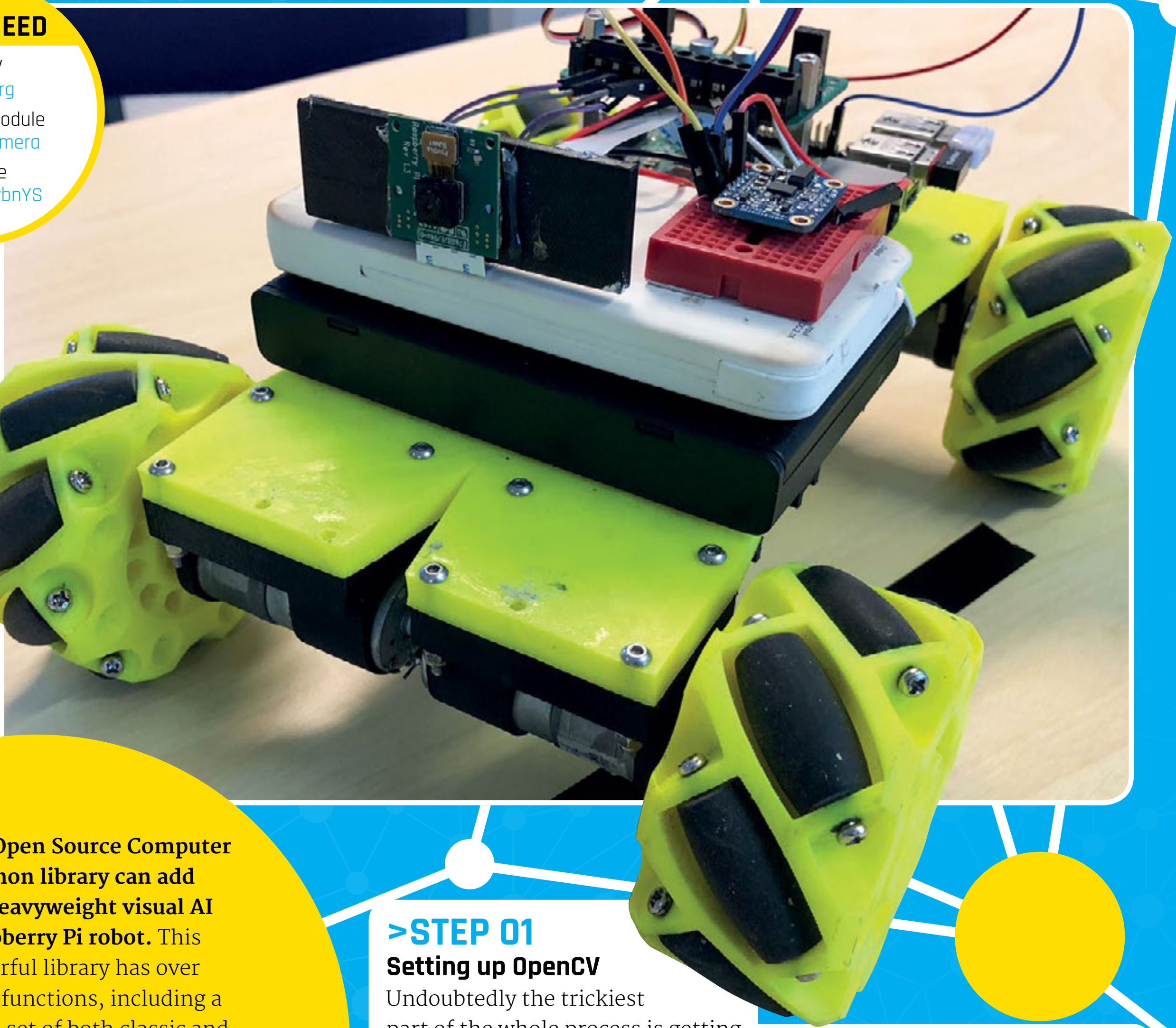
Is your robot just a bit too… robotic?

YOU'LL NEED

OpenCV
opencv.org

Pi Camera Module
magpi.cc/camera

Full code
magpi.cc/FwbnYS



The OpenCV (Open Source Computer Vision) Python library can add some fairly heavyweight visual AI to your Raspberry Pi robot. This hugely powerful library has over 2500 different functions, including a comprehensive set of both classic and state-of-the-art computer vision and machine-learning algorithms. These algorithms can be used to detect and recognise faces and identify and track objects. In this example, we'll get you started by showing how OpenCV can be used to detect a line and how we can then train a robot to follow it.

>STEP 01

Setting up OpenCV

Undoubtedly the trickiest part of the whole process is getting OpenCV installed and running on your Raspberry Pi. The installation process depends on which Pi / OS setup you are using. Fortunately, there are some very good guides out there, including this one by the very awesome Adrian Rosebrock: magpi.cc/PwLKfE.

>STEP 02

Capture an Image

Capturing images is done with the `picamera.capture()` function. We set the resolution fairly low (320×240) as this keeps the image size small and lowers the processing power required to analyse each one. Once the image is captured, we also then crop it to ensure we get just the central part that we're interested in.

>STEP 03

Blur the image

As we're not really interested in the actual details of the image, we apply a Gaussian blur on the whole thing to blur out unnecessary details. This helps level out any noise in the image. Don't worry about the name – this is just an OpenCV function.

>STEP 04

Use findContours() function to find the line

Firstly, we make a negative of the image as this makes it easier for OpenCV to detect edges. We then use the `findContours()` function to find the edges of the line. With some simple maths, we can then work out exactly where in the image the line appears.

>STEP 05

Let's get moving

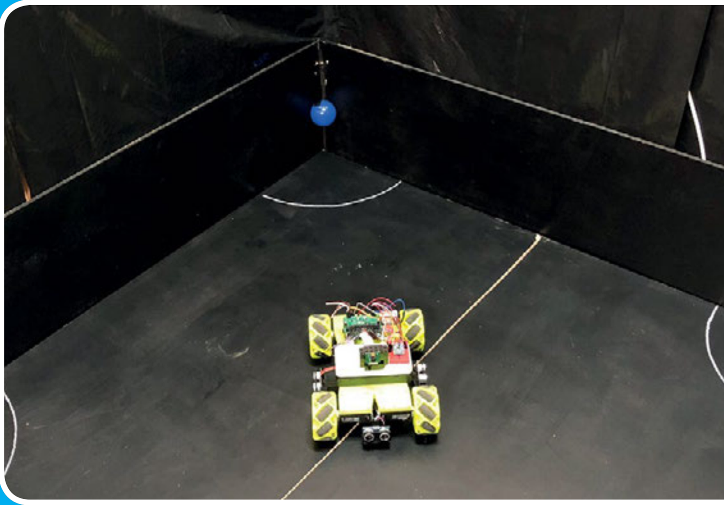
Now we know where in the image the edges of the line are, we can instruct our robot to move accordingly. There is some very simple logic that can be applied at this point. The pseudocode below should help.

```
if LINE in centre of image:
GO STRAIGHT ON

if LINE is on left of image:
TURN LEFT

if LINE is on the right of
the image:
TURN RIGHT
```

This will work, but you can get a lot more subtle and complex than this if you want. For example, speed for both turning and driving can be altered depending on how far away from the centre the line is.

Now, it doesn't matter what robot platform you use, (think CamJam EduKit or the Tiny 4WD from Coretec Robotics) all of the above will remain the same. It is only the instructions to actually drive the motors that will change depending on your setup.

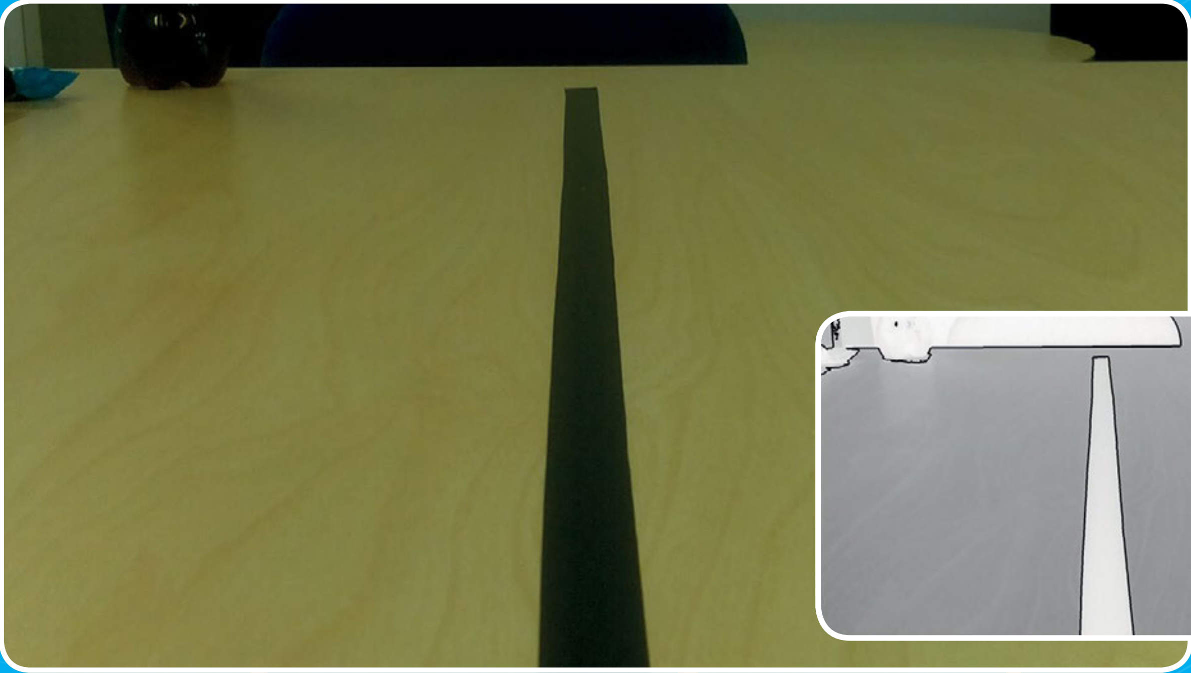
A full version of the code described above can be found here: magpi.cc/FwbnYS.

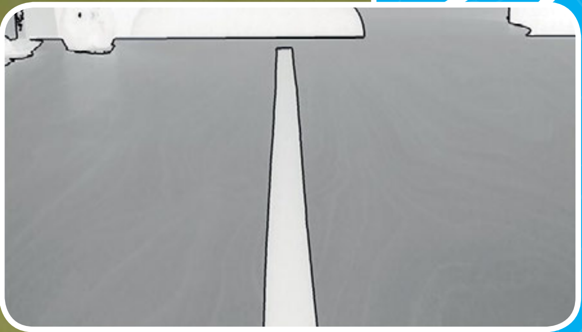


FURTHER ADVENTURES SOMEWHERE OVER THE RAINBOW!

There is much, much more that you can do with OpenCV. In the 'Somewhere Over the Rainbow' event in Pi Wars this year, robots had to autonomously locate and then drive to four different coloured balls set in the corners of a 1.2 m square 'arena' (Some teams were more successful at this than others!) The colour/object tracking teams used was mainly OpenCV-based and takes a similar approach to finding a line. Except it's a ball. And there's four of them. And they're all different colours!

magpi.cc/aUzwfk





Left The picture the Pi Camera takes of the line in front of it

Below Performing some editing techniques makes it readable to the Pi

IMAGE RECOGNITION WITH MICROSOFT AI

It's remarkably simple to use this computer vision service to give you a description of what the Raspberry Pi can 'see'

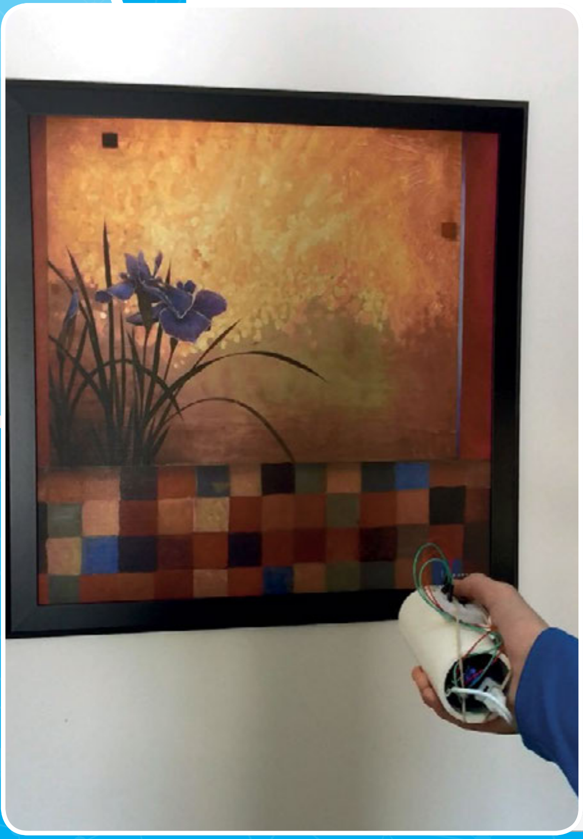
Microsoft's Computer Vision Cognitive Service is a feature-rich, cloud-based API providing image analysis that can be easily and quickly integrated into your project. In addition to providing a description of what it 'sees', the service is able to categorise and tag images, detect human faces, and recognise text, among other features. Pricing is free for up to 5000 transactions per month, and thereafter $1 or less per 1000 transactions for core features.

Send a picture

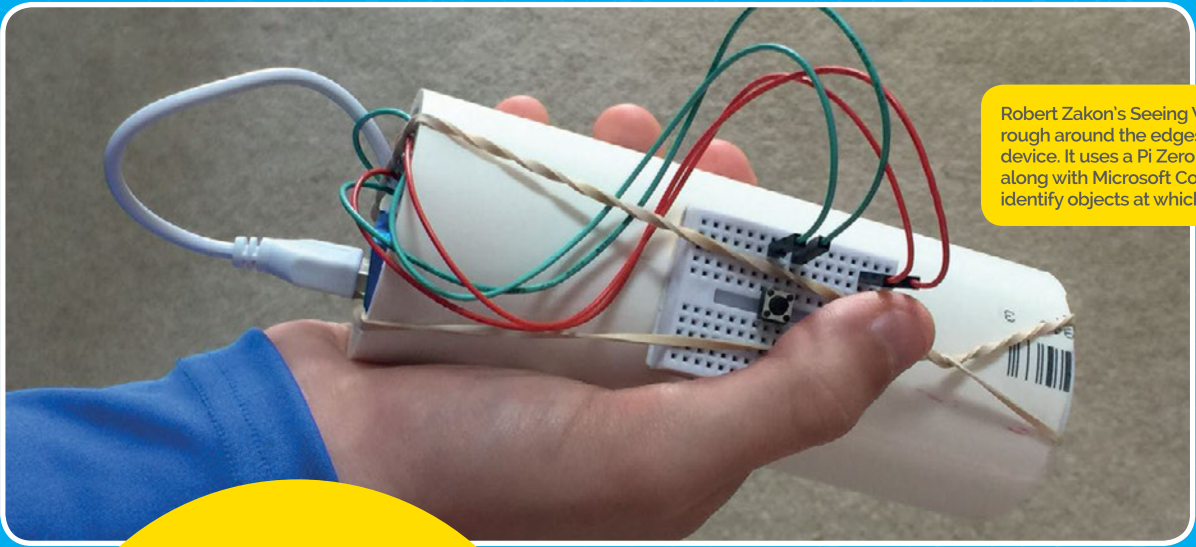
In order to get started with the Computer Vision service, an API key is required and may be obtained at **magpi.cc/dMRkhi**. Using the API is simply a matter of sending a POST request with the API key, an image, and a list of the desired visual analysis features, then processing the returned result.

The image may either be a URL or a file. The visual features include: Categories, Tags, Description, Faces, ImageType, Colour, and Adult. Additional details may also be requested such as identifying celebrities and landmarks. A full list of all the options and what each provides is available on the API documentation page at **magpi.cc/vOeFzE**.

The **recognition.py** listing is an example in Python for requesting a description of an image stored locally in the file **/tmp/image.jpg**. The returned result will be of the form shown below:



```
{
  "description": {

    "captions": [
          {
            "text": "The description of the image appears here",
            "confidence": 0.9234897234987234
          }
    ]
  },
  "requestId": "c11894eb-de3e-451b-9257-7c8b168073d1",
  "metadata": {
    "height": 600,
    "width": 400,
    "format": "Jpeg"
  }
}
```

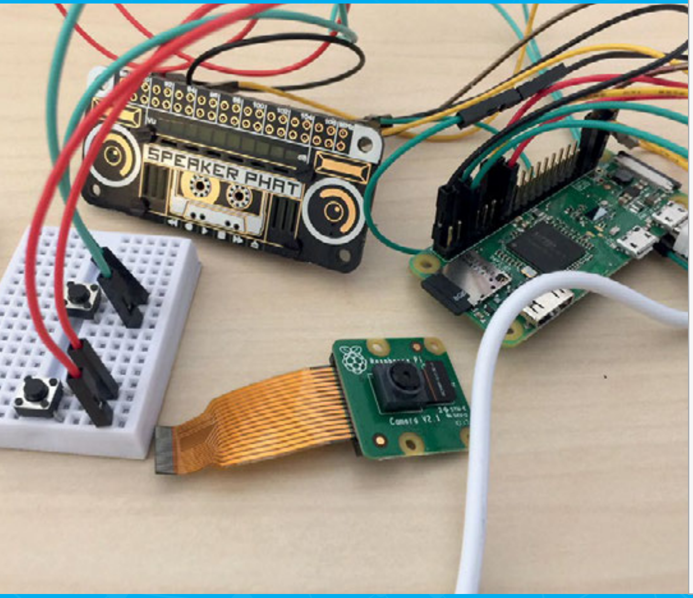


Robert Zakon's Seeing Wand may look a little rough around the edges, but it's a very smart device. It uses a Pi Zero and Camera Module, along with Microsoft Cognitive Services, to identify objects at which it is pointed.

SEEING WAND

The basics of this project can be applied to build a 'Seeing Wand', a project that tells you what it's being pointed at! You can find out more details on how to build one here: **magpi.cc/pfOPwB**. The wand was built to help a blind person 'see' a bit more of the world around them – it's a bit of a makeshift contraption though, using a breadboard and a PVC tube. Still, it's pretty wonderful.

In order to learn more about the Computer Vision service and test it out using your own images (without having to write code if you wish), check out: **magpi.cc/fFLtpJ**.



recognition.py

CODE:

LANGUAGE: PYTHON

DOWNLOAD: magpi.cc/DWcGRT

```python
#!/usr/bin/python
import httplib, urllib, base64, json, re

# CHANGE {MS_API_KEY} BELOW WITH YOUR MICROSOFT VISION API KEY
ms_api_key = "{MS_API_KEY}"

# setup vision API
headers = {
    'Content-Type': 'application/octet-stream',
    'Ocp-Apim-Subscription-Key': ms_api_key,
}
params = urllib.urlencode({
    'visualFeatures': 'Description',
})

# read image
body = open('/tmp/image.jpg', "rb").read()

# submit request to API and print description if successful or error otherwise
try:
    conn = httplib.HTTPSConnection('westcentralus.api.cognitive.microsoft.com')
    conn.request("POST", "/vision/v1.0/analyze?%s"%params, body, headers)
    response = conn.getresponse()
    analysis=json.loads(response.read())
    image_caption = analysis["description"]["captions"][0]["text"].capitalize()
    conn.close()
    print image_caption

except Exception as e:
    print e.args
```

SPEECH RECOGNITION WITH ZAMIA

YOU'LL NEED

- Active speakers
- USB microphone
- eliza.py
- magpi.cc/DUkkTT

AI projects don't always have to be connected to the internet, and Zamia shows us how to do offline voice recognition

Voice assistants are all the rage these days – but most of them rely on cloud services requiring an internet connection, which may cause data privacy issues and obscure their inner workings.

>STEP 01

Installation

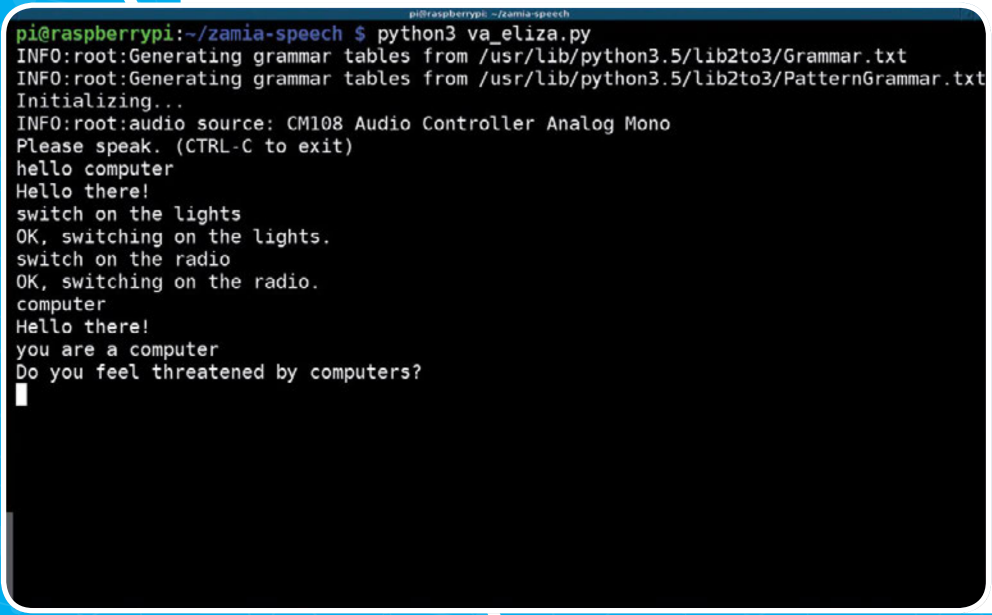
Most of the more advanced speech-recognition software we are using here is not yet part of Raspbian. We will therefore rely on a third-party repository from the Zamia-Speech project.

To set up the additional APT source, execute with root permissions (**sudo -i**):

```
echo "deb http://goofy.zamia.org/repo-ai/raspbian/stretch/armhf/ ./" >/etc/apt/sources.list.d/zamia-ai.list
wget -qO - http://goofy.zamia.org/repo-ai/raspbian/stretch/armhf/bofh.asc | sudo apt-key add -
```

With that in place, you can install the required packages using APT (again with root permissions):

```
apt-get update
apt-get install python3-nltools python-kaldiasr-doc kaldi-chain-zamia-speech-en pulseaudio-utils pulseaudio
```



```
pi@raspberrypi:~/zamia-speech $ python3 va_eliza.py
INFO:root:Generating grammar tables from /usr/lib/python3.5/lib2to3/Grammar.txt
INFO:root:Generating grammar tables from /usr/lib/python3.5/lib2to3/PatternGrammar.txt
Initializing...
INFO:root:audio source: CM108 Audio Controller Analog Mono
Please speak. (CTRL-C to exit)
hello computer
Hello there!
switch on the lights
OK, switching on the lights.
switch on the radio
OK, switching on the radio.
computer
Hello there!
you are a computer
Do you feel threatened by computers?
```

>STEP 02

ASR on a WAV file

We start out small: to try out the Kaldi-ASR speech-recognition engine, we will feed it a simple recording of a spoken sentence. We will use one of the example files that comes with it (or any other 16 kHz mono WAV file, for that matter):

```
zcat /usr/share/doc/python-kaldiasr-doc/examples/dw961.wav.gz > dw961.wav
```

The code to run Kaldi ASR on this example is shown in the **wav_decoder.py** code.

>STEP 03

Live speech recognition

The **live_vad.py** program is the foundation of our voice assistant: we record samples from our microphone using Pulse Recorder, feed it into a Voice Activity Detector (VAD), and decode it using Kaldi ASR. The **MODELDIR** variable points to the embedded English Zamia-Speech speech-recognition

Ideal for offline speech recognition, Zamia Speech offers pre-built Kaldi ASR packages for Raspbian, complete with pre-trained models for English and German. Everything is free, cloudless, and open source

CODE:

LANGUAGE:
PYTHON

DOWNLOAD:
magpi.cc/zbtuSx

model which we installed from their APT source earlier. You can experiment with different volume settings if you are not satisfied with the recognition results.

>STEP 04

Text to speech

A voice assistant should not only recognise natural language, it should also be able to produce spoken answers to communicate back to the user. We will use eSpeak NG here for speech synthesis since it is free software and easy to set up. The **espeakng_tts.py** code is a short example of how to use it.

>STEP 05

Intents and actions

We need to figure out what the user wants us to do: the user's intent. We want to keep things simple for now so we will have just three intents: **`HELLO`** (say hello), **`LIGHTS`**, and **`RADIO`** (toggle the lights and the radio on or off). **`add_utt`** is a utility function that we use to create a static mapping between natural language utterances such as 'switch on the radio' and corresponding intents.

With that in place, we can simply look up the user's utterance to find the intent and generate a response. The **va_simple.py** code is our complete bot at this stage.

>STEP 06

Patterns and ELIZA

To make our bot tolerant against input variations, we can use patterns: 'switch the (radio|music) (on|off)' will already generate four patterns. Instead of a simple mapping between input and patterns, we will use their edit distance (the minimum set of insertion, deletion, and substitution operations to transform one into the other) to compute the closest match – this will make our bot tolerant against utterances we did not foresee. If we do not find a close enough match, we will turn to ELIZA to generate a response. The **va_eliza.py** code comprises the completed bot.

>STEP 07

Taking it further

There are endless possibilities to improve upon this voice assistant. You can get creative: make it tell jokes, say quotes from your favourite movies, or come up with your own witty responses. Or make it more practical: add a wake word and attention mechanism, make it tell the current time or date; or, if you do not need it to run offline, add online skills like fetching the weather report or looking up definitions on Wikipedia.

You can also improve its language capabilities: use more advanced speech synthesis tools like SVOX Pico or Google's Tacotron for more natural-sounding responses, or go to **zamia-speech.org** to learn how to adapt the speech-recognition model to your domain.

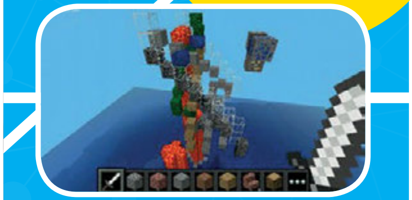
> “There are endless possibilities to improve upon this voice assistant”

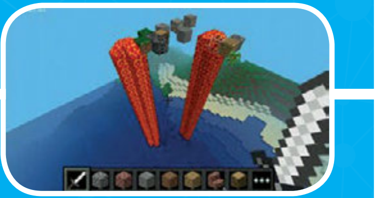
INSPIRATIONAL AI PROJECTS

More examples of some of the amazing things you can do with AI on Raspberry Pi

ARTIFICIAL LIFE PROJECT

MAKER: MICHAEL DARBY
AI TYPE: SIMULATION
magpi.cc/ZKcLUY





Something you'd more traditionally associate with AI, this simulation creates artificial lifeforms in Python code, giving them different traits and allowing them to 'breed' and die and such. It would be quite horrifying an experiment if they were real, but they're not, so it's fine. Anyway, it's all displayed beautifully on a Unicorn HAT.

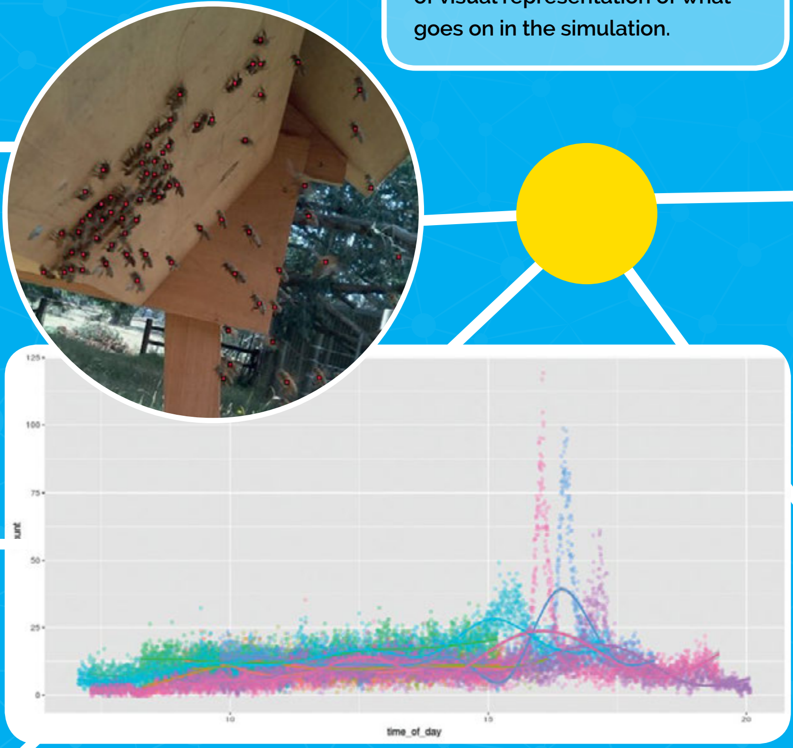
As a bit of a bonus, Michael figured out a way to plug it into Minecraft Pi to get a different kind of visual representation of what goes on in the simulation.

COUNTING BEES

MAKER: MAT KELCEY
AI TYPE: VISION
magpi.cc/BHXFpo

Like the name suggests, this project is used to count bees, specifically in a beehive. According to Mat on his blog about it, he couldn't find a decent enough, non-intrusive system to the trick. So he made one!

It uses a standard Raspberry Pi Camera Module, and the Python code uses TensorFlow (and some manual training) to detect the bees in the image. Over time it's become a lot more accurate, through a mixture of machine learning and image optimisation.



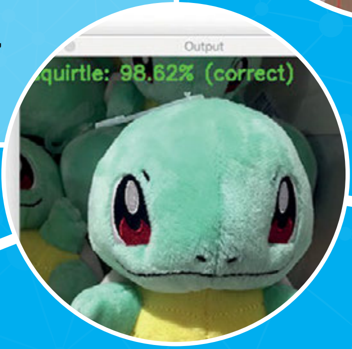
POKÉDEX

MAKER: ADRIAN ROSEBROCK
AI TYPE: VISION
magpi.cc/gYUpxN

There have been several Pokédex toys in the past 20 or so years since the Pokémon games were released. However, none possessed the actual capability to use the image of a Pokémon to identify it like in the games and TV show.

Using hundreds of images, from official artwork to renders and plushes and fan-works, Adrian managed to train a Raspberry Pi to be able to identify five of the 807 Pokémon currently in existence (minus different forms, regional variants, Mega evolutions, etc.).

The results? From his blog about it, pretty accurate. Maybe Nintendo will license it and put it in Pokémon GO?

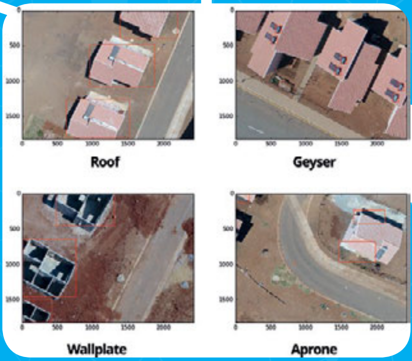


DRONE OBJECT DETECTION

MAKER: ARUN GANDHI
AI TYPE: VISION
magpi.cc/vZXAXa

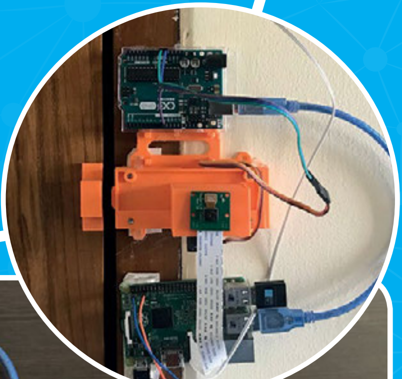
Drone technology, and drone automation, is always improving. Pairing drones with image recognition seems like a no-brainer, and this fusion of technology has been used in Africa to monitor house construction projects.

The images capture the various building stages of the houses in an attempt to make sure that if any problems arise, the team will know about it. Using nearly 1500 images, they trained the AI to expertly look out for the signs of different parts of house construction so it could be relayed back for inspection. This project was created with the NanoNets API (nanonets.com).



INTELLIGENT DOOR LOCK

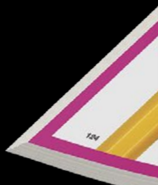
MAKER: KHAIRUL ALAM
AI TYPE: VISION
magpi.cc/Cbwaih



This project is pretty complex, but basically it's a smart lock. All you need to do is train it to recognise people you know and if they come to your door, it will let you know who is there. It's more complicated than just installing a camera you can stream over the web, but it means you can receive a text message about who it is rather than squint at a camera from across town.

This particular project uses Alexa, and it can also open the door without you being physically present. It will even welcome your guest. Proper *Star Trek* stuff.

MAKE | BUILD | HACK | CREATE
HackSpace
TECHNOLOGY IN YOUR HANDS
hsmag.cc
August 2018
Issue #09
ULTIMATE LED CUBE
50 TOOLS FOR BETTER MAKING
Everything you need to build (almost) anything
Remap project
Electronics for beginners
Pi Wars
RACHEL WONG
HYDROPONICS MASTER MIDI NAIL CLIPPERS MEASURE INERTIA



ISSUE **#09**

OUT NOW

REVIEW
TinyLily Mini by TinyCircuits
A wearable microcontroller, suitable for dolls
FORM VERSUS FUNCTION
SMALL, MODULAR, AND POWERFUL
VERDICT
8/10

Available on the App Store

GET IT ON Google Play

ANTHONY DIPILATO

Based in the Miami/Fort Lauderdale area of Florida, Anthony is a full-stack developer who enjoys designing, building, and coding things.
anthonydipilato.com

GHOST DETECTOR

Equipped with all manner of sensors, this Ghost Detector is designed to discover paranormal presences. A spooked **Phil King** investigates

Quick Facts

- Build details are at **magpi.cc/bLOSrp**
- Code is on GitHub: **magpi.cc/zbDYrV**
- The 'wooden' case is actually 3D-printed from PLA
- The project took two months to build
- The detector includes a cooling fan

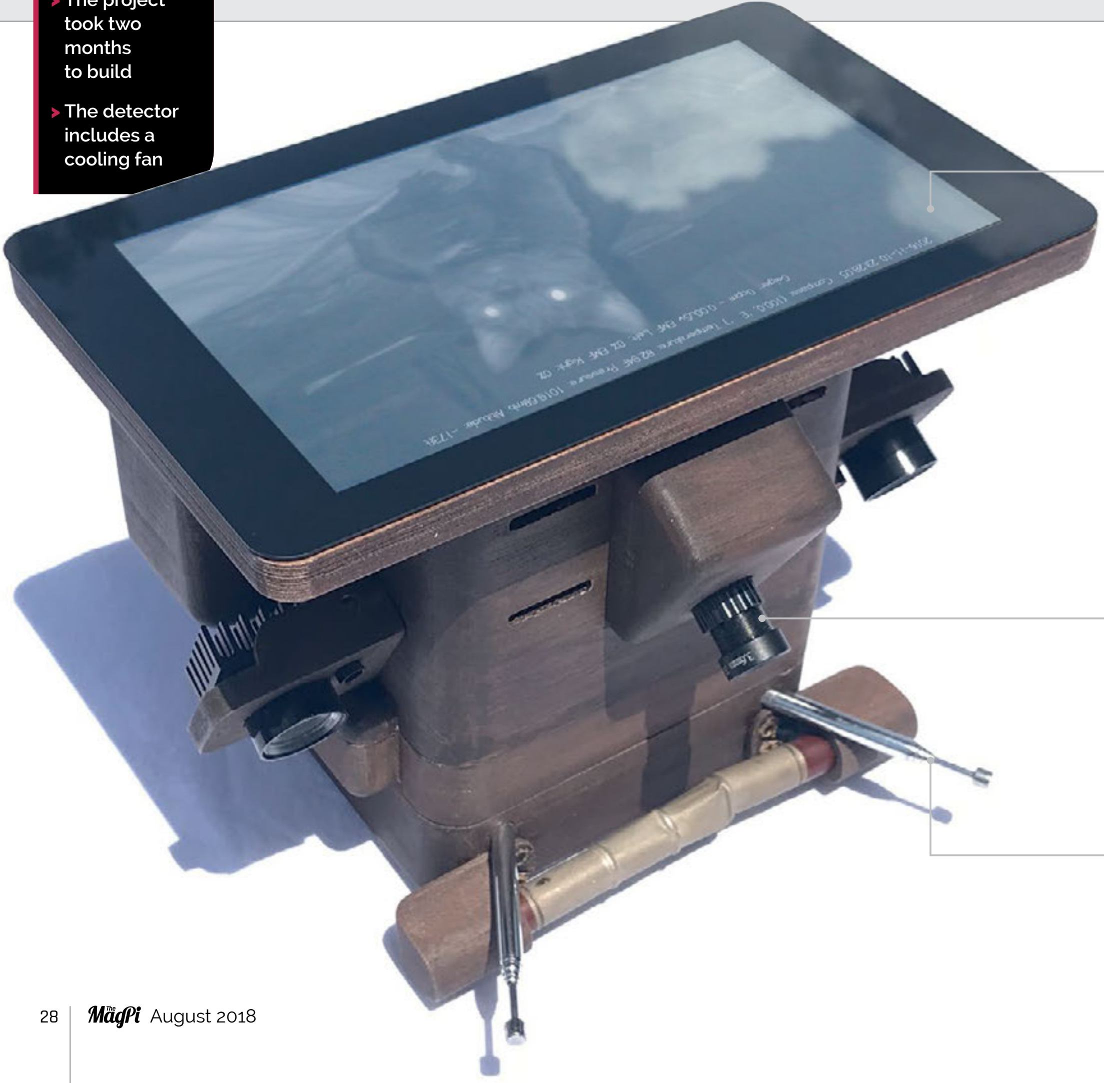
The truth is out there... At least that's what they used to say on *The X-Files*. Well, if there is any paranormal activity around, Anthony DiPilato's sensor-packed Ghost Detector aims to find it. While he built the device around two years ago, this top-secret project has only just been shared with the wider world.

The idea stemmed from a desire to create a home-made present for his father. "My dad watches a lot of paranormal investigation shows," says Anthony, "so I thought it would be a fun project to give as a Christmas gift."

Infrared camera

While the project started off as a simple Arduino-based EMF (electromagnetic field) meter, it quickly evolved into something far more ambitious. "I saw Raspberry Pi offers an infrared camera," recalls Anthony, "so I decided to build something that could record video with overlaid sensor data."

The Raspberry Pi records video, audio, and sensor data, then saves it to a USB flash drive. Mounted on top of the device, an official Raspberry Pi 7-inch touchscreen provides a user interface, while also displaying the data from the



A 7-inch touchscreen shows the live camera view overlaid with sensor data

Aided by two sets of IR LEDs, an infrared camera captures images in the dark

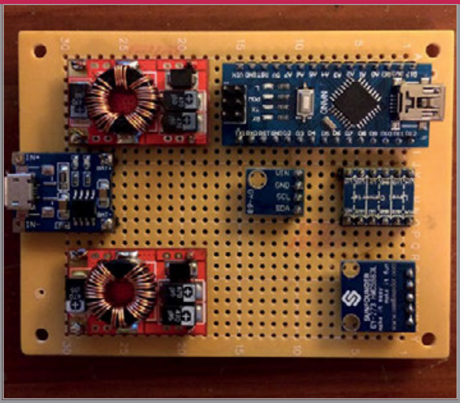
Twin antennas amplify the signal for the EMF sensors

BUILD A GHOST DETECTOR

>STEP-01

3D-printed enclosure

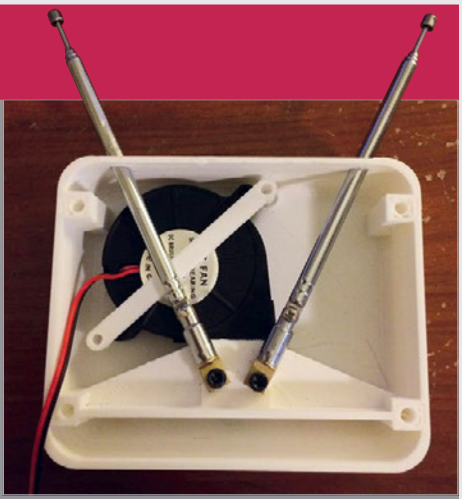
After creating several prototypes and making adjustments, Anthony 3D-printed the final enclosure from Hatchbox Wood PLA, then sanded and stained it for a wood-style finish.



>STEP-02

Add loads of sensors

A magnetometer and temperature/pressure sensor are mounted on stripboard, along with an Arduino Nano connected to dual EMF sensors. The Geiger counter is a separate board.



>STEP-03

Keep your cool

A cooling fan blows air into a duct that vents up the back of the enclosure. Added partly for aesthetic effect, twin telescopic antennas are linked to the EMF sensors.

numerous sensors and a live video view from the infrared camera.

Featuring a pistol-grip handle, the body of the detector was 3D-printed on Anthony's newly acquired Monoprice Maker Select. He designed the enclosure using the Autodesk Fusion 360 CAD software, which offers a free licence for hobbyists.

> Since it is a pseudoscientific instrument, I wanted to make it look as ridiculous as possible

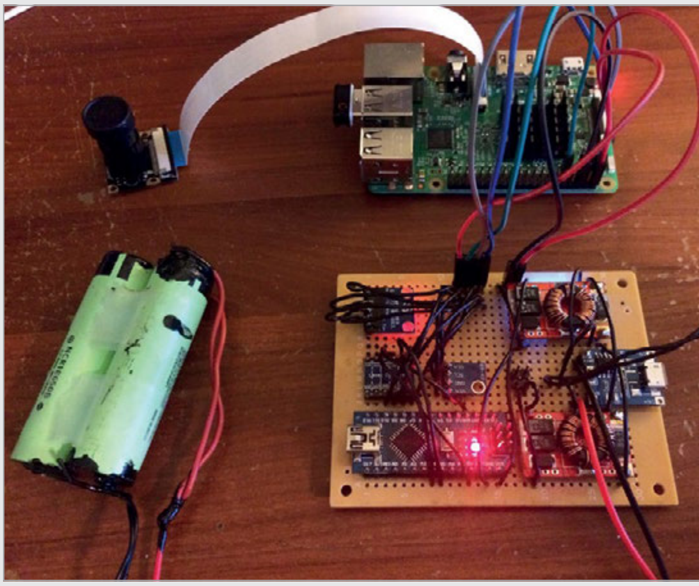
"Since it is a pseudoscientific instrument, I wanted to make it look as ridiculous as possible," he tells us. "So I included rabbit-ear telescopic antennas [for the EMF sensors] and a Geiger tube. I thought the stained wood enclosure would match that aesthetic."

