THE OFFICIAL RASPBERRY PI MAGAZINE

The MagPi

The official Raspberry Pi magazine

Issue 58 June 2017

raspberrypi.org/magpi

BUILD A WEB SERVER

Step-by-step guide to running a simple server

DEVELOP A GAME

Learn code the fun way by designing games

MINECRAFT MAKER GUIDE

Incredible ideas for projects to hack, code & make

MAKE A MUSIC BOX

Build a hands-free theremin music synth

HACK YOUR LIGHTS

Install weather-responsive lighting for your home

BRILLIANT AIY PROJECTS

Design dazzling devices with Google Cloud Speech API

Also inside:

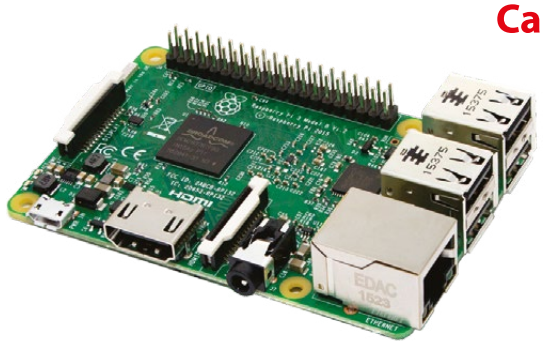
- ZX SPECTRUM REBOOTED WITH RASPBERRY PI
- MAKE A HARRY POTTER-STYLE MAGIC NEWSPAPER
- REVIEWED: JOY BONNET POCKET GAME SYSTEM
- NEVER FORGET WITH A PI ZERO CALENDAR DEVICE

Issue 58 • Jun 2017 • £5.99

9 772051 998001 06

THE ONLY MONTHLY MAGAZINE WRITTEN BY AND FOR THE PI COMMUNITY

CanaKit™



CanaKit Raspberry Pi 3 Ultimate Starter Kit

Model B | 1 GB RAM | 1.2 GHz | Quad-Core CPU

> Learn to Code
> Explore Computing
> Get started with Electronics

KIT INCLUDES RASPBERRY PI 3 AND ...

- PREMIUM CASE & HEAT SINKS

- 2.5A POWER ADAPTER

- 32 GB CLASS 10 MICROSD CARD

PRE-LOADED WITH OPERATING SYSTEM

- USB MICROSD CARD READER

- PREMIUM HDMI CABLE

- QUICK-START GUIDE

- GPIO TO BREADBOARD INTERFACE BOARD

- RIBBON CABLE

- FULL-SIZE BREADBOARD

- JUMPERS

MALE TO MALE & MALE TO FEMALE

- LEDS

- RESISTORS & PUSH-BUTTONS

Available for worldwide shipping at:

WWW.CANAKIT.COM

Raspberry Pi Zero W

Now available at CanaKit!

Wi Fi™ Bluetooth®

ELECTRONIC KITS • ELECTRONIC PARTS • RASPBERRY PI • ARDUINO

WELCOME TO THE OFFICIAL MAGAZINE

Digital makers are the most creative, industrious and innovative group of people in the world.

What other group builds chess robots, weather-responsive lights, or looks at the magical newspaper in Harry Potter and thinks: 'I could make that!'

Life is about what you build, craft, and create.

Minecraft is a lovely creation, but the version for Raspberry Pi is special. Only Minecraft Pi allows makers to hack and code it.

Minecraft Pi is digital making on a virtual level. We recommend it to everybody, and this month we have a massive feature on making with Minecraft.

This issue is special for me. I'm taking over as editor for *The MagPi*. It's a huge responsibility, and I'm tremendously grateful.

The MagPi is a rare publication. A community magazine that is loved by a small band of makers, but big enough to shake the tech world.

Despite its huge success, *The MagPi* remains a community mag. It's your publication. So tell me what you love, and like, and want to see.

Remember to make things. And don't forget to share with the community.

Lucy Hattersley
Editor - The MagPi
lucy@raspberrypi.org

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

CC BY NC SA

EDITORIAL
Editor: **Lucy Hattersley**
Features Editor: **Rob Zwetsloot**
Sub Editors: **Rachel Churcher and Phil King**

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

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

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

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

CONTRIBUTORS
Alex Bate, Mike Cook, Richard Jarvis, Phil King, Simon Long, Joshua Lowe, Mehdi Imani Masouleh, Opemipo Ogunkola , KG Orphanides, Les Pounder, Matt Richardson, Marc Scott, Richard Smedley, Lorraine Underwood, John M. Wargo, Clive Webster

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

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

Contents

Issue 58 June 2017 raspberrypi.org/magpi

TUTORIALS

IN THE NEWS

COVER FEATURE

MINECRAFT MAKES

AIY KIT

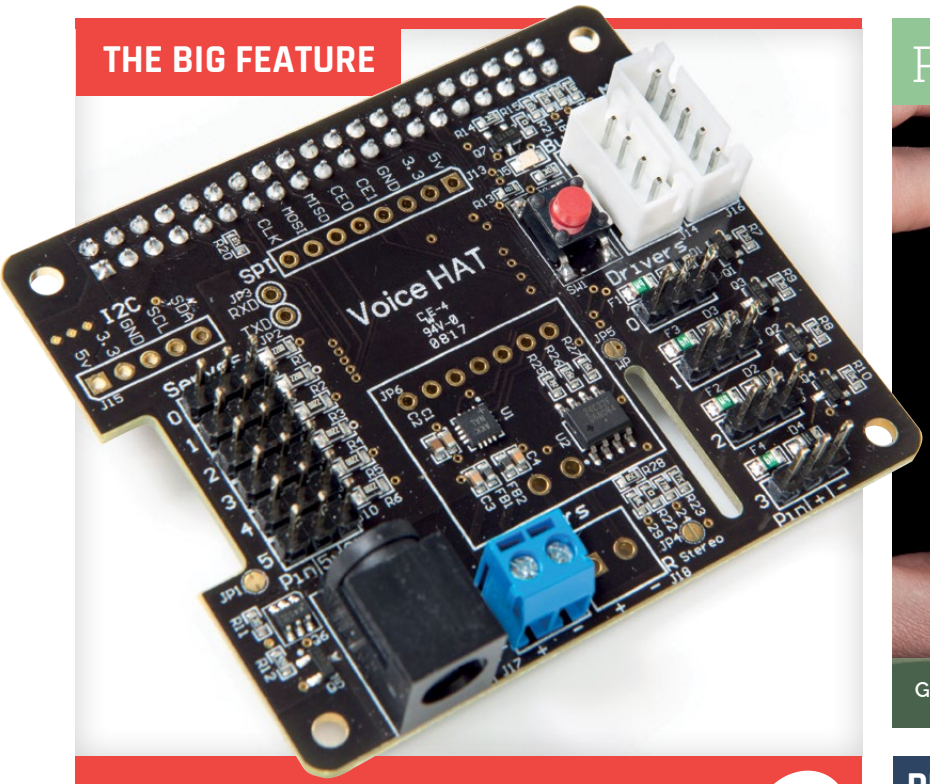
A lot of you enjoyed our voice kit last month! **6**

PIONEERS

Make it outdoors with the new challenge **10**

Contents

THE BIG FEATURE



YOUR PROJECTS

POCO ZERO INTERVIEW

REGULARS

COMMUNITY

REVIEWS

AIY PROJECTS A HUGE SUCCESS

How to get hold of an AIY Projects Voice Kit

Issue 57 of *The MagPi* proved so popular that people quickly resorted to asking staff to look under counters and in back rooms for spare copies to buy. The reason is our giveaway: a free AIY Projects Voice Kit developed by Google.

The AIY Projects Voice Kit was a free gift with issue 57, allowing you to add natural voice interactions to your projects easily and for free. The kit comprises an AIY Projects Voice HAT, a stereo microphone, large arcade-style button, and a selection of wires. In typical Google style, you make the case out of cardboard.

Everybody who got a copy of the magazine seemed to love building the natural language recognizer.

"I had fun building the #AIYProjects this weekend with my son," tweeted *The MagPi* reader Stephen Keep. "It was his first electronics project."

"What a wonderful unexpected surprise," said Simon Patterson.

The magazine quickly sold out at WHSmith, Tesco, Sainsbury's, and Asda. We hope you were one of the lucky ones who got a copy.

The magazine was also available in the United States at Barnes & Noble, Micro Center, and other stockists.

> "Sign up for The MagPi newsletter for alerts"

"I picked up my copy of *The MagPi* last night," said Brian Krohn on Twitter. "Barnes & Noble only had two copies left."

With copies quickly selling out, many readers wanted to know if the kit will be available on its own. "Where can I order one?" asked Will Bunker.

Left The AIY Projects Voice Kit given away free with *The MagPi*

Above If you want to buy an AIY Projects Voice Kit, make sure you sign up for The MagPi newsletter

If you missed the initial release of issue 57 and just want the exclusive AIY Projects Voice Kit on its own, you might be in luck. Raspberry Pi and Google are . working to figure out a way to make the kits available in the longer term.

If you're interested in buying the AIY Projects Voice Kit, sign up for The MagPi newsletter at **magpi.cc**. We will email you if AIY Projects Voice Kit becomes available for purchase. Our newsletter is a good way to be the first to know about special issues of The MagPi.

We can't say for sure that the AIY Projects Voice Kit will be available for purchase, but we are pretty confident that a longer term solution can be found.

Many people subscribe to *The MagPi* (**magpi.cc/Subs1**) and these folks got the AIY Projects kit with their monthly issue.

Remember that subscribing to The MagPi is the best way to ensure you get special projects, like this.

Don't forget: you also get a free Pi Zero W, case, and cable bundle with new 12-month subscriptions.

250,000TH PI ZERO W SHIPPED

Plus 13 new distributors now selling the Pi Zero W

Raspberry Pi has shipped over 250,000 Raspberry Pi Zero W boards, only three months after release.

The tiny computer only costs £10, yet has wireless networking, Bluetooth, a 1GHz processor, and 512MB of RAM – no wonder demand has been so high.

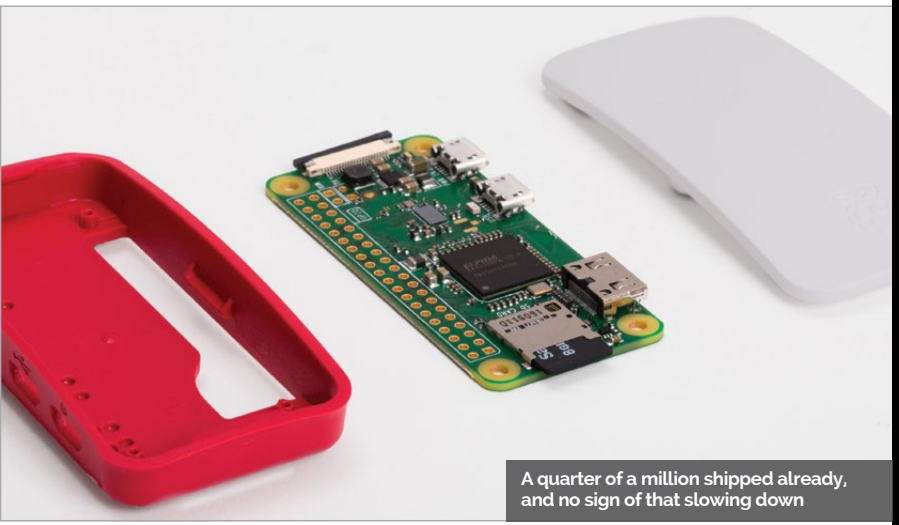
Malaysia, Norway, South Africa and Sweden.

Fans in Malaysia, Japan, and South Africa will have to wait for the Zero W to achieve certification first, however.

Mike Buffham,Director of Product Management, says: "We are further strengthening our network in the USA, Canada, and Germany, where demand continues to be very high. We are hoping that adding these new distributors will make it much easier for Pi fans across the world to get hold of their favourite tiny computer."

> "We are further strengthening our network in the USA, Canada, and Germany"

Raspberry Pi has also signed up 13 new distributors around the world for the Pi Zero W. The new distributors serve Australia and New Zealand, Italy, Finland, Poland, Greece, Japan, Switzerland, Denmark,



A quarter of a million shipped already, and no sign of that slowing down

PI ZERO IN NUMBERS

250,000
PI ZERO W UNITS SHIPPED WORLDWIDE

3 MONTHS SINCE LAUNCH

13 NEW DISTRIBUTORS TO GET ZERO W TO PI FANS

4,000 UNITS SOLD **ON DAY ONE**

NEXT-GEN ZX SPECTRUM

The Spectrum reborn with Raspberry Pi as an add-on board

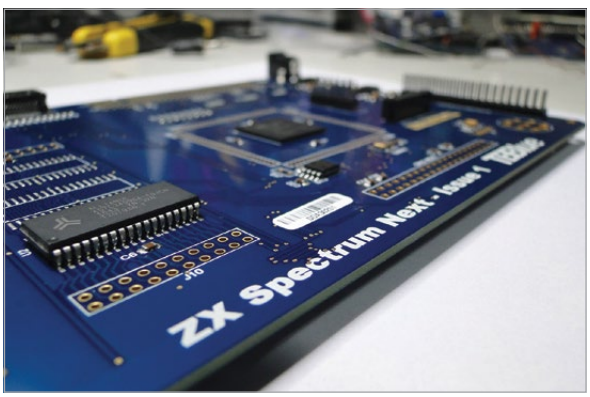
The Kickstarter campaign for an updated version of the Sinclair ZX Spectrum has just closed successfully.

Called the ZX Spectrum Next, the new retro console is backwards compatible with all Spectrum software, yet features hardware updates such as HDMI output, a faster processor, and a quick menu system to load games in seconds.

> "Add a Raspberry Pi Zero as a 'slave co-accelerator board'"

However, if you miss the whines and screeches of loading software on a Spectrum from tape, the Spectrum Next is compatible with all original ZX Spectrum hardware.

Below The Pi Zero connects via the white GPIO-sized connector to add extra processing power



While the main unit is powered by an SLX16 processor on a custom-made circuit board, you can add a Raspberry Pi Zero as a 'slave co-accelerator board'. This novel use of a Raspberry Pi means that the Pi Zero becomes the cheapest graphics card ever made.

Even without the Pi Zero, the Spectrum Next features a 'boost mode'. Henrique Olifiers, co-founder of Bossa Studios and one of the creators of Spectrum Next, says that games like Hard Drivin' and Starstrike II had such low frame rates on the original Spectrum that they were unplayable. Hit the boost mode button and "all of a sudden, this is playable now."

It's not just more speed that the Spectrum Next offers, but improved visuals too. Some classic games have a new 'enhanced colour mode' that

Above The new case adds modern flair, but the bare circuit board will fit inside an old ZX Spectrum case if you prefer

"makes the games look much better than they did before." Henrique says it adds: "Colours that we didn't have originally, with oranges and so on, but it's still the same game!"