Sensory overload

Continuing the theme of making it as ludicrous as possible, Anthony crammed the detector with "as many sensors as I could fit." Along with the EMF sensors, there's a magnetometer (compass), altimeter, temperature and barometric pressure sensor, microphone, and a Geiger counter to measure radioactivity. Most of the sensors and other electronics are mounted on stripboard, including two 5 V 3 A step-up power supplies, an Arduino Nano, and a logic level converter to interface the Nano to the Raspberry Pi.

The Geiger counter is a separate board, while its Geiger tube is mounted on the front along with the camera and two lots of infrared LEDs either side. To power the device, two Panasonic 18650 3400 mAh batteries are housed in the handle.

From start to finish, the Ghost Detector took Anthony around two months to build: "The only major issue I encountered was the control board for my 3D printer burned out, and needed to be replaced before I could finish the project."



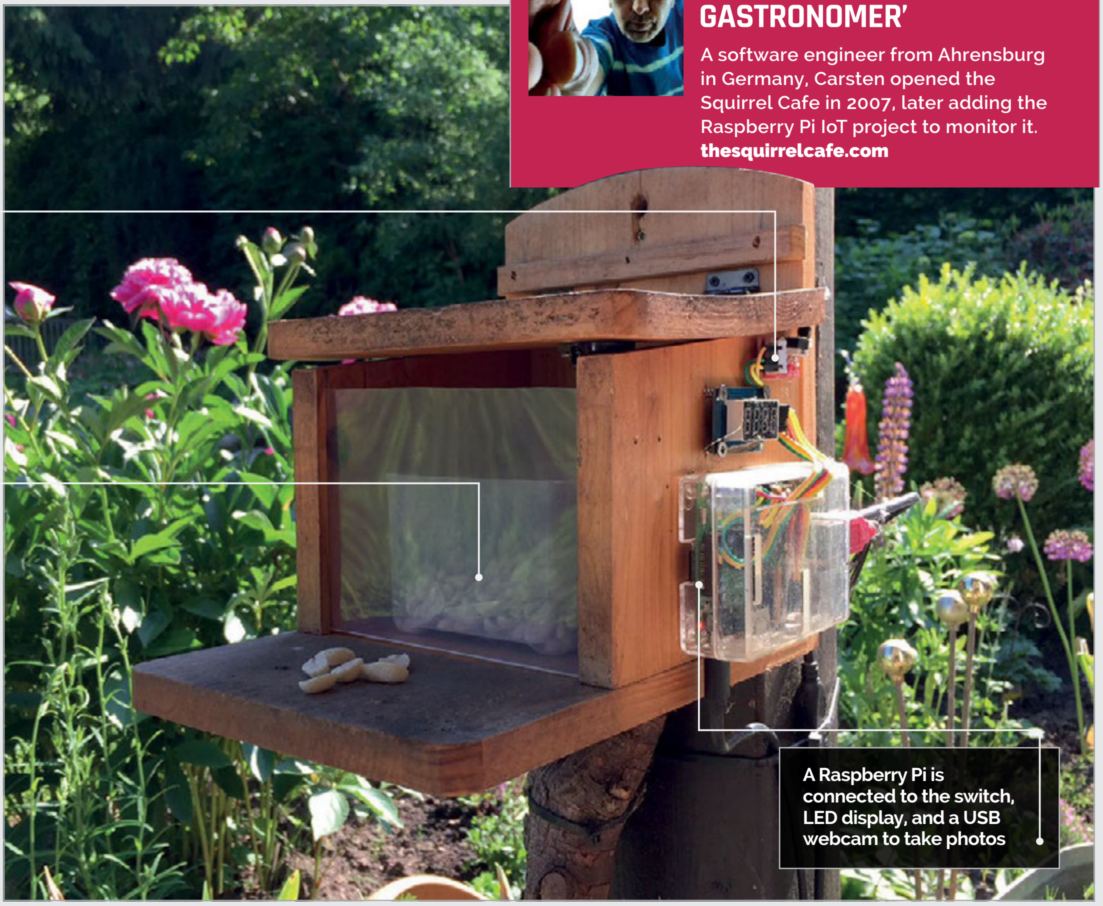
Above Anthony tested the electronics out before cramming them into the 3D-printed enclosure

It took him a mere two days to program the software, mainly comprising Python scripts.

Asked if he has detected any unearthly presences, Anthony replies, "I only tested at a few local places that are supposedly haunted, but I was not able to record any conclusive evidence of hauntings." He did discover that blood absorbs infrared light, however, showing up all the veins in his arm on the camera view – which looks pretty spooky.

CARSTEN DANNAT AKA 'THE SQUIRREL GASTRONOMER'

A software engineer from Ahrensburg in Germany, Carsten opened the Squirrel Cafe in 2007, later adding the Raspberry Pi IoT project to monitor it.
thesquirrelcafe.com



A mechanical switch is pressed whenever the lid is opened by a squirrel

The nuts are visible behind a glass panel

A Raspberry Pi is connected to the switch, LED display, and a USB webcam to take photos

THE SQUIRREL CAFE

Predict the weather with... squirrels and nuts!? **Nicola King** lifts the lid on an ingenious project

Quick Facts

- It correctly predicted the cold winter of 2017-18
- Carsten plans to add a scale to weigh the nuts...
- ...for more accurate measuring of nut consumption
- Raccoons have broken into the feeder
- A video 'security camera' now monitors all visitors

Back in 2012, Carsten Dannat was at a science summit in London, during which a lecture inspired him to come up with a way of finding correlations between nature and climate. "Some people say it's possible to predict changes in weather by looking at the way certain animals behave," he tells us. "Perhaps you can predict how cold it'll be next winter by analysing the eating habits of animals? Do animals eat more to get additional fat and excess weight to be prepared for the upcoming winter?"

An interesting idea, and one that Germany-based Carsten was determined to investigate further. "On returning home, I got the sudden inspiration to measure the nut consumption of squirrels at our squirrel feeder", he says. Four years later and his first prototype of the 'The Squirrel Cafe' was built, incorporating a first-generation Raspberry Pi.

A tough nut to crack

A switch in the feeder's lid is triggered every time a squirrel opens it. To give visual feedback on how often the lid has been opened, a seven-segment LED display shows the number of openings per meal break. A USB webcam is also used to capture images of the squirrels, which are tweeted automatically, along with stats on the nuts eaten and time taken. Unsurprisingly perhaps, Carsten says that the squirrels are "focussed on nuts and are not showing interest at all in the electronics!"

So, how do you know how many nuts have actually been eaten by the squirrels? Carsten explains that "the number of nuts eaten per visit is calculated by counting lid openings. This part of the source code had been reworked a couple of times to get adjusted to the squirrel's behaviour while grabbing a nut out of the feeder. Not always has a nut been taken out of the feeder, even if the lid has been opened." Carsten makes an assumption that if the lid hasn't been opened for at least 90 seconds, the squirrel went away. "I'm planning to improve the current design by implementing a scale to weigh the nuts themselves to get a more accurate measurement of nut consumption," he says.

Just nuts about the weather!

The big question of course is what does this all tell us about the weather? Well, this is a complicated area too, as Carsten illustrates: "There are a lot of factors to consider if you want to find a correlation between eating habits and the prediction of the upcoming winter weather. One of them is that I cannot differentiate between individual squirrels currently [in order to calculate overall nut consumption per squirrel]."

He suggests that one way around this might be to weigh the individual squirrels in order to know exactly who is visiting the Cafe, with what he intriguingly calls "individual squirrel recognition" – a planned improvement for a future incarnation of The Squirrel Cafe.

Fine-tuning of the system aside, Carsten's forecast for the winter of 2017/18 was spot-on when he predicted, via Twitter, a very cold winter compared to the previous year. He was proven right, as Germany experienced its coldest winter since 2012. Go squirrels!

> Some people say it's possible to predict changes in weather by looking at the way certain animals behave

Left A squirrel enjoying a tasty treat at the Squirrel Cafe

Below The results of a raccoon's rampage

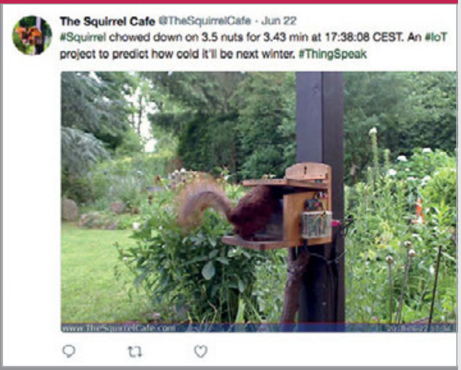
SECRET SQUIRREL

>STEP-01
Lid switch

When a squirrel opens the lid, a mechanical switch is triggered. This replaced a magnetic reed switch, which Carsten says wasn't totally reliable.

>STEP-02
Nut supply

The feeder is filled with peanuts. Since these are split into halves, it's assumed that each lid opening results in half a nut being consumed by the squirrel.



The Squirrel Cafe @TheSquirrelCafe · Jun 22
#Squirrel chowed down on 3.5 nuts for 3.43 min at 17:38:08 CEST. An #IoT project to predict how cold it'll be next winter. #ThingSpeak

>STEP-03
Tweepy tweet

After each meal visit, the Tweepy Python library is used to automatically tweet the details and a photo taken by a connected USB webcam.

ROBIN NEWMAN

Now retired, Robin was once the Head of IT at Oundle School in Peterborough. He currently enjoys experimenting with his Raspberry Pi boards.
magpi.cc/Dwyabx

SONIC PI GLOCKENSPIEL

This project strikes a chord for its clever use of Sonic Pi and LEGO to make sweet music, as **David Crookes** discovers

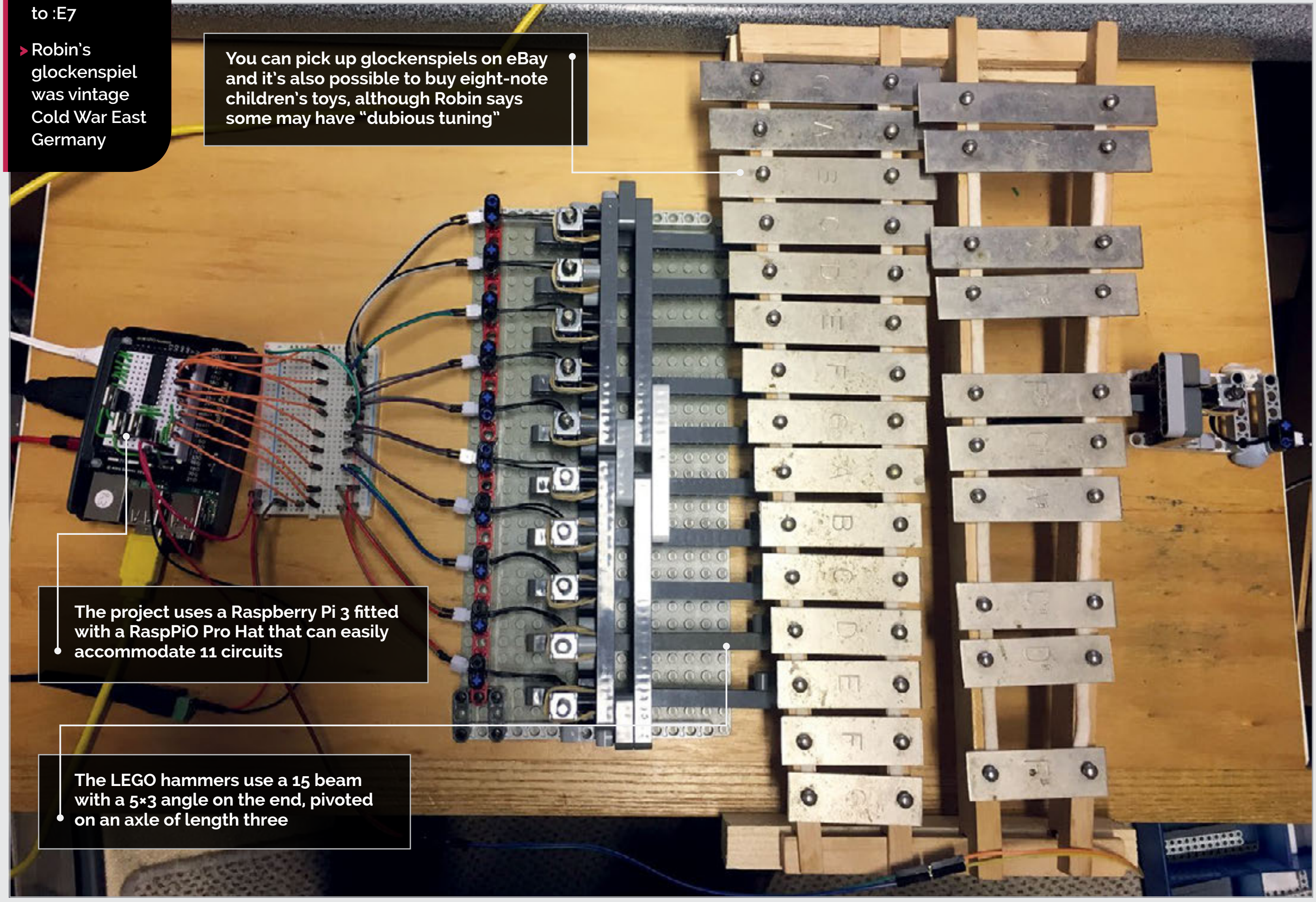
Quick Facts

- It uses Sonic Pi version 3.0.1
- LEGO forms the Pi-controlled hammers
- Spacing between the centre of notes is 24 mm
- The note range used is :C6 to :E7
- Robin's glockenspiel was vintage Cold War East Germany

Robots have already blown their own trumpet: Toyota developed a humanoid in 2004 which could play *When You Wish Upon a Star* by clasping the instrument in its hands and blowing air through its mouth. Since then, we've seen robots play the drums and guitar; even going as far as recording an album. But now we've heard the results of a Raspberry Pi playing a glockenspiel and it's been music to our ears.

It's all thanks to Robin Newman whose love of computers and music goes way back. In the 1980s, he networked 24 BBC Micros and had them play parts of a Bach Brandenburg Concerto. "Today, 80 percent of my work with Raspberry Pi boards involves Sonic Pi," he says. He got the idea for a Sonic Pi-controlled glockenspiel after seeing similar projects online that used an Arduino.

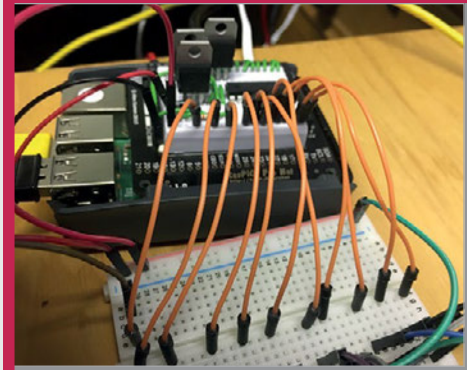
"Version 3.1 was a game changer because it allowed Sonic Pi to communicate with the outside world using either MIDI signals or Open Sound Control messages," he explains. "It enables Sonic Pi to interact with Python-controlled devices and to interact easily with


You can pick up glockenspiels on eBay and it's also possible to buy eight-note children's toys, although Robin says some may have "dubious tuning"

The project uses a Raspberry Pi 3 fitted with a RaspPiO Pro Hat that can easily accommodate 11 circuits

The LEGO hammers use a 15 beam with a 5×3 angle on the end, pivoted on an axle of length three

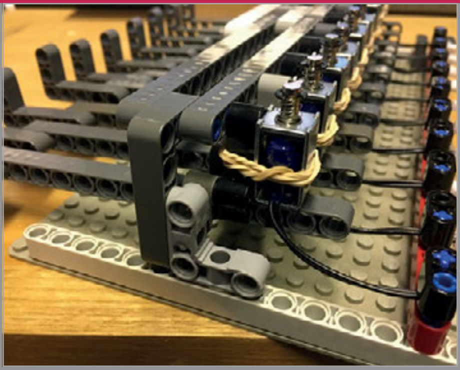
START MAKING MUSIC



>STEP-01

Wire the Pro Hat

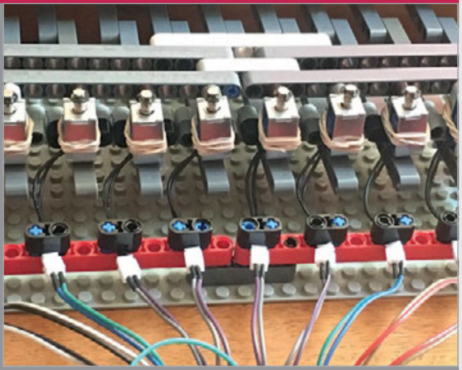
After grabbing a glockenspiel, Robin restricted the notes he'd use and accommodated the driver circuitry on a RasPiO Pro Hat. He used three TP102 Darlington power transistors. A breadboard was used to distribute signals to the solenoids.



>STEP-02

Get out the LEGO

The hammers were built from LEGO and, by happy coincidence, the width of the basic hammer mechanism was three LEGO beams (or 24 mm). Since ten notes on Robin's glockenspiel occupied 234 mm, it made for a good fit.



>STEP-03

Completing the setup

After wiring up the solenoids, Robin propped the glockenspiel on wooden blocks to allow the hammers to fit under the instrument. He used Sonic Pi to send OSC information to the Python script and specify the notes.

signals to and from the Pi's GPIO pins. I wanted to use the fact that it could control a glockenspiel and play itself at the same time to accompany the instrument."

Setting up

Robin already had a glockenspiel. A 30-year-old gift to his son, it was languishing in his attic. As such, he sought to produce an easily constructed project that could be added to the instrument. The Pi, he envisaged, would control hammers to strike the glockenspiel's metal bars and he decided to use solenoids as the actuators.

"I bought a 5V, Adafruit-sourced solenoid and I already had a suitable power supply to hand," he recalls. "I also had a power transistor and projection diode from an Arduino starter kit. I was able to connect them up to a GPIO pin and use the GPIO Zero **LED** command to switch it on and off. This worked fine and so the question was how could this small movement be used to hit the keys." It was then that he turned to LEGO.

> "80 percent of my work with Raspberry Pi boards involves Sonic Pi"

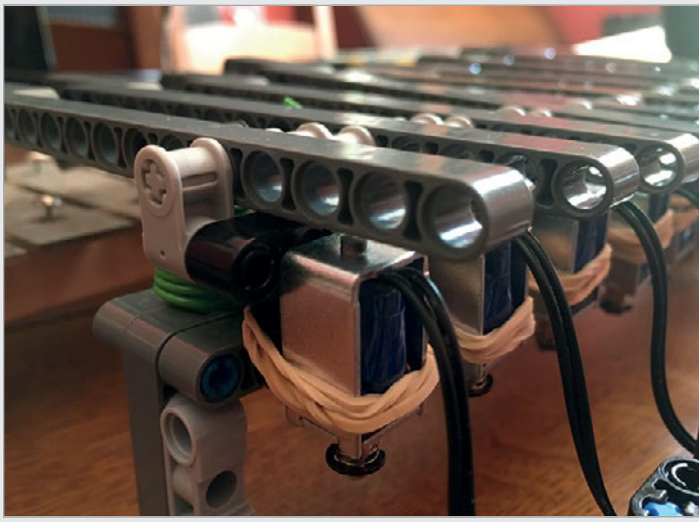
Hammer time

Before the Pi was launched, Robin had spent a few years working with the LEGO EV3 system, mainly producing colour-sorting robots. "After some experimentation, it turned out to be beautifully simple to produce a hammer mechanism out of LEGO which could then be driven by the solenoid, providing just the right kick," he says. To do this, he had the LEGO hammers strike the notes from underneath, allowing gravity to return them to their resting position. "It's important that the hammers strike the notes and then immediately move away from them so that they don't dampen the sound," he explains. From then on, the software could work its magic.

Robin wrote a Python script to drive the GPIO pins and therefore the solenoids, and he delayed the notes sent to the glockenspiel by 0.3 seconds. This compensated for the delay between the Sonic Pi playing a note and the note sounding in a speaker, allowing the Pi to accompany the glockenspiel without it sounding odd.

"I'm now looking at a system of overhead hammers," Robin reveals, keen to continue refining the project. "This will open up the range of glockenspiels that can be used."

Below Robin has been experimenting with overhead hammers, winding 38 cm of insulated 1/0.6 wire around a biro to form a weak spring, allowing them to return to a rest position clear of the metal notes



DIEMO NIEMANN

Diemo Niemann is the CEO of the Save-Nemo Foundation in Germany which is helping to save coral reefs from damage.
save-nemo.org

NEMO-PI

Coral reefs are threatened by divers and climate change, but one organisation hopes to save them by using a Raspberry Pi as an underwater 'weather station'. **David Crookes** reports

Quick Facts

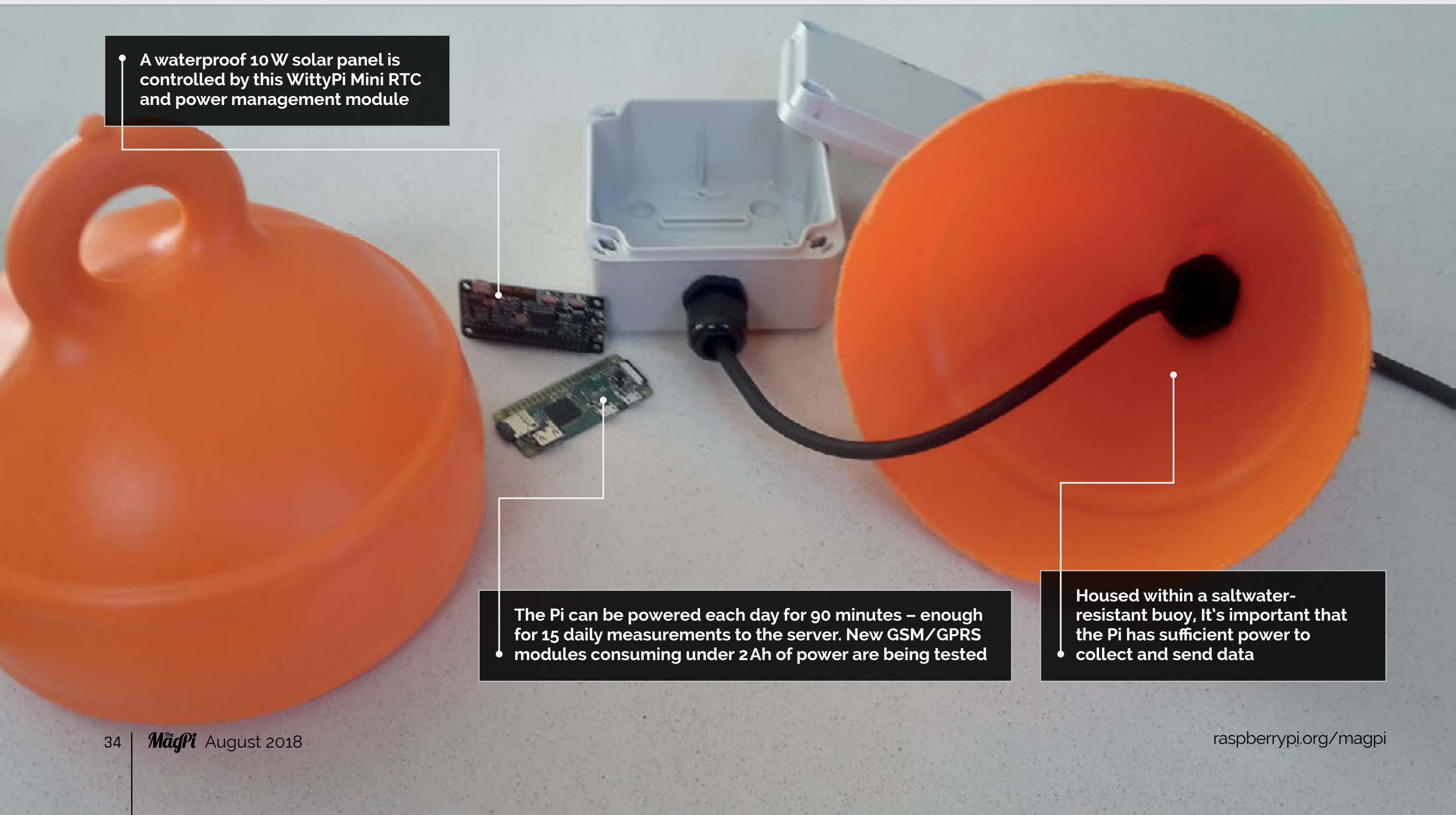
- Nemo-Pi has been tested to a depth of up to 50 m
- The first Nemo-Pi has been running for a year
- Each Nemo-Pi setup costs $600 max
- The price depends on the number of sensors
- Eight sensors are currently being tested

For the past two years, the Save Nemo Foundation has worked hard to protect coral reefs off the coast of Thailand and Indonesia. Its members have been sinking concrete blocks next to the reefs, allowing diving and snorkelling boats to safely moor by using them as anchor points. In doing so, they've eased the problem of boat crews dropping anchor directly into the reefs, which has already caused significant damage.

But while that has had a positive effect on the creeping destruction, the organisation spotted another opportunity. "We realised we could do more by making these moorings smart," says its CEO Diemo Niemann. "So we created a plan to collect underwater physical and chemical data that is not only essential for science, but helpful for local people and their business." The result? Nemo-Pi, a device able to measure temperature, visibility, pH levels, and the concentration of CO_2 and nitrogen oxide at each anchor point.

Oh buoy

Every one of the concrete moorings has a buoy attached on top and, as it bobs in the water, it shows boat crews where they can anchor. The idea behind Nemo-Pi is to put an encased Raspberry Pi into the buoy, attach it to a solar panel for power, include a GPS device so that its location can be determined, and run an array of sensors from



A waterproof 10 W solar panel is controlled by this WittyPi Mini RTC and power management module

The Pi can be powered each day for 90 minutes – enough for 15 daily measurements to the server. New GSM/GPRS modules consuming under 2 Ah of power are being tested

Housed within a saltwater-resistant buoy, It's important that the Pi has sufficient power to collect and send data

The Nemo-Pi project, which won a Google Impact Challenge 2018 award, needs volunteer programmers. Interested? Email **nemopi@save-nemo.org**

the computer into the sea that can then feed back vital information.

A team of programmers has been busy coding in Python and C++ on a slimmed Debian environment to create the Nemo-Pi itself. Meanwhile, much testing has been carried out to ensure the project is saltwater resistant and able to withstand high levels of UV irradiation. It is important that the entire setup is simple, sustainable, affordable and reliable, not to mention energy-efficient.

> “We created a plan to collect underwater physical and chemical data”

Saving energy

“The Nemo-Pi has a modified real-time clock and GPRS/GPS hub,” Diemo explains. “With this, the device is powered up and down to save energy and send its data direct to our server, which takes care of the visualisation and processing. During the night, Nemo-Pi is automatically powered off and we have developed a library to optimise data transmission against sunlight, power consumption and battery load. It means a Pi can operate endlessly – or at least until the battery or solar gives up.”

Although the project is still in a pre-production stage (“we have eight sensors under long-term alpha and beta testing,” says Diemo), the setup works well on a number of levels. Since it allows water conditions to be monitored, holidaymakers – as well as underwater, hotel, and cruising businesses – can determine whether it’s worth making a trip.

“Hundreds of dives and snorkelling trips have been cancelled each day while out on the water because of bad conditions,” Diemo continues. “If you know where to be at the right time, you can save gasoline, working hours, and unnecessary anchoring, and because of that we can help generate corals and sea life.”

Nemo-Pi also assists scientists, universities, and governments by picking up on signs of climate change and dangers to marine wildlife. “It is a small but important step against global warming and pollution of the sea and it all helps to make our ocean more blue again,” Diemo concludes.

MONITORING THE SEAS

>STEP-01

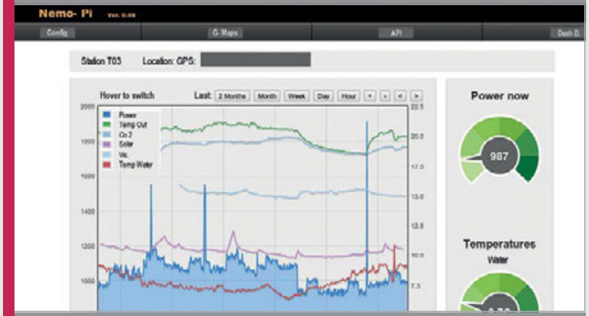
On the surface

Buoys are attached to concrete moorings. Inside the prototypes for Nemo-Pi is a Raspberry Pi – chosen over an Arduino following a year of research. A flexible and robust solar panel and battery unit is also included.

>STEP-02

Senses from the deep

Sensors connect to the Pi and run out from the buoy. They are submerged, measuring different physical and chemical data at various depths. New sensors are being added, such as one measuring refraction of light underwater – testing suspended matter/visibility.



>STEP-03

Processing the data

Readings taken from the sensors are uploaded live to a public web server. A dashboard displays the data and allows observers to figure whether tours are feasible or whether the conditions need further scientific analysis.

MASTER THE RASPBIAN FILE MANAGER

Find and manage files and folders on your Raspberry Pi and connected servers

You'll Need

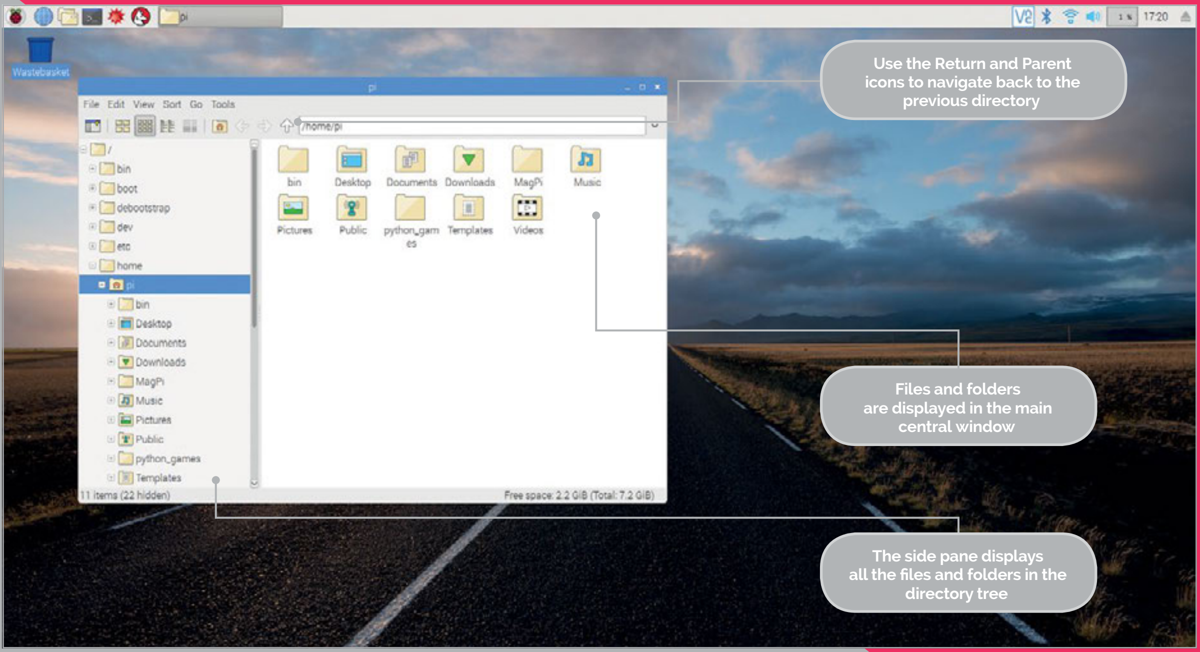
- Raspberry Pi running Raspbian
- Keyboard
- Mouse or trackpad

File Manager sits at the heart of the Raspbian operating system, giving you a graphical, easily navigated view of your files, folders, and drives. Switchers from Windows or macOS might find the unfamiliar Linux folder structure daunting at first, but – as we'll demonstrate here – you can easily slim down what's on show for a clearer view of your own personal assets.

Once you get used to the structure and start to learn where Raspbian places files that are stored outside your user folder – like web assets at **/var/www/html/**, for example – you can skip directly to those folders without stepping out of the graphical environment. Use **CTRL+L** or **ALT+D** to move the cursor to the path box and type in the location directly.

You can also connect to remote folders on WebDAV and FTP servers, thus managing websites directly within File Manager, without the kind of FTP software commonly required on other platforms.

Don't confuse File Manager with the system menus. These are shortcuts to specific apps or utilities installed on your Raspberry Pi. The File Manager is far more flexible, giving you direct access to specific locations, and putting a friendly, graphical face on the text commands like **ls** (to list files), **cd** (to change directory), and **mv** (to move or rename files) used in the Terminal environment.

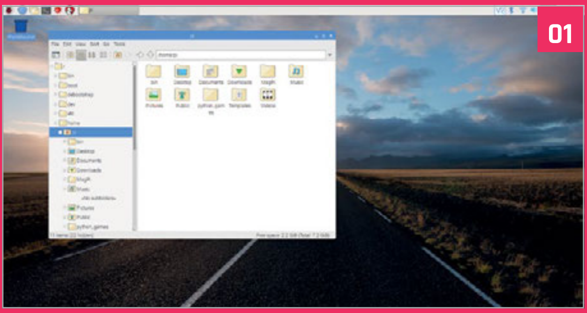


Use the Return and Parent icons to navigate back to the previous directory

Files and folders are displayed in the main central window

The side pane displays all the files and folders in the directory tree

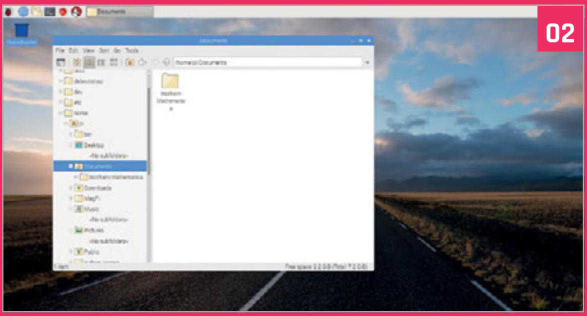
HOW TO: NAVIGATE FILES AND FOLDERS IN RASPBIAN



>STEP-01

Open File Manager

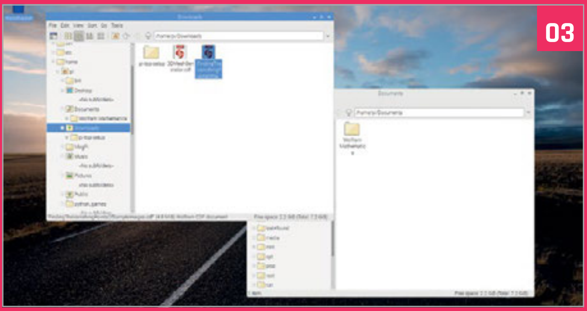
Click the File Manager icon in the Menu bar. By default, File Manager shows large icons for each file and folder from your Home directory. All the files and folders are also displayed in the sidebar.



>STEP-02

Moving around

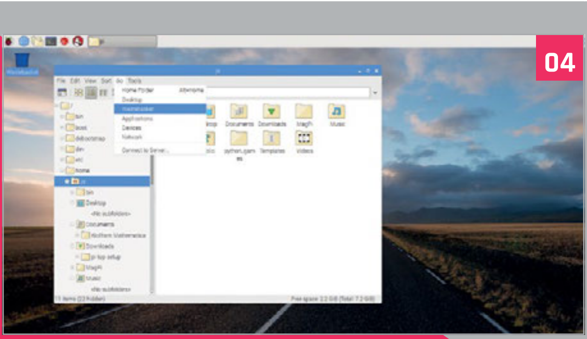
Double-click a folder (such as Documents) to move the main window to that folder. Now use the Return (left) or Parent (up) arrow to move back to the Home directory. You can also click on folders in the sidebar.



>STEP-03

Drag and drop with ease

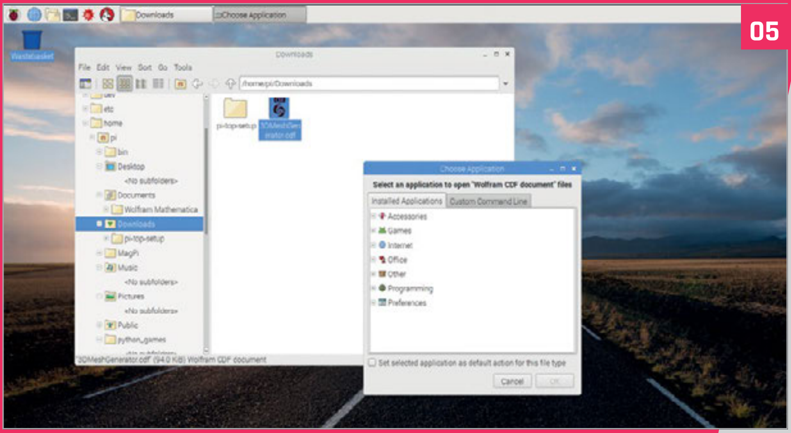
The easiest way to copy a file is to open a second window with **CTRL+N**. Now click and drag the file from one window to another.



>STEP-04

Quickly Go

Click on the Go menu to find shortcuts to the Home folder, Desktop, and Wastebasket. You can also click the down arrow to the right of the directory path to reveal a history of recently visited places.



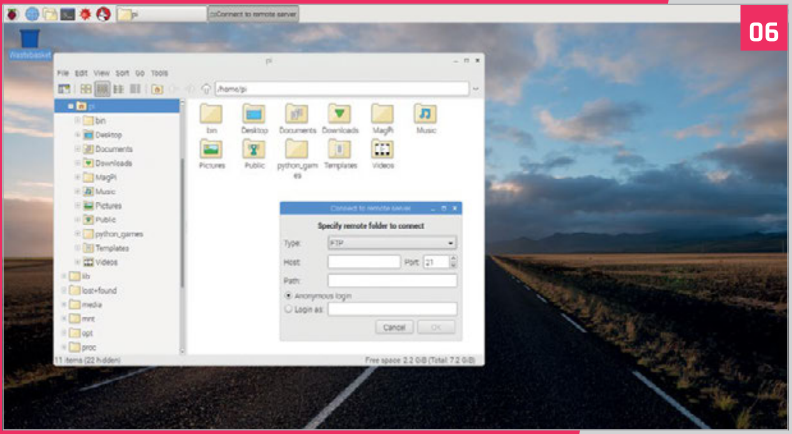
Select an application to open 'Wolfram CDF document' files

☐ Set selected application as default action for this file type

>STEP-05

Open files

Double-click a file to open it in the default program. You can choose a different program to handle the file by right-clicking the file and picking 'Open with…'. Close the file using the Close ('X') icon when you are done.



Specify remote folder to connect

Type: FTP

Host:

Port: 21

Path:

● Anonymous login

○ Login as:

>STEP-06

Remote file management

Manage remote files on FTP and WebDAV servers or via SSH by picking 'Connect to Server' from the Go menu. Pick the server type from the drop-down, then provide login details.

MIKE'S PI BAKERY

MIKE COOK

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

magpi.cc/259aT3X

KNIT YOUR OWN STRETCH SENSOR

Make a Stretch Armstrong out of code and some thread

You'll Need

- Knitting Nancy
- Conducting thread – smooth
- Wool or other thread
- Reel holder
- A/D converter, either commercial based on the MCP3008 or as we made in *The MagPi* #68
- 2 × Crocodile clips
- 270 Ω resistor
- Crochet hook, snipe-nose tweezers, darning needle (optional)

Initially it seems improbable that you could knit your own sensor, but you can. With the right sort of knitting and some conductive thread, you can make a stretch sensor quite simply. There is a touch more of the experimental nature about this project than most of the ones we do, but it is none the less rewarding for that. Here at the Bakery we are old enough to have been brought up in a time when everyone was taught needlework at primary school, but unfortunately this seems no longer to be the case. We are quite firm believers in the opinion that there is no such thing as gender-specific activities.

Knitting Nancy

A Knitting Nancy can be made from an old wooden cotton reel and four very small nails, but for not very much more you can purchase a fancy one. These are often built in the form of a cartoon-like figure. Their purpose is to produce 'French knitting', which is tubular. Due to the way the thread is looped back on itself, it acquires a sort of springiness. By incorporating a conducting thread along with the basic thread, we can make a sensor that changes its resistance as it is stretched.

How it works

Conductive thread is made from tiny slivers of stainless steel foil, spun together to make a thread. We used the smooth variety for these experiments; this type is not so 'hairy' as the normal type, but there is nothing to stop you using that.