Games can be stored on an SD card, and the 512kB of RAM can be expanded to 2.5MB. There's also a PS/2 port to add a mouse, and a VGA output as well as the HDMI to connect to any monitor.

The ZX Spectrum Next has received over £440,000 of crowdsourced funding from more than 1,900 backers on Kickstarter, smashing its target of £250,000. We expect the final product to start shipping in July 2018.

More information can be found at **magpi.cc/2rc3FjT**.

A PALM-SIZED PI SERVER

WD PIDRIVE NODE ZERO

ONLY
$44.95

Build a personal file server, wireless mobile storage device, a Pi Music Box with local storage and so much more.

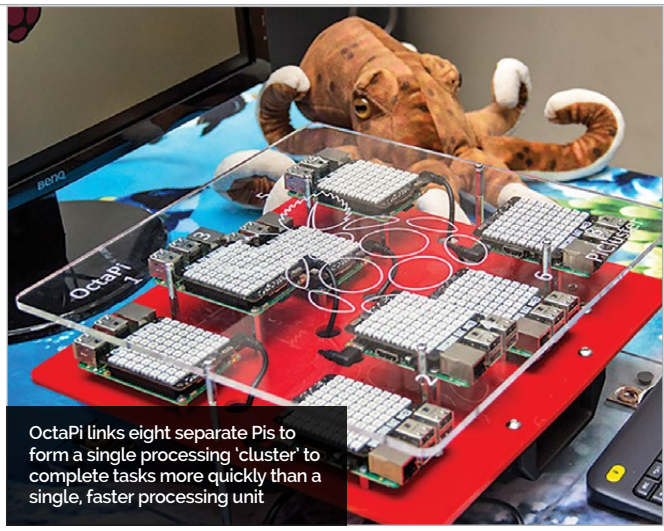
Includes:

- 314GB WD PiDrive, Raspberry Pi Zero, and custom adapter board
- SD card, starter software, and mini HDMI® adapter cable

WDLABS

Start building your mini file server now at **wdlabs.io/mp58a**

GCHQ SHOWS OFF ITS OCTAPI



OctaPi links eight separate Pis to form a single processing 'cluster' to complete tasks more quickly than a single, faster processing unit

Spy agency takes its Raspberry Pi cluster computer on tour

UK security agency GCHQ brought a few of its toys to the Edinburgh International Science Festival in April, among them an eight-way Pi cluster dubbed the OctaPi.

GCHQ's aim with the OctaPi is to show how processing performance and capability can be 'scaled out' instead of the typical approach of 'scaling up'. The latter is proving increasingly hard to achieve, as single processing units can only operate reliably up to certain frequencies. This limits the maximum possible level of performance.

Instead, 'scaling out' adds more processing units, allowing tasks to be divided and processed in parallel. This, GCHQ tells us, requires a change in approach in how applications and programs are coded.

The Science Festival lasts for two weeks every year – see **sciencefestival.co.uk** for details.

PIONEERS HEAD OUTDOORS

Next Pioneers challenge is to 'Make it outdoors'

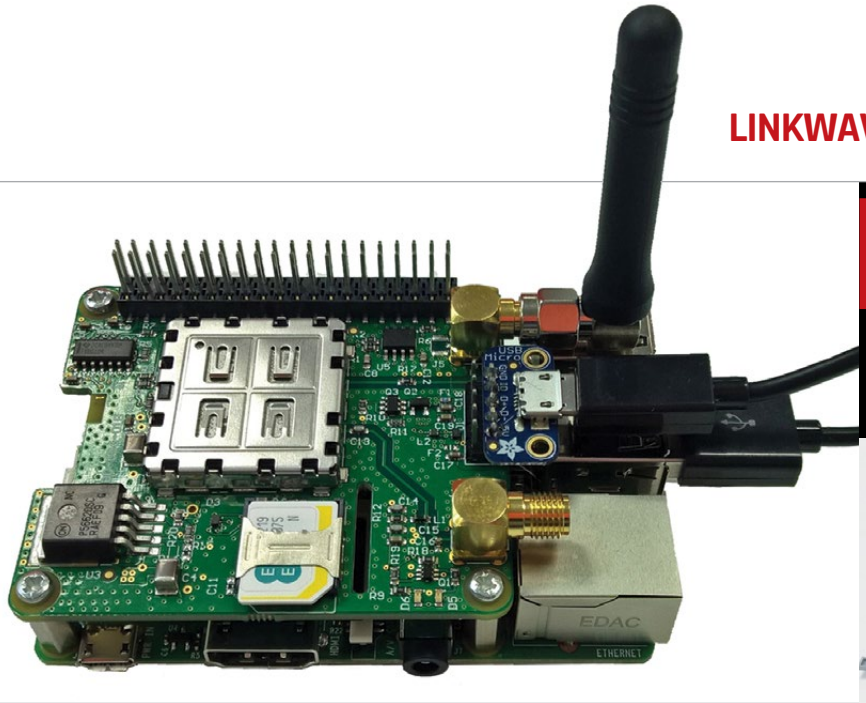
As long it's something outdoors, you can make anything!

The next challenge for Pioneers is to 'Make it outdoors'. This could mean anything from using a Raspberry Pi to track your pet or making a speedometer for your bike.

Pioneers are UK-based teams of between two and five people, all aged 11 to 16. That might be a school-based group, a group of friends with a mentor, or a team attending a Pioneers event. If you're over 18, you could mentor a Pioneers team: see **rpf.io/pioneers** for details.

To help you Make it outdoors, Pimoroni is offering a 15 percent discount on its 'Getting started with wearables' (**magpi.cc/2pY406W**) and 'Getting started with Pi Camera' kits (**magpi.cc/2pXNpzZ**). The Shell Centenary Scholarship Fund (**magpi.cc/2pXGsPf**) is offering bursaries to cover the costs of these kits.

For more information on Pioneers, see 'This time it's outdoors,' Page 86.



LINKWAVE LAUNCHES 3G HAT

Add anywhere-internet to your Pi

Linkwave has launched a 3G HAT for the Raspberry Pi, compatible with the Pi 2, 3, and Zero. The new HAT, called the PiloT, aims to make building a Pi-based Internet of Things (IoT) device easier and quicker.

UMTS, HSDPA, HSUPA, and HSPA+ mobile data standards for quick uploads and downloads on mobile data networks.

The PiloT is powered by the Pi, using a small USB to micro-USB cable (supplied). The PiloT aims to be easy to use and implement,

> "The PiloT is powered by the Pi, using a small USB to micro USB cable"

There are two versions of PiloT. The standard model costs £95 inc. VAT while the HL8548-G model adds GNSS location information (via the SiRF V GPS and GLONASS standards) and costs £107 inc. VAT. Both versions support with a CDC-ECM mode allowing the PiloT to act "as an Ethernet-like device. In this mode PPP is not required; a simple command initiates the session."

See **magpi.cc/2pXPrjv** for more details.

NOW TRENDING

The stories we shared that flew around the world

BUILD A LEGO NES CONSOLE

magpi.cc/2p2Ktni

Supplied with its own brick-by-brick instruction manual, this kit allows you to build a retro Nintendo NES case out of LEGO. It's perfect for housing a Pi 3 running the RetroPie classic games emulator.

PI WARS 2017

magpi.cc/2npYSoW

Veterans and newcomers alike competed in Pi Wars at the start of April. Exa Coding Club's ExaBot finally won the overall Schools competition, while Brian Corteil's '2 wheels or not 2 wheels' was the Pros / Veterans champion.