Stainless steel is not the best electrical conductor, but it is good enough when used to stitch circuit paths over a short distance; this is because when you stitch with it, the thread is under tension. This means that the short foil pieces make good contact with others that surround it. However, when the thread is not under tension, and even more when

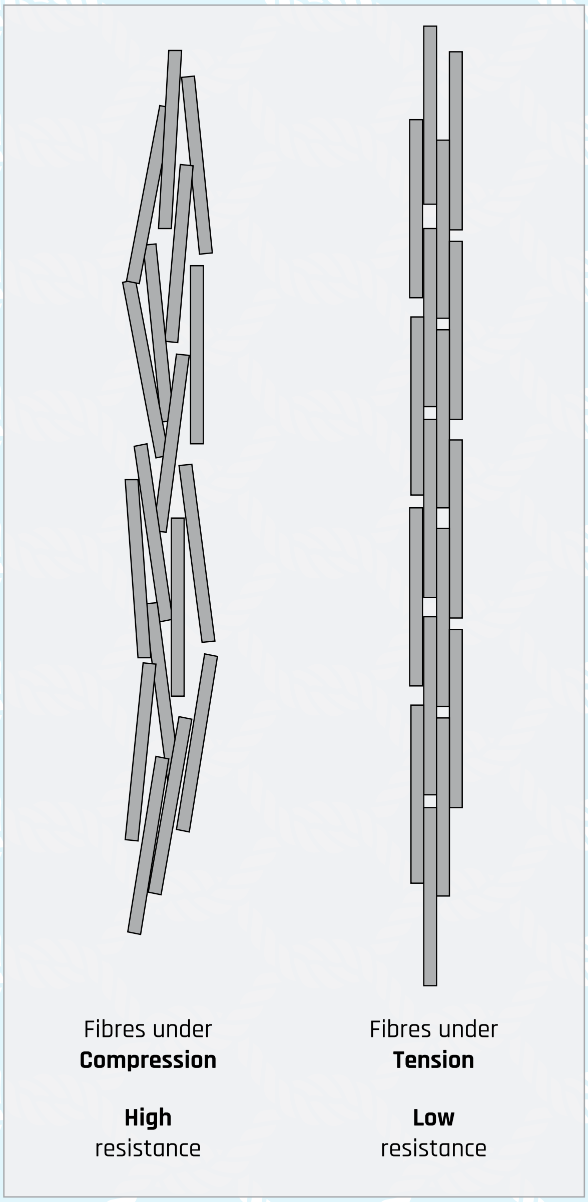
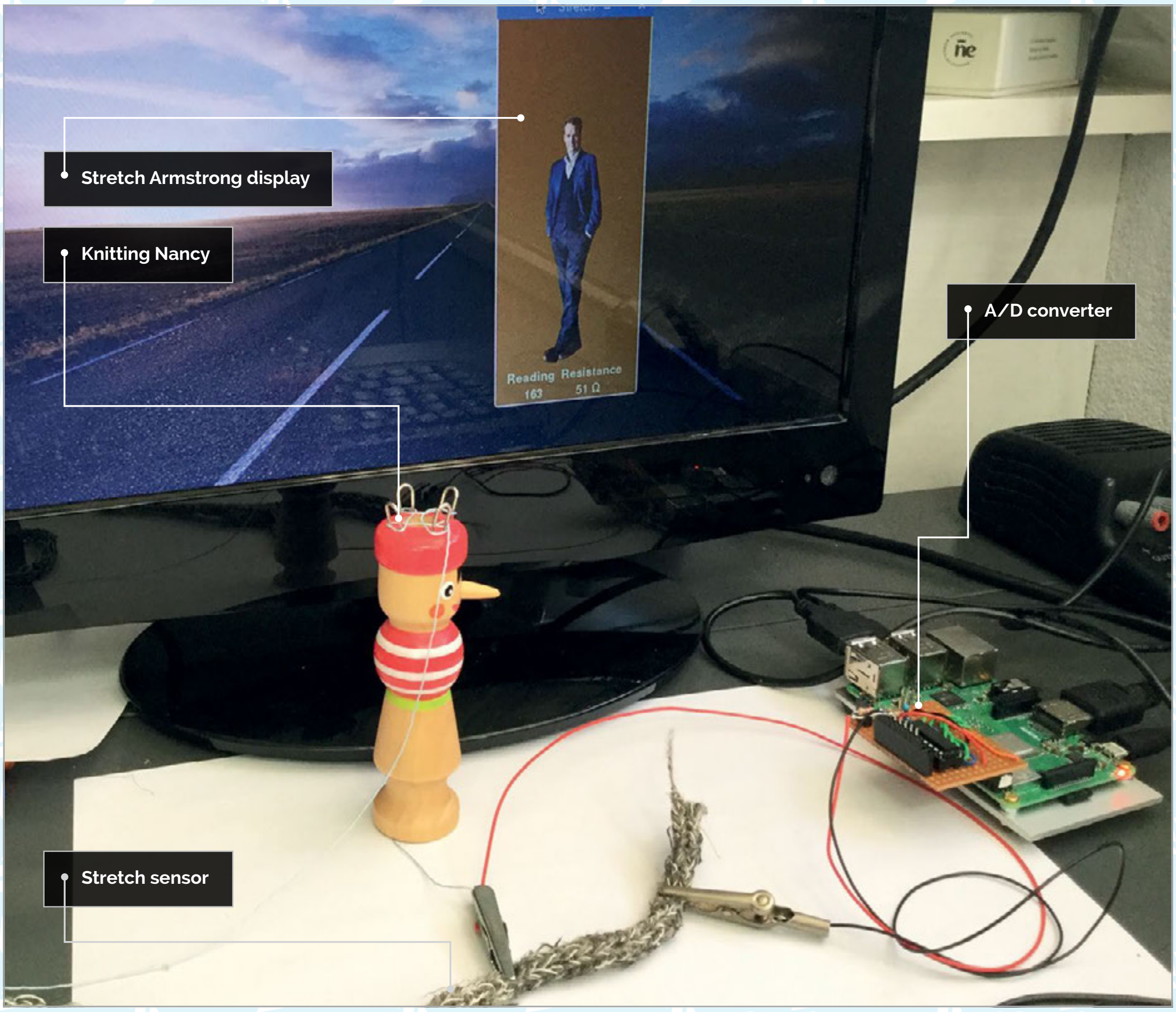


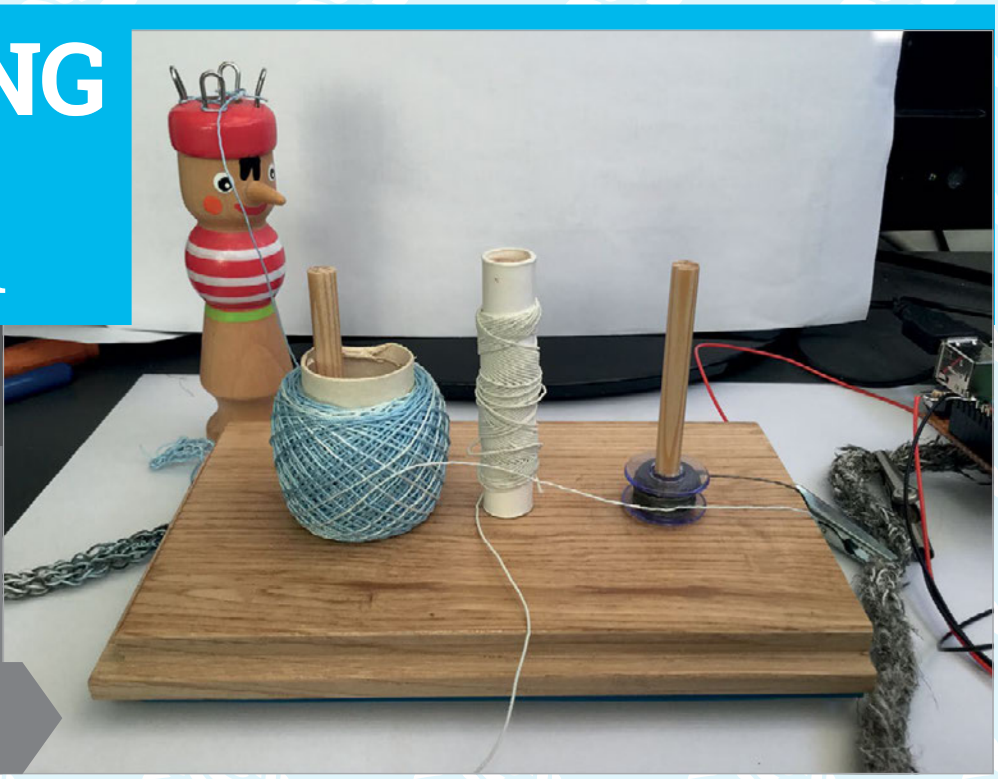
Figure 1 A representation of how fibres in conducting thread are aligned when under tension and are bunched up under compression

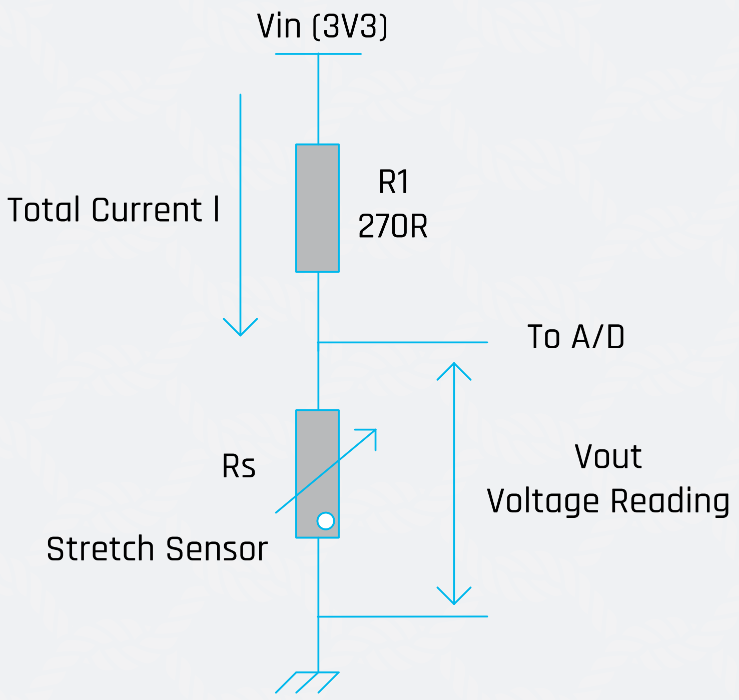


KNITTING YOUR SENSOR

>STEP-01
Making a bobbin holder

On a 150 mm by 90 mm piece of wood (actually a floorboard off-cut) we drilled three 6 mm holes part way through and glued in three 6 mm diameter 80 mm long dowels. A piece of 2 mm foam was glued to the underside to ensure a firm grip on the table and also to prevent it scratching the table.





$$\text{Total Current I} = 3.3 / (R1 + Rs)$$

$$\text{Voltage across Rs} = I \times Rs$$

$$Rs = R1 \times \frac{1}{\left(\frac{Vin}{Vout} - 1\right)}$$

Figure 2 Calculating and measuring the sensor resistance

it is under a slight compression, the thread's fibres bunch up and tend to not make contact over their whole length. This leads to an increase in resistance of the thread. This effect is illustrated in **Figure 1**. The result is that as the knitted tube is stretched, its resistance drops.

Details of how to knit your own sensor are shown in the numbered steps. We got a resistance range of about 140 Ω between the stretched and unstretched sensor; this was for tube lengths of about 24 cm.

There are many videos on French knitting on the internet. One of the first we found (**magpi.cc/frZJnV**) shows you how to cast on and off.

Reading the resistance

The stretch sensor is connected to an input channel of an A/D converter using crocodile clips, which are known in the US as alligator clips. In the Drum Sequencer project of *The Mag Pi* #68, we showed you how to make an A/D converter. **Figure 2** shows you not only how to connect this up, but also the formula needed to calculate the resistance of the sensor. We can find the resistance of the sensor if we know the voltage across it, and divide that by the current passing through it. This is called Ohm's law.

We have a resistor, R1, of 270 Ω connected between the 3V3 supply and the analogue input, with the sensor wired between the analogue input and ground. Resistors in series simply add up to

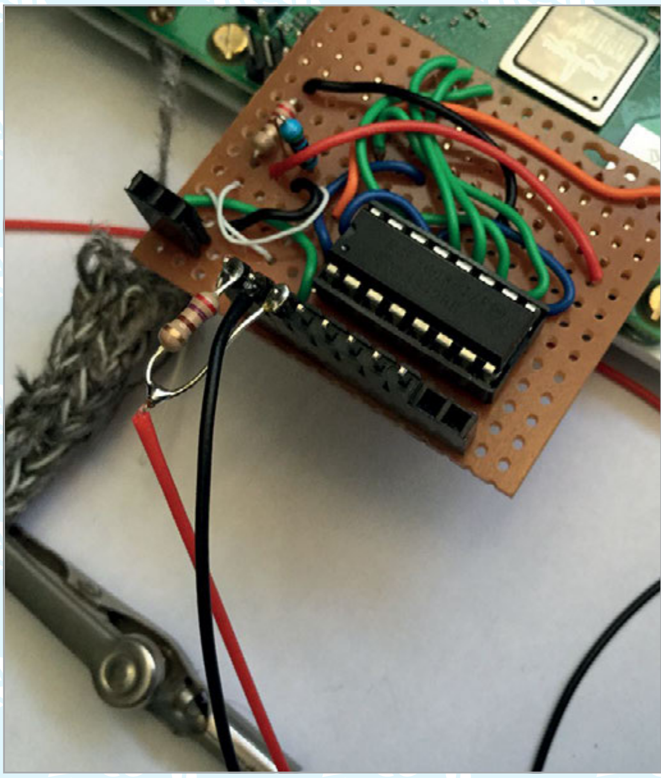
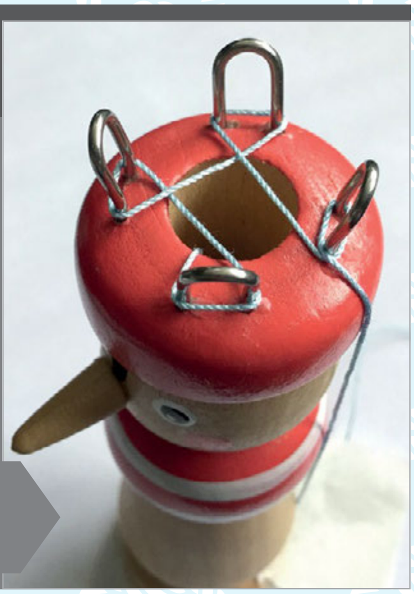


Figure 3 Physical wiring attaching the resistor and sensor to the A/D converter

>STEP-02
Start knitting

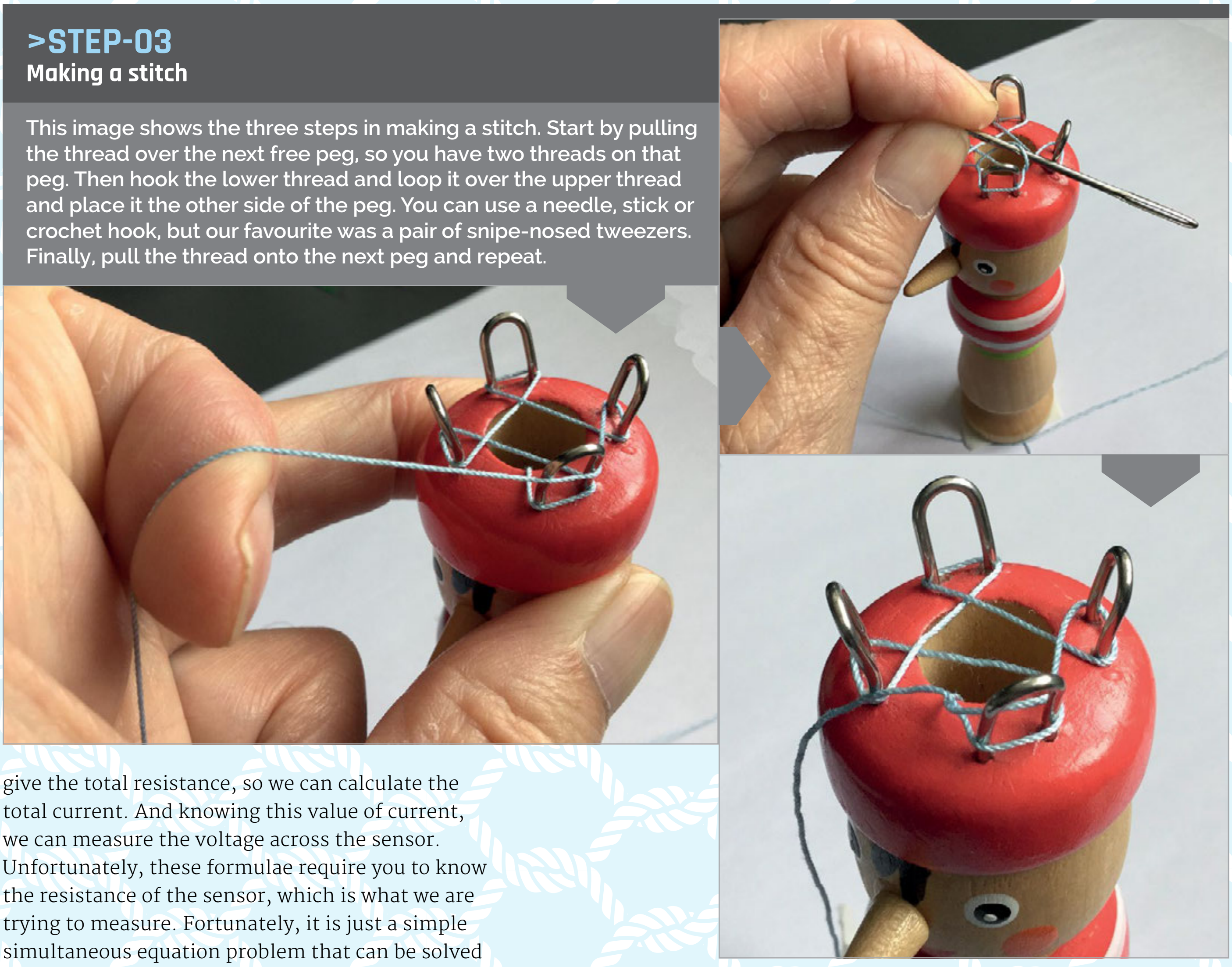
The initial setup of the threads is shown in the photo. We started knitting a few rounds with a base thread before laying the conducting thread beside it. You then have to lift the two threads over the hooks together. Occasional gentle tugs on the thread out of base stops the threads bunching up. Be careful not to make the tension too tight, as it will make lifting the threads too difficult and you could snap the conducting thread.



>STEP-03

Making a stitch

This image shows the three steps in making a stitch. Start by pulling the thread over the next free peg, so you have two threads on that peg. Then hook the lower thread and loop it over the upper thread and place it the other side of the peg. You can use a needle, stick or crochet hook, but our favourite was a pair of snipe-nosed tweezers. Finally, pull the thread onto the next peg and repeat.



give the total resistance, so we can calculate the total current. And knowing this value of current, we can measure the voltage across the sensor. Unfortunately, these formulae require you to know the resistance of the sensor, which is what we are trying to measure. Fortunately, it is just a simple simultaneous equation problem that can be solved by the substitution method.

> “Resistors in series simply add up to give the total resistance, so we can calculate the total”

Finally, you can manipulate the resulting equation so that Rs is the subject of the equation. We will leave this as an exercise to the reader to verify the final formula. **Figure 3** shows the physical wiring of the sensor to the A/D converter, with R1 being connected to 3V3 and A0, and the sensor's connection wires to ground and A0.

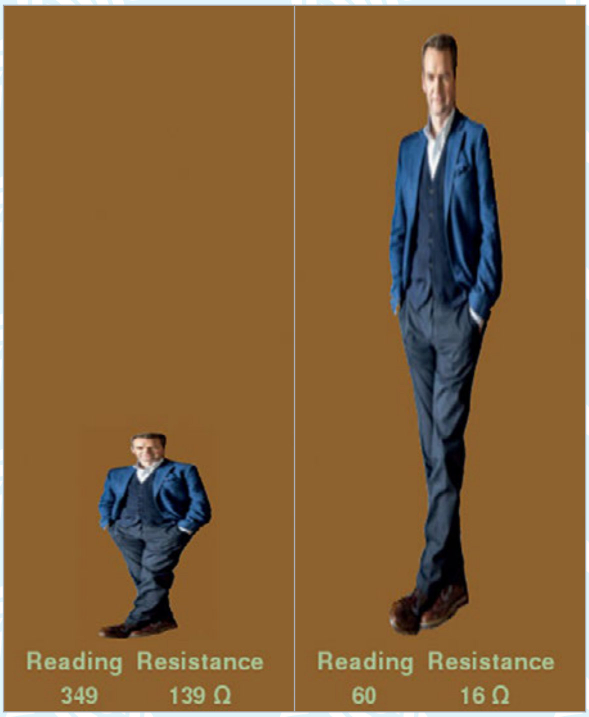


Figure 4 The Stretch display for two different stretches

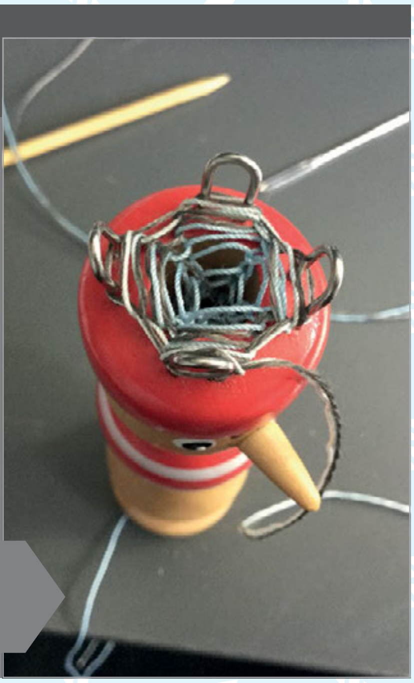
Stretch Armstrong

Stretch Armstrong was a doll that could be stretched to a very large extent and then recover. This was about 40 years ago, but it made a reappearance a few years back. We are going to do a software version of this, and you can use an image of your favourite

>STEP-04

Casting off

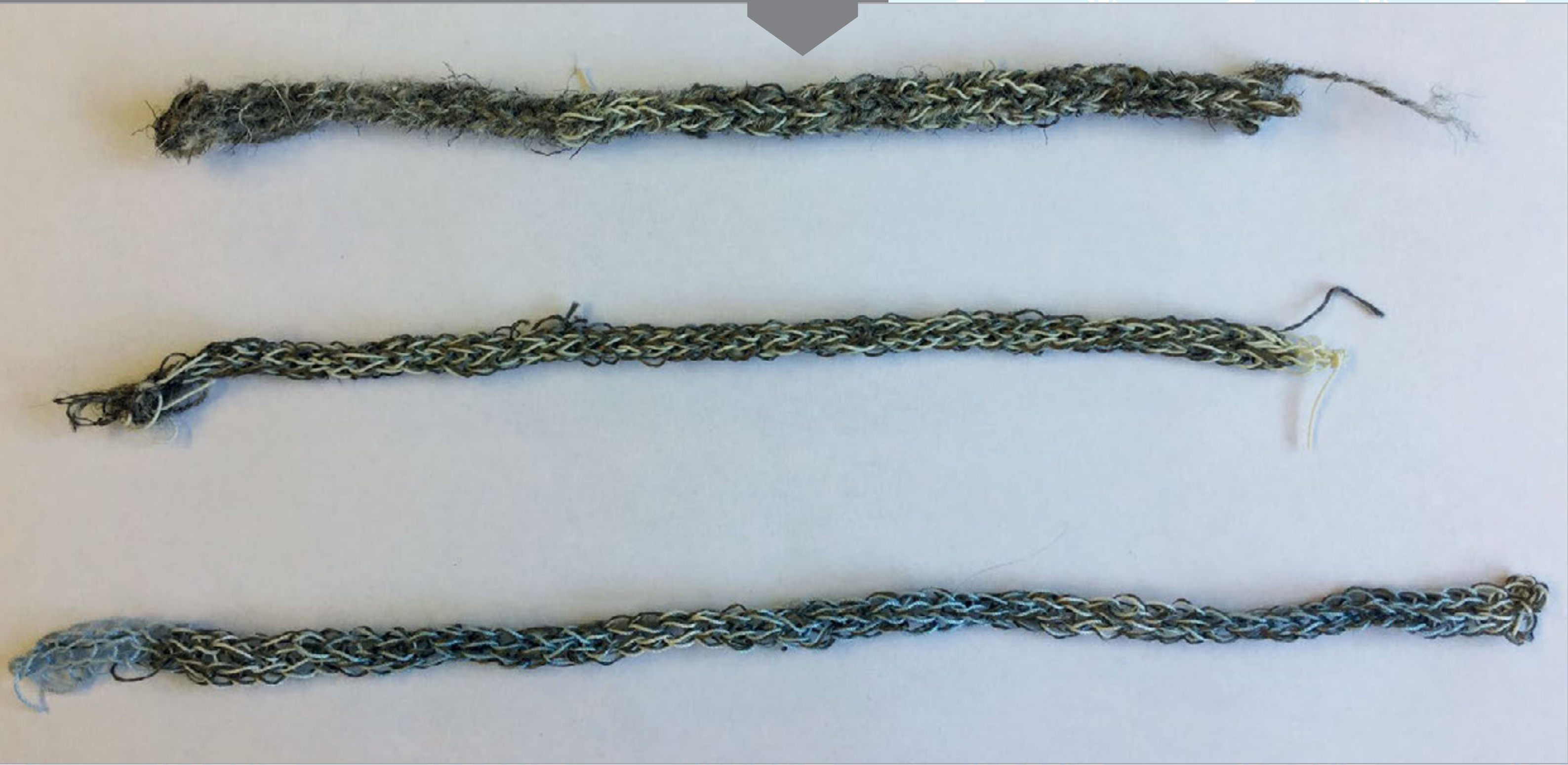
When you have knitted a long enough sensor, you can cast off. Lift the stitch off the last peg you used and loop it over the peg next to it. Then, just like a normal stitch, pull the thread that was already on the peg over it. Repeat this with the next peg and so on until you only have one stitch on one peg. Then cut the thread about 8 cm from the stitch, and thread it through a darning needle. Use this to darn the end so it is knotted and will not fray. If this sounds complicated then just view one of the many online French knitting videos.



>STEP-05

Mixing the threads

This shows three sensors we made. The top one was a combination of hairy wool, knitting elastic, and conducting thread; the middle one just knitting elastic and conducting thread; and finally, non-stretchy crochet thread and conducting thread. This last one worked the best. Note that knitting elastic is elastic with a cotton thread wrapped round it; it was first used in the 1940s/1950s for knitting swimming costumes, but the least said about that the better. We found the bottom example worked best.



Armstrong; we chose Alexander Armstrong from *Pointless*, and created an image 300 pixels high by 96 wide, with a transparent background, and placed it in a folder called **images**. Alongside this we put the code shown in the **stretch.py** listing.

This Python 3 program is written under the Pygame framework, and is quite simple. The **Vin** and **R1** variable values are defined at the start of the program. The code shows the standard values – but, for more accuracy, you can use measured values. The calculation of the sensor's resistance is done in the **readSensor** function, and a couple of **if** statements stop the code from trying to divide by zero. The **drawScreen** function actually does the rescaling of the image and works out where to put the head so the feet always stay in the same place. The Ω (ohm) and ∞ (infinity) symbols are printed using their Unicode character numbers (0x3a9 and 0x221E). The results of two different stretch measurements are shown in **Figure 4**.

Taking it further

While Stretch Armstrong is a simple example of the use of the sensor, there are many others. One that springs to mind is to use several sensors to control an on-screen virtual puppet, or even a servo-controlled real puppet. It could be used to control effects in a MIDI sound setup, or even used to weigh things. You can do this by attaching the sensor to a piece of bungee elastic so that it takes a lot more force to stretch it. Note that for most applications you don't have to calculate the resistance, just use the raw A/D readings.

stretch.py

Language

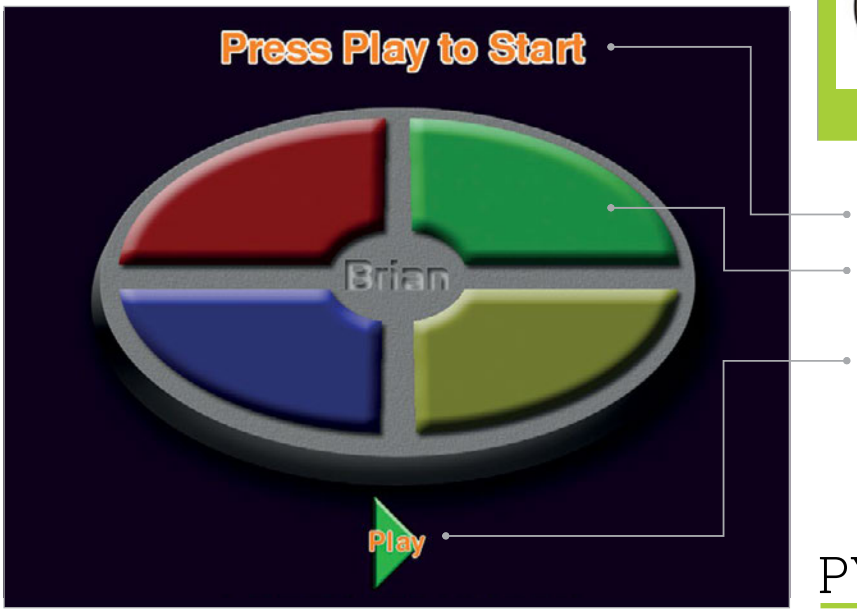
>PYTHON 3

DOWNLOAD:
magpi.cc/1NqJjmV

PROJECT VIDEOS

Check out Mike's Bakery videos at:
magpi.cc/DsjbZK

```python
import pygame, os, time, random
import spidev

pygame.init()  # initialise graphics interface

os.environ['SDL_VIDEO_WINDOW_POS'] = 'center'
pygame.display.set_caption("Stretch")
pygame.event.set_allowed(None)
pygame.event.set_allowed([pygame.KEYDOWN,pygame.QUIT])
textHeight=26 ; font = pygame.font.Font(None, textHeight)
screenWidth = 206 ; screenHeight = 500
screen = pygame.display.set_mode([screenWidth, screenHeight],0,32)
background = (150,100,40) ; pramCol = (180,200,150)
lastReading = 0.0 ; reading = 200
Vin = 3.3 ; R1 = 270 # or replace with measured values

def main():
    global lastReading, reading
    loadResource()
    initScreen()
    while(1):
        checkForEvent()
        readSensor()
        if lastReading != reading:
          drawScreen()
        lastReading = reading
        time.sleep(0.1)

def drawScreen():
   pygame.draw.rect(screen,background,(0,474,screenWidth,screenHeight),0)
   pygame.draw.rect(screen,background,(0,0,screenWidth,450),0)
   drawWords(str(reading),40,478,pramCol,background)
   if reading > 600:
       drawWords(chr(0x221E),116,478,pramCol,background) # infinity
   else:
       drawWords(str(Rs)+" "+chr(0x3a9),116,478,pramCol,background)
   stretch = 500 - reading
   if stretch > 2 :
      stretched =  pygame.transform.smoothscale(armstrong , (96,stretch) )
      screen.blit(stretched,(54,450 - stretch))
   pygame.display.update()

def drawWords(words,x,y,col,backCol) :
    textSurface = font.render(words, True, col, backCol)
    textRect = textSurface.get_rect()
    textRect.left = x
    textRect.top = y
    screen.blit(textSurface, textRect)

def initScreen():
   pygame.draw.rect(screen,background,(0,0,screenWidth,screenHeight),0)
   drawWords("Reading  Resistance",16,454,pramCol,background)

def loadResource():
   global spi, armstrong
   spi = spidev.SpiDev()
   spi.open(0,0)
   spi.max_speed_hz=1000000
   armstrong = pygame.image.load("images/Armstrong.png").convert_alpha()

def readSensor():
      global reading,Rs
      adc = spi.xfer2([1,(8)<<4,0]) # request channel
      reading = (adc[1] & 3)<<8 | adc[2] # join two bytes together
      if reading !=0:
         Rs = R1*( ( 1/ (Vin / (reading * Vin /1024) -1 ) ) )
         Rs = int(Rs) # convert into interger
      else:
         Rs = 0

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

def checkForEvent(): # handle events
    global reading
    event = pygame.event.poll()
    if event.type == pygame.QUIT :
         terminate()
    if event.type == pygame.KEYDOWN :
       if event.key == pygame.K_ESCAPE :
          terminate()
       if event.key == pygame.K_d : # screen dump
          os.system("scrot -u")

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



Instructions are displayed here

The game has four coloured buttons that light up when they are pressed

Let's get this Brian battle underway

MARK VANSTONE

Educational software author from the nineties, author of the ArcVenture series, disappeared into the corporate software wasteland. Rescued by the Raspberry Pi!
technovisualeducation.co.uk
twitter.com/mindexplorers

PYGAME ZERO PART 02

VIDEO GAME LOGIC WITH SIMPLE BRIAN

Recreate a classic electronic game using Pygame Zero

You'll Need

- Raspbian Jessie or newer
- An image manipulation program such as GIMP, or images from **magpi.cc/keaiKi**
- The latest version of Pygame Zero (1.2)
- A good memory

Long, long ago, before the Raspberry Pi existed, there was a game. It was a round plastic box with four coloured buttons on top and you had to copy what it did. To reconstruct this classic game using Pygame Zero, we'll first need a name. To avoid any possible trademark misunderstandings and because we are using the Python language, let's call it 'Brian'. The way the game works is that Brian will show you a colour sequence by lighting up the buttons and then you have to copy the sequence by pressing the coloured buttons in the same sequence. Each round, an extra button is added to the sequence and you get a point for each round you complete correctly. The game continues until you get the sequence wrong.

>STEP 01

Run, run as fast as you can

In the first part of this series (**magpi.cc/71**), we ran our Pygame Zero code by typing the **pgzrun** command into a Terminal window, as previously there was no way of running a Pygame Zero program directly from IDLE. With the 1.2 update of Pygame Zero, there is now a way to run your programs directly from IDLE without using a Terminal window. You will need to make sure that you are running version 1.2 or later of Pygame Zero. If the code in **Figure 1** does not run straight from IDLE then you may need to upgrade Raspbian on your Pi.

figure1.py

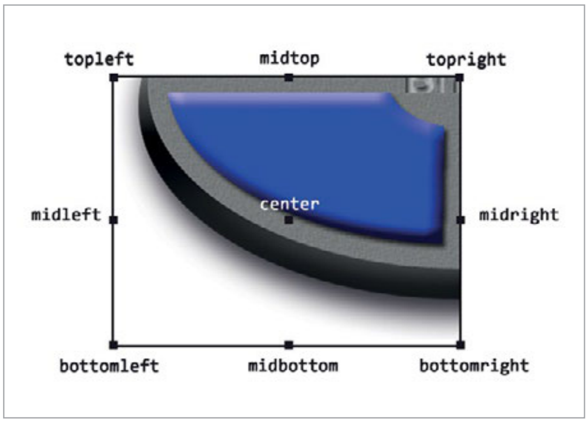
```python
import pgzrun

# Your program code will go here

pgzrun.go()
```

Figure 1 With Pygame Zero V1.2 you can now run your program from IDLE with these extra lines top and bottom

>STEP 02

The stage is set

If we are using Pygame Zero, the point is to get things happening quickly, so let's get some stuff on the screen. We are going to need some images that make up the buttons of the Brian game. You can make your own or you can get ours from GitHub at **magpi.cc/keaiKi**. The images will need to be in an **images** directory next to your program file. We have called our starting images **redunlit**, **greenunlit**, **blueunlit**, and **yellowunlit** because all the buttons will be unlit at the start of the game. We have also got a play button so that the player can click on it to start the game.



Above There are nine positions that an Actor's co-ordinates can be aligned to when the Actor is created

>STEP 03

Getting the actors on stage

As we saw in part one, we can create Actors by simply supplying an image name and a position on the screen for it to go. There are several ways of supplying the position information. This time we are going to use position handles to define where the character appears. We will use the same co-ordinates for each quadrant of the whole graphic, but we will change the handle names we use. For these Actors we can use **`bottomright`**, **`bottomleft`**, **`topright`**, and **`topleft`**, as well as the co-ordinates (400,270), which is the centre point of our whole graphic. Have a look at **Figure 2** and see how this is written.

>STEP 04

Look at the state of that

We now need to add some logic to determine if each button is on or off and show it lit or unlit accordingly. We can do this by adding a variable to each of our Actors. We want it to be either on or off, so we can set this variable as a Boolean value, i.e. True or False. If we call this variable 'state', we can add it to the Actor by writing (for the first button): **`myButtons[0].state = False`**. We then do the same for each of the button Actors with their list numbers 1, 2, and 3 because we defined them as a list.

>STEP 05

Light goes on, light goes off

We have defined a state for each of our buttons, but now we have to write some code to react to that state. First let's make a couple of lists which hold the names of the images we will use for the two states. The first list will be the images we use for the buttons being lit, which would be: **`buttonsLit = ['redlit', 'greenlit', 'bluelit', 'yellowlit']`**. We then need a list of the unlit buttons: **`buttonsUnlit = ['redunlit', 'greenunlit', 'blueunlit', 'yellowunlit']`**.

Language

>PYTHON

DOWNLOAD:
magpi.cc/hPLqsi

figure2.py

```python
import pgzrun

myButtons = []
myButtons.append(Actor('redunlit', bottomright=(400,270)))
myButtons.append(Actor('greenunlit', bottomleft=(400,270)))
myButtons.append(Actor('blueunlit',topright=(400,270)))
myButtons.append(Actor('yellowunlit',topleft=(400,270)))
playButton = Actor('play', pos=(400,540))

def draw(): # Pygame Zero draw function
    screen.fill((30, 10, 30))
    for b in myButtons: b.draw()
    playButton.draw()

pgzrun.go()
```

Figure 2 Set up the screen by creating Actors and draw them in the draw() function

figure3.py

Figure 3 Adding a state to the buttons means that we can change the image in our update() function and draw them as lit

```python
import pgzrun

myButtons = []
myButtons.append(Actor('redunlit', bottomright=(400,270)))
myButtons[0].state = False
myButtons.append(Actor('greenunlit', bottomleft=(400,270)))
myButtons[1].state = False
myButtons.append(Actor('blueunlit',topright=(400,270)))
myButtons[2].state = False
myButtons.append(Actor('yellowunlit',topleft=(400,270)))
myButtons[3].state = False
buttonsLit = ['redlit', 'greenlit', 'bluelit', 'yellowlit']
buttonsUnlit = ['redunlit', 'greenunlit', 'blueunlit', 'yellowunlit']
playButton = Actor('play', pos=(400,540))

def draw(): # Pygame Zero draw function
    screen.fill((30, 10, 30))
    for b in myButtons: b.draw()
    playButton.draw()

def update(): # Pygame Zero update function
    bcount = 0
    for b in myButtons:
        if b.state == True: b.image = buttonsLit[bcount]
        else: b.image = buttonsUnlit[bcount]
        bcount += 1

pgzrun.go()
```

GLOBALS

You can read global variables inside a function, but if you change the value of the variable, you must declare it as global in the function.

Then we can use these lists in an **`update()`** function to set the image of each button to match its state. See **Figure 3** to see how we can do this.

>STEP 06

Switching images

We can see from **Figure 3** that each time our **`update()`** function runs, we will loop through our button list. If the button's state is **`True`**, we set the image of the button to the image in the **`buttonsLit`** list. If not (i.e. the state variable is **`False`**) then we set the image of the button to the image in the **`buttonsUnlit`** list. At this stage you can test to see if your logic is working by setting the state variables at the top of the code to **`True`** and see if when you run the program, the buttons display lit.

>STEP 07

What happens if I press this button?

The idea of a button is that you can press it and something happens, so the next thing we need to write is a way to allow the user to press the buttons and make them light up. We can do this with the Pygame Zero functions **`on_mouse_down()`** and **`on_mouse_up()`**. What we need to test in these functions is if the mouse has been clicked down; if it has, we should set our button state to **`True`**. We also need to test if the mouse button has been released, in which case, all the buttons should be set to **`False`**. We can test the value we are passed (**`pos`**) into these functions with the method **`collidepoint()`**, which is part of the Actor object.

figure4.py

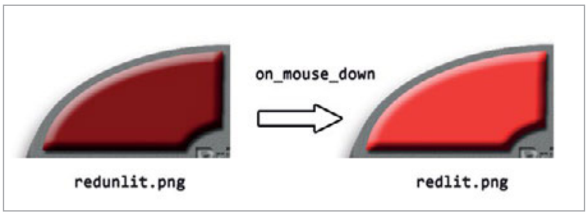
Figure 4 Check for mouse clicks on mouse down, and set all buttons back to unlit on mouse up

```python
def on_mouse_down(pos):
    global myButtons
    for b in myButtons:
        if b.collidepoint(pos): b.state = True

def on_mouse_up(pos):
    global myButtons
    for b in myButtons: b.state = False
```

figure5.py

```python
def playAnimation():
    global playPosition, playingAnimation
    playPosition = 0
    playingAnimation = True

def addButton():
    global buttonList
    buttonList.append(randint(0, 3))
    playAnimation()
```

Figure 5 addButton() generates a random number between 0 and 3, then adds this number to the end of the list called `buttonList`. Then we call the `playAnimation()` function



Above When the mouse is clicked on a button, we switch the unlit image for the lit image

>STEP 08

Ups and downs

We can write a test in **`on_mouse_down()`** for each button, to see if it has been pressed, and then change the state of the button if it has been pressed. We can then write code to set all the button states to **`False`** in the **`on_mouse_up()`** function, and our **`update()`** and **`draw()`** functions will now reflect what we need to see on the screen from those actions. Look at **Figure 4** and you will see how we can change the state of the buttons as a response to mouse events. When you have added this to your program, test it to make sure that the buttons light up correctly when clicked.

>STEP 09

Write a list

Now that we have our buttons working, we will need to make a way to use them in two different ways. The first will be for the game to display a sequence for the player to follow, and the second is to receive input from the player when they repeat the sequence. For the first task we will need to build a list to represent the sequence and then play that sequence to the player. Let's define our list at the top of the code with **`buttonList = []`** and then make a function **`def addButton():`** which will create an additional entry into the sequence each round.

>STEP 10

That's a bit random

We can generate our sequence by generating random integers using the **`random`** module. We can use this module by importing it at the top of our code with: **`from random import randint`**. We will only need the **`randint()`** function, so we can import that function specifically. To add a new number to the sequence in the **`addButton()`** function, we can write **`buttonList.append(randint(0, 3))`**, which will add a number between 0 and 3 to our list. Once we have added our new number, we will want to show the player the sequence, so add a line in the **`addButton()`** function: **`playAnimation()`**.

>STEP 11

Playing the animation

We have set up our function to create the sequence. Now we need a system to play that sequence so that the player can see it. We will do this with a counter variable called **`playPosition`**. We define this at the start of our code: **`playPosition = 0`**. We will also need a variable to show that our animation is playing: **`playingAnimation = False`**, also written at the start of the code. We can then define our **`playAnimation()`** function that we used in the previous step. Look at **Figure 5** to see how the **`addButton()`** and **`playAnimation()`** functions are written.

>STEP 12

Are we playing?

So, once we have set our animation going, we will need to react to that in our **`update()`** function. We know that all we need to do is change the state of the buttons and the **`draw()`** function will handle the visuals for us. In our **`update()`** function, we have to say: "If the animation is playing then increment our animation counter, check that we haven't reached the end of the animation and if not then light the button (change its state to True) which is indicated by our sequence list." This is a bit of a mouthful, so in **Figure 6** we can see how this translates into code.

>STEP 13

Getting a bit loopy

We can see from **Figure 6** that we are incrementing the play position each time **`update()`** is called. What we want to do is keep each button in the sequence lit for several refreshes, so we divide the **`playPosition`** by a predefined number (**`LOOPDELAY`**) to get our list position that we want to display. We round the result downwards with the **`math.floor()`** function. To use this function, you will have to import the **`math`** module at the top of your code. So if **`LOOPDELAY`** is 80 then we will move from one list position (**`listpos`**) to the next every 80 times **`update()`** is called.

>STEP 14

A dramatic pause

Still in **Figure 6**, we check to see if we have reached the end of the **`buttonList`** with **`listpos`**. If we have then we stop the animation. Otherwise, if we are still running the animation, we work out which button should be lit from our **`buttonList`**. We could just say "light that button and set the rest to unlit", but before we do that we have a line that basically says: "If we are in the second half of our button lighting loop, set all the buttons to unlit." This means that we will get a pause in between each button being lit when no buttons are lit. We can then just loop through our buttons and set their state according to the value of **`litButton`**.

>STEP 15

Testing the animation

Now, ignoring the fact that we have a play button ready and waiting to do something, we can test our animation by calling the **`addButton()`** function.

figure6.py

```python
def update(): # Pygame Zero update function
    global myButtons, playingAnimation, playPosition
    if playingAnimation:
        playPosition += 1
        listpos = math.floor(playPosition/LOOPDELAY)
        if listpos == len(buttonList):
            playingAnimation = False
            clearButtons()
        else:
            litButton = buttonList[listpos]
            if playPosition%LOOPDELAY > LOOPDELAY/2: litButton = -1
            bcount = 0
            for b in myButtons:
                if litButton == bcount: b.state = True
                else: b.state = False
                bcount += 1
```

Figure 6 The update() function with a check to see if the animation is playing. If it is, we need to work out which buttons to light and move the animation on

This function adds a random button number to the list and sets the animation in motion. To test it, we can call it a few times at the bottom of our code, just above the **`pgzrun.go()`**. If we call the **`addButton()`** function three times then three numbers will be added to the **`buttonList`** list and the animation will start. If this all works, we are ready to add the code to capture the player's response.

> "We can collect the player's clicks on the buttons just by adding another list"

>STEP 16

I need input

We can collect the player's clicks on the buttons just by adding another list, **`playerInput`**, to the definitions at the top of the code and adding a few lines into our **`on_mouse_down()`** function. Add a counter variable **`bcount = 0`** at the top of the function and then add one to **`bcount`** at the end of the loop. Then, after **`if b.collidepoint(pos):`** we add **`playerInput.append(bcount)`**. We can then test the player input to see if it matches the **`buttonList`** list we are looking for. We will write this as a separate function called **`checkPlayerInput()`** and call it at the end of our **`on_mouse_down()`** function. As we now have the basis of our game, refer to the full listing to see how the rest of the code comes together as we go through the final steps.

MODULO OR %

The % symbol is used to get the remainder after a division calculation. It's useful for creating smaller repeats within a larger loop.

CHANGING THE DELAY

We have used the constant **LOOPDELAY** for timing our loops, if the game is running too slow, decrease this value at the top of the code.

>STEP 17

Game over man

The **checkPlayerInput()** function will check the buttons that the player has clicked against the list held in **buttonList** which we have been building up with the **addButton()** function. So we need to loop through the **playerInput** list with a counter variable – let's call it **ui**, and write **if playerInput[ui] != buttonList[ui]: gameOver()**. If we get to the end of the list and both **playerInput** and **buttonList** are the same length then we know that the player has completed the sequence and we can signal that the score needs to be incremented. The score variable can be defined at the top of the code as **score = 0**. In our **on_mouse_up()** function, we can then respond to the score signal by incrementing the score and setting the next round in motion.

>STEP 18

Just press play

We still haven't done anything with that play button Actor that we set up at the beginning. Let's put some code behind that to get the game started. Make sure you have removed any testing calls at the bottom of your code to **addButton()** (Step 15). We will need a variable to check if the game is started, so put **gameStarted = False** at the top of the code with the other variables and then in our **on_mouse_up()** function we can add a test: **if playButton.collidepoint(pos) and gameStarted == False:** and then set the **gameStarted** variable to **True**. We can set a countdown variable when the play button is clicked so that there is a slight pause before the first animation starts.

> "Let's put some code behind that to get the game started"

>STEP 19

Finishing touches

We're nearly there with our game: we have a way to play a random sequence and build that list round by round, and we have a way to capture and check user input. The last things we need are some instructions for the player, which we can do with the Pygame Zero **screen.draw.text()** function. We will want an initial 'Press Play to Start' message, a 'Watch' message for when the animation is playing, a 'Now You' message to prompt the player to respond, and a score message to be displayed when the game is over. Have a look in the **draw()** function in the complete listing to see how these fit in. There are many ways we can enhance this game; for example, the original electronic game had sound too, but that will be covered in another part of this series.

USING TEXT

If you are going to use text, it's a good idea to display some on the first screen as it can take a while to load fonts for the first time. You may get unwanted delays if you load it later.

brian.py

```python
import pgzrun
from random import randint
import math
WIDTH = 800
HEIGHT = 600

myButtons = []
myButtons.append(Actor('redunlit', bottomright=(400,270)))
myButtons[0].state = False
myButtons.append(Actor('greenunlit', bottomleft=(400,270)))
myButtons[1].state = False
myButtons.append(Actor('blueunlit', topright=(400,270)))
myButtons[2].state = False
myButtons.append(Actor('yellowunlit', topleft=(400,270)))
myButtons[3].state = False
buttonsLit = ['redlit', 'greenlit', 'bluelit', 'yellowlit']
buttonsUnlit = ['redunlit', 'greenunlit', 'blueunlit', 'yellowunlit']
playButton = Actor('play', pos=(400,540))
buttonList = []
playPosition = 0
playingAnimation = False
gameCountdown = -1
LOOPDELAY = 80
score = 0
playerInput = []
signalScore = False
gameStarted = False

def draw(): # Pygame Zero draw function
    global playingAnimation, score
    screen.fill((30, 10, 30))
    for b in myButtons: b.draw()
    if gameStarted:
        screen.draw.text("Score : " + str(score), (310, 540), owidth=0.5, ocolor=(255,255,255), color=(255,128,0) , fontsize=60)
    else:
        playButton.draw()
        screen.draw.text("Play", (370, 525), owidth=0.5, ocolor=(255,255,255), color=(255,128,0) , fontsize=40)
        if score > 0:
            screen.draw.text("Final Score : " + str(score), (250, 20), owidth=0.5, ocolor=(255,255,255),
```

Language

>PYTHON

DOWNLOAD:
magpi.cc/hPlqsl

```
color=(255,128,0) , fontsize=60)
        else:
            screen.draw.text("Press Play to Start", (220, 20), owidth=0.5, ocolor=(255,255,255), color=(255,128,0) , fontsize=60)
    if playingAnimation or gameCountdown > 0:
        screen.draw.text("Watch", (330, 20), owidth=0.5, ocolor=(255,255,255), color=(255,128,0) , fontsize=60)
    if not playingAnimation and gameCountdown == 0:
        screen.draw.text("Now You", (310, 20), owidth=0.5, ocolor=(255,255,255), color=(255,128,0) , fontsize=60)

def update(): # Pygame Zero update function
    global myButtons, playingAnimation, playPosition, gameCountdown
    if playingAnimation:
        playPosition += 1
        listpos = math.floor(playPosition/LOOPDELAY)
        if listpos == len(buttonList):
            playingAnimation = False
            clearButtons()
        else:
            litButton = buttonList[listpos]
            if playPosition%LOOPDELAY > LOOPDELAY/2: litButton = -1
            bcount = 0
            for b in myButtons:
                if litButton == bcount: b.state = True
                else: b.state = False
                bcount += 1
    bcount = 0
    for b in myButtons:
        if b.state == True: b.image = buttonsLit[bcount]
        else: b.image = buttonsUnlit[bcount]
        bcount += 1
    if gameCountdown > 0:
        gameCountdown -=1
        if gameCountdown == 0:
            addButton()
            playerInput.clear()

def gameOver():
    global gameStarted, gameCountdown, playerInput, buttonList
    gameStarted = False
    gameCountdown = -1
    playerInput.clear()
    buttonList.clear()
    clearButtons()

def checkPlayerInput():
    global playerInput, gameStarted, score, buttonList, gameCountdown, signalScore
    ui = 0
    while ui < len(playerInput):
        if playerInput[ui] != buttonList[ui]: gameOver()
        ui += 1
    if ui == len(buttonList): signalScore = True

def on_mouse_down(pos):
    global myButtons, playingAnimation, gameCountdown, playerInput
    if not playingAnimation and gameCountdown == 0:
        bcount = 0
        for b in myButtons:
            if b.collidepoint(pos):
                playerInput.append(bcount)
                b.state = True
            bcount += 1
        checkPlayerInput()

def on_mouse_up(pos):
    global myButtons, gameStarted, gameCountdown, signalScore, score
    if not playingAnimation and gameCountdown == 0:
        for b in myButtons: b.state = False
    if playButton.collidepoint(pos) and gameStarted == False:
        gameStarted = True
        score = 0
        gameCountdown = LOOPDELAY
    if signalScore:
        score += 1
        gameCountdown = LOOPDELAY
        clearButtons()
        signalScore = False

def clearButtons():
    global myButtons
    for b in myButtons: b.state = False

def playAnimation():
    global playPosition, playingAnimation
    playPosition = 0
    playingAnimation = True

def addButton():
    global buttonList
    buttonList.append(randint(0, 3))
    playAnimation()

pgzrun.go()
```

WESLEY ARCHER

Raspberry Pi fan and man behind Raspberry Coulis. Enjoys building Pi-related projects and writing up guides.

raspberrycoulis.com
twitter.com/RaspberryCoulis

MINI PI-POWERED SMART MIRROR

Build your own mini smart mirror to display the temperature and humidity

You'll Need

- Raspberry Pi Zero W
- Micro Dot pHAT **magpi.cc/2cfq7Ob**
- BME280 sensor
- Prototyping wire and soldering iron
- MOSSEBO 13×18 cm photo frame – IKEA
- Reflective mirror window film
- Beebotte account – **beebotte.com**

New parents can be obsessed with ensuring their child is not too hot or cold. Sure, they could use a standard thermometer, but that's not as fun as using a Raspberry Pi. Smart mirrors are all the rage, and we'll show you how to build a mini smart mirror to display the temperature and humidity in a room. And without breaking the bank.

>STEP-01

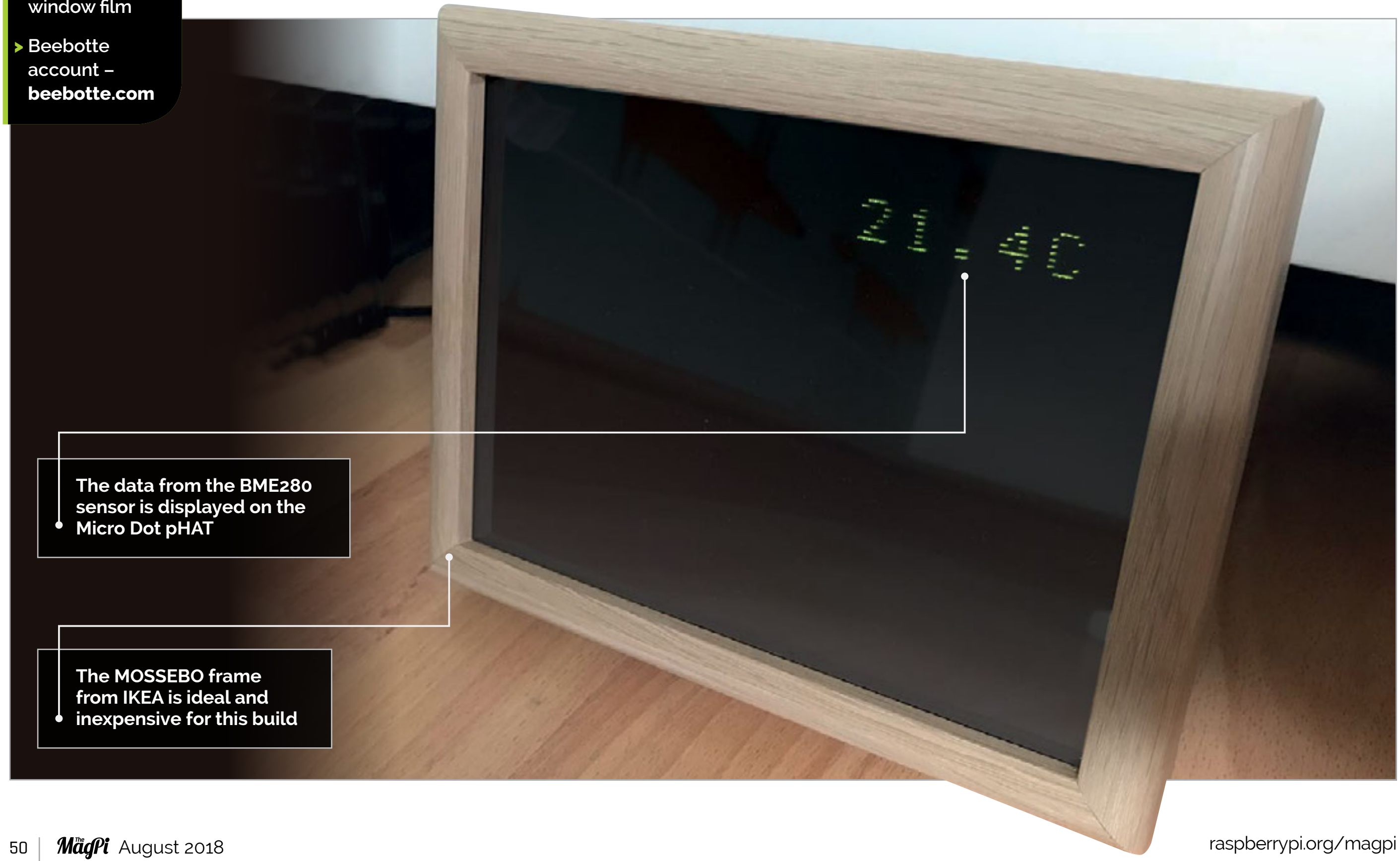
Prototype your circuit first

We recommend testing your circuit with a breadboard before soldering them. The BME280 sensor is 3 V tolerant, and connects to the SDA and SCL pins on the Pi, but shares them with the Micro Dot pHAT, so will require intricate wiring and soldering – we found that stripping a small part of the wire in the middle and soldering to that so the wire is shared worked well. Or you could solder two wires to the one GPIO pin; i.e. one at the top, one at the bottom.

>STEP-02

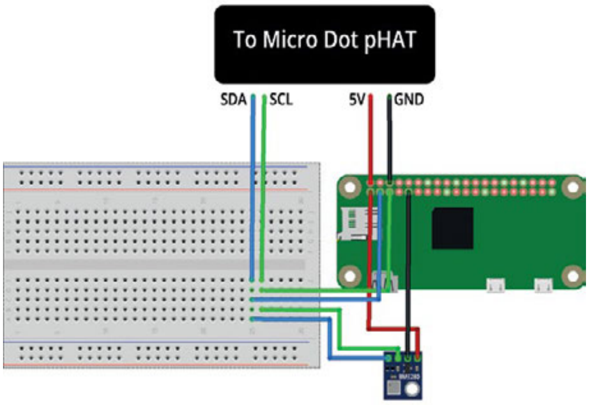
Applying the mirrored window film

We found a UK supplier of mirrored window film that issued free A5-sized samples – they've since started charging 99p, which is still great value. Applying the mirrored window film is fairly straightforward, but make sure you follow instructions! We used soapy water, a squeegee, and patience when applying ours to make sure no air bubbles were trapped, and we applied two layers to add to the effect. Make sure it dries!



The data from the BME280 sensor is displayed on the Micro Dot pHAT

The MOSSEBO frame from IKEA is ideal and inexpensive for this build



Above The BME280 is connected to 3V, GND, SDA, and SCL pins on the Pi. The Micro Dot pHAT is connected to 5V, GND, SDA, and SCL as well

>STEP-03

Prepare the picture frame

We chose the MOSSEBO picture frame as it was fairly deep and housed the Pi and Micro Dot pHAT perfectly. Inside the frame is a white internal frame, which we used to hold our Micro Dot pHAT in place by roughly cutting a slot into the corner. Don't worry if this is not picture perfect, as it will be hidden anyway. We also cut a small gap for the cables to pass through.

>STEP-04

Download our code to your Pi

Now your BME280 sensor and Micro Dot pHAT are all connected, and your frame is built, you'll need to install the software. Install the Micro Dot pHAT libraries (**magpi.cc/NtPlsE**) as well as sign up for a Beebotte XS account (**beebotte.com**) and then you'll need to install our code. With Beebotte you can keep track of the data sent from your Pi and graph it out.

>STEP-05

Install the Beebotte module

Beebotte works via an API, which is very well suited to life on a Pi, but you'll need to install the Beebotte module with **`sudo pip install beebotte`**. The guide at **magpi.cc/AiWqCZ** is a great starting point. Once installed, you'll need to create a channel so you can store the data sent from your Pi. This generates a unique channel token that you'll need to add to our code for it to work, as well as the channel name used.

>STEP-06

Test it all out!

Once you've added the Beebotte channel token and name, you should be able to test it all out. Type **`python bme280-mdot-beebotte.py`** at the command prompt and you should see the Micro Dot pHAT spring into life and cycle through the temperature and humidity. If you have created a dashboard to display your data, you should start seeing it appear in your account straight away. Pressing **CTRL+C** will exit the code when running.

bme280-mdot-beebotte.py

Language
>PYTHON

DOWNLOAD:
magpi.cc/FXuLSi

```
#!/usr/bin/env python

import bme280
import time
import sys
import threading
from microdotphat import write_string, set_decimal, clear, show
from beebotte import *

### Replace CHANNEL_TOKEN with that of your Beebotte channel and
### YOUR_CHANNEL_NAME with the name you give your Beebotte channel
bbt = BBT(token = 'CHANNEL_TOKEN')
chanName = "YOUR_CHANNEL_NAME"

# These resources needed to be added to your Beebotte channel - i.e.
# Temperature and Humidity.
temp_resource    = Resource(bbt, chanName, 'temperature')
humidity_resource = Resource(bbt, chanName, 'humidity')

# Sends data to your Beebotte channel
def beebotte():
    while True:
        temp_resource.write(round(temperature,1))
        humidity_resource.write(round(humidity,0))
        time.sleep(900)    # 15 mins to prevent maxing API limit

# Display stats on the Micro Dot pHAT
def microdot():
    clear()
    write_string( "%.1f" % temperature + "C", kerning=False)
    show()
    time.sleep(5)
    clear()
    write_string( "%.0f" % humidity + "% RH", kerning=False)
    show()
    time.sleep(5)

try:
    # Get the first reading from the BME280 sensor - ignore pressure for now.
    temperature,pressure,humidity = bme280.readBME280All()
    # Start the Beebotte function as a thread so it works in the background
    beebotte_thread = threading.Thread(target=beebotte)
    beebotte_thread.daemon = True
    beebotte_thread.start()
    # Run a loop to collect data and display it on the Micro Dot pHAT
    while True:
        temperature,pressure,humidity = bme280.readBME280All()
        microdot()

# Attempt to exit cleanly - not quite there, needs work!
except (KeyboardInterrupt, SystemExit):
    sys.exit()
    pass
```

MARTIN O'HANLON

Martin loves technology and creates projects and learning resources for Raspberry Pi
raspberrypi.org

DOCUMENTING YOUR CODE

Create a website that automatically creates documentation for Python programs

When you've created a really useful project and you want to share it with other people, a crucial step is creating documentation that helps people understand what the code does, how it works, and how they can use it.

To share the code you've created and help people use it, you can create a website that documents your project.

This guide will teach you how to build a website for your project that automatically creates documentation for your Python code.

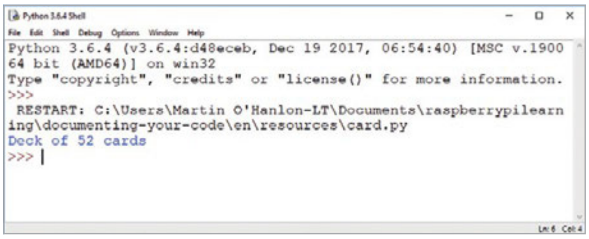
Imagine you've created a software masterpiece that will make countless coders' lives a little easier, and you've made it available, but nobody is using it because they don't know how to! Creating documentation is key when you share your code.

In this project, you will be documenting some example code based on a deck of cards project (**magpi.cc/CEtozb**). It's not essential for you to have worked through this project, but it would be useful, because it would help you understand what the code you'll document does.

Download the project code called **card.py** from here: **magpi.cc/WdQwNP**. Then open the downloaded **card.py** program using Python 3 IDLE.

Look over the code. You will see that there are two classes: **Card** and **Deck**. **Card** represents a single playing card, while **Deck** is a collection of cards stacked in order. At the bottom of the program, a **deck** object is created from **Deck**, and a description is printed to the screen.

Run the **card.py** program to see how it works. You should see a message describing the deck of cards printed to the screen (**Figure 1**).



```
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:54:40) [MSC v.1900
64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:\Users\Martin O'Hanlon-LT\Documents\raspberrypilearn
ing\documenting-your-code\en\resources\card.py
Deck of 52 cards
>>> |
```

Figure 1 After running the card.py program, you should see this in IDLE

Documenting your code

If you look through the **card.py** program, you will notice that there is no additional information in the code that describes how it works or how to use it.

The program is small, so you could probably review the code and work out how the interface works and what each function does. But what about programs that have a thousand lines of code? Or even a million? It would be extremely difficult and time-consuming to figure out how such big programs work without some additional information.

Python allows you to add information about your program into the code using docstrings (**magpi.cc/XuSYYW**). Docstrings are the basis for creating documentation for your code. You add docstrings at the start of a module, class, or function in the form of a string framed by three double quotation marks **"""** on either side:

```
def helloworld():
    """ prints 'hello world' to the screen """
    print("hello world")
```

Docstrings can be a single line (as in the example above), or they can span multiple lines:

```
def helloworld():
    """
    This function prints 'hello world' to the
screen.
    It accepts no arguments and returns
nothing.
    """
    print("hello world")
```

Your first task is to add a docstring to the **Card** class to describe the class and what it is for.

```
class Card:
    """
    The Card class represents a single playing
card and is initialised by passing a suit and
number.
    """
    def __init__(self, suit, number):

        self._suit = suit
        self._number = number
```

Do the same for the **Deck** class

Generating documentation

Now that your code contains some information about itself in the form of docstrings, you can use Python's pydoc module (**magpi.cc/cLunif**) to automatically create HTML documentation about your code.

Open a Terminal window and navigate to the folder containing your **card.py** program:

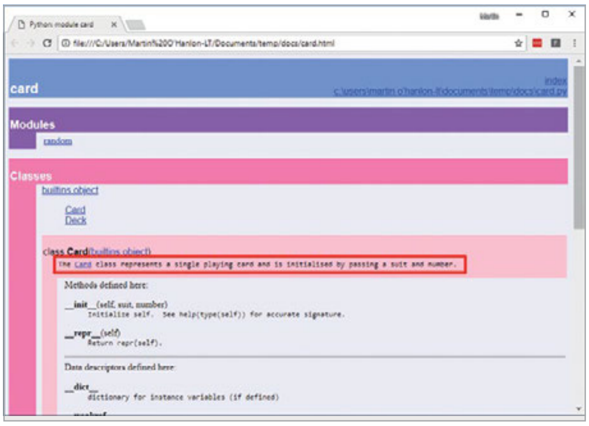
Language

>PYTHON

DOWNLOAD:
magpi.cc/WdQwNP

card.py

```
import random

class Card:

    def __init__(self, suit, number):
        self._suit = suit
        self._number = number

    def __repr__(self):
        return self._number + " of " + self._suit

    @property
    def suit(self):
        return self._suit

    @suit.setter
    def suit(self, suit):
        if suit in ["hearts", "clubs", "diamonds", "spades"]:
            self._suit = suit
        else:
            print("That's not a suit!")

    @property
    def number(self):
        return self._number

    @number.setter
    def number(self, number):
        valid = [str(n) for n in range(2,11)] + ["J", "Q", "K", "A"]
        if number in valid:
            self._number = number
        else:
            print("That's not a valid number")


class Deck:

    def __init__(self):
        self._cards = []
        self.populate()

    def populate(self):
        suits = ["hearts", "clubs", "diamonds", "spades"]
        numbers = [str(n) for n in range(2,11)] + ["J", "Q", "K", "A"]
        self._cards = [ Card(s, n) for s in suits for n in numbers ]

    def shuffle(self):
        random.shuffle(self._cards)

    def deal(self, no_of_cards):
        dealt_cards = []
        for i in range(no_of_cards):
            dealt_card = self._cards.pop(0)
            dealt_cards.append(dealt_card)
        return dealt_cards

    def __repr__(self):
        cards_in_deck = len(self._cards)
        return "Deck of " + str(cards_in_deck) + " cards"

deck = Deck()
print(deck)

```



Python: module card

file:///C:/Users/Martin%20O'Hanlon-LT/Documents/temp/docs/card.html

card

index
c:\users\martin o'hanlon-lt\documents\temp\docs\card.py

Modules

random

Classes

builtins.object

Card
Deck

class Card(builtins.object)

```
The Card class represents a single playing card and is initialised by passing a suit and number.
```

Methods defined here:

__init__(self, suit, number)

```
    Initialize self.  See help(type(self)) for accurate signature.
```

__repr__(self)

```
    Return repr(self).
```

Data descriptors defined here:

__dict__

```
    dictionary for instance variables (if defined)
```

Figure 2 Your online documentation will look something like this

```
cd name_of_folder
```

Enter the command to run the pydoc module to create documentation:

```
python3 -m pydoc -w ./card.py
```

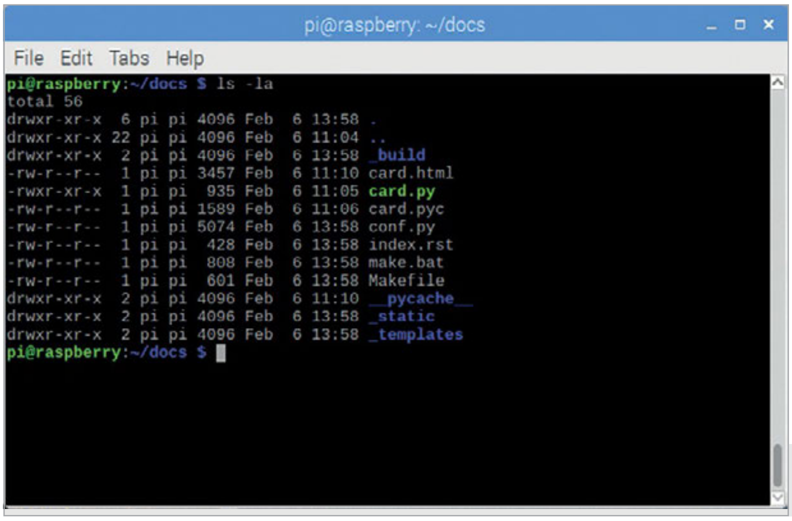
When the command has finished running, you will be presented with the message 'wrote card.html'. Note that pydoc uses the name of the Python program you input as the name for the HTML file.

Open the **card.html** file using your web browser to see the documentation that's been created. You will see a page that shows the **Card** and **Deck** classes, their methods, and properties, including the descriptions you added as docstrings (**Figure 2**).

You could upload this basic HTML page to a hosting service to provide your users with information about your software. The next steps will show you how to use the Sphinx tool to build a documentation website to which you can add additional content and information.

Complete the documentation

Add the rest of the docstrings to the methods and properties of the **Card** and **Deck** classes to describe what they do.



```
pi@raspberry:~/docs $ ls -la
total 56
drwxr-xr-x  6 pi pi 4096 Feb  6 13:58 .
drwxr-xr-x 22 pi pi 4096 Feb  6 11:04 ..
drwxr-xr-x  2 pi pi 4096 Feb  6 13:58 _build
-rw-r--r--  1 pi pi 3457 Feb  6 11:10 card.html
-rwxr-xr-x  1 pi pi  935 Feb  6 11:05 card.py
-rw-r--r--  1 pi pi 1589 Feb  6 11:06 card.pyc
-rw-r--r--  1 pi pi 5074 Feb  6 13:58 conf.py
-rw-r--r--  1 pi pi  428 Feb  6 13:58 index.rst
-rw-r--r--  1 pi pi  808 Feb  6 13:58 make.bat
-rw-r--r--  1 pi pi  601 Feb  6 13:58 Makefile
drwxr-xr-x  2 pi pi 4096 Feb  6 11:10 __pycache__
drwxr-xr-x  2 pi pi 4096 Feb  6 13:58 _static
drwxr-xr-x  2 pi pi 4096 Feb  6 13:58 _templates
pi@raspberry:~/docs $ 
```

Figure 3 The file structure of Sphinx once it's all installed

Add a docstring directly under the method's definition **def**:

```
def shuffle(self):
    """ A description of what the shuffle 
method does """
    random.shuffle(self._cards)
```

Only add a docstring to a property's **getter** that describes what the property is – the **setter** doesn't need one:

```
@property
def suit(self):
    """ Gets or sets the suit of the Card 
"""
    return self._suit

@suit.setter
def suit(self, suit):
    # there is no need for a docstring on 
a setter
    if suit in ["hearts", "clubs", 
"diamonds", "spades"]:
        self._suit = suit
    else:
        print("That's not a suit!")
```

Adding a docstring to the setter will not cause an error, but pydoc will ignore it when generating the documentation. When you have added docstrings to your code, you can run the **pydoc** command again to generate updated documentation.

Creating a project website with Sphinx

Using pydoc and docstrings is a great way to create structured documentation about your code, but it has limits because you can't add additional information and content.

On a less basic website, similar to for instance the one for GPIO Zero (**magpi.cc/2pQmvfl**), you can add a lot more information in addition to documentation, such as recipes, FAQs, images, and code snippets.

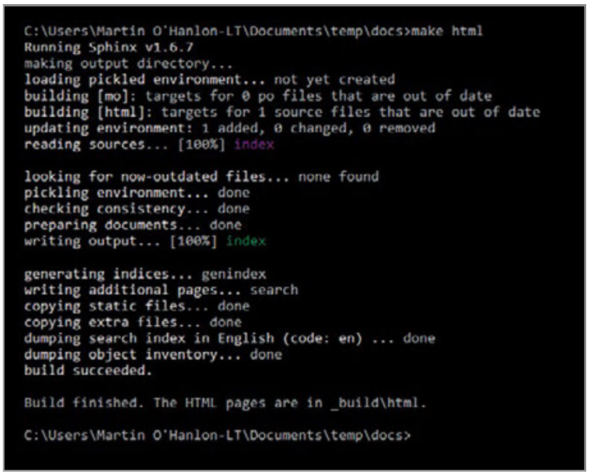
You will use Sphinx (**sphinx-doc.org**) to create such a project website. This tool was originally created for documenting the Python language.

To create content for your Sphinx website, you will write text files formatted using the simple but powerful markup language reStructuredText (reST): **magpi.cc/yDHiZh**.

Install Sphinx

Open a Terminal or command prompt window and use pip3 to install the Sphinx module:

```
pip3 install sphinx
```



```
C:\Users\Martin O'Hanlon-LT\Documents\temp\docs>make html
Running Sphinx v1.6.7
making output directory...
loading pickled environment... not yet created
building [mo]: targets for 0 po files that are out of date
building [html]: targets for 1 source files that are out of date
updating environment: 1 added, 0 changed, 0 removed
reading sources... [100%] index

looking for now-outdated files... none found
pickling environment... done
checking consistency... done
preparing documents... done
writing output... [100%] index

generating indices... genindex
writing additional pages... search
copying static files... done
copying extra files... done
dumping search index in English (code: en) ... done
dumping object inventory... done
build succeeded.

Build finished. The HTML pages are in _build\html.

C:\Users\Martin O'Hanlon-LT\Documents\temp\docs>
```

Figure 4 Building your website should look a little like this

Sphinx includes a quick-start utility that creates a template Sphinx project. Navigate to your project directory and run the Sphinx quick-start utility by entering:

```
sphinx-quickstart
```

Use the following responses to complete the quick-start questionnaire that will determine properties and settings Sphinx will use for your website – while there's a fair number of questions, you'll use the default answer for all but four of them by pressing **ENTER**.

The Sphinx quick-start program will make a number of files and directories where your documentation will be created (**Figure 3**).

The key files and directory are:

conf.py – a Sphinx configuration file that describes how your documentation should be created

index.rst – the main page and index of your documentation

_build – the directory where your documentation will be created

Build your website

To view your project website, you will need to build it. This converts all the project files into HTML files. Run the following command to build your website:

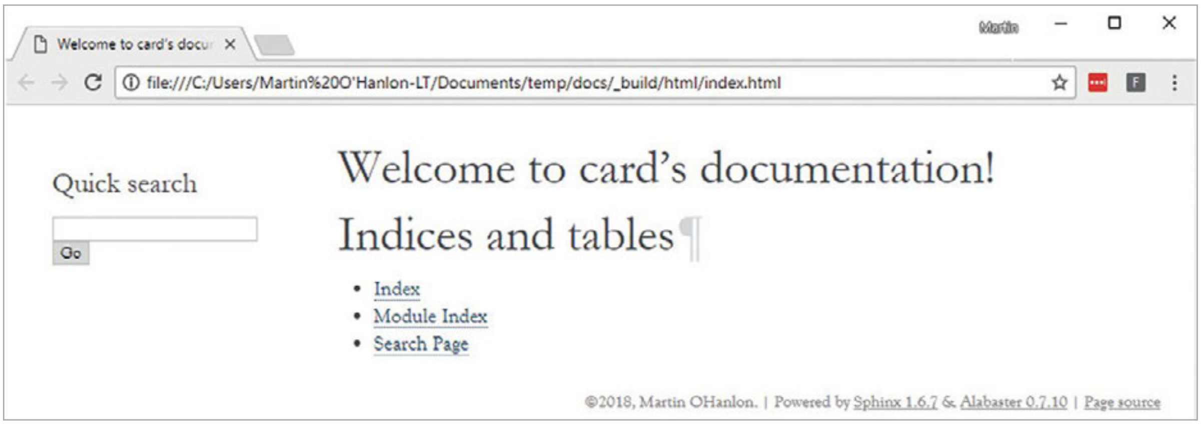
```
make html
```

This will create an **html** folder within the **_build** directory that will hold the project website files (**Figure 4**).

Open the **index.html** file in the **_build/html** directory using a web browser. You will see an empty project site like in **Figure 5** (overleaf).

Next, you'll add your own content to the project site, including auto-generated documentation for your code.

QUESTION	USE DEFAULT	RESPONSE
Root path for the documentation [.]	Yes	.
Separate source and build directories (y/n) [n]	Yes	n
Name prefix for templates and static dir [_]:	Yes	_
Project name:	No	card
Author name(s):	No	your name
Project version []:	No	0.1
Project release [0.1]:	Yes	0.1
Project language [en]:	Yes	en
Source file suffix [.rst]:	Yes	.rst
Name of your master document (without suffix) [index]:	Yes	index
Do you want to use the epub builder (y/n) [n]:	Yes	n
autodoc: automatically insert docstrings from modules (y/n) [n]:	No	y
doctest: automatically test code snippets in doctest blocks (y/n) [n]:	Yes	n
intersphinx: link between Sphinx documentation of different projects (y/n) [n]:	Yes	n
todo: write "todo" entries that can be shown or hidden on build (y/n) [n]:	Yes	n
coverage: checks for documentation coverage (y/n) [n]:	Yes	n
imgmath: include math, rendered as PNG or SVG images (y/n) [n]:	Yes	n
mathjax: include math, rendered in the browser by MathJax (y/n) [n]:	Yes	n
ifconfig: conditional inclusion of content based on config values (y/n) [n]:	Yes	n
viewcode: include links to the source code of documented Python objects (y/n) [n]:	Yes	n
githubpages: create .nojekyll file to publish the document on GitHub pages (y/n) [n]:	Yes	n
Create Makefile? (y/n) [y]:	Yes	y
Create Windows command file? (y/n) [y]:	Yes	y


Welcome to card's docu ×
Martin
file:///C:/Users/Martin%20O'Hanlon-LT/Documents/temp/docs/_build/html/index.html
Quick search
Go

Welcome to card's documentation!

Indices and tables¶

- Index
- Module Index
- Search Page

©2018, Martin OHanlon. | Powered by Sphinx 1.6.7 & Alabaster 0.7.10 | Page source

Figure 5 The empty project website ready to have code uploaded to it

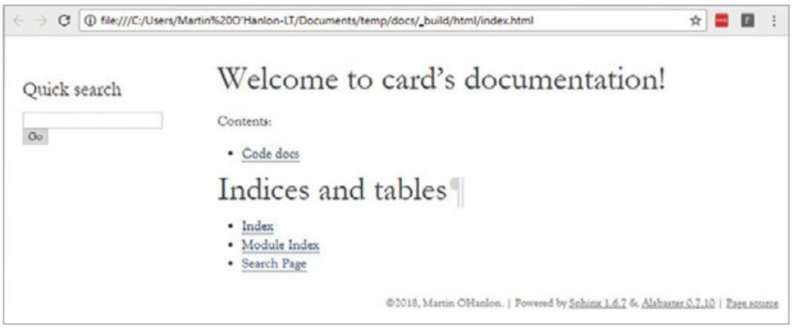
Adding your code documentation

So it can include the code documentation, Sphinx needs to know where to find it. You tell Sphinx this by modifying the **conf.py** file. Open **conf.py** and find this section of code:

```
# import os
# import sys
# sys.path.insert(0, os.path.abspath('.'))
```

Uncomment this code by removing the **#** characters.

```
import os
import sys
sys.path.insert(0, os.path.abspath('.'))
```


file:///C:/Users/Martin%20O'Hanlon-LT/Documents/temp/docs/_build/html/index.html
Quick search
Go

Welcome to card's documentation!

Contents:

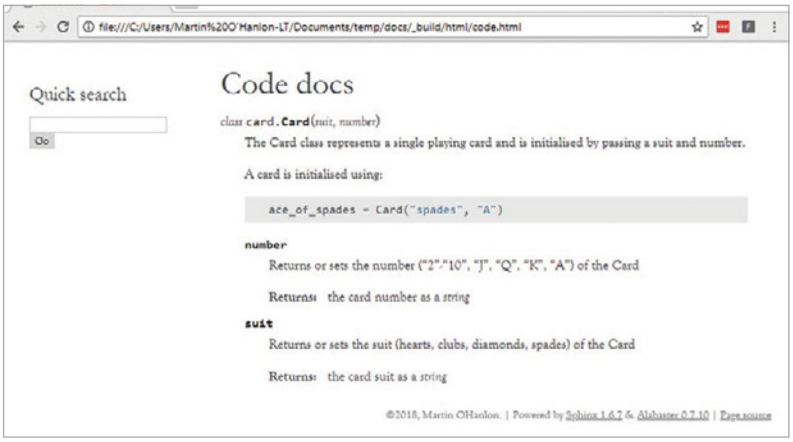
- Code docs

Indices and tables¶

- Index
- Module Index
- Search Page

©2018, Martin OHanlon. | Powered by Sphinx 1.6.7 & Alabaster 0.7.10 | Page source

Figure 6 Documentation now appearing in your index


file:///C:/Users/Martin%20O'Hanlon-LT/Documents/temp/docs/_build/html/code.html
Quick search
Go

Code docs

class card.Card(suit, number)

The Card class represents a single playing card and is initialised by passing a suit and number.

A card is initialised using:

```
ace_of_spades = Card("spades", "A")
```

number

Returns or sets the number ("2"-"10", "J", "Q", "K", "A") of the Card

Returns: the card number as a string

suit

Returns or sets the suit (hearts, clubs, diamonds, spades) of the Card

Returns: the card suit as a string

©2018, Martin OHanlon. | Powered by Sphinx 1.6.7 & Alabaster 0.7.10 | Page source

Figure 7 Documentation is auto-generated from the info we gave it

This adds the current directory (**.**) to Sphinx's configuration so it can find your code files. Now, create a new file called **code.rst**. This will be your code documentation page. Add a title to the page:

```
Code docs
=========
```

Add the following code to import the **card.py** module.