TRACKING TELESCOPE MOUNT

magpi.cc/2p4QmNl

With some precision engineering, a Raspberry Pi 3 and some Python coding, Mike Hamende (known online as '8PumpkinDonuts') built this telescope mount to automatically track celestial bodies.

AI CAMERA TAKES AWARD-WINNING SHOTS

With a Raspberry Pi inside, this camera knows whether a shot it has taken is award-worthy

Two Belgian creatives have designed and made a camera that 'can only make award-winning pictures', thanks to an AI algorithm running on a Raspberry Pi.

The 'Trophy Camera' has been 'trained' to identify what makes an award-winning image by analysing all the previous winning entries for the World Press Photos of the Year. When you take a picture with the Trophy Camera, the camera is programmed to 'recognise, make, and save only winning photos.'

Intriguingly, the Trophy Camera doesn't even have a viewscreen, merely a two-line readout that tells you whether the image you have taken is of award-winning standard or not. The project uses a Pi Zero W and the Camera Module v2.

Asked how the Trophy Camera works, Dries Depoorter, the engineer of the project, tells us, "Point the camera at your subject and press the red button… the LCD screen will then show what it sees in the form of labels, for example: inside, room, kitchen, blender. Then it will give the photograph a rating based on how much it looks like a World Press Photo winner. If the photograph has 90% of more correlation with the algorithm it has created based on the history of World Press Photo winners, it automatically uploads the picture to the website."

With no viewscreen, the Trophy Camera ironically demonstrates that an AI can judge photographic merit better than a human

"The idea for the camera came from a concern for the development of repetitive visual and aesthetic strategies in photojournalism," says Max Pinckers, the other half of the Trophy Camera team, who is currently researching tropes in photojournalism for his PhD at the School of Arts in Ghent. "Press photography appears to be becoming a self-referential medium dominated by tropes, archetypes, and pop-culture references."

> "The technology functions on a deeper level that we humans cannot see or understand"

The Trophy Camera, therefore, seems to be a tongue-in-cheek demonstration of the lack of creativity in what should be an intensely creative medium.

But can an AI algorithm discern arresting images from poor-quality ones? Dries tells us, "The technology in the camera searches for patterns in photographs and creates its own standards for evaluating them, but also functions on a deeper level that we humans cannot see or understand."

The Trophy Camera is described as 'v0.9', and Max has great plans for a future update: "I wanted to build the camera so it's real-time and checks real-time [for] award-winning pictures. For the moment, you have to press a red trigger. The next version is going to be real-time."

He originally constructed the Trophy Camera with a Pi 2, "but then the new Raspberry Pi Zero W got released, and I ordered right away." Fitting everything inside the case was tricky, "so the battery is really small… the biggest challenge was the software then the physical camera. I made it in ten days."

The Trophy Camera is on display in Tetem in The Netherlands until 30 July. More info (in Dutch) can found at **magpi.cc/2pXK6sw**.

Below It took Dries and Max ten days to make the AI-powered Trophy Camera, on display at the Tetem in The Netherlands

NEW ROBOT KIT FROM LEGENDARY PI WARS DESIGNER

Make a Tiny 4WD robot rover

Perennial Pi Wars champion Brian Corteil has partnered with Pimoroni to launch the £55 Tiny 4WD kit.

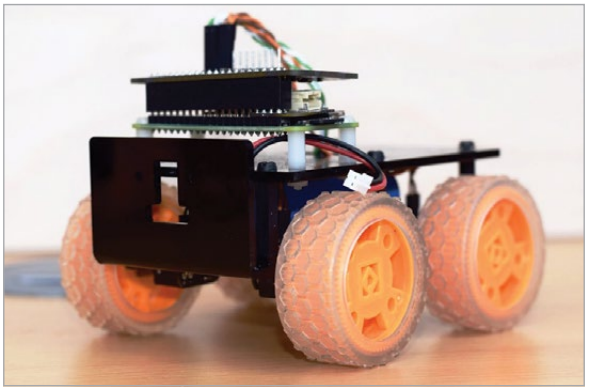
The Tiny 4WD is close, but not idential, to Brian's remote-controlled robot detailed in issue 51 of *The MagPi*. "The reason for the change to the Pimoroni pHAT," Brian tells us, "was to get the price of the kit below £60. If a ZeroBorg was used, the kit would have been closer to £90 to £100."

The Explorer pHAT also "has buffered 5V inputs/outputs plus four analogue inputs, allowing a range of sensors to be added."

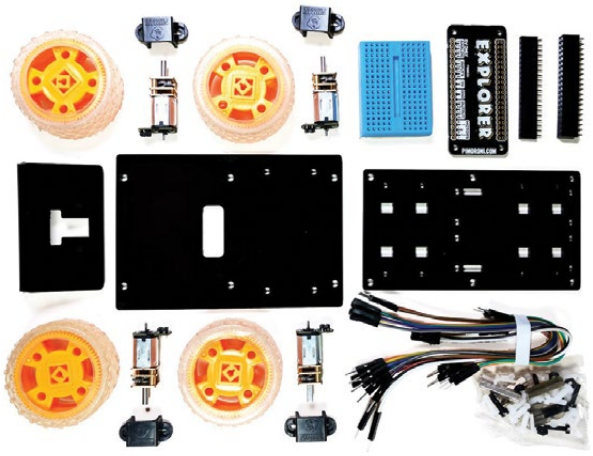
When selecting motors, Brian advises "the 50:1 ratio [motors] are a great choice for general use. A higher ratio, say 20:1, makes the robot into a more sporty model, [but] turning on the spot and climbing is not as good due to lower torque. For computer vision and sensors, the 298:1 motors are more suitable, due to the lower speed."

Emma Norling was lucky enough to win a Tiny 4WD kit recently and shared her experience with it at **magpi.cc/2pXHCu8**. The kit is "really simple to put together, and nice and robust. I've had experience before with kits that are tricky to put together, but ultimately robust; or simple to put together but [of] poor build quality - this one wins on both counts."

You can buy the CoreTec Tiny 4WD Robot Rover kit from Pimoroni at **magpi.cc/2r8e70S** for £55.



Left The rugged Tiny 4WD is quick and easy to assemble



Images credit: shop.pimoroni.com

Above You only need to add a Pi, SD card, and a power source

THE FUTURE OF STEAM EDUCATION

POWERED BY RASPBERRY PI

pi-top Complete STEAM Education solution

Transform your classroom. pi-top provides access to world-class educational resources for the Computer Science (CS) and STEAM (Science, Technolgy, Engineering, Art and Math) curriculum, designed to improve digital literacy with an all-inclusive simple to deliver solution.

www.pi-top.com

bett AWARDS 2017

pi-top

The modular build-it-yourself Raspberry Pi powered laptop

Colors: green or gray
Includes Raspberry Pi

10 Hour Battery Life

13.3" HD Screen

Modular Components

pi-topCEED

The modular all-in-one Raspberry Pi powered desktop

Colors: green or gray
Includes Raspberry Pi

Adjustable Viewing Angles

14" HD Screen

Modular Components

OCR Oxford Cambridge and RSA

The only maker product endorsed by OCR - a leading UK awarding body

modular add-on boards

Available from our distributors:

BT More power to you

CPC

Findel Education Resources The choice for schools for classrooms for specialists

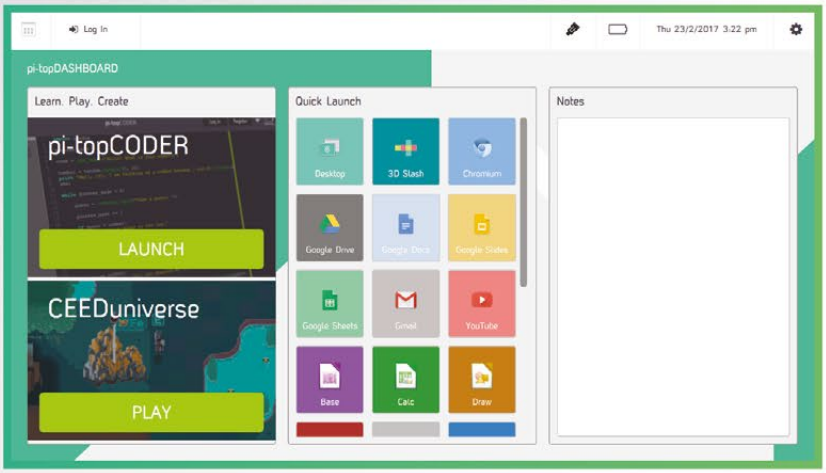
MISCO

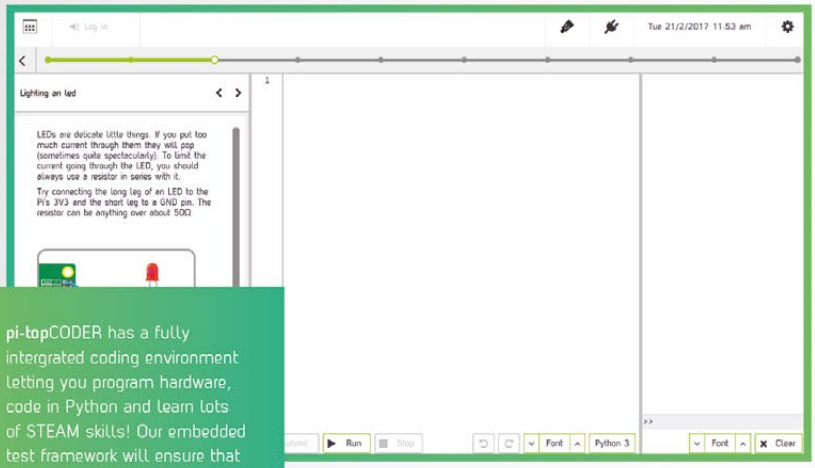
RS

ypo

pi-topOS

The OCR* endorsed **pi-top**OS (Operating system) platform comes pre-installed on the SD card shipped with every unit. **pi-top**OS software suite lets you - browse the web, - check emails, - create and edit Microsoft Office compatible files. It includes **pi-top**CODER and comes with the revolutionary educational game CEED**universe**.





pi-topCODER has a fully intergrated coding environment letting you program hardware, code in Python and learn lots of STEAM skills! Our embedded test framework will ensure that you understand all concepts.

pi-topCODER

pi-topCODER is the interface that allows you to access worksheets and pre-built lesson plans. It's the easiest way to deliver computer science lessons providing step-by-step guides for computer science and STEAM worksheets. Programming languages that can be taught: Python and Sonic Pi (a variant of Ruby) 100+ hours of lesson plan content.

pi-topCODER

CEEDuniverse is our educational game. It's a world of fantasy developed in line with the computing curriculum – taking science fiction and transforming it into science. It is a FREE massive role-play game carefully crafted by **pi-top**. The game teaches students to solve computational puzzles, how to code in Python and build physical circuits which interact with the game. Exploring the planet, the students first encounter 'drag & drop' coding puzzles and move on to writing text based code.

MINIGAMES

Learn programing concepts through our minigames, for example learn problem decomposition by solving visual programming puzzles.

pi-top

Worldwide shipping available in green or grey at www.pi-top.com
Stay up to date with our latest news by following our social media

MINECRAFT MAKER GUIDE

Want to do more with Minecraft Pi?
We've got some excellent projects for you to try, whether you're a novice or a pro!

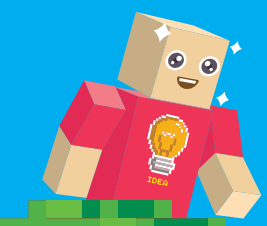
Bare Essentials

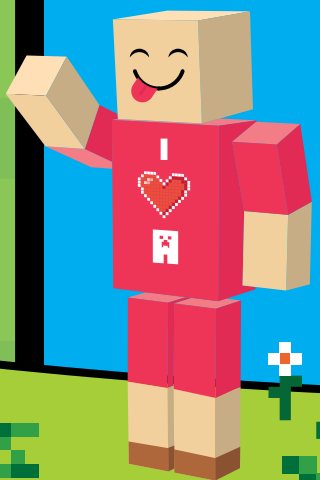
- Raspberry Pi (any model!)
- Latest version of Raspbian
- Monitor
- Keyboard and mouse

We love how easy it is to hack Minecraft on the Raspberry Pi. With built-in Python libraries that let you modify the world you're playing in, the possibilities are nearly endless!

In issue 41 we covered some Minecraft mash-ups, and this issue we're bringing Minecraft hacking back with five new projects that make use of some different aspects of the game. Whether it's using a different programming method, such as EduBlocks, or using RFID cards or a camera to connect Minecraft to the real world, we've got it covered.

Fire up your Raspberry Pi, and get ready to bend Minecraft to your will.





MINECRAFT PI BASICS

Your first steps into hacking Minecraft Pi

Programming Minecraft in Python makes use of the special API that allows you to control, alter, and interact with the Minecraft world. It even works as you play the game and it lets you do the following:

- Get the player's position
- Change (or set) the player's position
- Idenitfy the type of block
- Change a block
- Change the camera angle
- Post messages to the player

Hello world!

The most basic thing you can do is print a message to the player (you!) in the Minecraft world. Here's how you can do that...

01. Go to the Minecraft menu by pressing the **ESC** key, but leave the game playing.

02. Open IDLE by clicking Menu > Programming > Python 3.

03. Use File > New Window to create a new program and then save it as **hellominecraftworld.py**.

04. At the top of your program, type the following code to import the 'minecraft' module, which will allow you to use the API and talk to the game:

```
import mcpi.minecraft as
minecraft
```

05. Create a connection from your program to Minecraft and call it **mc**:

```
mc = minecraft.Minecraft.
create()
```

06. Use your Minecraft connection and the function **postToChat()** to put a message in the chat window:

```
mc.postToChat("Hello
Minecraft World")
```

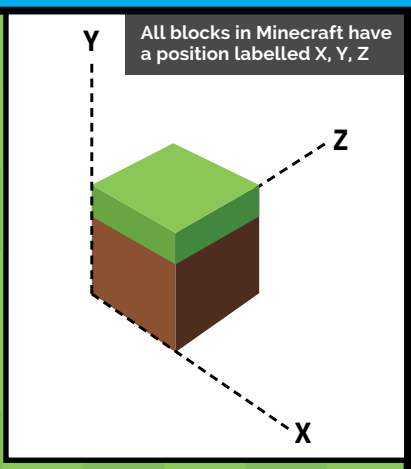
07. Run your program by clicking Run > Run Module or pressing **F5**.

Back on Minecraft, you'll see the message 'Hello Minecraft World' on your screen. You'll have to be quick, though, as it only lasts for ten seconds. Try printing other words and phrases to Minecraft.