```
.. module:: card
```

Next, add this code to automatically generate the documentation for the **Card** class:

```
.. autoclass:: Card
    :members:
```

In order for your **code.rst** page to appear on your project website, it needs to be added to the index. Open **index.rst** and modify it to add the code page under the table of contents **.. toctree::** – the code should look like this:

```
Welcome to card's documentation!
================================

.. toctree::
   :maxdepth: 2
   :caption: Contents:

   code
```

Now rebuild your project website:

```
make html
```

Reopen the **index.html** file in the **_build/html** directory using your browser. Your code page will now appear in the index (**Figure 6**). When you open the code page, you will see your auto-generated documentation about the **Card** class (**Figure 7**).

Update **code.rst** to auto-generate the documentation for the **Deck** class by repeating the step where you used **autoclass** to generate the docs for the **Card** class. The **code.rst** file should look like this:

```
Code docs
=========

.. module:: card

.. autoclass:: Card
    :members:

.. autoclass:: Deck
    :members:
```

Rebuild your website and check that the code page now includes documentation for **Card** and **Deck** – it should look like **Figure 8**.

Creating pages

In addition to pages that describe your code, you can include pages with other content on your project website.

All the pages you want to add need to be formatted using the reST markup language. This primer (**magpi.cc/pBzQUQ**) will show you how to use the most common reST syntax elements.

To get started, let's create an About page that will display some details about the project and where to find more information. Create a new file called **about.rst** and add a title to the page:

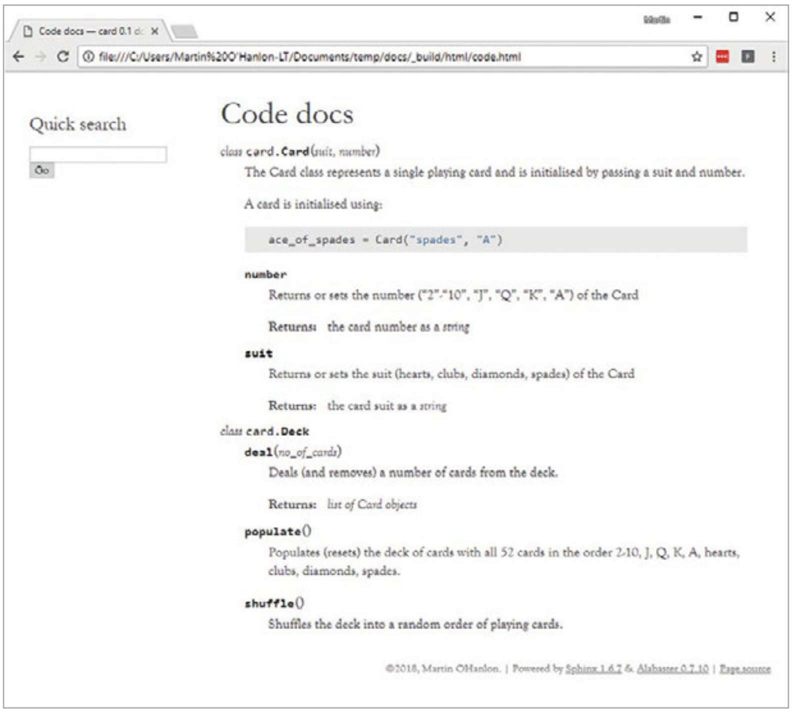
```
About this project
==================
```

Add some text about the project; for example:

```
Deck of cards is a set of classes for creating
playing cards.
```

Perhaps add a hyperlink to the project's source code:

```
You can create this project yourself by
visiting `this page <https://projects.
raspberrypi.org/en/projects/deck-of-cards>`_.
```


Code docs — card 0.1 d
file:///C:/Users/Martin%20O'Hanlon-LT/Documents/temp/docs/_build/html/code.html
Quick search
Go

Code docs

class **card.Card**(*suit, number*)

The Card class represents a single playing card and is initialised by passing a suit and number.

A card is initialised using:

```
ace_of_spades = Card("spades", "A")
```

number

Returns or sets the number ("2"-"10", "J", "Q", "K", "A") of the Card

Returns: the card number as a *string*

suit

Returns or sets the suit (hearts, clubs, diamonds, spades) of the Card

Returns: the card suit as a *string*

class **card.Deck**

deal(*no_of_cards*)

Deals (and removes) a number of cards from the deck.

Returns: *list of Card objects*

populate()

Populates (resets) the deck of cards with all 52 cards in the order 2-10, J, Q, K, A, hearts, clubs, diamonds, spades.

shuffle()

Shuffles the deck into a random order of playing cards.

©2018, Martin OHanlon. | Powered by Sphinx 1.6.7 & Alabaster 0.7.10 | Page source

Figure 8 You're almost done!

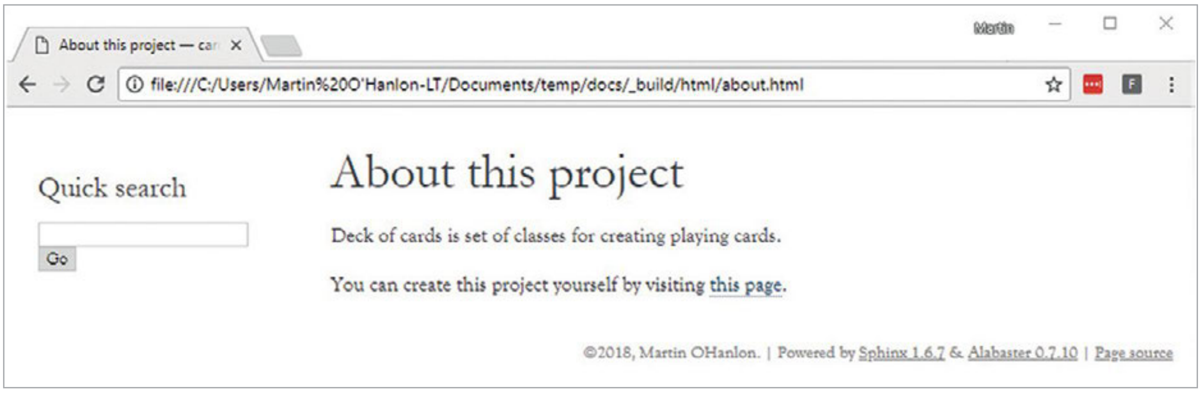
When you have finished creating your **about.rst** page, add it to the **index.rst** file.

```
Welcome to card's documentation!
================================

.. toctree::
   :maxdepth: 2
   :caption: Contents:

   about
   code
```

Rebuild your project to see your new page using the following command:

```
make html
```

And you're done! Check **Figure 9** to see how it should all look. You can add more pages to your project site now, e.g. starting with FAQs.


About this project — car
file:///C:/Users/Martin%20O'Hanlon-LT/Documents/temp/docs/_build/html/about.html
Quick search
Go

About this project

Deck of cards is set of classes for creating playing cards.

You can create this project yourself by visiting this page.

©2018, Martin OHanlon. | Powered by Sphinx 1.6.7 & Alabaster 0.7.10 | Page source

Figure 9 Now you can add separate pages to your website

JON MCLOONE

Jon McLoone holds a degree in mathematics from the University of Durham and is the Director of Technical Services, Communication and Strategy at Wolfram Research Europe.

jon.mcloone.info

PROGRAMMING MINECRAFT PI WITH WOLFRAM LANGUAGE

You'll Need

- Wolfram Language
- Wireless connection
- MinecraftLink Library

Combining Wolfram Language and Minecraft provides a fun playground for learning coding

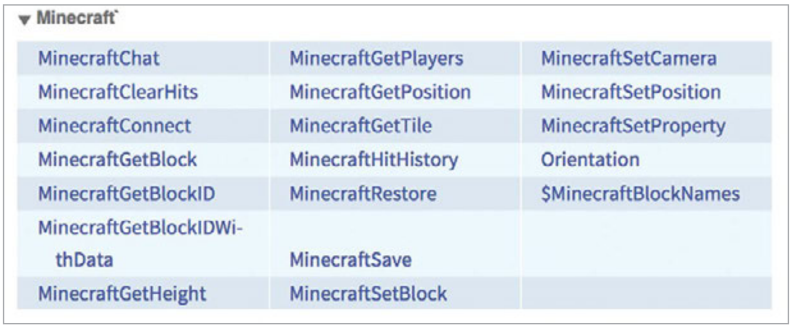
If you are a gamer, you can use the richness of the Wolfram Language to programmatically generate all kinds of interesting structures in the game world of Minecraft, or to add new capabilities to the game. If you are a coder, then you can consider Minecraft just as a fun 3D rendering engine for the output of your code.

First, make sure that your Raspberry Pi is up to date and then open Mathematica. Install the MinecraftLink Library by entering the following code, followed by **SHIFT+RETURN** to evaluate:

```
PacletInstall["MinecraftLink"]
```

The MinecraftLink library adds a set of new commands to the Wolfram Language for connecting to a running Raspberry Pi Minecraft game. You will find basic documentation on them by searching for 'Minecraft' in the Wolfram Help system. Start by launching Minecraft on the Raspberry Pi, and then start a fresh game or open an existing one. In Mathematica, load the library by evaluating:

Figure 1 Adding the MinecraftLink library to the Wolfram Language extends it with these new commands



▾ Minecraft`

MinecraftChat	MinecraftGetPlayers	MinecraftSetCamera
MinecraftClearHits	MinecraftGetPosition	MinecraftSetPosition
MinecraftConnect	MinecraftGetTile	MinecraftSetProperty
MinecraftGetBlock	MinecraftHitHistory	Orientation
MinecraftGetBlockID	MinecraftRestore	$MinecraftBlockNames
MinecraftGetBlockIDWithData	MinecraftSave	
MinecraftGetHeight	MinecraftSetBlock	

```
<< MinecraftLink`
```

Don't overlook the grave accent ("`") at the end. You can find the new commands in **Figure 1**. If you are connecting from a different computer, you must tell Mathematica the IP address of the Raspberry Pi:

```
MinecraftConnect["10.10.163.22"]
```

You don't need to do this if both programs are on the same machine, but if you need to reset the connection, you can use `MinecraftConnect` or `MinecraftConnect["localhost"]`.

Let's test to see if that worked by evaluating:

```
MinecraftChat["Hello from the Wolfram Language"]
```

You should see the message 'Hello from the Wolfram Language' appear briefly in the game.

Minecraft uses a simple {x, y, z} co-ordinate system, where x and z are the horizontal directions and y is the vertical direction. You can see the co-ordinates in the top-left corner of the screen, but to get them programmatically you can use:

```
MinecraftGetPosition[]
```

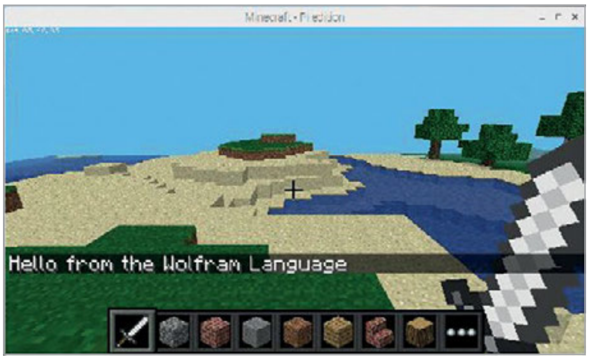
We can teleport the character to a new location (in this case, up in the air):

```
MinecraftSetPosition[{0, 50, 0}]
```

Language

>WOLFRAM LANGUAGE

DOWNLOAD:
magpi.cc/GTpzNv



Hello from the Wolfram Language

Above After evaluating your code, this message should appear briefly in the game chat area

If we have just started a game, then 10 blocks in front of us is {0,10,0}. But depending on how mountainous the terrain is, that block might be above or below ground. We can find the surface level with:

```
y = MinecraftGetHeight[{0, 8}]
2
```

We can test that by looking at the block at that position. It should be Air:

```
pos = {0, y, 8}
MinecraftGetBlock[pos]
```

Now we can start building. We can place blocks of any type – for example, 'Wood':

```
MinecraftSetBlock[pos, "Wood"]
```

You can specify the block material by name, BlockID number, or using an Entity. We remove them by just overwriting the block with something else, such as 'Air':

```
MinecraftSetBlock[pos, "Air"]
```

One reason to use the Wolfram Language for this is that it handles all kinds of interesting 2D and 3D objects, and we have set up the **SetBlock** command to handle these fairly automatically. For example, let's paint a letter X in the sky in gold:

```
MinecraftSetBlock[pos, "GoldBlock", "X"]
```

By default, rasterized content will be 12 blocks wide, so if you need more detail, you can increase that with an option.

```
  MinecraftSetBlock[pos, "GoldBlock", "🐺",
RasterSize → 50]
  MinecraftSetBlock[pos, "Air", "🐺",
RasterSize → 50]
```

Use "\[WolframLanguageLogo]" to type: 🐺 Anything you make can become blocks. Here is a plot of the function sin[x] (see **Figure 2**):

```
  MinecraftSetBlock[pos, "Dirt",
Plot[Sin[x], {x, 0, 12}, Axes → False],
RasterSize → 18]
```

If the content is a 3D geometry, then it will be rasterized in 3D:

```
  MinecraftSetBlock[pos, "Wood", Sphere[],
RasterSize → 50]
```

There are lots of 3D geometry primitives and they can be combined in many ways, including cuboids, a pyramid, and a sphere to make a basic house. We can look at our creation from the air by controlling the camera:

```
MinecraftSetCamera["Fixed"];
MinecraftSetCamera[{0, 25, 6}];
```

We can interact with blocks that you hit using the right mouse button while holding a sword. The left mouse button just places and smashes blocks, but the right mouse button creates events that wait for us to read and act on them. You can read these with:

```
MinecraftHitHistory[]
```

This shows that since the game started, we have done two of these special hits, each time on the same block at {-1,2,2}, on face number 1 (the top of the block). We are player 1, but there could be multiple players. We can fetch these pieces of information by position and name. For example, **MinecraftHitHistory[-1]** is the last hit, and we can extract its 'Position' information and use that co-ordinate in **MinecraftGetBlock** to discover the type of block that was most recently hit.

This shows that since the game started, we have done two of these special hits, each time on the block at {-1,2,2}, on Face 1 (the top of the block). We can fetch these pieces of information by position and name.

As well as giving you a simple programming interface to Minecraft, similar to other languages, the Wolfram Language contains hundreds of high-level functions that let you develop much more exciting projects quickly. You might want to check out some of the 3D geometry, 3D image processing, and built-in data sources as a starting point.

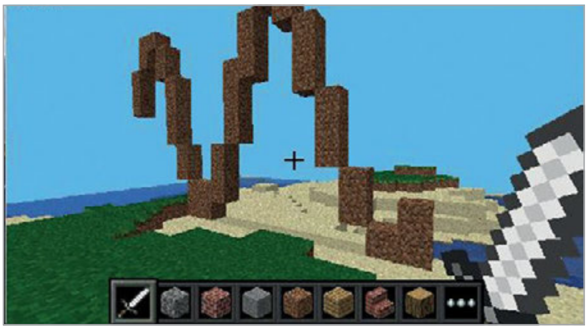


Figure 2 Anything you can create in the Wolfram Language can be made into blocks, like this plot of the function sin[x]

BRIAN BEUKEN

Very old game programmer now teaching very young game programmers a lot of bad habits at Breda University of Applied Science in Breda NL.
scratchpadgames.net

CODING GAMES ON THE RASPBERRY PI IN C/C++ PART 08

Optimising your game so it doesn't quite run as slow

You'll Need

- Code::Blocks
- sudo apt-get codeblocks
- stb_image.h installed previously

Last time we had a **`#define`** that seemed to give us a massive speed boost: it totally changed the game from being a slow and unplayable but clearly-all-there game, to actually being a little too fast – all by setting one value.

Why was there such a massive difference? Well, it's nothing to do with the **`#define`** itself – all that did was tell our compiler that if the defined value was true, to compile a different chunk of code between the **`#if`**, **`#else`**, and **`#endif`** directives.

This is known as a compiler directive; it causes different code to be produced and is frequently used to selectively test and debug code or add extra output features we don't want to include in a final build.

Why so slow?

But why was the code initially so slow? It harks back to that concept we already described of a bottleneck.

Every time we get the CPU to talk to the GPU, we need to wait for the GPU to wake up from its intense dreams of shifting triangles; hit its snooze button (twice); get its dressing gown on; find its slippers; stumble down the stairs; check through the spy-hole to see who's there; recognise the CPU delivery man; fumble for the door keys; take a couple of bits of info at a time, closing the door and stacking the parcels of info, then ask the CPU for another couple. When done, it goes back to bed to continue to dream.

It's only fast when it's left to its own devices doing what it loves to do. And while the GPU is doing its getting-up routine, our CPU is waiting around.

There's nothing really wrong with the code we wrote in terms of it doing what we want it to do. Tiles are on screen, sprites are on screen, Bob moves as he's supposed to, it's all there and working! It just didn't take into account the fact that our GPU and our CPU really don't like to talk to each other, and we made them talk to each other *a lot*.

So game programmers become more than a little OCD over how long things take, but also, when and where to do things – and whether or not the same results apply, which allows us not to repeat the same things. By setting the **`#define FastUpdate`** flag to true, we altered the way our graphics were set up and updated. Rather than talk to the GPU each object update, we let the CPU focus on the logic and the tiles didn't need to draw themselves until after the logic was done.

We also took note that our tiles aren't actually going anywhere, they're always visible at the same point at the same time, so why should we ever be transferring their data from the CPU to the GPU? Much better to set up a GPU-side storage buffer called a variable buffer object (VBO), to hold all that position info and avoid the transfer each frame. The GPU is happy to work with data that it already has access to, which lets it work at its fastest rate, not talking to the CPU – which doesn't need to wait.

Optimising is king, but...

If optimising is so important, then why didn't we optimise from the start? Mainly, we did that

USE THE TEXTURE SPACE WELL

Having lots of small textures is good for visualising, but bad for drawing group images in as few textures as possible.

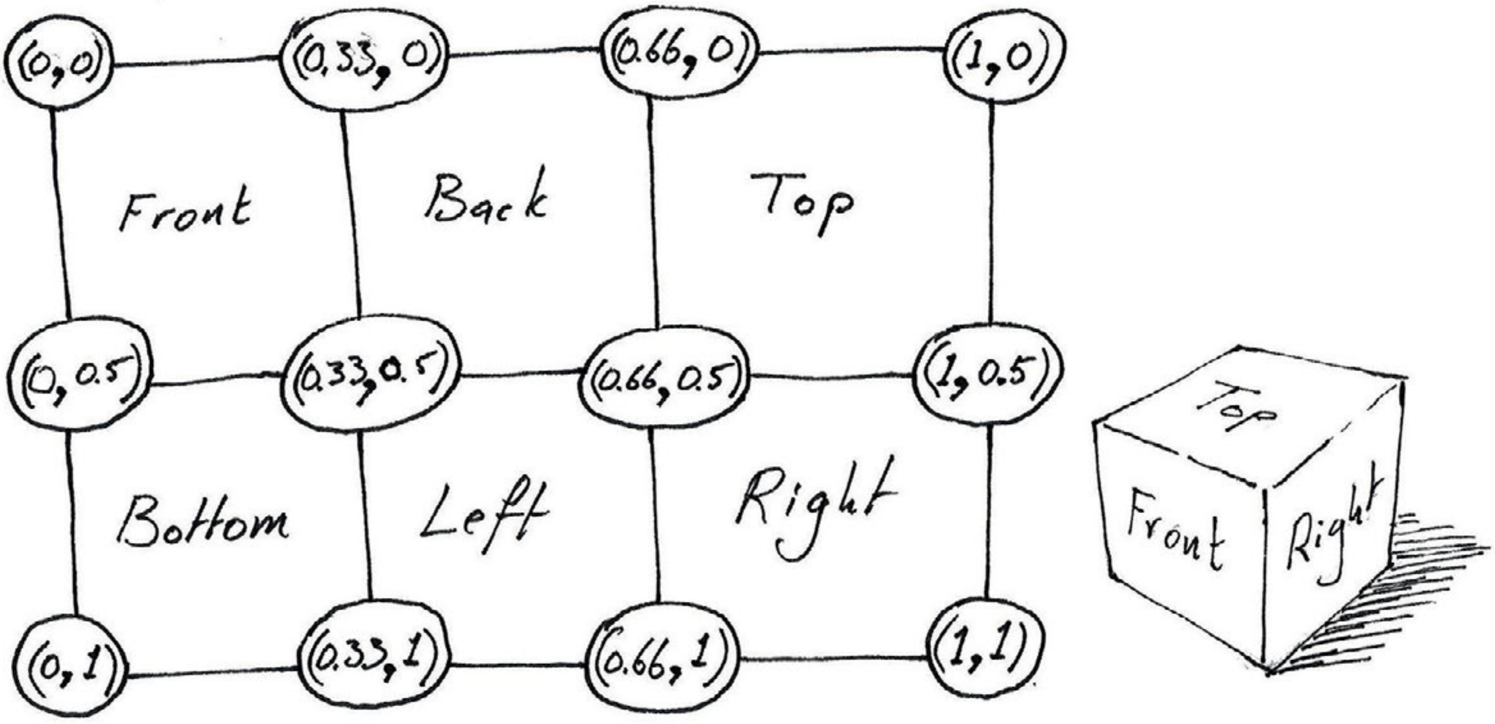


Figure 1 Simple individual frames

to really impress on you as a new coder the quite devastating impact of badly designed code, but also to encourage you that usually it only needs a simple change and a think about how we do things to get our performance back.

Over time, as you get better at coding, you will amass a small toolkit of dos and don'ts; this is one of your first don'ts.

Optimising also has a bit of a downside: the code can become less readable and clear. Our intent to draw is very obvious when we see it in our update functions, but now we're delaying the actual draw to some time after the updates are all done. So are we sure they are ready to go?

Therefore, what we've done is to move most of the setting up of the tiles to a one-time VBO initialisation, which enables us keep the data in one place that the GPU can access instantly. Then, rather than do a draw call for each tile object – all 2560 of them – we draw all our tiles with one draw call. The differences are very clear to see. The conclusion should be obvious: avoid doing too many draw calls!

Planning for bigger things

Only the sprites – in this case Bob – really need to move on screen, making it impossible to keep an unchanged VBO of his position info, and that's the only object that currently still has a CPU-to-GPU update in its code update, but… what if we have 100 sprites?

We are lucky here, because we are really only going to have a few sprites on screen, so we can probably get away with continuing the update/draw cycle. However, it's really much better if we take the hint that faster is better.

That means that rather than draw them every object update, we need to find a way to draw them just after we draw the tiles. One solution is to keep a note of the updated sprites data needed to draw, in a CPU buffer, which can be sent direct to the GPU in one draw call.

> “Optimising also has a bit of a downside: the code can become less readable and clear”

Doing a draw call from a CPU buffer is much slower than doing a draw call from a VBO, but sadly the sprites move, so their positioning is variable, as is their current frame for animation. So there's just no benefit to setting up the CPU buffer, making a VBO, then calling the draw. We will just allow the CPU to send the data directly.

This is just one of those things we need to learn to live with. Variable or dynamic data can't easily be stored in the GPU memory, only in the CPU's. So we just have to try to limit it to the least amount needed, store it until the data is complete, and then send when ready.

KEEP CLASSES SMALL

Keep only what you need to keep in a class; try to separate objects into individual classes and avoid catch-all types.

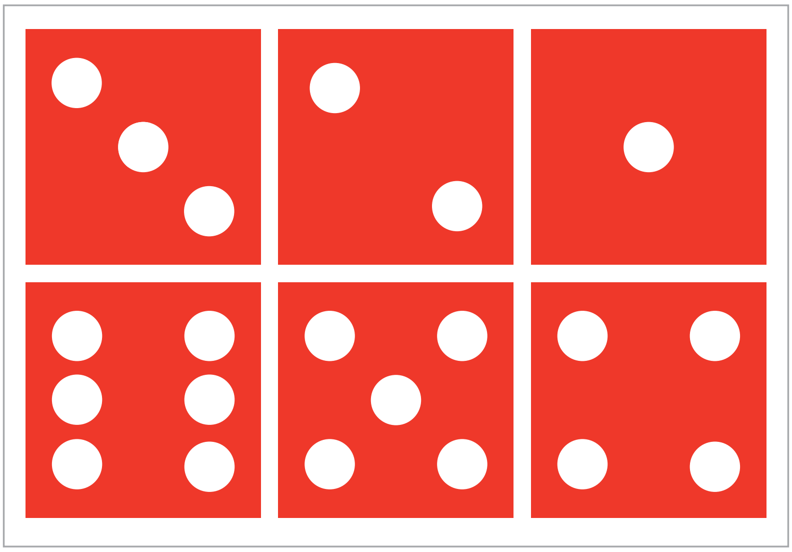


Figure 2 So many maps, so few tiles

If the sprites share so much info, then we can in principle define a data pattern that consists of:

- **X and Y co-ordinates, and the defunct Z**
- **Texture co-ordinates**
- **Quad vertices**

If we define a small data structure to contain this info, then we have a fairly simple way to create an array, or vector, of the structure and, when the time comes, we set it up as the 'attributes' we want to send to the shaders to draw.

MAKE IT WORK FIRST

Now that we can see that `FastUpdate` is so much better, we should actively remove the old slow code.

We also need some optimising on our textures. One texture per frame means we set up the texture before each draw call, but if we arrange all our images into one texture, we only need one set up.

It does force us to use a different way to address each frame, but that's not too hard to visualise. See **Figures 1** and **2**. We keep the six faces/frames of a die in one texture, which allows us to reference them using what are known as UV cords (we use UV to avoid confusion with XY for position). These represent fractions of the size of the texture, from 0.0f to 1.0f along its U and V co-ordinates. Imagine you said halfway along and halfway down is the start of the image we want; that means a UV of 0.5f, 0.5f.

One more limitation: all the sprites need to have the same shader, since we set up the shader we need before the draw call. If we want to have a couple of different shaders, for different effects, it's OK to do so within reason, but don't chop and change.

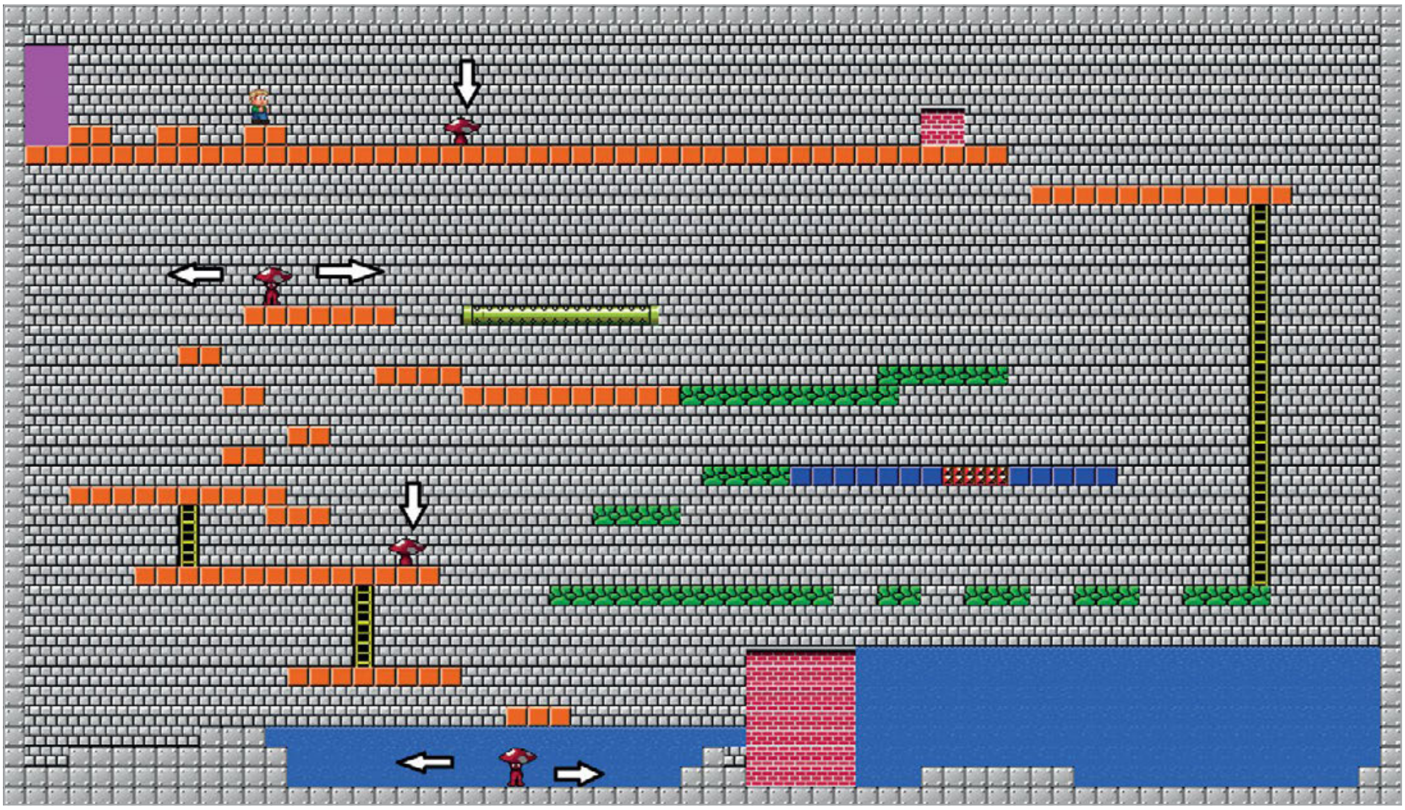
Review the source code to see how our new faster but slightly less readable code is doing its thing.

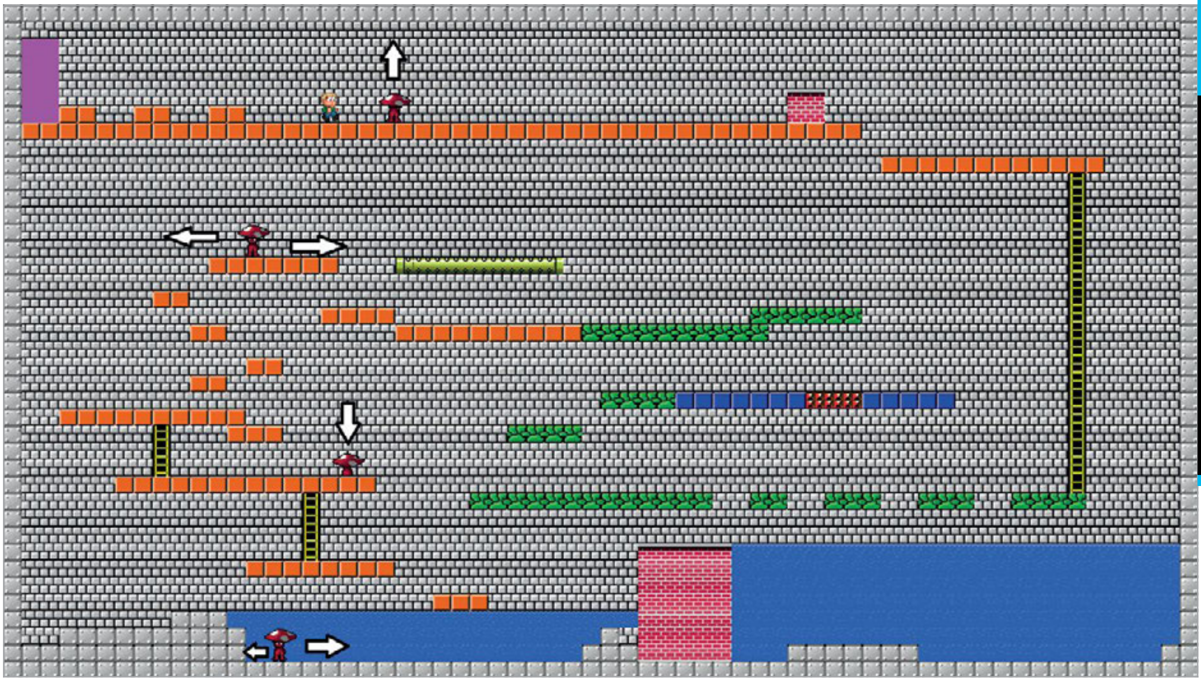
Adding the baddies

Just like Bob, our baddies interact with the map, and obey some basic rules of gravity and platform solidity. Though we don't make them climb, as they don't really need to.

We could make a class called 'Baddies', but we want a number of them to do different things, which would mean keeping code for different types of baddies in the same class and having flags or type variables to let your update and draw systems know what to work on.

Below Baddies will be baddies





Left Enemies wake up when close

Language

>C++

NAME:
MagPiHelloTriangle.cpp
SimpleObj.h/cpp
Game.h/cpp
OGL.h/cpp
Bob.h/cpp
HideAndChase.h/cpp
PointToPoint.h/cpp
Input.h/cpp

DOWNLOAD:
magpi.cc/hOZQbG

The baddies work in almost the same way as Bob, except of course they don't use key inputs, they use their own logic. They use our **`SimpleObject`** base class and all we need to do is focus on how they work for their own logic. Nothing too fancy here, we're going to define two basic types of baddie which have a few simple logic states they can switch between. So now we introduce the wait-and-chase baddie, and a point-to-point baddie.

The wait-and-chase baddie is a mushroom graphic with a walk sequence and a hide graphic, and has only two states: wait and, erm, chase. Waiting is not an idle 'do nothing' kind of wait, though: the mushroom is testing each frame to see if Bob is close to him; if so, he switches to a chase state and if he escapes, goes back to a wait state.

Point-to-point, as you can imagine, is simply a case of our baddie walking to one point, changing direction, then walking to another. Since many types of baddies may want to use this, it can be an inherited class for others who use the logic, but not the draw systems.

Both baddies are simple contact enemies which we can signal to the main loop, or to Bob's code, that he's been hit. For now we will just recognise it, and act on it soon. With that done, all we need to do is make absolutely sure we set them up in the game initialisation, give them some starting info on position and frames, and add them to the update and draw cycles. Feel free to add your own types of baddies!

Now review the code. Notice how the update code really is very small, and quite focused on the specific logic needed for the specific class.

Next time

We've tackled most the key elements, but our game looks a bit naff: because of our hi-res screens, our full-screen display and small sprites are really rather small and hard to see. We want something a bit more chunky and retro, and that means we need to consider the view area of our screen and how much we make visible. These are the basics of scrolling.

> We want something a bit more chunky and retro, and that means we need to consider the view area of our screen and how much we make visible

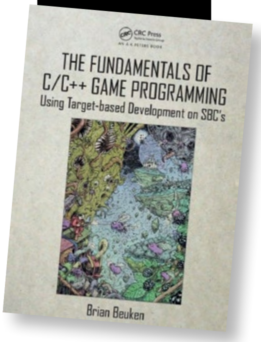
Because our game logic is running detached from any concept of what the screen looks like, we can zoom our screen to any part of the map we want. And if we choose to zoom to a variable point, for example around where Bob is moving, we can effectively get scrolling which follows Bob as he moves around.

We need to make quite some changes to the way our shaders work and how we view the play area to get this to work well, so that's our job next time.

Again, we ran out of space to do text, so once we get scrolling up and running, we'll dedicate the rest of the space to doing text.

LEARN MORE ABOUT C

Brian has a whole book on the subject of making games C and C++, called *The Fundamentals of C/C++ Game Programming: Using Target-based Development on SBCs*. Grab it here: **magpi.cc/nUkjEt**



FREQUENTLY ASKED QUESTIONS

Your technical hardware and software problems solved…

NEED A PROBLEM SOLVED?

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

EMBEDDED ELECTRONICS

WHAT ARE EMBEDDED SYSTEMS?

Embedded

An embedded system is a (usually small) computer system that performs a dedicated function within a large system. They basically control or monitor a smaller part of a larger system, and are generally only used for one specialised thing.

Benefits

Computers are designed to perform multiple functions, either at once or separately. Why this grants them great flexibility for day-to-day use, it means they're not always optimised to perform a single task. Dedicated embedded systems/chips can not only work better for a specific task; they're usually cheaper in bulk as well.

Raspberry Pi

Due to the tiny size of the Raspberry Pi, it's great for prototyping systems that can then be made embedded. Because it's also pretty cheap, some people tend to just stick with the Raspberry Pi as the embedded device in their system – there's even a specialised version of the Pi that aids in this, called the Compute Module.

WHAT RASPBERRY PI TYPES ARE GOOD FOR EMBEDDED SYSTEMS?

Full-size Pi

Raspberry Pi models like the 2, 3, 3B+, and even the Model As to a lesser extent, are widely available so are great for prototyping. Although they're still small, they're not the smallest Pi boards, and the more powerful Pi models require more electricity as well. Depending on the project, this could be a serious weakness.

Pi Zero

The Zero is an excellent choice for embedded systems as it's tiny and requires very little power to run in comparison to its bigger siblings. It also runs a little faster than the Model A+ by default. The main issue you might find with it is buying in bulk, though, and it's nowhere near as powerful as something like a 3B+.

Compute Module

The Compute Module is a special version of the Raspberry Pi specifically made for industrial applications. It's tiny and includes a standard DDR2 SODIMM connector to plug into projects, flash memory, and more. The full dev kit is a bit more expensive than a normal Pi, but it's a lot better suited for these kind of tasks.

WHAT ABOUT A RASPBERRY PI KIT SO I CAN MAKE A CUSTOM VERSION?

No kits

Unfortunately, you can't get a kit to make a Pi. Raspberry Pi boards come with all the components pre-soldered, so you can't get a basic kit and then pick and choose what components to add. This has been the case since the Pi was first released in early 2012.

Desoldering

This doesn't mean you can't desolder components from your Raspberry Pi, though – although we'll say that this is not recommended and can be dangerous unless you know exactly what you're doing (and don't mind voiding your warranty as well). We've seen people make incredibly slim Raspberry Pis by removing some of the more bulky connectors from the top part of the board.

Customisation service

The Raspberry Pi can be customised by element14, however, depending on the customer's needs. This way, you get the kind of Pi you need without having to do any of the hard work yourself. You can find more info about this (and whether or not it's for you) here: **magpi.cc/eyipPm**.

FROM THE RASPBERRY PI FAQ RASPBERRYPI.ORG/HELP

CAN I EXTEND THE PI CAMERA MODULE RIBBON CABLE?

Yes. We have reports of people using cables up to 4 m in length and still receiving acceptable images, although your experience may differ.

CAN I HAVE A CAMERA WITH MORE MEGAPIXELS?

No; the Raspberry Pi Camera Modules are the only cameras that are compatible with the Raspberry Pi's Camera port. There are currently no plans to release a higher-resolution sensor.

HOW MUCH POWER DOES THE CAMERA MODULE USE?

The Raspberry Pi Camera Modules requires 250 mA to operate. Ensure that your power supply can provide enough power for the connected Camera Module, as well as for the Raspberry Pi itself and any peripherals directly attached to it.

DOES THE RASPBERRY PI COME WITH A CASE?

An official case for the Raspberry Pi is available from our distributors. There are also lots of home-brew case discussions on the forum, as well as several third-party cases available. We suggest stopping by the cases sub-forum and reading some of the threads about cases you can purchase or build yourself.

DOES IT FIT IN AN ALTOIDS TIN?

It is possible to make a case for the Raspberry Pi out of an Altoids tin: there are instructions given in a forum post at **magpi.cc/gLTrYB** (the directions are also available as an Instructables guide). However, you may find that this is not possible with older boards, as their form factors are not as neat. You can, of course, fit a Pi Zero into an Altoids tin with plenty of room to spare.

THE MAGPI PRINT SUBSCRIPTION

Having trouble with your subscription to *The MagPi*? Here are your most common questions answered:

How can I find out details about my print subscription?
For all print subscription enquiries, please contact **rpipresshelp@raspberrypi.org** – this includes changes of address, and finding out when your subscription ends.

I seem to be missing an issue of my subscription, what should I do?
Unfortunately, things do get lost in the post. If that's the case, make sure to email **rpipresshelp@raspberrypi.org**. Include your name, the email address you signed up with, and which issue you're missing – all being well, they will send you a replacement issue.

If I re-subscribe for a year, do I get the free Pi Zero W bundle again?
Yes. We think the world of re-subscribers and we treat them just as well as new subscribers. You'll get whatever free gift we include for people taking out a year's subscription (currently a Pi Zero W starter kit), whether it's your first sub to *The MagPi* or your fourth.

K.G. ORPHANIDES

K.G. is a writer, developer and lapsed session musician. They're currently working on environmental audio for the Codename Caerus game project.
twitter.com/kgorphanides

MAKING MUSIC

TURN YOUR RASPBERRY PI INTO A COMPLETE HOME RECORDING STUDIO WITH DIGITAL AUDIO WORKSTATION SOFTWARE

You'll Need

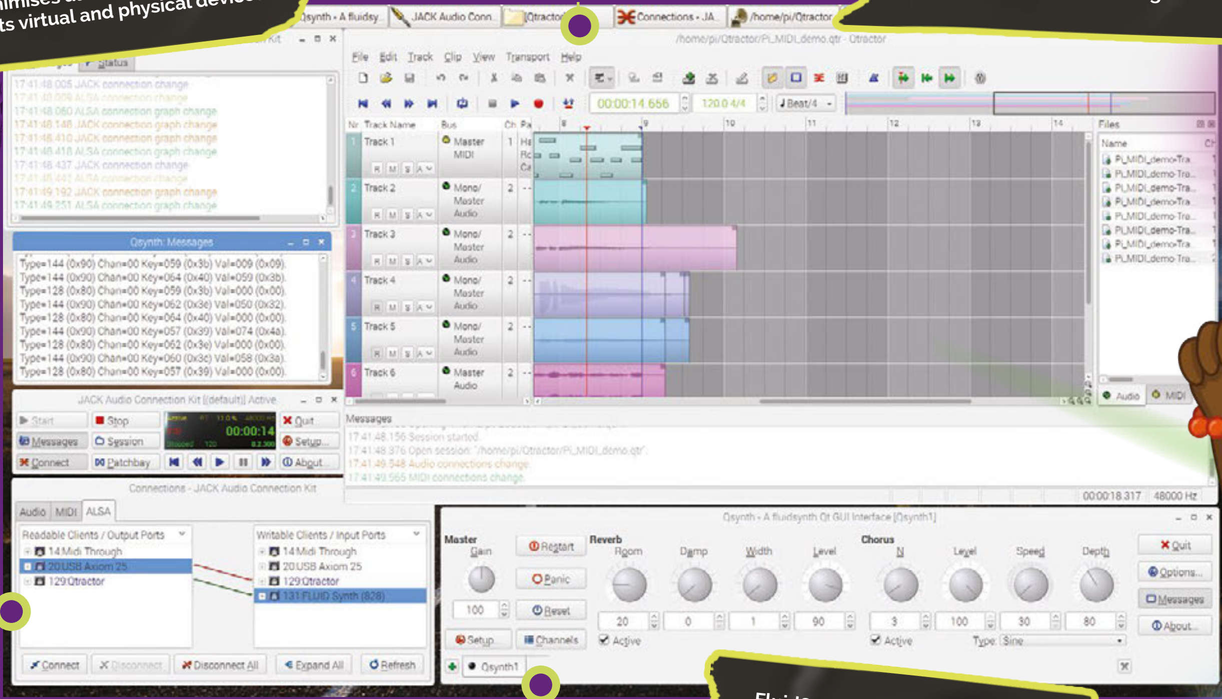
- Speakers/ headphones for audio monitoring – use the best ones you have
- MIDI keyboard (optional)
- USB audio interface (optional)
- MIDI instrument definition file **magpi.cc/LLbvwo**

Compact, silent, and capable of seamlessly integrating with professional-quality USB audio devices, your Raspberry Pi can become a complete home recording studio. We'll introduce the free tools you'll need to get started with audio production and show you how to set up your Pi for low-latency digital audio recording and MIDI synthesis.

In the process, you'll use software including the JACK Audio Connection Kit, FluidSynth software synthesizer, and Qtractor DAW (digital audio workstation). We'll also introduce tools for playing MIDI instruments using your computer keyboard, editing notation with your mouse, and exporting your recordings into easily distributable formats.

JACK is a low-latency audio daemon that minimises audio delay and connects virtual and physical devices

Qtractor is a stable tracker-style digital audio workstation for MIDI and digital audio recording