Display messages in Minecraft using a simple piece of code

Blocks and positions

Minecraft is a world of blocks, all about 1 m × 1 m × 1 m. The player and every block in the world has a position made up of x, y, and z; x and z are the horizontal positions, and y is the vertical. The player starts at position x = 0, y = 0, z = 0, which is the spawn point, and the player's current position is shown at the top left of the screen.



All blocks in Minecraft have a position labelled X, Y, Z

Add the following code to your Hello Minecraft World program to teleport the player (called Steve) to position x = 0, y = 50, z = 0, which will put him 50 blocks up in the air:

```
mc.player.setPos(0, 50, 0)
```

You can use similar bits of code to change blocks at certain positions on the map as well. To delete a block, you can make it turn into an air block – that's about as empty as it gets in Minecraft!

MINECRAFT ESSENTIALS

Loved this feature and want to do more modding with Minecraft? Check out our Essentials book, *Hacking and Making with Minecraft*: magpi.cc/Minecraft-book

MARC SCOTT

Marc is Head of Curriculum Development at the Raspberry Pi Foundation. He also likes pyrotechnics.
raspberrypi.org

THE BIG MINECRAFT PIANO

Unleash your inner Tom Hanks on this giant piano

In this project, you're going to create a piano keyboard in Minecraft that can be played when Steve (the player) walks over the keys.

This project uses Sonic Pi to play the music, Minecraft to visualise the piano and act as your input, and Python to build the piano and allow Sonic Pi and Minecraft to communicate with each other.

MINECRAFT WITH: SONIC PI

A programming language built into Raspbian, Sonic Pi lets you create music. It can hook into Minecraft Pi, making it perfect for modding.

STEP 1
Receiving messages in Sonic Pi

The first step in this project is to try to send notes from Python to Sonic Pi. This is possible because Sonic Pi uses Open Sound Control (OSC). This is a way for sound synthesizers to communicate with each other over a network.

The first thing to do is to tell Sonic Pi to listen out for messages. Load up Sonic Pi by clicking on Menu > Programming > Sonic Pi, and then click into Buffer 0 to start writing code.

You just need a few lines of code in your Sonic Pi file – found in the **MC_piano_sound** listing (page 21). This tells Sonic Pi to listen out for notes, and to play them straight away. You can run the script, but nothing will happen just yet!

STEP 2
Sending messages to Sonic Pi

Open up a new Python 3 file by clicking Menu > Programming > Python 3 (IDLE), then clicking on File > New File. You'll need the python-osc module for this project, so install it with:

```
sudo pip3 install python-osc
```

The first two lines of **piano.py** (page 21) import the necessary methods from the module.

```
from pythonosc import osc_
message_builder
  from pythonosc import udp_
client
```

Next, you need to create an object that will send the message. Open Sound Control

allows for computers to talk to each other, but we're going to use it to get Python to talk to Sonic Pi.

As both programs are on the same Raspberry Pi, you can use the home address of the Raspberry Pi to tell Python where to send the message: this is 127.0.0.1, and it will be on port 4559. In **piano.py**, you'll see a line like this:

```
sender = udp_client.
SimpleUDPClient('127.0.0.1',
4559)
```

This sends out the signal to the right place. When the `play_note` function activates, it then knows where to send the note.

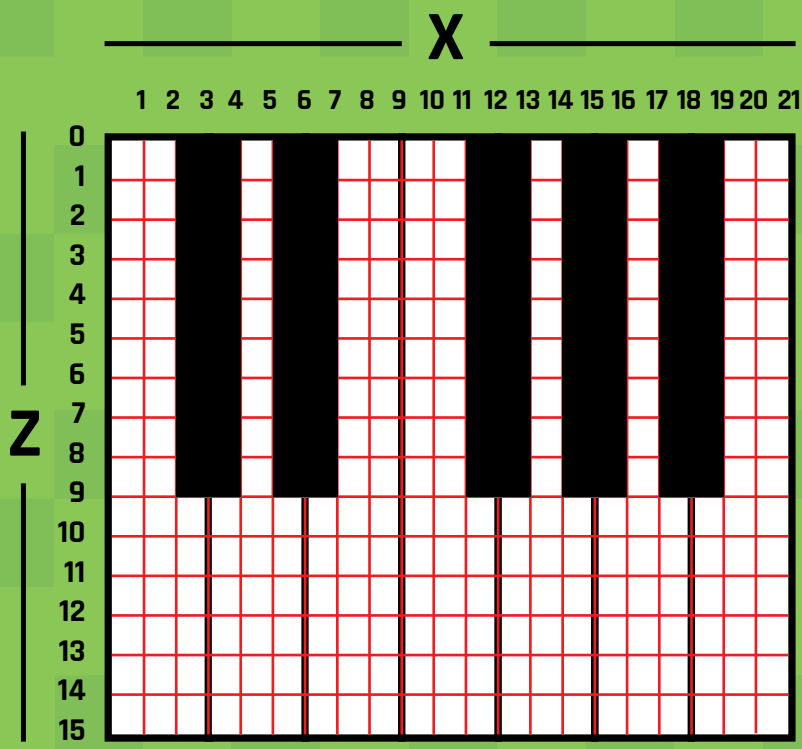
STEP 3
Building piano keys

It may seem a little daunting attempting to build a piano in Minecraft, so it's easier to try to break down the problem into much smaller chunks. This is a process that computer scientists call decomposition.

A piano keyboard comprises repeating groups of seven white keys and five black keys – i.e. octaves. Building each of these elements one at a time will allow you to easily build a keyboard.

Our code will work by checking the location of the player on all three planes so we can allocate a key press to specific coordinates. This is done with the line:

```
player_x, player_y, player_z
= mc.player.getTilePos()
```



STEP 4
Planning the keyboard

It's always a good idea to quickly sketch out what you want to build before you start throwing blocks into the Minecraft world. Here's a diagram (above) of an octave of a keyboard, showing the x and z block positions.

STEP 5
Clearing some space

Depending on where you are in the Minecraft world, you might find your piano could be created in the middle of a mountain. To prevent this, you can clear some space with a bulldozer function that will fill a cube around the player with air. Find this in **piano.py** as:

```
def bulldozer(x, y, z):
    mc.setBlocks(x - 30, y -
3, z - 30, x + 30, y + 20, z
+ 30, 0)
```

STEP 6
Building black keys

The code uses a function called `black_key` to build the black piano keys. The function will need to know where to build the black piano key, so it will need three parameters. These parameters will be the x, y, and z position in the Minecraft world where the key needs to be built.

The next step is to use the `setBlocks` function, to set a few black Minecraft blocks. If you look at the black key on the far left, you can see that it's two blocks wide

and nine blocks long. So if the first block is placed at an x and z coordinate, then you need the one to its right to be placed at x + 1, and the ones below it to be placed at z + 1 up to z + 8. All the blocks can be placed at 1 block below the player's position: y – 1.

Obsidian seems like a sensible material to build the blocks from. This has a block ID of 49, so the **setBlocks** code will look like:

```
mc.setBlocks(x, y - 1, z, x
+ 1, y - 1, z + 8, 49)
```

STEP 7
Building white keys

Have a look at the first white key in the diagram (page 19). It's three blocks wide and 15 blocks long. This time, you need to set blocks from x up to x + 2, and from z up to z + 14. We'll call this function **white_key** to do this using the white tile block, which has a block ID of 44, 7. The 44 is the tile block, and the 7 tells Minecraft that it should be white.

```
mc.setBlocks(x, y - 1, z, x
+ 2, y - 1, z + 14, 44, 7)
```

STEP 8
Making an octave

One octave consists of seven white notes and five black notes. As in the diagram (page 19), the blocks stretch from x to x + 18. The **for** loop needs to place a white key every three block-units on the x axis, from 0 up to 18.

Now you can start making your octave function, placing a white key at every position provided by **i**. Look in **piano.py** for this function:

```
def make_octave(x, y, z):
    for i in range(0, 19, 3):
        white_key(player_x +
i, player_y, player_z)
```

Next is the black keys. You can use the same system for placing these. Look at the diagram again. This time, the black keys need to be placed starting at x = 2. Within the **make_octave** function we can add another **for** loop.

> "One octave consists of seven white notes and five black notes"

At the moment there's just one key too many. A key has been placed at x = 8, and you need to make sure that this key is missed out. A little bit of conditional selection will help with this. If the value of **i** is 8 then the **black_key** function should not be called. Another way of putting this is if **i** isn't equal to 8, the **black_key** function should be called. So we add the conditional **if i != 8:** to the function.

STEP 9
Making the octave again

Let's tie all that together now. At the end of all the functions we've made, we can now call the functions in the code, and use three lines to set it all up. First bulldoze the area, then make the piano, and then on the last line set the player's position, so that Steve moves to the middle of the piano.

Now when you save and run your code, a piano octave should appear beneath your feet. Each time you run the code, a new octave will be produced.

STEP 10
Playing your piano

The next step is to have the piano play a note when Steve walks over a key. This is handled by the big **while** loop. It starts by constantly checking for Steve's current position.

Next, it finds the block below Steve's feet. The problem is that the white keys are only half a block in height.

If Steve is standing on a white tile, because of their smaller height, **block_below** ends up being the air that's beneath the piano. We handle this with a conditional, and check if the block below is not a white or black key, which is what this bit of code does:

BIG

Why is it called the big piano? Well first of all, it's big, and secondly it's from a famous scene in an old film called *Big* where two characters played music on a big piano in FAO Schwarz, a toy store in New York. A version of the piano stayed there until the store was recently shut down.

```
block_below =
mc.getBlock(new_x, new_y - 1,
new_z)
if block_below != 44 and
block_below != 49:
    block_below =
mc.getBlock(new_x, new_y,
new_z)
```

Next, we find Steve's position relative to the piano's position. The piano was placed at **player_x**, but Steve is now standing at **new_x**. Subtracting one from the other will tell you where Steve is standing on the piano octave.

After this is a list of notes to be played. Starting from middle C, the white notes have MIDI values of 60, 62, 64, 65, 67, 68, and 71. The black notes are the MIDI values in between the white notes. You can place a 0 into the **black_notes** as there are only five of them on the keyboard.

The specific white note to play, if Steve is standing on the white note, can be found by dividing his relative x position by -3 and then ignoring the remainder. This is called floor division, and can be done in Python using the **//** operator, like so:

```
if block_below == 44:
    notes_along = relative_
position // -3
    play_note(white_
notes[notes_along])
```

To find the black note to play, we subtract 1 from Steve's relative position, floor dividing by 3, and then subtracting 1 again. This is because the notes are only two blocks wide.

And that's it. Try running the code and then moving over the blocks. So long as Sonic Pi is open and running your initial script, you should hear the piano being played each time Steve steps on a particular key.

MC_piano_sound

```
set_sched_ahead_time! 0
live_loop :listen do
  message = sync "/play_this"
  note = message[:args][0]
  play note
end
```

piano.py

```
from pythonosc import osc_message_builder
from pythonosc import udp_client
from mcpi.minecraft import Minecraft
from time import sleep

sender = udp_client.SimpleUDPClient('127.0.0.1', 4559)
mc = Minecraft.create()

player_x, player_y, player_z  = mc.player.getTilePos()

def bulldozer(x, y, z):
  mc.setBlocks(x - 30, y - 3, z - 30, x + 30, y + 20, z + 30, 0)

def black_key(x, y, z):
  mc.setBlocks(x, y - 1, z, x + 1, y - 1, z + 8, 49)

def white_key(x, y, z):
  mc.setBlocks(x, y - 1, z, x + 2, y - 1, z + 14, 44, 7)

def make_octave(x, y, z):
  for i in range(0, 19, 3):
    white_key(player_x + i, player_y, player_z)
  for i in range(2, 18, 3):
    if i != 8:
      black_key(player_x + i, player_y, player_z)

def play_note(note):
  sender.send_message('/play_this', note)
  sleep(0.5)

bulldozer(player_x, player_y, player_z)
make_octave(player_x, player_y, player_z)
mc.player.setPos(player_x + 8, player_y + 3, player_z + 12)

while True:
  new_x, new_y, new_z = mc.player.getTilePos()
  block_below = mc.getBlock(new_x, new_y - 1, new_z)
  if block_below != 44 and block_below != 49:
    block_below = mc.getBlock(new_x, new_y, new_z)
  relative_position = player_x - new_x
  white_notes = [60, 62, 64, 65, 67, 69, 71]
  black_notes = [61, 63, 0,  66, 68, 70]
  if block_below == 44:
    notes_along = relative_position // -3
    play_note(white_notes[notes_along])
  if block_below == 49:
    notes_along = ((relative_position - 1) // -3) - 1
    play_note(black_notes[notes_along])
```

Language
>PYTHON 3

FILE NAMES:
MC_piano_sound
piano.py

DOWNLOAD:
magpi.cc/
MinecraftMaker

MARC SCOTT

Marc is Head of Curriculum Development at the Raspberry Pi Foundation. He also likes pyrotechnics.
raspberrypi.org

MINECRAFT SELFIES

Smile! You're on camera and in the Minecraft world

You'll need

- A Raspberry Pi Camera Module
  magpi.cc/28ljlsz

In this tutorial you'll use the Pi Camera Module to take a selfie of yourself, and then with a bit of Python 3 code, you'll render the picture in a gigantic wall of Minecraft blocks.

STEP 1
Importing some modules

For this project you'll need to begin by importing a few modules. Most of them are pre-installed in Raspbian, but you need to install skimage yourself by opening the Terminal and entering:

```
sudo apt-get install
python3-skimage
```

Open Python 3 (IDLE) from the Menu. Create a new file by clicking on File > New File. Copy the code from the **minecraft_selfie.py** listing (page 25). For this, we're importing the picamera module to control the camera, and skimage to analyse the image. Save your file as **minecraft_selfie.py**.

STEP 2
Taking a selfie

The first stage is fairly simple. You're just going to use the Pi Camera Module to take a selfie.

Below the Python module imports we've set up the `camera` object, and set its resolution with the following two lines:

```
camera = PiCamera()
camera.resolution = (80,60)
```

You could use a larger resolution, but the code takes much longer to run, even on a Pi 3.

Next, the code starts a preview of the camera, waits a little bit, then captures an image which is saved as **selfie.jpg**. That's the first part of the script finished.

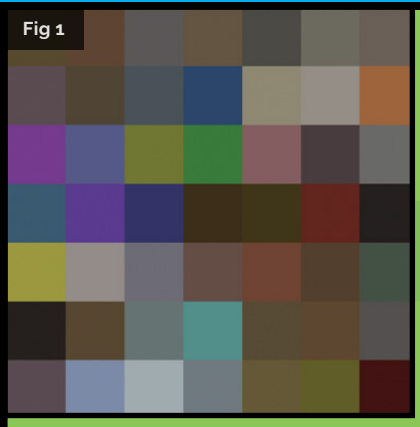
STEP 3
Mapping colours to blocks

Download the colour map (**magpi.cc/2pQJaHS**) and place it into the same directory as your Python script. The image is tiny

MINECRAFT WITH: RASPBERRY PI CAMERA MODULE

The official camera add-on board for the Raspberry Pi. It works well with Python, which means we can use it in Minecraft

Fig 1



(only 7 × 7 pixels), but we've enlarged it to show you what it looks like: **Fig 1**.