```
17:41:48.005 JACK connection change
17:41:48.009 ALSA connection change
17:41:48.060 ALSA connection graph change
17:41:48.148 JACK connection graph change
17:41:48.410 JACK connection graph change
17:41:48.418 ALSA connection graph change
17:41:48.437 JACK connection change
17:41:48.441 ALSA connection change
17:41:49.192 JACK connection graph change
17:41:49.251 ALSA connection graph change
```

Qsynth: Messages

```
Type=144 (0x90) Chan=00 Key=059 (0x3b) Val=009 (0x09).
Type=144 (0x90) Chan=00 Key=064 (0x40) Val=059 (0x3b).
Type=128 (0x80) Chan=00 Key=059 (0x3b) Val=000 (0x00).
Type=144 (0x90) Chan=00 Key=062 (0x3e) Val=050 (0x32).
Type=128 (0x80) Chan=00 Key=064 (0x40) Val=000 (0x00).
Type=144 (0x90) Chan=00 Key=057 (0x39) Val=074 (0x4a).
Type=128 (0x80) Chan=00 Key=062 (0x3e) Val=000 (0x00).
Type=144 (0x90) Chan=00 Key=060 (0x3c) Val=058 (0x3a).
Type=128 (0x80) Chan=00 Key=057 (0x39) Val=000 (0x00).
```

JACK Audio Connection Kit [(default)] Active

/home/pi/Qtractor/PI_MIDI_demo.qtr - Qtractor

Messages

```
17:41:48.156 Session started.
17:41:48.376 Open session: "/home/pi/Qtractor/PI_MIDI_demo.qtr".
17:41:49.548 Audio connections change.
17:41:49.565 MIDI connections change.
```

Connections - JACK Audio Connection Kit

Readable Clients / Output Ports:
14:Midi Through
20:USB Axiom 25
129:Qtractor

Writable Clients / Input Ports:
14:Midi Through
20:USB Axiom 25
129:Qtractor
131:FLUID Synth (828)

Qsynth - A fluidsynth Qt GUI Interface [Qsynth1]

Master: Gain 100
Reverb: Room 20, Damp 0, Width 1, Level 90, ☑ Active
Chorus: N 3, Level 100, Speed 30, Depth 80, ☑ Active, Type Sine

FluidSynth's Qsynth front end helps to output MIDI through a standard audio device

The Raspberry Pi has limited memory and processor power and lacks an integrated line input, but with some external hardware and judicious software choices, the latest Raspberry Pi 3 B+ is easily up to the task of producing and editing complex multitrack music.

> CD-quality audio has a sample rate of 44.1 kHz and a 16-bit bit depth

HOW DOES DIGITAL AUDIO RECORDING WORK?

When you record sound digitally, an analogue-to-digital converter (ADC) in your recording device cuts the analogue waveform into sections that will be encoded as data using a technique called pulse-code modulation (PCM), with its accuracy determined by its sample rate and bit depth.

Sample rate, also called sample frequency, refers to the number of slices that an analogue waveform is divided into while it's recorded digitally. Bit depth is a measurement of how much data is used to encode the digital signal.

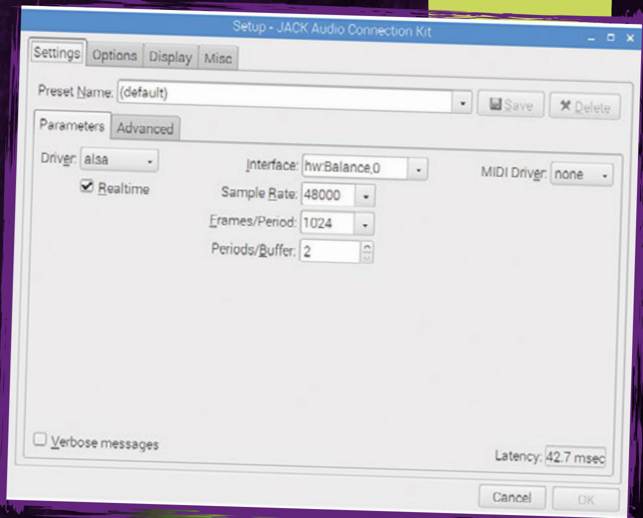
CD-quality audio has a sample rate of 44.1kHz and a 16-bit bit depth. When recording digital audio, you should treat this as your minimum quality requirement. Higher sample rates and bit depths are supported by many audio devices, and are useful in specific recording situations, but we're going to stick with our software's default settings, with a sample rate of either 44.1kHz or 48kHz, depending on your hardware.

WHAT IS MIDI?

MIDI (Musical Instrument Digital Interface) is a standardised method of transmitting and encoding musical signals, first devised in the early 1980s. It records input from MIDI instruments as a series of instructions that encode the pitch, duration, velocity – the dynamics of the note, such as the force with which a key is struck – and other audio characteristics of each note, including effects such as pitch bends, tuning, and decay.

While older MIDI hardware used five-pin DIN connectors, most modern devices send MIDI over USB. For this tutorial we'll use a software synthesizer, or soft synth, to read MIDI input and play it through a digital-to-analogue converter (DAC) such as those built into USB audio interfaces or the Pi's Broadcom BCM2835 SoC, using software instrument banks called SoundFonts.

The first step in configuring your audio environment is to tell JACK to use ALSA and your interface – a Propellerhead Balance USB device in this example



Setup - JACK Audio Connection Kit

Preset Name: (default)

Driver: alsa

☑ Realtime

Interface: hw:Balance,0

Sample Rate: 48000

Frames/Period: 1024

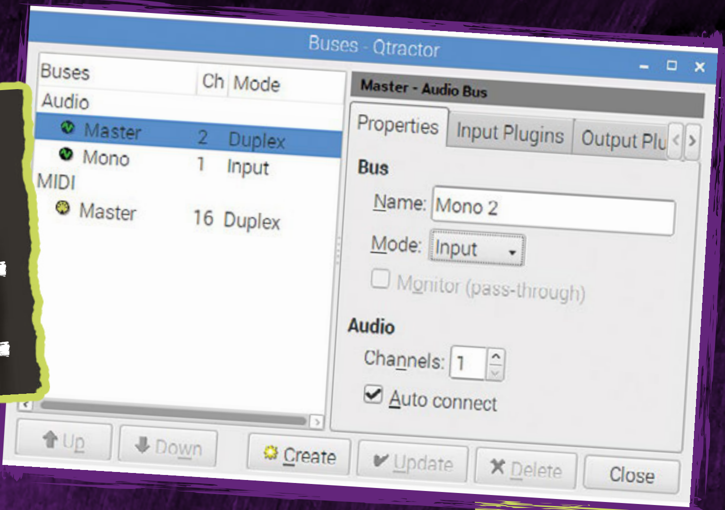
Periods/Buffer: 2

MIDI Driver: none

□ Verbose messages

Latency: 42.7 msec

HARDWARE AND SOFTWARE



Audio
- Master 2 Duplex
- Mono 1 Input

MIDI
- Master 16 Duplex

Master - Audio Bus

Bus
Name: Mono 2
Mode: Input
☐ Monitor (pass-through)

Audio
Channels: 1
☑ Auto connect

To record mono instruments like guitars in Qtractor, you should create a dedicated mono bus

If you just have a Raspberry Pi, with no USB audio or MIDI input devices of any kind, you can still start making music. For MIDI composition, you can enter notes using a mouse or use virtual keyboard software to play in real time using your PC keyboard as an instrument.

For digital audio, libraries of royalty-free samples are available online to edit together into tracks of your own. You can also import recordings made on other devices, such as your smartphone. The only hardware you definitely can't do without is a pair of speakers or some decent headphones.

> "You can also import recordings made on other devices"

GEAR UP

To record a range of live instruments, though, you'll want a USB audio interface with multiple inputs to handle analogue-to-digital conversion. Popular entry-level options include the Behringer U-Phoria UMC404HD and the IK Multimedia iRig Pro Duo. Our screenshots show the – sadly discontinued – Propellerhead Balance audio interface.

If you want something even cheaper and simpler, look for a ¼-inch TRS (guitar) lead to USB converter, which includes a tiny sound card to digitise your audio output, such as the Behringer Line 2 USB.

For vocals or acoustic instrument recordings, a USB condenser mic – which acts as its own dedicated USB audio input device – is a good place to start. The Samson Meteor Mic is a solid budget choice, while Blue Microphones' Yeti Pro is fantastic and can double as a standard XLR mic if you have more to spend.

THE SOFTWARE MACHINE

Digital audio workstation (DAW) software gives you a fully featured environment to record, edit, and combine both audio tracks and input from MIDI devices. They'll typically include modules that can record MIDI data live as you play it in via a keyboard, notation or piano roll editing tools, digital audio recording and editing tools, and mixing capabilities to help you balance audio from various sources.

In this tutorial, we'll use Qtractor. Although it isn't as fully featured as some of its rivals, with the conspicuous absence of a notation editor and MIDI step entry, it's particularly simple to use and is extremely stable on the Pi.

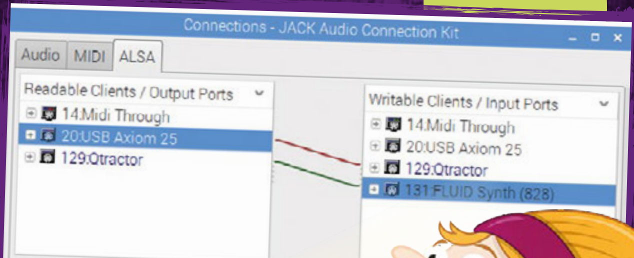
To ensure optimal recording quality and avoid lag, particularly when playing along to previously recorded digital audio tracks, we'll be using the JACK Audio Connection Kit (JACK for short) low-latency audio daemon to handle audio routing and connection, and FluidSynth to give voices to our MIDI keyboard.

We'll also install a handful of other useful musicians' tools: the Lingot guitar tuner; MuseScore notation editor and VMPK (the Virtual MIDI Piano Keyboard) to control a MIDI synth using your computer keyboard; and the Audacity audio editor, which provides a cleaner environment to edit digital audio waveforms.

To install all the software you'll need for this tutorial, run the following in a Terminal window:

```
sudo apt install
qjackctl qsynth qtractor
fluid-soundfont-gs vmpk
musescore lingot audacity
```

We'll also need a MIDI instrument definition file to go with our SoundFonts, so download the open-source **Standard.ins** file from **magpi.cc/Llbvwo.**



Readable Clients / Output Ports
- 14:Midi Through
- 20:USB Axiom 25
- 129:Qtractor

Writable Clients / Input Ports
- 14:Midi Through
- 20:USB Axiom 25
- 129:Qtractor
- 131:FLUID Synth (828)

JACK lets you run virtual wires between devices like your MIDI keyboard, DAW, and soft synth

SET UP YOUR DIGITAL AUDIO WORKSTATION

Linux has integrated support for class-compliant USB audio hardware. Plug in your audio interface and MIDI keyboard, boot your Pi, and you're ready to go. We'll start by loading our software and setting up the JACK audio daemon to correctly route our MIDI devices.

QjackCtl – the GUI control interface for the JACK low-latency audio driver – is installed by default on Raspbian, but isn't visible in the menus. To make it appear, go to Preferences > Main Menu Editor in the Raspbian menu, select Sound & Video, and place a tick next to QjackCtl before clicking OK.

Alternatively, you can start it by running QjackCtl in a Terminal window, which we're going to do, to simultaneously launch it alongside the other software we'll need: Qsynth to run a software synthesizer that outputs MIDI via our audio device, and the Qtractor DAW.

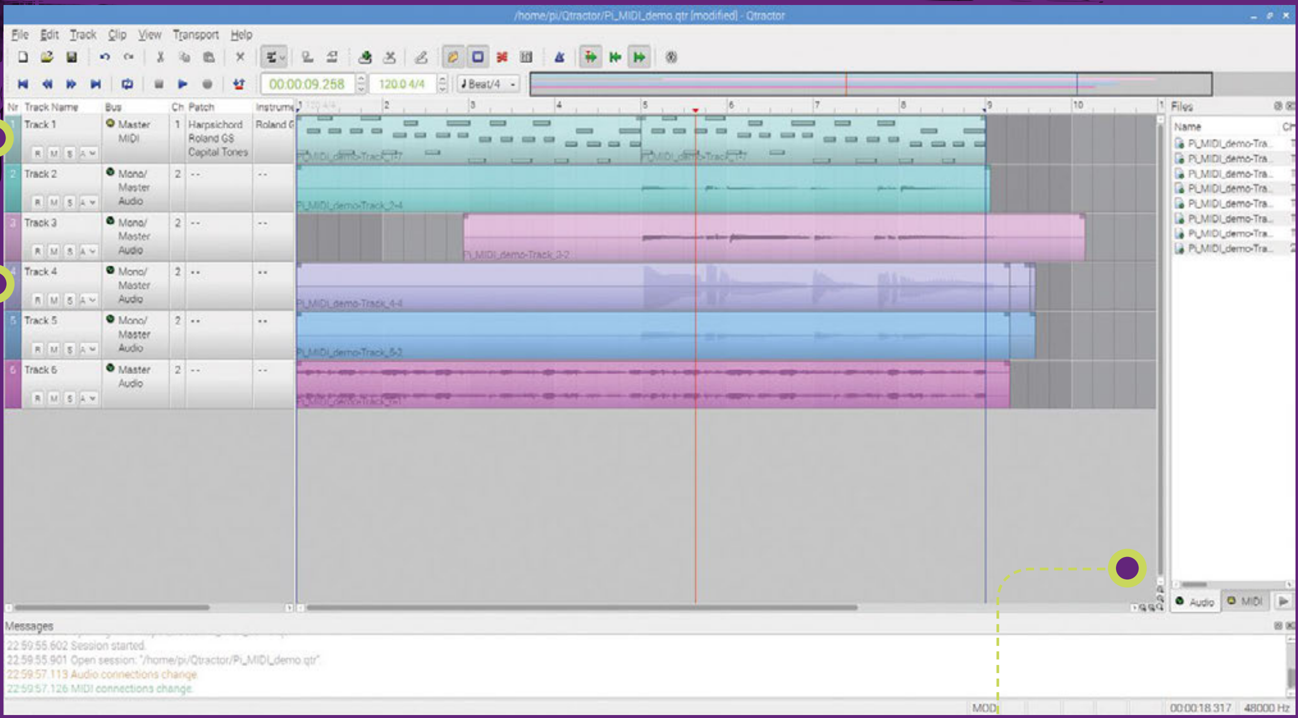
```
qjackctl & qsynth & qtractor
```

The QjackCtl window may spawn with its title bar off screen. If this happens, hold **ALT** on the keyboard and use the mouse to drag it into a more convenient location.

Click the Setup button in QjackCtl, make sure the ALSA driver is selected, and choose your audio output device from the Interface pull-down. If you're using the Pi's built-in sound, you want **hw:ALSA bcm2835**. If you're using a USB output device, select that instead. Click OK and then press Start on the JACK control panel.

Go to Qsynth and select Setup. Click on the SoundFonts tab and click Open. It should automatically open the **/usr/share/sounds/sf2** directory, where you'll find the General MIDI (GM) and Roland General Standard (GS) SoundFonts we installed earlier. Select both of them and click Open.

Having returned to the setup menu, highlight the FluidR3_GS SoundFont and click the up

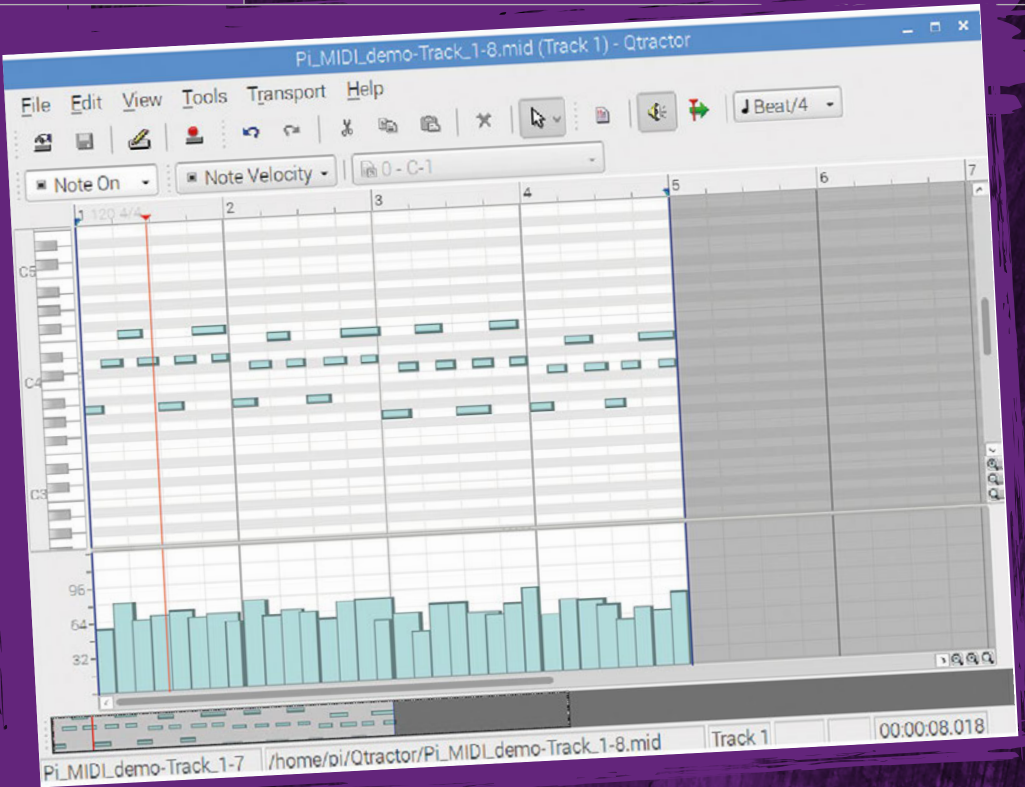


```
22:59:55.602 Session started.
22:59:55.901 Open session: "/home/pi/Qtractor/Pi_MIDI_demo.qtr".
22:59:57.113 Audio connections change.
22:59:57.126 MIDI connections change.
```

MIDI input is shown as a sequence of discrete tones

Digital audio recordings are displayed as a waveform

Five magnifying glass icons allow you to zoom in and out on tracks



Qtractor's piano roll lets you carry out note-by-note MIDI editing

button to make it the default SoundFont, then click OK. Allow Qsynth to restart.

Go to Qtractor and, from the View menu, select Instruments. Click Import and browse to where you saved the **Standard.ins** file we downloaded earlier. Select it and click Open, then select Close and save the instrument changes. You'll now be able to select GM and GS MIDI instruments by name.

USING YOUR MIDI SETUP

Make sure your USB MIDI keyboard is connected – or, if you don't have one, open VMPK (Virtual MIDI Piano Keyboard) from the Sound & Video menu.

Go to QjackCtl and press the Connect button. This is where you string together virtual wires connecting your different devices to one another. Make sure the ALSA tab is selected. You should see a list of input and output devices. In our example, we want to connect our MIDI keyboard to Qtractor for input and Qtractor to FluidSynth for output.

To do this, click and drag from the keyboard in the Readable Clients / Output Ports list on the right, to Qtractor in the Writable Clients / Input Ports on the left. Then drag from Qtractor on the right to FluidSynth on the left.

Before you can use Qtractor to play or record MIDI input, you'll have to create a MIDI track. Right-click in its main workspace and select Add Track, select MIDI as its type, Roland GS as its instrument, Roland GS Capital Tones as its Bank, and any voice you fancy in Program. Then click OK.

You can change a MIDI track's instrument at any time by right-clicking on it and choosing a voice from the Instrument submenu.

READY TO RECORD

The first time you create or start recording a track, you'll be prompted to name your session and select a save directory. It's advisable to create a folder called **Qtractor** in your home directory for this. Once you've started working

on a piece of music, remember to save often by pressing **CTRL+S**.

To record MIDI input in real time, select the track you want to record to and arm it by pressing the **R** button just below its name. Make sure the Play-head transport – indicated by a red line – is at the beginning of the track by using the **BACKSPACE** key, the Backward toolbar button, or dragging it into place with the mouse. Then press the red Record button in the toolbar and, finally, press Play to begin recording.

Press Stop or hit **SPACE** when you've finished. When you finish recording on that track, remember to press the **R** button again to un-arm it.

If you need a click track to play along to, press the Metronome icon on the toolbar or select Metronome from the Transport menu. You can change the tempo and time signature of your piece in the Current tempo and Snap/Beat sections of the toolbar.

Once you've recorded a MIDI passage, you can remove blank space or unwanted bars from the beginning and end by dragging the blue Edit-head and Edit-tail markers to the beginning and end of the section you wish to remove and pressing **CTRL+SHIFT+DELETE**. The Loop function lets you use similar markers to define a section you want to listen to repeatedly.

If you need to manually tidy up your recording, double-click on the track to open the piano roll. Here, you can see individual notes and their characteristics, such as pitch, timing, and velocity. You can select, move, and delete notes, or apply effects to entire sections of music, such as quantization to make their timing line up precisely with the beat. You can also draw in new notes using your mouse.

> “To record MIDI input in real time, select the track you want to record to and arm it”

KNOW THE SCORE

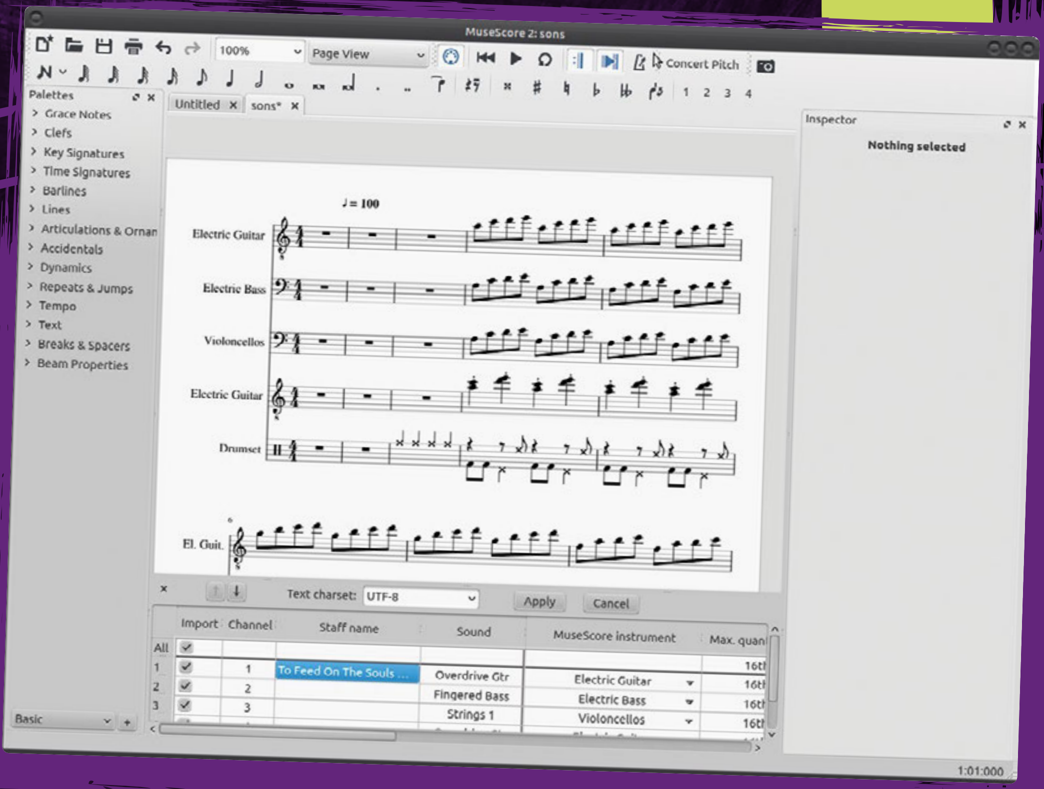
If you're more comfortable working with a traditional musical score than a piano roll, or if you want to print and share your composition for live performance, you'll want a notation editor. Qtractor doesn't have one built in, but we installed MuseScore earlier.

Qtractor saves and exports its MIDI sections as .mid files, which MuseScore can open and edit. MuseScore gives you fine control over the pitch and length of notes, and can also play back MIDI files through JACK: go to Edit > Preferences, choose the I/O tab, and select JACK audio server before saving these settings and restarting the program.

DIGITAL AUDIO RECORDING

Qtractor can also handle digital audio recording from microphones or the instrument inputs of a USB audio interface. For most instruments, such as electric basses, violins, and guitars, you'll want to configure a mono input for

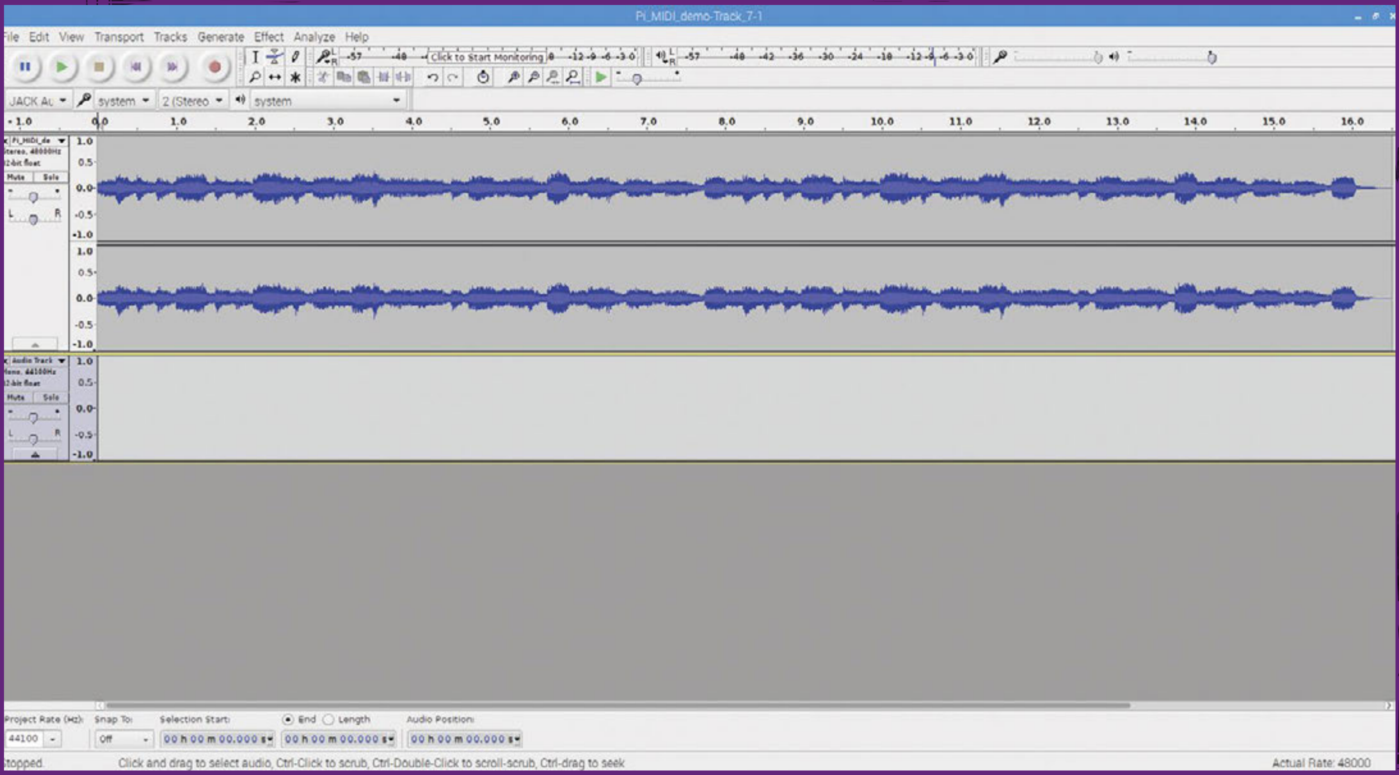
If you read traditional musical notation, tools like MuseScore let you edit and step-enter MIDI music on a stave



Text charset: UTF-8

Apply | Cancel

	Import	Channel	Staff name	Sound	MuseScore instrument	Max. quan
All	✓					16th
1	✓	1	To Feed On The Souls ...	Overdrive Gtr	Electric Guitar	16th
2	✓	2		Fingered Bass	Electric Bass	16th
3	✓	3		Strings 1	Violoncellos	16th


Pi_MIDI_demo-Track_7-1
File Edit View Transport Tracks Generate Effect Analyze Help
Click to Start Monitoring
JACK Au
system
2 (Stereo
system
Project Rate (Hz):
44100
Snap To:
Off
Selection Start:
End
Length
Audio Position:
00 h 00 m 00.000 s
Stopped.
Click and drag to select audio, Ctrl-Click to scrub, Ctrl-Double-Click to scroll-scrub, Ctrl-drag to seek
Actual Rate: 48000

For detailed waveform editing, Audacity makes it easy to cut, paste, delete, and apply effects to digital audio tracks

your file before you record. Open the View menu and select Buses. We're going to make a copy of the Audio Master bus.

To do this, select it and then, in the Properties box, type in a new name to replace Master – Mono would be best. In the Mode pull-down, replace Duplex with Input and under Audio, reduce the number of Channels to 1. Then click the Create button.

Save your file once you've done this, as bus configuration is per-project. Adding a bus can sometimes interfere with JACK, so it's a good idea to set your bus up before you start work on your project. If you stop being able to hear MIDI or audio, you may have to shut down and restart Qtractor, Qsynth, and QjackCtl.

> “We're going to make a copy of the Audio Master bus”

With your bus configured, return to the main interface and press **SHIFT+INSERT** or use the right-click menu to create a track. Set its type as audio, its Input as Mono, and its Output as Master. You should be able to hear your instrument through your monitors.

Recording audio tracks works much like recording live MIDI input: press **R** on the track to arm it, move the Play-head transport to wherever you'd like to start, press the red Record button, press Play, and start strumming.

EXPORT MIDI TO AUDIO

Because Qtractor uses FluidSynth to handle its MIDI voices, you can't directly export MIDI tracks as PCM audio suitable for compression and distribution as an MP3.

However, once you're happy with the MIDI elements of your track, you can record them as PCM audio by creating a new audio track using the Master bus for both input and output. Press **F8** to open Qtractor's JACK Audio connection interface; expand the Qsynth outputs on the right, the Qtractor inputs on the left; and drag Qsynth left to Qtractor Master/in_1 and Qsynth right to Qtractor Master/in_2.

Return to Qtractor's main interface, arm the audio track by pressing **R**, rewind to the beginning of your piece, press the red Record button in the toolbar, and then press Play.

While it's a little circuitous, this method creates a perfect PCM audio rendition of one or more MIDI tracks, which you can then easily mix, apply effects to, publish, or export for further editing.

ANOTHER POINT OF VIEW

The default view for audio waveforms in Qtractor is rather cramped but, hidden in the lower right corner of the main workspace, you'll find a set of five magnifying glass icons. You can use these to zoom in on your waveform so you can cut and paste parts of the track, which you select by enclosing the section you want between the edit heads.

Qtractor's design is heavily informed by tracker-style sequencer software, which makes it well suited to building up a track from short loops and samples. However, if you have a particularly long recording that you want to clean up, the Audacity audio editor, which we installed earlier, provides a quick and easy range of dedicated tools for fast waveform editing.

To export digital audio tracks from Qtractor, select the track, go to Track > Export tracks > Audio and save it as a WAV file, which can be opened in Audacity.

Audacity uses ALSA audio by default, so don't forget to switch it to JACK in the Preferences settings or via the right-hand pull-down menu just above the main workspace.

Using Audacity, it's easy to cut away the parts you don't want: just select a section of your track by dragging the mouse and press **CTRL+X** to cut it, **CTRL+C** to copy it, or **DELETE** to remove it entirely. Cut or copied audio will be pasted in wherever you click or over any section of audio you select.

A powerful effects menu can radically or subtly alter your recording. When you're happy, you can use the File > Export menu to save your edited composition as a WAV to reimport into Qtractor for further mixing.

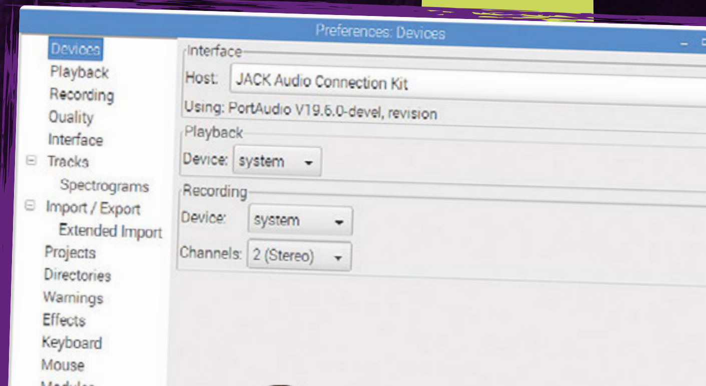
> “A powerful effects menu can radically or subtly alter your recording”

When you've finished working on a composition in Qtractor, you can use its Export Tracks feature to mix down every selected track in your workspace into a single WAV file, which you can again open in Audacity to export into a smaller, more listener-friendly format.

MUSIC PUBLISHING

It's important to hang on to a lossless, multitrack copy of your recordings, but for online distribution it's usual to compress files using either a lossy codec such as MP3 – which throws out parts of audio data likely to be inaudible to the human ear – or, for higher quality, lossless codecs like FLAC, which retains all your sound data but results in larger file sizes.

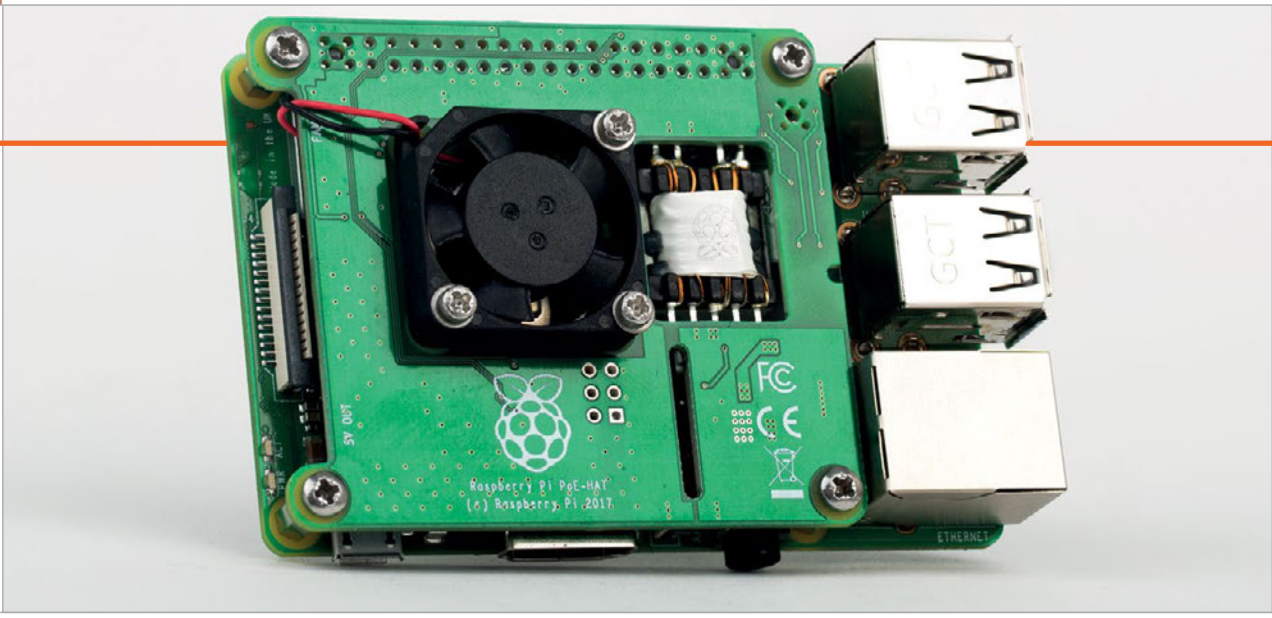
Whichever you choose, don't forget to add appropriate track name and credit metadata. You can publish and stream your music for free on **bandcamp.com** and **soundcloud.com**, while independent distribution services such as **cdbaby.com** are able to publish your music on platforms including Spotify and iTunes for a small fee.


Preferences: Devices

Devices
Playback
Recording
Quality
Interface
⊟ Tracks
Spectrograms
⊟ Import / Export
Extended Import
Projects
Directories
Warnings
Effects
Keyboard
Mouse
Modules

Interface
Host: JACK Audio Connection Kit
Using: PortAudio V19.6.0-devel, revision
Playback
Device: system
Recording
Device: system
Channels: 2 (Stereo)

Audacity also allows you generous save preferences, so audio details can come up on playback



Maker Says

Powers a Raspberry Pi 3 Model B+ via an Ethernet cable

Raspberry Pi

RASPBERRY PI POE HAT

Power your Raspberry Pi directly from a PoE-capable Ethernet socket

Power over Ethernet (PoE) has been a much requested Raspberry Pi feature from the community. The ability to access power and network along a single cable could prove tremendously useful in a range of deployments.

Until now, third-party solutions have been available, most of which add a second Ethernet socket via the HAT and then pass power on to the USB power port.

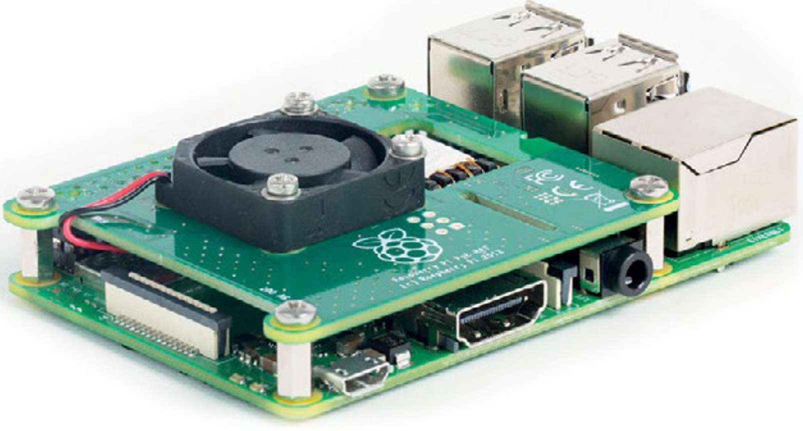
However, the Raspberry Pi 3B+ added four extra GPIO pins specifically to support PoE options and subsequently announced this PoE HAT, a Class 2 device for drawing power directly from a network.

This is a more elegant solution that draws its power directly from the Ethernet connection on the Raspberry Pi 3B+ and transfers the power via the four new GPIO pins on the board, while simultaneously enabling a direct network connection.

The PoE HAT board uses a custom-built isolation transformer (this is the small white circuit) to turn the 37–57 V DC from Ethernet to a Raspberry Pi-acceptable 5 V.

The isolation transformer doesn't need much space, and the designers have decided to double up with another oft-requested feature: a fan.

All of this makes the PoE HAT an appealing addition to the Raspberry Pi, especially for industrial users. Being able to draw power from a network makes the Raspberry Pi easily deployable away from electrical sockets. And the presence of a fan provides a layer of security when using a Raspberry Pi long-term in a hot environment.



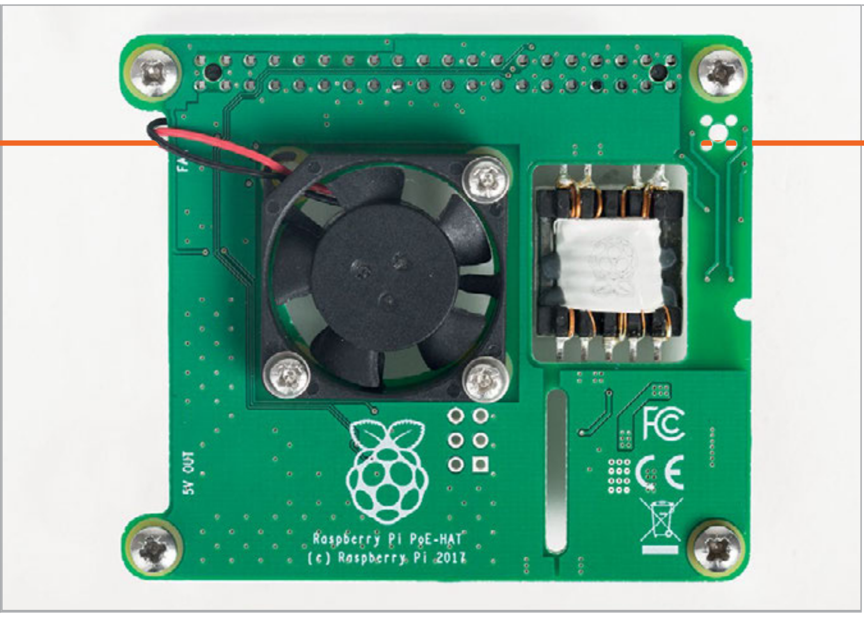
Related

PI POE SWITCH HAT

Features an Ethernet socket on the board and a pass-through cable. The PoE Switch HAT is more complex, expensive, and it lacks the fan. However, it is also compatible with older Raspberry Pi Model B-style devices.

£30 / $40

magpi.cc/yAcjMp



magpi.cc/Gd0kuq

£18 / $20

Because the 3B+ is the only Raspberry Pi to feature these additional PoE pins, you must use this HAT with a Raspberry Pi 3B+ (no other model is currently supported).

Setting up

Connecting the Raspberry Pi PoE HAT to the Raspberry Pi 3B+ is a simple 30-second job. The HAT connects directly to the 40-pin GPIO header (and four-pin PoE header) on the 3B+ board.

Four standoffs and eight screws are used to keep it in place. Then it's simply a case of connecting a PoE-enabled Ethernet cable to the Raspberry Pi board. It then boots up without the requirement of the USB power cable. A slot for the Pi Camera Module enables you to feed its cable through.

It is possible to add a riser to the HAT to extend the GPIO pins out, in case you wanted to add a second HAT or access the GPIO on the Raspberry Pi. The HAT uses I²C to interact with the Raspberry Pi.

At this point, we must stress the requirement that you have a powered Ethernet connection. Most household routers provided by the likes of Virgin and BT do not supply power over the Ethernet connection. Neither do many power-line adapters. Check that you have your PoE system in place before setting up the PoE HAT.

For this test, we used a TP Link Gigabit PoE Injector (**magpi.cc/LkGEMO**) connected to a Virgin Media HUB 3.0 broadband router (**magpi.cc/kDKBac**) The engineering team used a NETGEAR GS108PEv3 8-Port Gigabit PoE Smart Managed Plus Switch for testing (**magpi.cc/LIaBWo**).

> An appealing addition to the Raspberry Pi, especially for industrial users

Once assembled, the Raspberry Pi 3B+ and PoE HAT would not fit inside the Raspberry Pi official case. It may well be that a case option is available down the line, but the product brief says 'This product should be operated in a well-ventilated environment and, if used inside a case, the case should not be covered.'

Testing the fan

The PoE HAT conforms to the 802.3af PoE standard (**magpi.cc/cDxYYm**) and on the board is an isolated switched-mode power supply. It's a 37–57 V DC, Class 2 device that provides the 5 V/2.5 A DC output used by a Raspberry Pi.

In terms of testing, we powered it up and used it without a USB connection plugged in.

In terms of networking, the Ethernet connection remains a Gigabit connection running via the USB 2.0 sockets and thus limited to 200Mbps. We found no network difference using the PoE HAT.

A more interesting test is the fan, which in our testing knocked roughly 10 °C off the CPU temperature.

The Raspberry Pi without a fan idled at 40.8 °C; with the fan it idled at 32.2 °C. Without a fan and playing high-definition YouTube video, it ran at 60.1 °C; with a fan it dropped to 42.9 °C.

We even ran a Raspberry Pi with all four cores stressed to 100% for half an hour and the temperature peaked at 53.7 °C (we don't recommend doing this on your own Raspberry Pi incidentally, so we're not disclosing the test – email us if you want to know how).

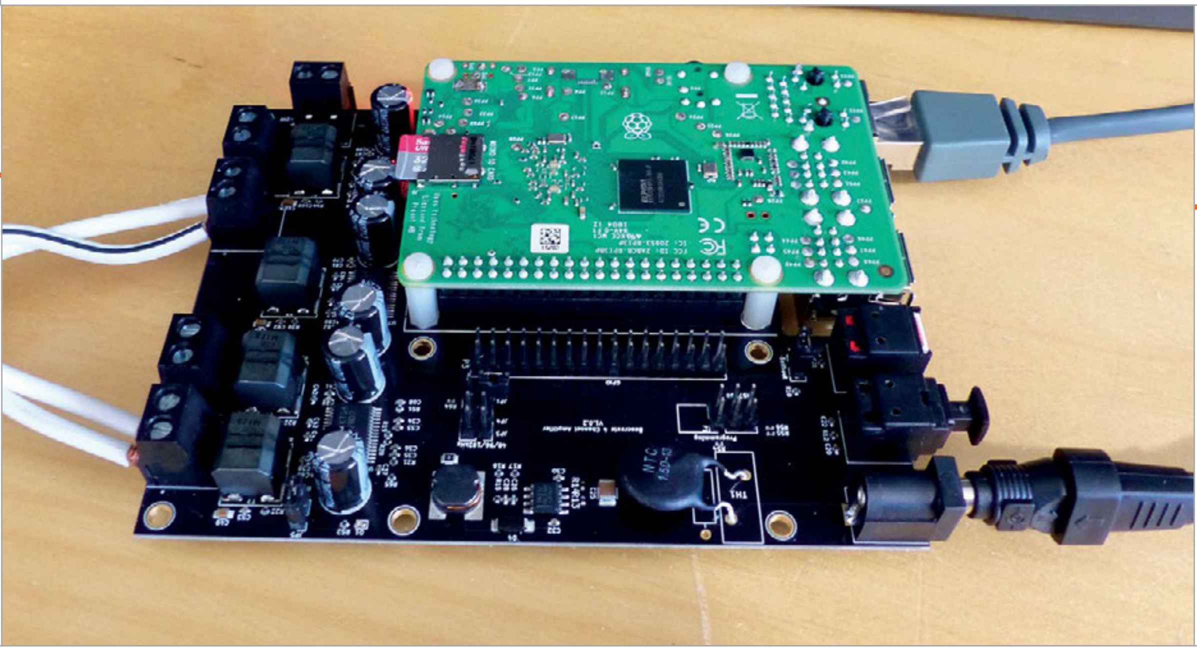
Neither the HAT, the powered Ethernet cable, nor the isolated switched-mode power supply generate any additional heat, so the fan isn't a required component for PoE Ethernet support, but it will be a good addition for users in high-temperature environments.

This is ideal for folks who use the Pi intensely and worry about the long-term effects of stressing the CPU without any form of cooling.

Last word

A fantastic solution to a long-requested feature. The PoE HAT enables the Raspberry Pi to draw power directly from a powered Ethernet connection. The bonus inclusion of a fan enables you to run a Raspberry Pi for long periods in hot environments with confidence.

★★★★★



Maker Says

“We are putting a new lease of life into vintage B&O speakers”

HiFiBerry

BEOCREATE 4-CHANNEL AMPLIFIER

Upcycle vintage speakers into the digital age with this audiophile-grade HAT amplifier from Bang & Olufsen

Somewhere, in the darkest recesses of your garage or loft, there may be speakers. Not speakers like today’s tiny, tinny Bluetooth excuses, but proper speakers: large, heavy, loud. Although these monoliths are still capable of outperforming today’s examples, they are becoming increasingly useless. Why? Well, unless you’re holding on to your vintage amplifiers, there’s nothing left to drive them. So your wonderful, expensive pieces of engineering excellence sit gathering dust, playing only John Cage’s 4′33″. Until now.

Audiophile giant Bang & Olufsen has partnered with HiFiBerry to produce the Beocreate, a high-end digital sound processor and amplifier for the Raspberry Pi. Not so much a HAT as a full fedora, scarf, and cape, this monster upgrades your favourite tiny computer to audiophile quality. Four channels (a whopping 2×60 W and 2×30 W) can be configured to drive not only standalone speakers, but also their individual woofers and tweeters. The on-board DSP can be endlessly configured over the web, even in real-time as you’re listening, allowing for an incredible amount of tinkering that will have even the most dedicated perfectionist thinking ‘Oh, that’ll do’.

> “The sound quality is enough to get the credit card twitching”

The Beocreate also upcycles your classic speakers to work with 21st century music sources. Add a Pi, hook up the wires, install some software, and you’ve got a Bluetooth, Spotify, and AirPlay speaker like no other. B&O has targeted its classic CX50 and CX100 speakers, providing built-in profiles for the DSP to get the absolute best from them and guides to install everything inside the speakers. The good news is it also works with any passive speaker, providing a default ‘safe’ profile that you can then tweak if you see fit.

Get what you pay for

For a rather breathtaking £149, you do get very good build quality. On board are a power input (which also powers your Pi), four speaker

Related

PI-DIGIAMP+

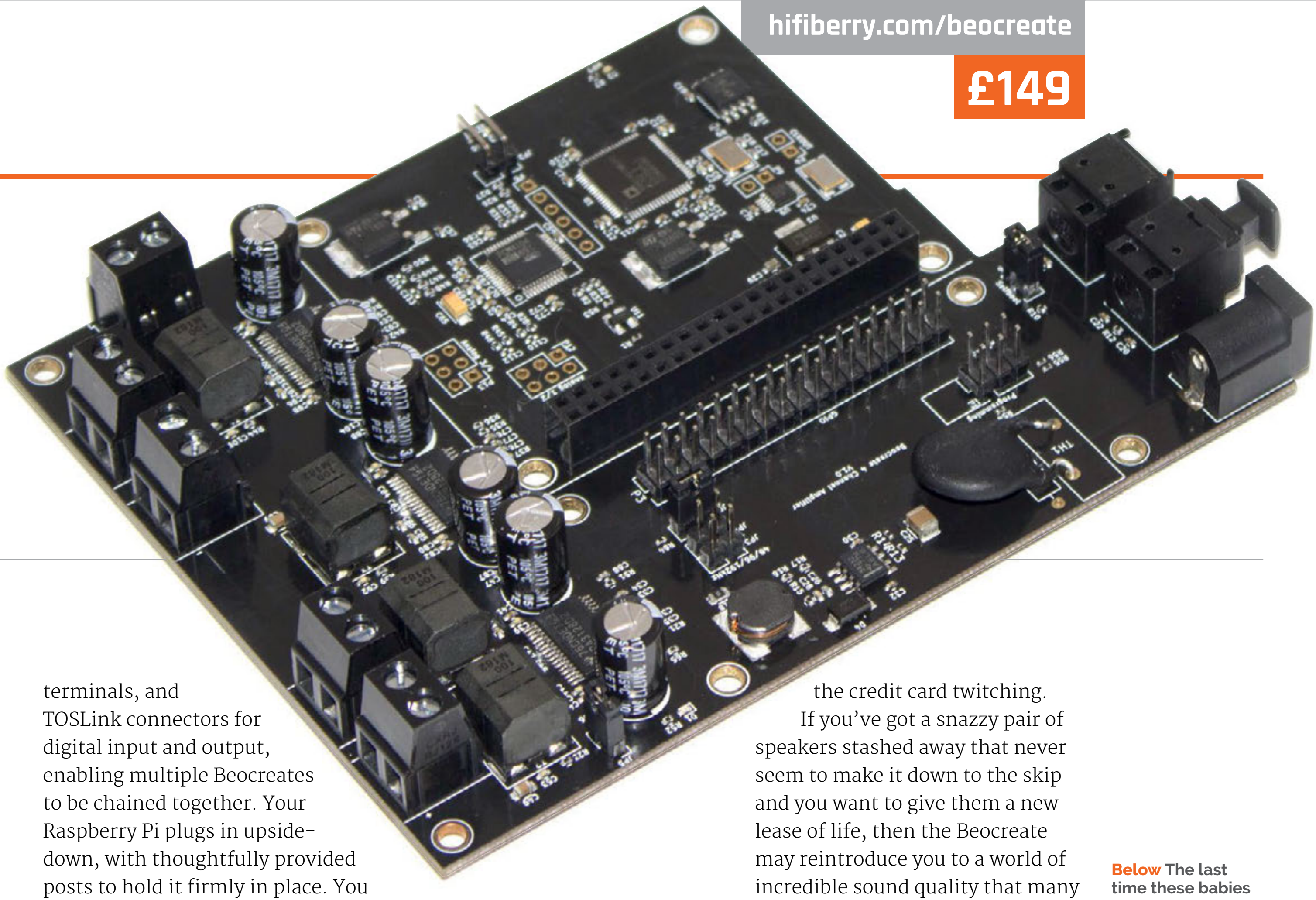
This HAT provides no-frills good audio quality. A DAC and 2×35W amplifier on a single board. Perfect for more entry-level projects.

£56

magpi.cc/MWwKNe

hifiberry.com/beocreate

£149



terminals, and TOSLink connectors for digital input and output, enabling multiple Beocreates to be chained together. Your Raspberry Pi plugs in upside-down, with thoughtfully provided posts to hold it firmly in place. You can then power both boards from a single power supply (not supplied).

All that remains is to wire in your speakers and load up some software, a choice between a read-to-go image for 'it just works' or a more customisable Raspbian build with drivers and configuration already in place. Setup is a breeze despite confusing documentation. You can set up your WiFi connection and add sources such as AirPlay and Spotify Connect within a few minutes.

Once your double-decker-bus-scale speakers are down from the loft, you may struggle to work out where to connect them. There is a distinct lack of labelling and we restored to trial-and-error. Once connected, the result was pretty astonishing. Some of us of a, ahem, 'middle-aged' persuasion have forgotten the golden age of speakers as convenience has compromised audio quality. The depth and range of sound was wonderful. The point of this became obvious.

Digging deeper into the capabilities of this device, it supports the creation of DSP 'profiles' using the provided SigmaStudio software. You can play endlessly with crossover frequencies and filters, create a setup you like, and then store it on the Beocreate so anything playing on the Pi uses it. This is a level of control that can only have been imagined just a few years ago.

Audio excellence

This is an unapologetically audiophile product – and in this case, that means expensive yet still value for money. If you just want to convert an old average speaker, there are cheaper options available. The documentation is described as in 'beta' and we couldn't agree more. Although installation was easy, it was not obvious where to start, causing some initial frustration. That said, the sound quality is enough to get the credit card twitching.

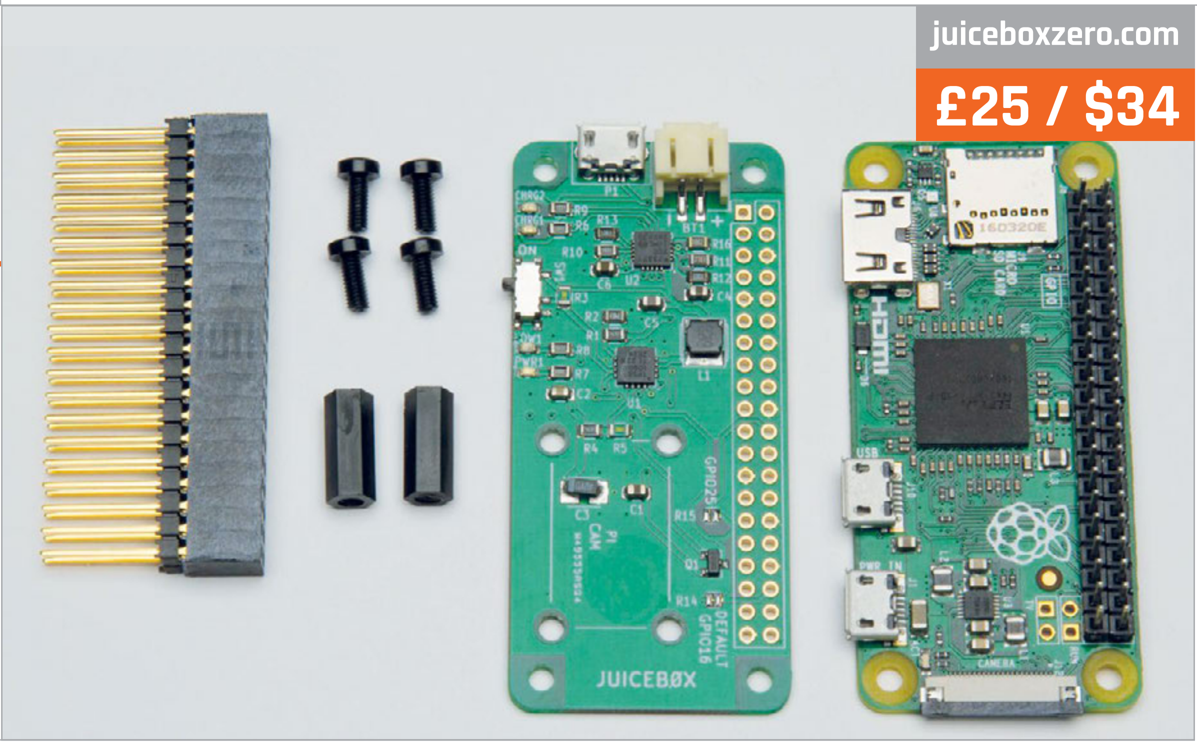
If you've got a snazzy pair of speakers stashed away that never seem to make it down to the skip and you want to give them a new lease of life, then the Beocreate may reintroduce you to a world of incredible sound quality that many thought had been lost in portable speakers, earbuds, and laptops.

Below The last time these babies made sound, there were still three living Beatles

Last word

An audiophile's dream, the Beocreate is the missing link between the worlds of classic high-end speakers and modern streaming technology. As documentation improves, this will be solid choice for bringing your digital music back to life.

★★★★☆



juiceboxzero.com

£25 / $34

Maker Says

"The simplest way to properly power your Pi Zero with a battery"

JuiceBox Zero

JUICEBOX ZERO

Portable power for your Pi Zero. **Phil King** recharges his battery

Billed as the only battery management board in this form factor that requires no code, JuiceBox Zero can be used as a plug-and-play device to provide portable power to your Raspberry Pi Zero. Well, it's not quite plug-and-play, since you first need to solder on a stacking GPIO header (not supplied) and also provide your own battery.

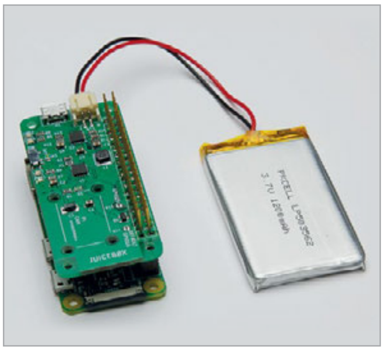
It's compatible with any 3.7 V or 4.2 V single-cell Li-ion or LiPo battery with a two-pole JST PH 2 mm connector, 1 A current output, and 1 A charge rate. Note: while we're informed that multi-cell batteries may work, they should only be used if you know for sure they have a reliable cell-levelling mechanism – if in any doubt, don't risk it!

Put it together

To mount the JuiceBox Zero on your Pi Zero's GPIO pins, you'll need to add your own stacking header: the online instructions (**magpi.cc/LjMmLh**) advise mounting this on the Pi Zero first, adding a couple of stand-offs (not supplied) to secure the JuiceBox Zero on top, then soldering the header pins. Using a stacking header will enable you to add a pHAT/HAT on top. There shouldn't be any pin conflict issues since the JuiceBox Zero only uses power, ground, and GPIO 16 (can be changed to GPIO 25).

That GPIO pin is used to trigger an optional automatic shutdown (after you add a single-line cron job) if the battery voltage falls below 3.2 V, at which point a low-battery red LED is also lit.

As well as providing power to the Pi, the JuiceBox Zero doubles as a battery charger. Just plug a standard 2 A or greater supply into its micro USB port; there are helpful LEDs to indicate when the battery is charging and fully charged. A nice bonus is the inclusion of a handy on/off slider switch and mounting holes for a Pi Camera Module.



Related

PIJUICE

If you need more sophisticated power management options, PiJuice offers plenty and it comes supplied with a swap-outable 1820 mAh battery.

£48

magpi.cc/v0lyQP

Last word

Unlike the rival PiJuice, there are no extensive software-based power management options on offer, but the JuiceBox Zero does its job well, providing portable power and doubling as a battery charger. When plugged into the mains, it can also be used as an uninterruptible power supply (UPS).

★★★★☆

CoderDojo

Can I start a CoderDojo club in my local area?

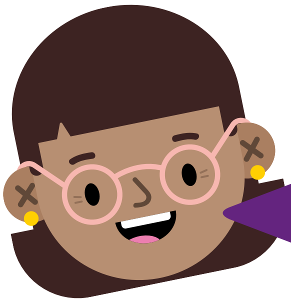
CoderDojo is a global network of free, volunteer-led, project-based programming clubs for children aged 7–17. Dojos are championed by individuals all around the world who are passionate about giving young people the opportunity to learn to code.

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

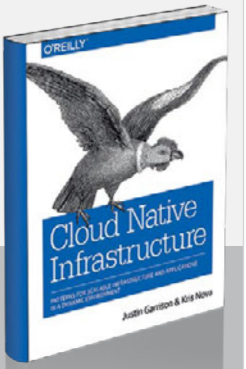
The CoderDojo Foundation is Irish Registered Charity (524255) CHY20812
and is part of the Raspberry Pi Foundation, UK registered charity 1129409

Part of Raspberry Pi

RASPBERRY PI BESTSELLERS DEVOPS

Serverless, cloud, containers – whatever you need to admin, O'Reilly probably has a concise intro guide for it.

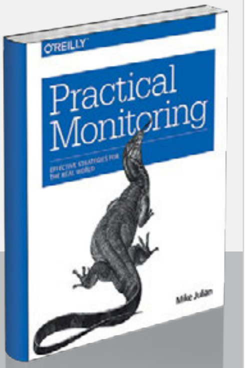
CLOUD NATIVE INFRASTRUCTURE



Authors: Justin Garrison & Kris Nova
Publisher: O'Reilly
Price: £31.99
ISBN: 978-1491984307
magpi.cc/LkIKxn

Infrastructure impacts business; get the overview of cloud native infrastructure, and how it can support a scalable business. From patterns to predictions, a high-level but technical view of design and best practice in the cloud.

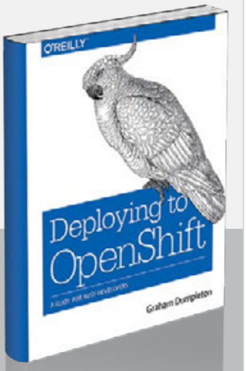
PRACTICAL MONITORING



Author: Mike Julian
Publisher: O'Reilly
Price: £21.50
ISBN: 978-1491957356
magpi.cc/LHbBFd

A near perfectly balanced overview of designing and implementing monitoring from your app down to infrastructure. Julian's neutral approach gives you practical knowledge that will work with your chosen tools, or get you to ask the right questions on tool choice.

DEPLOYING TO OPENSHIFT



Author: Graham Dumpleton
Publisher: O'Reilly
Price: £31.99
ISBN: 978-1491957165
magpi.cc/YuLldP

Containers, Security-Enhanced Linux, and Kubernetes are all brought together in OpenShift. This quick reference guide to Red Hat's container deployment and management platform will show you the magic that makes OpenShift so popular.

GET PROGRAMMING WITH HASKELL

Author: Will Kurt
Publisher: Manning
Price: £24.32
ISBN: 978-1617293764
magpi.cc/KULNCC

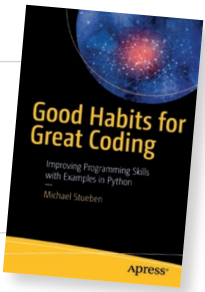
You know you want to learn Haskell, admit it. Maybe you've just done a bit of Python, but are intrigued by what you hear about functional programming. Maybe you code in a dozen other languages, but never got past the elitist talk of Applicative Functors and Monoids, and keep putting off a look at the language whose advanced features have been creeping into every other programming language for the last couple of decades. Perhaps the upfront load of theory learning in Haskell books prevents you really grokking it.

Worry no more, Kurt has written this book especially for you. You learn by doing, and absorb theory and abstract concepts only once you have practical examples to build the learning upon. This is what *Get Programming with Haskell* gives you.

Some choices may be questioned, such as using Haskell Platform (Stack is introduced later), but it's a sensible pragmatic choice for a book also aimed at non-UNIX users. Short lessons filled with exercises and interactive code snippets to run in the REPL keep you moving forward, and are organised into units which each finish with a project. An outstanding introduction to Haskell and functional programming for real-world use.

Score ★★★★★

GOOD HABITS FOR GREAT CODING



Author: Michael Stueben
Publisher: Apress
Price: £19.99
ISBN: 978-1484234587
magpi.cc/tFQWhq

Learning good programming style is not easy, but the quickest – and possibly only – way is to take on coding challenges, reflect on experience, and remember the lessons learned. So maintains Michael Stueben, who brings four decades of coding and teaching experience to this excellent but idiosyncratic guide.

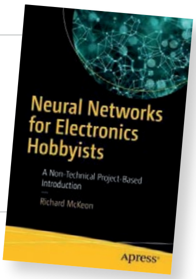
During a quick intro detailing what makes Python great (Python is used throughout this book, but it aims to teach good style to new coders using similar languages), Stueben also introduces refactoring and optimisation before the reader has even noticed starting, and dives into an interview code question. An allegorical diversion brings in unit testing, code reviews, and documentation, before delving into pain management for new programmers. There are many good practices that eventually experience can teach you – but you'll save a lot of pain by starting them early!

The writer holds strong opinions – e.g. scepticism of OOP – but an opinionated guide is just what's needed until the learner can justifiably argue from experience against their mentor's style. A number of rather simple errors that should have been caught in editing occasionally bring the reader up short, but overall this is a useful and enjoyable book which will considerably boost any beginner-to-intermediate Pythonista's skills.

Score ★★★★☆

NEURAL NETWORKS FOR ELECTRONICS HOBBYISTS

Author: Richard McKeon
Publisher: Apress
Price: £14.99
ISBN: 978-1484235065
magpi.cc/ezQzZJ



With the take-off of deep learning, there is no shortage of resources to teach you about neural networks – often modelled with Python. It's a strange concept to learn, as we're starting with an answer, then looking to train a network with cycles of back propagation until that answer can be reached: mimicking the way the brain works. Something that can perhaps be more clearly shown by a direct analogue, rather than matrix multiplication in code.

McKeon takes that analogue approach, and shows how to construct a modest but functioning neural network from simple electronic components on a breadboard. Although primarily aimed at makers and electronic hobbyists who want to understand neural networks, there's enough info provided for electronics newcomers to be able to build the circuit.

The project that the network is being trained on is to build an XOR logic gate. This is accomplished with careful training, well illustrating the neural network concept. Voltmeters and adjusting potentiometers make the process so much more hands-on than software, and this would be a great school project. Additional functions are tried at the end of the book, and the possibilities of a simple three-layer network leave the reader hungry for more learning in deep learning.

Score ★★★★☆

FLASK WEB DEVELOPMENT

Author: Miguel Grinberg
Publisher: O'Reilly
Price: £35.99
ISBN: 978-1491991732
magpi.cc/WIuELF

Flask is the web framework that doesn't get in your way. Its simple, extensible design makes a great foundation for any web-based project, but also makes it hard to get a grip on just what the programmer should do to get the best from it. You should certainly start with Flask before Django if you're considering your first Python web project, as you'll be coding closer to the action and learning more as you go.

Grinberg does a great job in a few short chapters of giving you the basic nuts and bolts of Flask, from templates – separating business and presentation logic – through database connections, to the culmination of part I in the organisation of large projects (which in turn is an example of the book's concern with good software engineering).

The real learning happens as you work through the example in part II, a social blogging application. Having one thoroughly worked-through project enables the reader to see how it all fits together as she works through authentication, roles, social relationships and database joins, content posting, etc. The final section covering testing, performance analysis, and deployment is full of practical advice. This is essential reading in your Python journey.

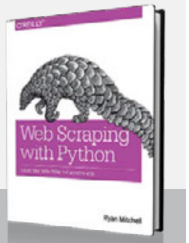
Score ★★★★☆

ESSENTIAL READING: UPSKILLING ESSENTIALS

Use the quiet days of summer to improve an area of your programming skills – from ML to concurrency

Web Scraping with Python

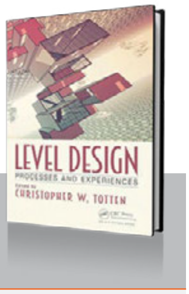
Author: Ryan Mitchell
Publisher: O'Reilly
Price: £31.99
ISBN: 978-1491985571
magpi.cc/alhfCh



Until every source of internet data has a well-crafted API, web scraping will be an essential need – and this an essential guide.

Level Design

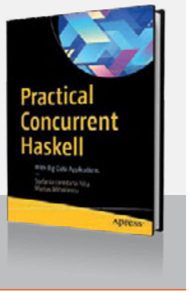
Author: Christopher W Totten
Publisher: CRC
Price: £40.99
ISBN: 978-1498745055
magpi.cc/HAaDkY



Fascinating collection of essays from games developers and academics that could bring gamification in your IoT app to the next level!

Practical Concurrent Haskell

Authors: Stefania Loredana Nita & Marius Mihailescu
Publisher: Apress
Price: £35.99
ISBN: 978-1484227800
magpi.cc/pKBuLX



Dive into Haskell and concurrency, and you'll be equipped for big data from your Pi cluster on up to the cloud.

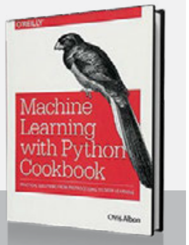
Modeling and Simulation of Everyday Things

Author: Michael W Roth
Publisher: CRC
Price: £53.99
ISBN: 978-1439869376
magpi.cc/JLkklW

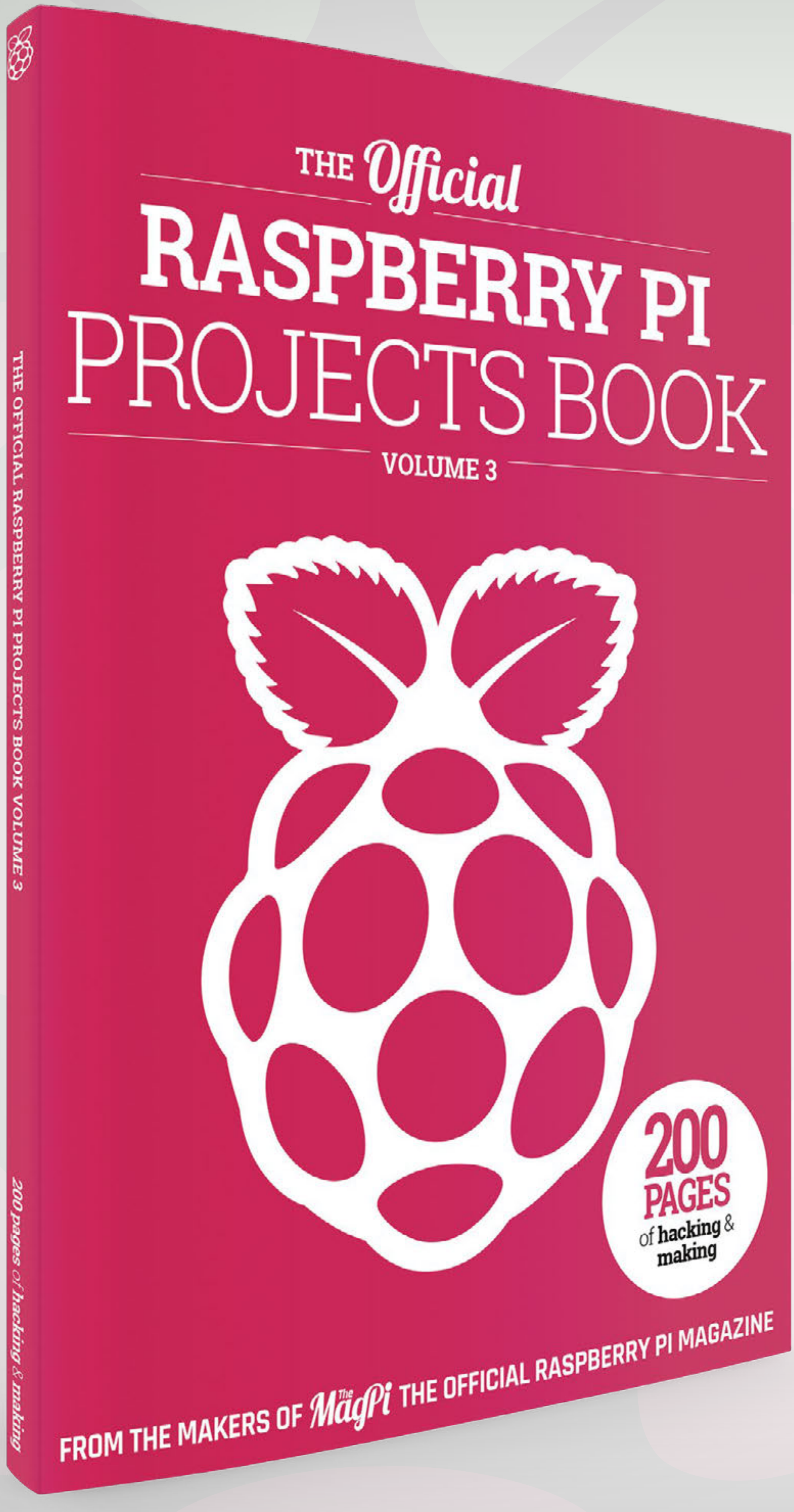
Bringing together maths, C++, and visualisation in an occasionally playful introduction to usefully modelling the real world on your computer.

Machine Learning with Python Cookbook

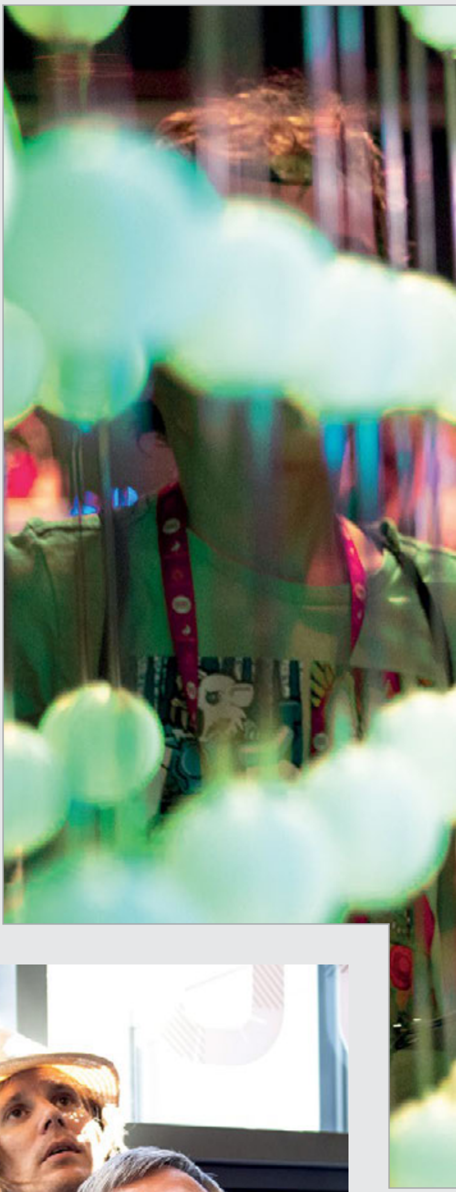
Author: Chris Albon
Publisher: O'Reilly
Price: £47.99
ISBN: 978-1491989388
magpi.cc/OjtfYy



Take your Pandas and Scikit-learn knowledge to real ML challenges with a cookbook that combines tutorial with reference qualities.


THE Official
RASPBERRY PI
PROJECTS BOOK
VOLUME 3
200 PAGES of hacking & making
FROM THE MAKERS OF The MagPi THE OFFICIAL RASPBERRY PI MAGAZINE
THE OFFICIAL RASPBERRY PI PROJECTS BOOK VOLUME 3
200 pages of hacking & making

FUN AT THE FIELDS



A picture tour of the amazing Raspberry Fields event!

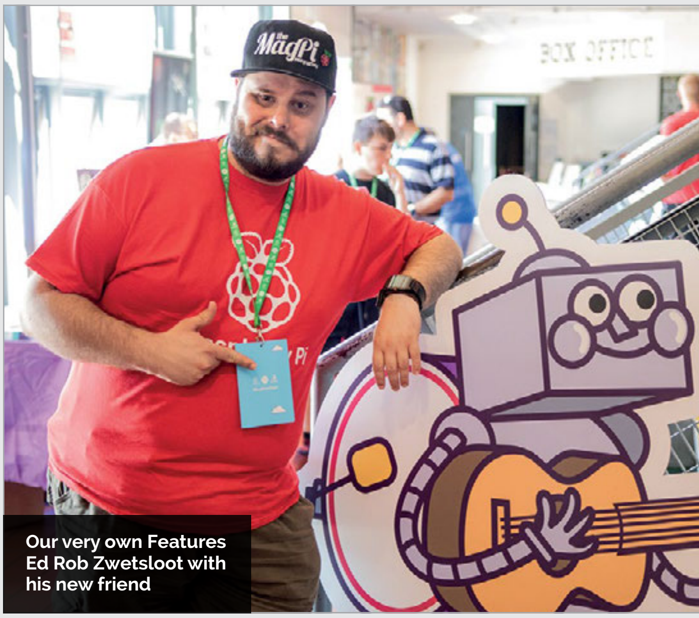
The first Raspberry Fields event occurred at the end of June, and we were there in the thick of it during the hot Cambridge weekend. Hundreds of people attended to see some amazing talks, shows, and demonstrations. If you missed the event, here are a load of excellent photos that should help make you feel like you were actually there...

Demos of amazing Pi projects were fun for all ages and types

Look, up in the sky! Is it a bird? A plane? No, it's the air-con

Cubert 2.0 was a huge draw for the crowds

Lots of Raspberry Pi folk helped out over the weekend



Our very own Features Ed Rob Zwetsloot with his new friend

Parents and teachers got to talk to Foundation representatives

Below Performances during the weekend ranged from interactive light shows to explosion-laden science!

Everyone loved the Pi robots being shown off

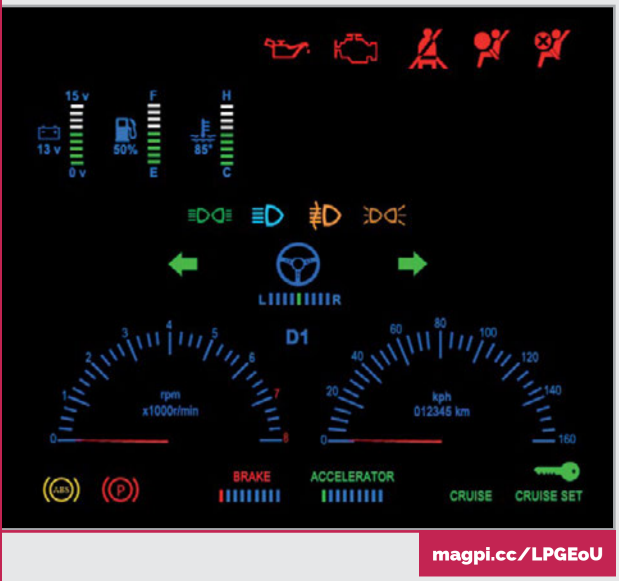
CROWDFUND THIS!

Raspberry Pi projects you can crowdfund this month

kck.st/2tE7mip

OFF-GRID MOBILE MESH NETWORK

These portable network routers allow you to create a mesh network while outdoors, or even within your own home. They run on Raspberry Pi boards and can also be very useful for using Raspberry Pi devices, especially when outdoors.



magpi.cc/LPGEoU

UCAN, A UNIVERSAL VEHICLE BUS DECODER

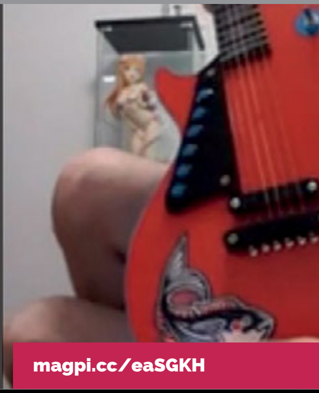
Want to see your in-car telemetry? The idea of this crowdfunding campaign is to finance decoding of the necessary data to a UCAN ecosystem. It runs on a Raspberry Pi, so you'll be able to display data that your (modern) car has but doesn't usually show.

BEST OF THE REST

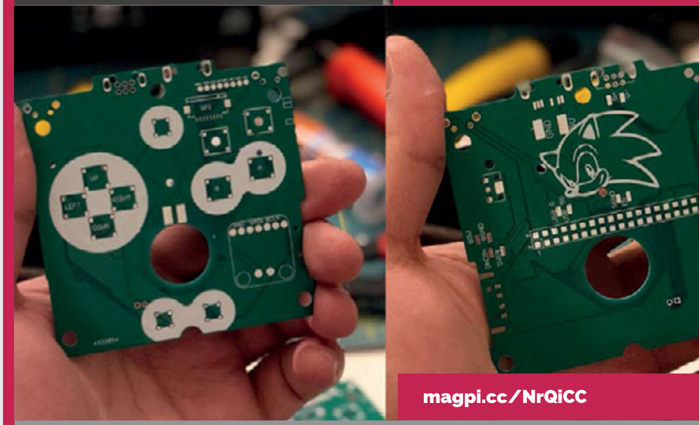
Here are some other great things we saw this month

RASPBERRY PUITAR

A custom Raspberry Pi-powered guitar, creating most excellent digital music through tapping (pressing the strings to create a note, as opposed to holding a chord and strumming). The design is great as well, with the Raspberry Pi visible through clear acrylic on the rear of the guitar.



magpi.cc/eaSGKH



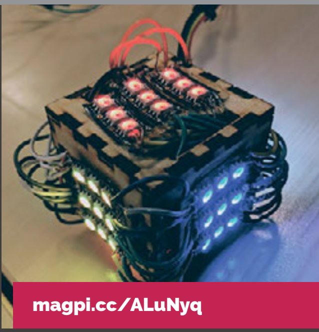
magpi.cc/NrQiCC

CUSTOM GBA SP

Game Boy and retro console hacks are very popular, usually gutting a machine and completely replacing the electronics. This build is different, though – using a custom PCB, Reddit user abhg2 has managed to hack a humble GBA SP to be able to connect a Raspberry Pi Zero while still using much of the same old hardware.

PI-POWERED RUBIK'S CUBE

This custom Rubik's cube may not quite look like the original puzzle, but it works just like it, albeit using a controller rather than your hands to physically spin it. We wonder how much this will throw Rubik's cube pros? Would their muscle-memory betray them? Either way, it looks pretty cool.



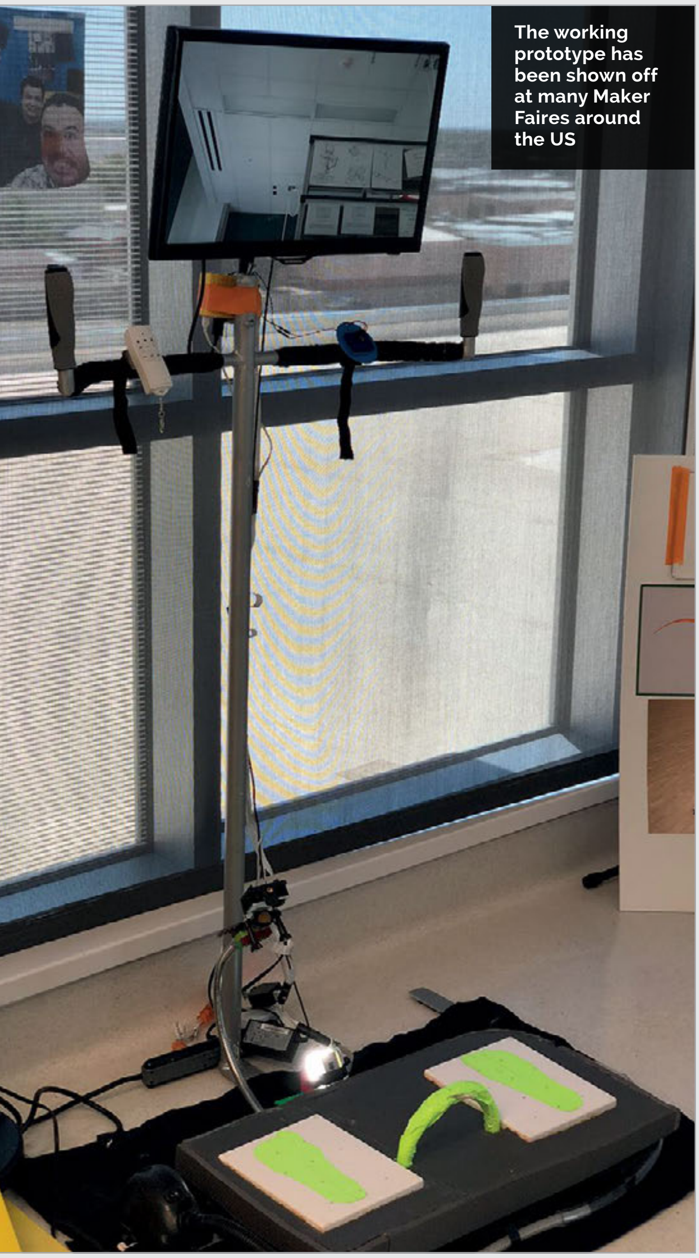
magpi.cc/ALuNyq

Ernesto Holguin

Occupation: Nurse

DIABETIC FOOT HEALTH WITH RASPBERRY PI

The Raspberry Pi takes another dip into medical computing



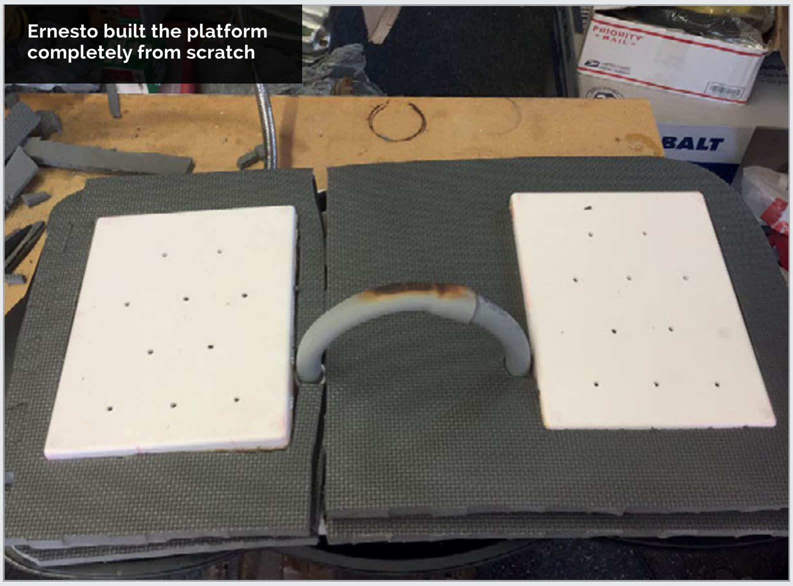
The working prototype has been shown off at many Maker Faires around the US

While we've seen the Raspberry Pi used in medical applications before, it's still quite rare. We met Ernesto Holguin from El Paso in Texas at the Htouston Maker Faire, showing off his fantastic new project, and decided we had to talk to him about how he too is improving medical technology with the humble Pi.

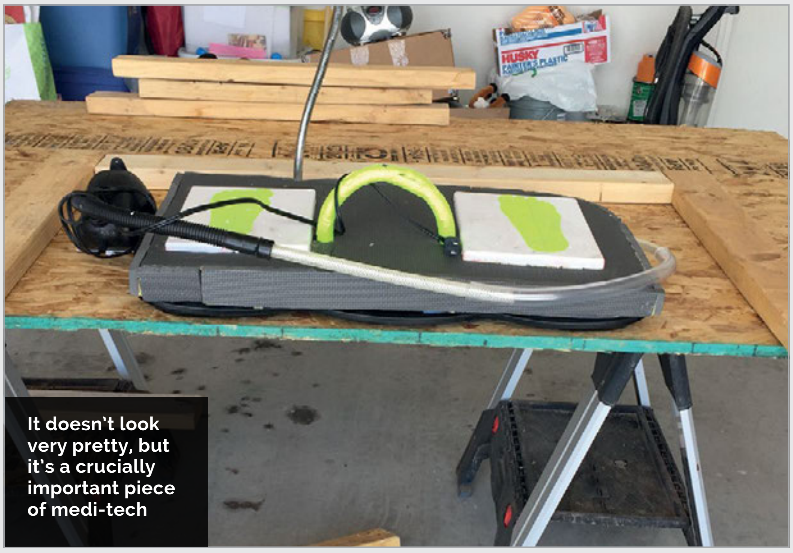
What is it you've made with a Raspberry Pi?

I invented a device that is dedicated to the telemonitoring and drying of diabetic foot ulcers.The patented apparatus – US Patent #9775474: **magpi.cc/vhseyj** – utilises the Raspberry Pi 3 to inspect, dry, take images, and communicate other vital sign information with clinicians and family members. It uses a Python-programmed application, forming a team approach to that person's care.

One in five diabetic patients visit a clinic/hospital for treatment of an ulcerative wound. From the time of discharge till the next clinician's visit, 30+ days on average, a patient's affected skin is not monitored, resulting in worsening ulcers/wounds, leading to amputations and deaths. Worsening of wounds primarily occurs due to the improper monitoring and drying in-between the toes. The diabetic



Ernesto built the platform completely from scratch



It doesn't look very pretty, but it's a crucially important piece of medi-tech

population suffers from high risk for infection, poor eyesight, and limited mobility. Limited family support and transportation adds to improper healing. My device would allow for patients to daily inspect their own feet … with the clinician overseeing the progress, from the comfort of the patient's home. Earlier medical intervention will be established, leading to better patient outcomes.

> I have five more projects lined up where I will be utilising the Raspberry Pi

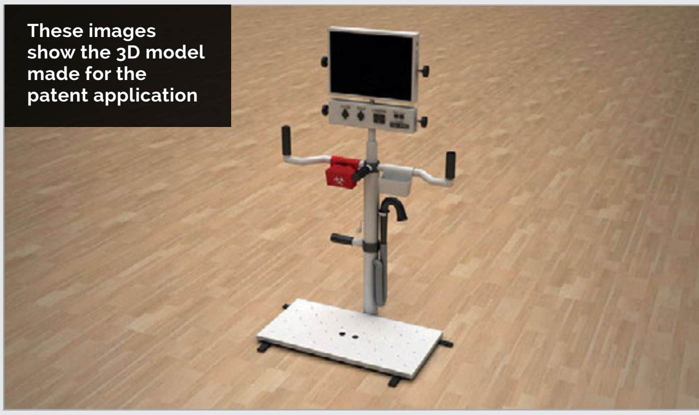
Why the Raspberry Pi?
The Raspberry Pi computer is small, fast, efficient, and is priced at an optimal value. It is pliable, allowing other applications to be easily added. I am a proud member of MakerNurse Community from MIT and they were instrumental in applying the Raspberry Pi with Python coding to my diabetic device.

How well does it perform?
The Raspberry Pi has worked consistently well while I have presented my device in Maker Faire New York 2017, Maker Faire Houston 2017, University of Texas in El Paso (UTEP) SIAM Symposium 2017, SXSW 2018, UTEP's System Engineering Symposium 2018, and Mini-Maker Faire Galveston 2018. It has gone through multiple airports, mail deliveries, and vigorous car trips. The Raspberry Pi continues to perform true to its original intentions.

How else do you think the Raspberry Pi can be used in medical tech?
The Raspberry Pi can be utilised in any upcoming medical device inventions in which the reliability of the system will be instrumental. The low-priced Pi will reduce the cost of production and make it more affordable to consumers.

Any future project plans using a Pi?
Yes! I have five more projects lined up where I will be utilising the Raspberry Pi. They're projects that will be used to improve the monitoring of patients and water conservation.

Technology is always advancing and making it more affordable to start projects people only dreamed of in the past. The Raspberry Pi is leading this new wave and has catapulted the idea I had in 2003 to a reality.



These images show the 3D model made for the patent application

AWARD WINNING

Ernesto's machine isn't just a neat thing he shows off at Maker Faires – the benefits of it have been well acknowledged with plenty of awards to back it up:

- **Infy Maker Award winner – June 2017**
- **MCA Proof-Of-Concept winner – July 2017**
- **Houston's Maker Faire: Merit Award winner – October 2017**

He's also been featured in local and national news outlets for his efforts. Find out more about the device at his website: **otenmedical.com**.

COMMUNITY PROFILE

SWAY GRANTHAM

How one primary school teacher's love of ICT made her legendary in the Raspberry Pi community

Sway

Category: Educator

Day job: Teacher

Website: swaygrantham.co.uk

twitter.com/swaygrantham

By day, Sway Grantham is a primary school teacher and Specialist Leader in Education for Primary Computing and ICT in Milton Keynes.

Outside of her day job, Sway has gone above-and-beyond to become one of the biggest names in the Raspberry Pi community.

She was invited as a 'lead learner' to attend the first ever Raspberry Picademy, becoming a Raspberry Pi Certified Educator; Sway organises TeachMeets in Milton Keynes and Northampton. She also hosts a Raspberry Pi networking group and supports a local CamJam.

"I decided to be a teacher when I was about eight," says Sway. At least that was according to her mum ("yes, I am that cool," she tells us).

"However, I didn't want to narrow my options that would keep me in teaching forever, so I did a Psychology and Computing degree at the University of Kent."

Sway was the only person in her entire university to do a Psychology and Computing combination, but it meant she could focus on developmental and educational psychology as well as her interest in computing.

Illustrious company

After becoming a primary school teacher, she became a Member of the Raspberry Pi Foundation. "I've been a Member for a couple of years, and it was a surprise to be asked!

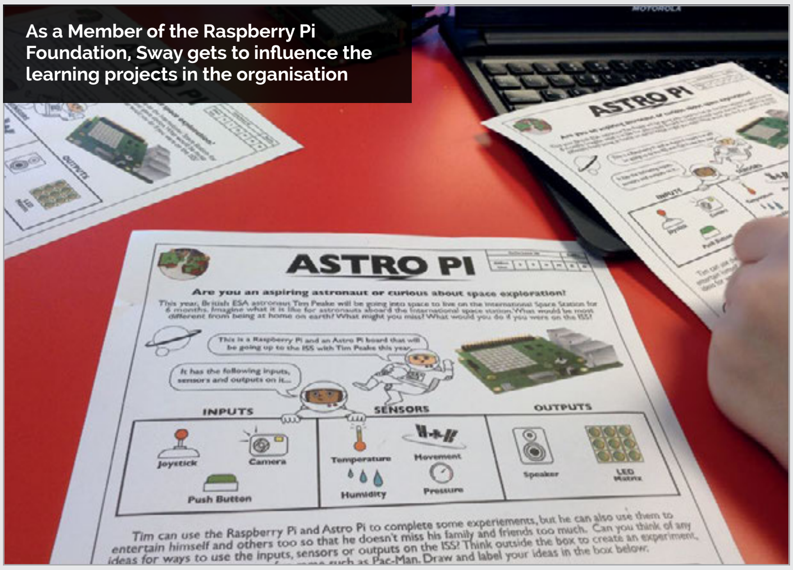
"I do suffer a little from imposter syndrome when you see the other members (cough…Tim

PI WARS FRANK

Sway takes part in school events, such as Pi Wars (where Giffard Park Primary School qualified to take part). "I had such a fantastic day at Pi Wars," says Sway. "The children I took were amazing. Here is their robot called Frank, short for Frankenstein."

Right Sway's students have made imaginative projects. But this one may turn out to be dangerous

We can assure you this is part of a lesson, not an order to the students



As a Member of the Raspberry Pi Foundation, Sway gets to influence the learning projects in the organisation

ASTRO PI

Are you an aspiring astronaut or curious about space exploration!

This year, British ESA astronaut Tim Peake will be going into space to live on the International Space Station for 6 months. Imagine what it is like for astronauts aboard the International space station. What would be most different from being at home on earth? What might you miss? What would you do if you were on the ISS?

This is a Raspberry Pi and an Astro Pi board that will be going up to the ISS with Tim Peake this year.

It has the following inputs, sensors and outputs on it...

Tim can use the Raspberry Pi and Astro Pi to complete some experiements, but he can also use them to entertain himself and others too so that he doesn't miss his family and friends too much. Can you think of any ideas for ways to use the inputs, sensors or outputs on the ISS? Think outside the box to create an experiment, such as Pac-Man. Draw and label your ideas in the box below.

Peake, cough). But I feel pleased that someone is representing primary education and is able to inform those who aren't in school every day what it's like in that environment – usually this is not what people imagine."

Sway is a keen maker, but her focus is on educating children rather than building for its own sake. "I am more of a functional user," she tells us. "I make things that help me teach stuff as opposed to cool projects.

"I am proud of everything that a child I teach creates," adds Sway. "From a lone LED that starts flashing, to a stop-motion camera program. For someone in a primary school to achieve such things is really amazing and it opens so many doors in their future."

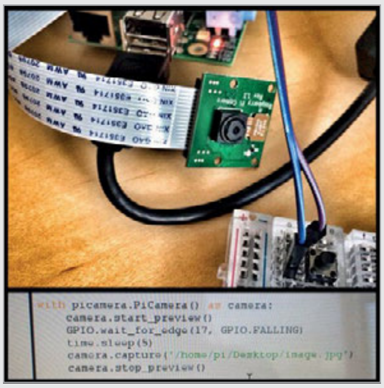
Opening doors

These projects help Sway to enrich the lives of her children. "I've seen the doors these skills can open and I want everyone to have the opportunity to try them. Particularly in primary schools, the main thing we need to do is show children what's possible – even if they're not ready to achieve that yet – and plant seeds that grow into ambitions for the future. The broader the experiences, the more rounded an individual you can become and Computer Science (CS) is my way to reach these children. It's so diverse, you can like computer games, music, art, science, etc. and there's still a CS-related project that can engage you.

> "I've been incredibly lucky with the support I've gotten from the community over the years"

"I've been incredibly lucky," says Sway, "with the support I've gotten from the community over the years. When I taught my first ever programming lesson, in 2013, I did it to win some Raspberry Pi devices. I had no idea what I was doing. I did it anyway. The amount of times I've had to tweet someone and ask what I've done, or what something means, or how do I do something, and there's always someone willing to take the time to explain it. I can only offer the opportunities I offer to children because those people were patient enough to explain it to me. Everyone should aspire to be a lifelong learner and ensure everyone knows you're still learning too!"

Left Sway attended the first ever Raspberry Picademy, becoming a Raspberry Pi Certified Educator



```
with picamera.PiCamera() as camera:
    camera.start_preview()
    GPIO.wait_for_edge(17, GPIO.FALLING)
    time.sleep(5)
    camera.capture('/home/pi/Desktop/image.jpg')
    camera.stop_preview()
```

Left Sway's pet projects include the code for a stop-motion camera

PI TOWERS

Sway was invited (along with other Digital Leaders) to Pi Towers for a day exploring and learning whilst sharing their own ideas and opinions with the Foundation. "When I explained the potential trip to the children, they were excited," recalls Sway. "When I handed out the letters, they nearly exploded. All the slips were back the following day – digital leaders were queuing at my door before school start and it wasn't even first come, first served!"

magpi.cc/wPuHYv

Left There's more to primary education than conkers and times tables

RASPBERRY JAM EVENT CALENDAR

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

5 ROANOKE RASPBERRY JAM
Roanoke, VA, USA
4 DALLAS YOUNG MAKERS CLUB
Dallas, TX, USA
1 OLIVE BRANCH RASPBERRY JAM
Olive Branch, MS, USA
2 SHE PI JAM
Lagos, Nigeria

FIND OUT ABOUT JAMS

Want a Raspberry Jam in your area? Want to start one? Email Ben Nuttall to find out more:
ben@raspberrypi.org

1-5 HIGHLIGHTED EVENTS

1 OLIVE BRANCH RASPBERRY JAM

When: Saturday 28 July
Where: B.J. Chain Public Library, Olive Branch, MS, USA
magpi.cc/WxeYYR
Join the Olive Branch team for a free family event showing off awesome inventions.

2 SHE PI JAM

When: Sunday 5 August
Where: Herbert Macaulay Way, Lagos, Nigeria
magpi.cc/jeLRKx
An initiative to build and train a community of young women makers with the use of Raspberry Pi.

3 DUBLIN RASPBERRY PI JAM

When: Saturday 11 August
Where: Science Gallery Dublin, Dublin, Ireland
magpi.cc/jXkusP
The first Dublin Raspberry Pi Jam will have something for everyone, from beginners to dedicated Pi hobbyists.

4 DALLAS YOUNG MAKERS CLUB

When: Saturday 11 August
Where: J. Erik Jonsson Central Library, Dallas, TX, USA
magpi.cc/UTJrCh
A new, regular free workshop for kids aged 8–18 that covers robotics and Raspberry Pi!

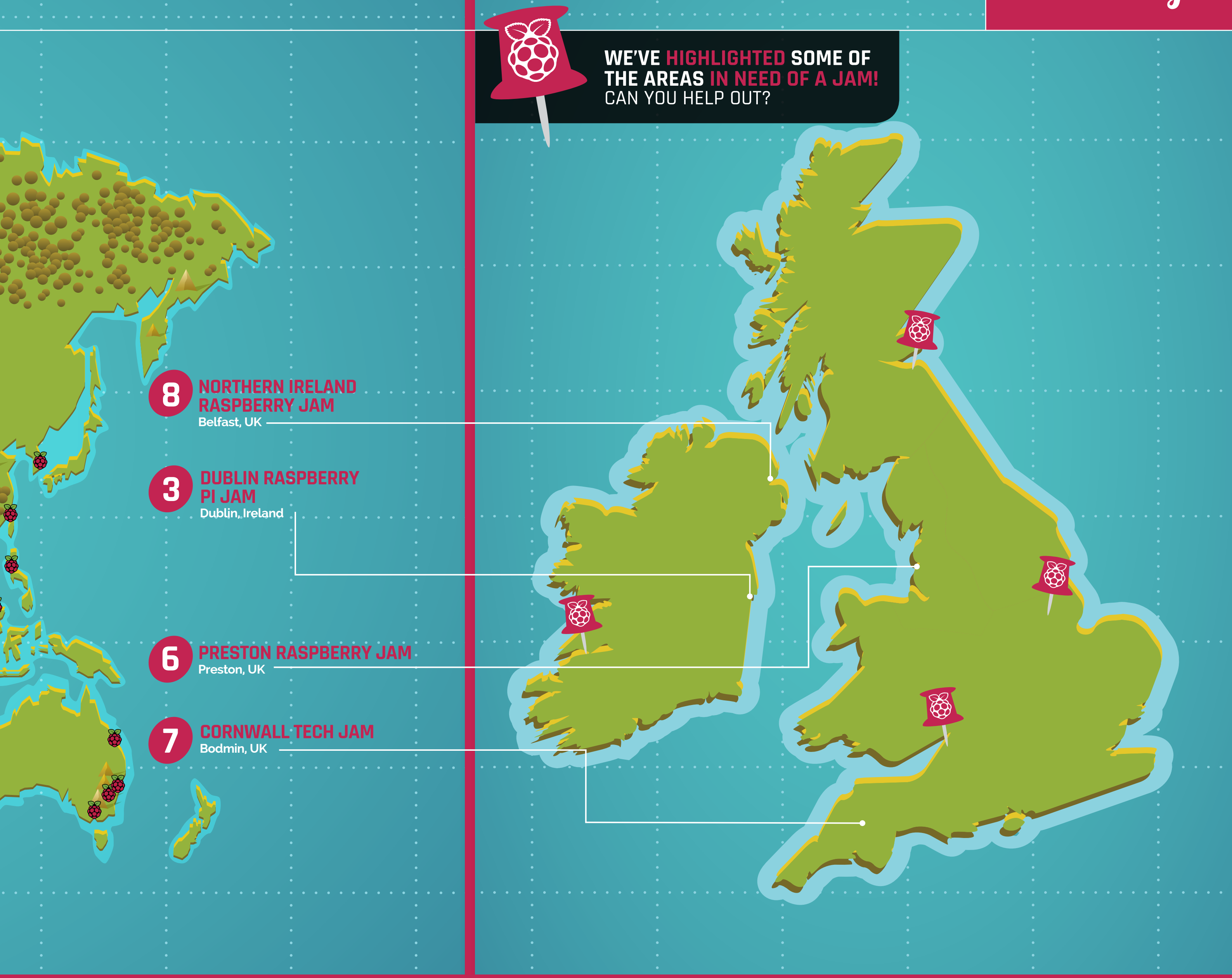
5 ROANOKE RASPBERRY JAM

When: Saturday 8 September
Where: Roanoke South County Library, Roanoke, VA, USA
magpi.cc/PAdbMf
A robotics-focused Raspberry Jam, with demon Pi robots and a chance to build a GoPiGo in teams.

6-8 REGULAR EVENTS

6 PRESTON RASPBERRY JAM

When: Monday 6 August
Where: Media Innovation Studio
magpi.cc/OYbCLu
A community of people who meet to learn, create, and share the potential of the Raspberry Pi.


WE'VE HIGHLIGHTED SOME OF THE AREAS IN NEED OF A JAM! CAN YOU HELP OUT?
8 NORTHERN IRELAND RASPBERRY JAM
Belfast, UK
3 DUBLIN RASPBERRY PI JAM
Dublin, Ireland
6 PRESTON RASPBERRY JAM
Preston, UK
7 CORNWALL TECH JAM
Bodmin, UK

7

CORNWALL TECH JAM

When: Saturday 8 September
Where: Bodmin Library, Bodmin, UK
cornwalltechjam.uk

Learn about programming on platforms including Arduino and Raspberry Pi, in various languages.

8

NORTHERN IRELAND RASPBERRY JAM

When: Saturday 8 September
Where: School of Maths and Physics Teaching, Belfast, UK
magpi.cc/yNUGdg

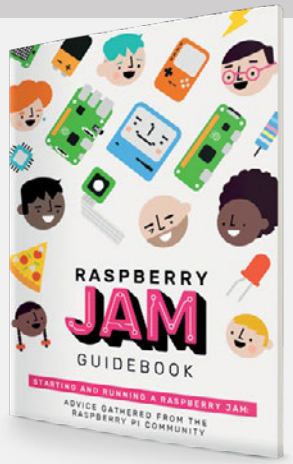
This free event involves tinkering, coding, electronics, and generally having fun making!

RASPBERRY JAM ADVICE

PROMOTING YOUR EVENT

> Our volunteers have badges that say 'Jam Maker' on them. Sometimes we have lanyards. Tim and I tend to wear our purple CamJam polo shirts. They weren't cheap, but we do stand out!
>
> **Michael Horne**
> **Cambridge 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 guidebook here: **magpi.cc/2q9DHfQ**


RASPBERRY JAM GUIDEBOOK
STARTING AND RUNNING A RASPBERRY JAM
ADVICE GATHERED FROM THE RASPBERRY PI COMMUNITY

YOUR LETTERS

TEACH ME

I love what the Raspberry Pi Foundation does, and have used a few of the projects on their website. I keep hearing about some kind of free teacher training, though – what exactly is this? Do I need to be a teacher to take it?
Ellen Y

You're thinking of Picademy, which is currently held in the UK and North America by the Foundation at various times and locations – check out the website for more info: **magpi.cc/picademy**. As long as you're helping kids learn about computing, you're eligible to apply for a session.

If there's not an event you can go to, you can always do the online courses on FutureLearn (you can find the links on the Picademy page linked above) which will give you more teacher-focused training.

3D-PRINTABLE

I loved the article on how to build a 3D printer in issue 69. However, now I've built it and tested it to make sure it works, I find myself unsure what to print! While I occasionally stumble across cool stuff on Thingiverse, I'd like to get some bits for my future Pi projects. Do you have any suggestions?
Bill C

Well, you can always get a 3D-printed case for your Raspberry Pi – including a wonderful 3D-printable version of the Astro Pi case that is the same size and shape as the cases used for the Raspberry Pi boards up on the ISS. You can find the details on how to do it here: **magpi.cc/fwxXmO**.

There are also the PiGRRL cases that you can use to make your own handheld retro console using the Raspberry Pi. There are many versions, but we're partial to the PiGRRL 2: **magpi.cc/2dbpjLx**.

There's also a brilliant recent article on the Raspberry Pi blog full of great 3D-printable projects and such. Find it here: **magpi.cc/ccGmWo**.

The PiGRRL projects are great uses of 3D-printed parts

SUMMER FUN

Last year I remember that you put out some tutorials for the summer – I loved doing them with my kids, and I was wondering if you'd be doing another series this year at all? They're really into using their Raspberry Pis, so it would be great to have more summery-themed projects to try out!

If not, do you know anywhere where we can find some guides to follow along to?
Cass

Unfortunately, other feature ideas took priority this year and we were unable to squeeze in any summer projects for a period that would make sense – sorry! We'll keep it in mind again for next year, though.

As for summer projects, there are a few on the Raspberry Pi projects website (**rpf.io/projects**) that you might find fun:

- Spinning Flower Wheel (**magpi.cc/MjmRJW**) shows you how to create a spinning flower with a motor that's controlled by your Raspberry Pi!
- The Infrared Bird Box (**magpi.cc/pCxjPR**) helps you make a home for birds that you can look at any time of day.
- Build a Robot Buggy (**magpi.cc/2zfbPbz**) and send it outdoors for testing (and racing).

There are also some great weather-related projects you can try, such as A Window on the Weather (**magpi.cc/QYREuF**), and a Weather Data Logger (**magpi.cc/fPIiix**).

Hopefully, you can use these to keep the little ones entertained throughout the holiday!

FROM THE FORUM: ARTICLE SUGGESTIONS

The Raspberry Pi Forum is a hotbed of conversations and problem-solving for the community – join in via **raspberrypi.org/forums**

WRITE TO US

Have you got something you'd like to say?
Get in touch via **magpi@raspberrypi.org** or on The MagPi section of the forum at: **raspberrypi.org/forums**

A while ago I posted up about how I would like *The MagPi* to do a robot build over several issues.

Several people liked this idea, but I was also wondering what else Pi fans might like to see as a project build in *The MagPi*?

Who knows, perhaps if we can get enough [support] for a certain type of build, *The MagPi* people may take us up on our request (perhaps they could run a vote system somehow on here – I did think of doing a poll, but would want to see what other ideas people have first)?

Otherwise, it would be fantastic if *The MagPi* did a wheeled robot build over a couple of issues or so.

Torcrayth

We're always happy to hear about article suggestions – we are the community magazine after all! Send suggestions to **magpi@raspberrypi.org** and we'll give them a look over! Otherwise, we may just use the idea of creating a poll to get an idea of what you folks would like to see more of.

Otherwise, if you're looking for a robot build in the magazine, we did do one in issues 51 and 52 with the Tiny robot from Brian Corteil.

YOUR OFFICIAL RASPBERRY PI MAGAZINE
The MagPi
BUILD A REMOTE CONTROL ROBOT
Create your own Raspberry Pi-powered rover with our latest guide
MAKE AN UNDERWATER CAMERA
TOTALLY WIZARD
SEND YOUR CODE TO SPACE
Also inside:
SPOOKY HALLOWEEN PROJECTS
THE ONLY PI MAGAZINE WRITTEN BY THE RASPBERRY PI COMMUNITY

If you're interested in an article about building robots, check out issue 51 of *The MagPi*

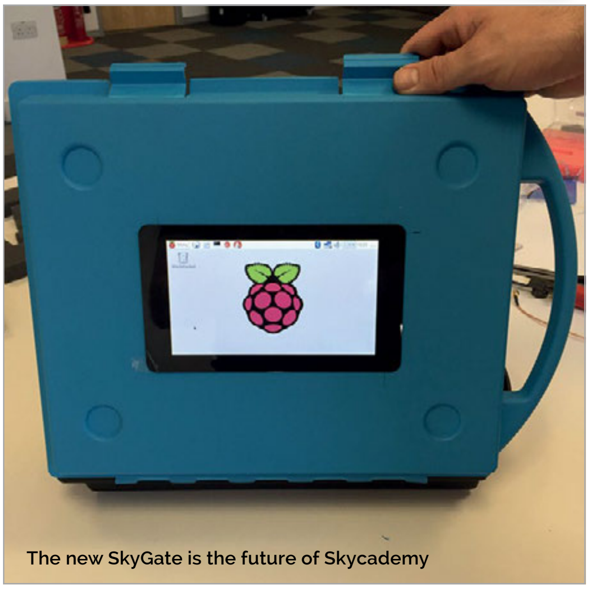
SKY HIGH

I've always been inspired by the high-altitude ballooning (HAB) that people have done with the Raspberry Pi, but have always found the process a little daunting. Are there any plans for a future article on making your own HAB with a Pi? Or are there any helpful resources you know of that would make it easier for a beginner?

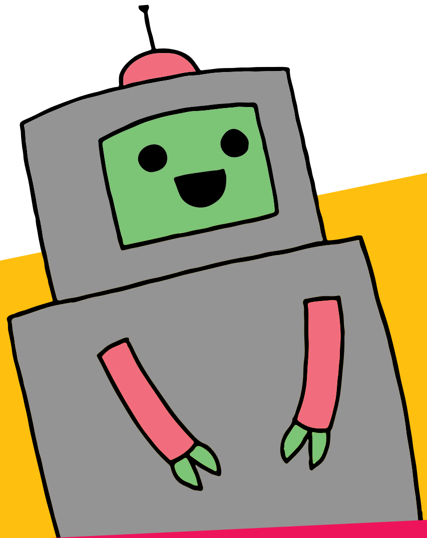
Benjamin

There's a lot to launching a HAB that we may not be able to adequately cover in the magazine (especially in regard to filling up the balloon), but we may give it a go one day!

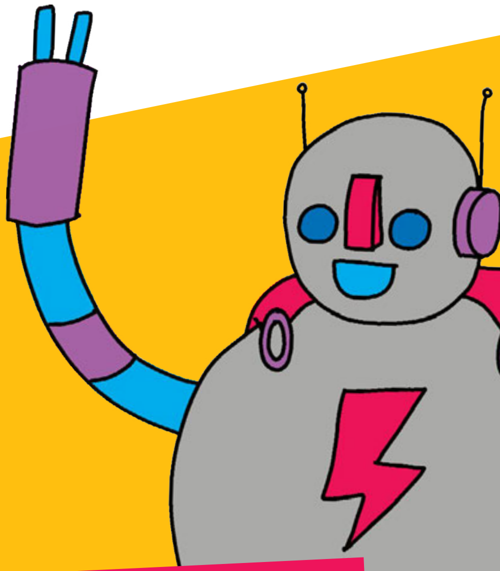
At the very least, we can point you towards the new developments going on with creating new software for Skycademy – the Raspberry Pi Foundation's free teacher training course to learn about HABs. You can read more about it on the Raspberry Pi blog here: **magpi.cc/MoZIYM**.



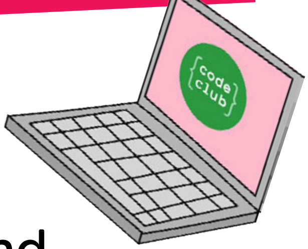
The new SkyGate is the future of Skycademy



code club



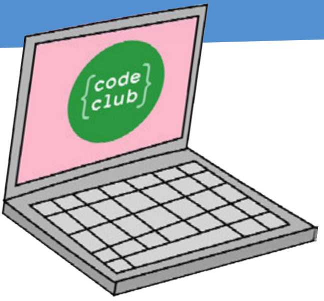
CAN YOU HELP INSPIRE THE NEXT GENERATION OF CODERS?



Code Club is a network of volunteers and educators who run free coding clubs for young people aged 9-13.

We're always looking for people with coding skills to volunteer to run a club at their local school, library, or community centre.

You can team up with friends or colleagues, you will be supported by someone from the venue, and we provide all the materials you'll need to help children get excited about digital making.



To find out more, join us at

www.codeclubworld.org

Code Club is part of the Raspberry Pi Foundation. Registered Charity Number 1129409

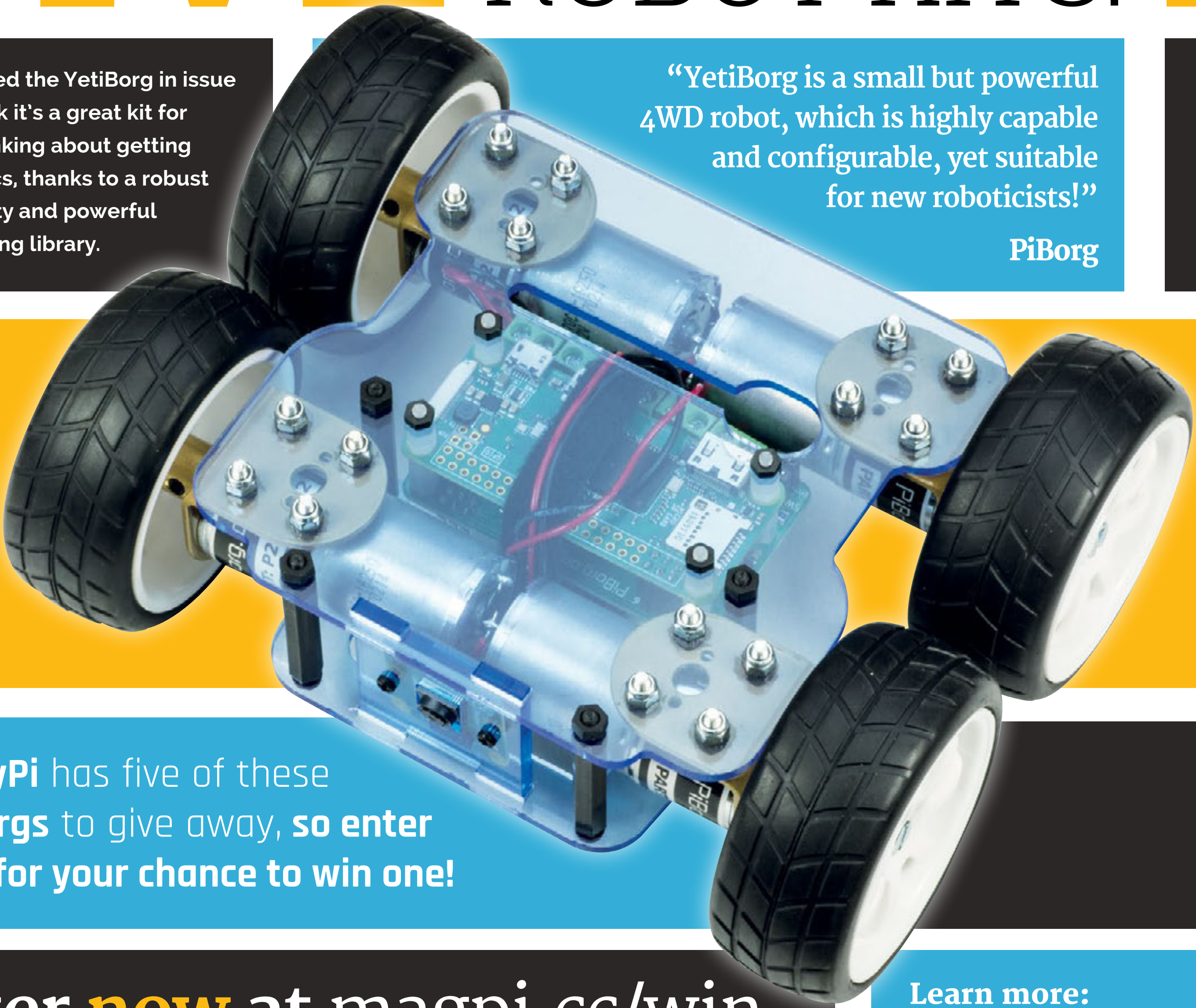
In association with

MODMYPI

WIN! ONE OF FIVE YETIBORG RASPBERRY PI ROBOT KITS!

We reviewed the YetiBorg in issue 71 and think it's a great kit for people thinking about getting into robotics, thanks to a robust build quality and powerful programming library.

> "YetiBorg is a small but powerful 4WD robot, which is highly capable and configurable, yet suitable for new roboticists!"
>
> **PiBorg**



ModMyPi has five of these **YetiBorgs** to give away, **so enter today for your chance to win one!**

Enter **now** at magpi.cc/win

Learn more: magpi.cc/mPKAMg

Terms & Conditions

Competition opens on **25 July 2018** and closes on **31 August 2018**. 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.

MATT RICHARDSON

Matt Richardson is the Executive Director of the Raspberry Pi Foundation North America and author of *Getting Started with Raspberry Pi*. Contact him on Twitter **@MattRichardson**.

THE MIGHTY CAMERA MODULE

Matt Richardson shares his favourite Raspberry Pi camera projects

One of my favourite Raspberry Pi accessories is the tried-and-true Camera Module. Pi enthusiasts have been using this legendary component in their projects since the early days of the Raspberry Pi and I still see tons of great Pi-based camera projects coming out on a regular basis.

I especially like the Camera Module because it can be used in so many different ways. It's great for tweeting photo booths, time-lapse projects, stop-motion rigs, security cameras, and wildlife monitoring projects. Makers also give new life to old antique cameras by retrofitting them with a Camera Module and a Raspberry Pi Zero, which is perfect for this purpose because of its small size. My friend Becky Stern used a Camera Module in an antique camera so that it makes animated GIFs and uploads them to the web via WiFi.

> Some of the most unusual Raspberry Pi projects I've seen use the Camera Module

Curious projects

Some of the most unusual Raspberry Pi projects I've seen use the Camera Module. For example, Dries Depoorter in collaboration with Max Pinckers created what he calls the Trophy Camera (**magpi.cc/NPdreS**). This AI-powered camera is trained with all of the winning photographs from the World Press Photo of the Year competition. Only if the photo you take has similar patterns to the award-winning photos will it be saved and uploaded to a dedicated website. I've never seen a camera like that before.

Recently Dan Macnish created a project called Draw This. It uses the Raspberry Pi 3, a Camera Module, and a thermal printer (another one of my favourite components). After you snap a photo, it's passed through a machine-learning model that identifies objects and their position in the frame. Then it submits that information to a massive dataset of doodles from Google. Those doodles make up a cartoonish version of the snapped frame. The result is printed out the front of the camera, as though it were a Polaroid.

Saving lives

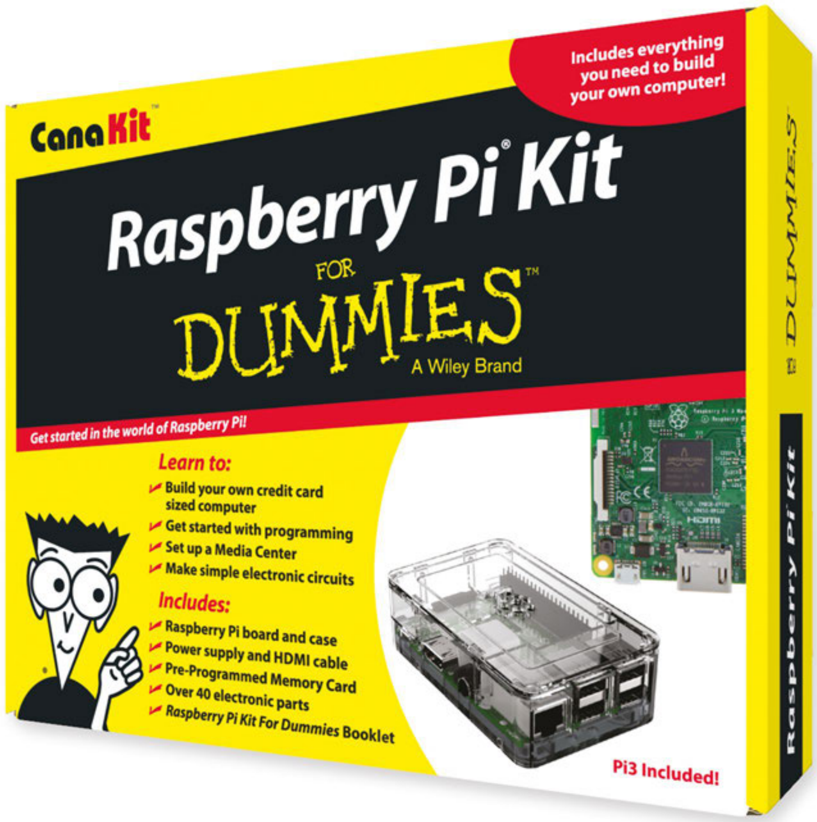
The Raspberry Pi Camera Module isn't only used for creative technology projects. It's also doing serious work and saving lives. Henry J Feldman, Chief Information Architect at Harvard Medical Faculty Physicians, created a Raspberry Pi-based robot that tests 3D-printed airway stents to ensure they're durable before they're implanted into a patient (**magpi.cc/nlvoET**). They test the stents by squeezing them 300 000 times. The Raspberry Pi controls a servo to repeatedly squeeze the stent, and the Camera Module takes a photo of the stent after each squeeze. The machine uses a computer vision algorithm to detect if the stent has failed and will stop the squeezing if so. This gives their team the opportunity to examine how it failed and iterate on the design.

At World Maker Faire New York a few years ago I met Joe Herman, who created his own Raspberry Pi-based telecine (**magpi.cc/NGIsVP**), a machine used to convert motion picture film to digital video. Joe used an old 8 mm film projector and outfitted it with stepper motors to advance the film after each frame is captured with the Camera Module. The Pi can then string those frames together to create video.

As you can see, there are so many amazing ways to use the Raspberry Pi Camera Module in a project. If this column has inspired you to start making your own, be sure to check out **magpi.cc/tutorials** and **rpf.io/projects** to see an abundance of camera-enabled projects. Don't forget to say cheese!

CanaKit™

CK


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
DUMMIES

FOR DUMMIES®
A Wiley Brand

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

Available for worldwide shipping at:

WWW.CANAKIT.COM

Available in Europe
through RS Components

RS

$89.99
US DOLLARS

£69.99
EXCLUDING VAT

Raspberry Pi is a registered trademark of the Raspberry Pi Foundation. For Dummies and the Dummies Man logo are trademarks or registered trademarks of John Wiley & Sons, Inc. Used under license. RS logo is a registered trademark of RS Components Ltd. CanaKit is a registered trademark of Cana Kit Corporation.

ELECTRONIC KITS • ELECTRONIC PARTS • RASPBERRY PI • ARDUINO