Each pixel on the colour map is the same average colour as a Minecraft block. The block in the top-left is dirt, for instance.

You need to load both this colour map and the selfie into the program, so that they become represented by lists of numbers. This is where the skimage module becomes useful:

```
selfie_rgb = io.imread("selfie.
jpg")
map_rgb = io.imread("colour_
map.png")
```

The variables become arrays which will be represented as something like this:

```
array([[[ 86,  74,  46],
        [ 93,  69,  49],
        [ 90,  87,  87],
        [ 99,  84,  65],
        [ 74,  73,  68],
        [108, 105,  95],
        [106,  95,  87]],
```

This is an extract of the representation of the colours in the colour map. The first row – 86, 74, 46 – represents the first pixel in the colour map. It's made up of three numbers: the first is the amount of red, the second the amount of green, and the third the amount of blue. Overall, this gives a brown colour. We call this RGB colour.

Now that you have the RGB values of the pixels in your selfie and the colours of the blocks from the colour map, if you could find the nearest colour from the map to the one in the selfie, you would know which block to place.

STEP 4

Finding the nearest colour

Now this is where it gets a bit complicated. Each colour is made up of three numbers, so you could plot the position of the colour on a graph. The colour R – 137, G – 164, B – 123 has been plotted on a 3D graph (**Fig 2**).

> "Each pixel on the colour map is the same average colour as a Minecraft block"

Now all the colours from the colour map can also be plotted on the same graph, using smaller points so you can still see the original colour (**Fig 3**).

It stands to reason that the closest dot in 3D space to the original colour would appear to be the closest colour visually. Unfortunately, this is not the case. While RGB values are useful for us when describing colours, they're not very useful for comparing colours. Have a look at **Fig 4** and **Fig 5**.

Fig 2

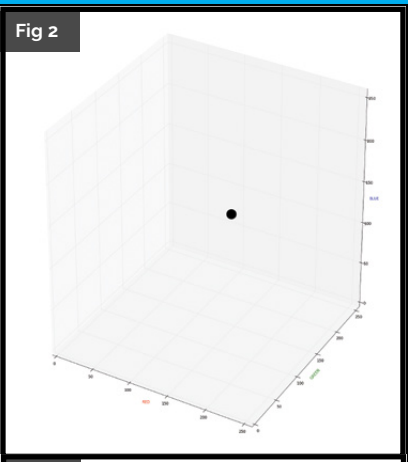


Fig 3

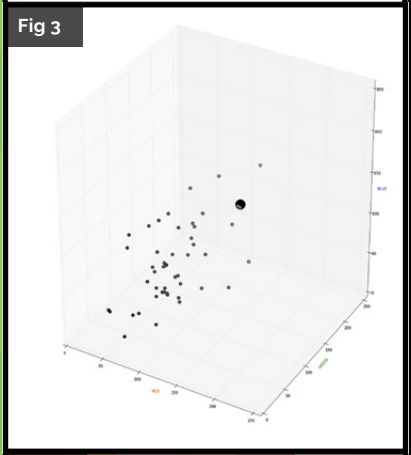


Fig 4

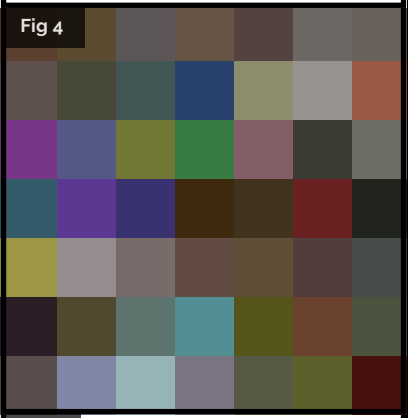
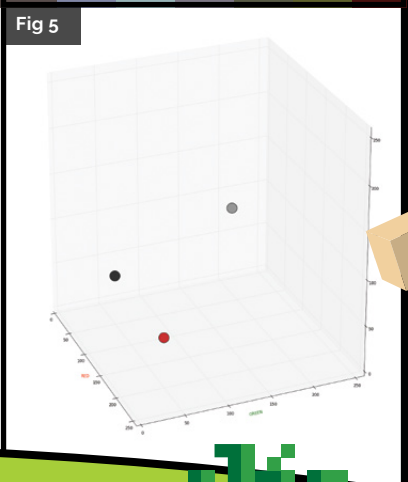


Fig 5



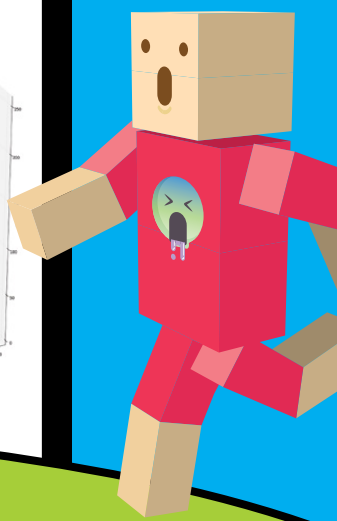


Fig 6

(0, 0): (2, 0),
(0, 1): (3, 0),
(0, 2): (4, 0),
(0, 3): (5, 0),
(0, 4): (7, 0),
(0, 5): (14, 0),
(0, 6): (15, 0),
(1, 0): (16, 0),
(1, 1): (17, 0),
(1, 2): (21, 0),
(1, 3): (22, 0),
(1, 4): (24, 0),
(1, 5): (35, 0),
(1, 6): (35, 1),
(2, 0): (35, 2),
(2, 1): (35, 3),
(2, 2): (35, 4),
(2, 3): (35, 5),
(2, 4): (35, 6),
(2, 5): (35, 7),
(2, 6): (35, 8),
(3, 0): (35, 9),
(3, 1): (35, 10),
(3, 2): (35, 11),
(3, 3): (35, 12),
(3, 4): (35, 13),
(3, 5): (35, 14),
(3, 6): (35, 15),
(4, 0): (41, 0),
(4, 1): (42, 0),
(4, 2): (43, 0),
(4, 3): (45, 0),
(4, 4): (46, 1),
(4, 5): (47, 0),
(4, 6): (48, 0),
(5, 0): (49, 0),
(5, 1): (54, 0),
(5, 2): (56, 0),
(5, 3): (57, 0),
(5, 4): (58, 0),
(5, 5): (60, 0),
(5, 6): (61, 0),
(6, 0): (73, 0),
(6, 1): (79, 0),
(6, 2): (80, 0),
(6, 3): (82, 0),
(6, 4): (89, 0),
(6, 5): (103, 0),
(6, 6): (246, 0)

Although the dark grey and light grey squares in **Fig 4** appear to be similar colours, **Fig 5** shows that they're actually 173 units apart. The light grey and dark grey dots are both closer to the red (150 units) than they are to each other. For this reason, comparing RGB values is not very useful, as colours that are close to each other in 3D space may visually appear to be very different.

STEP 5
Converting to Lab colour space

Because of this disparity in the 3D space, we convert the RGB values into what's known as Lab colour space. In Lab colour space, distance between colours in 3D space is very similar to our own perception of what could be called similar colours.

The skimage module makes conversion to Lab colour space from RGB colour space easy. You just need these two additional lines:

```
selfie_lab = color.
rgb2lab(selfie_rgb)
map_lab = color.
rgb2lab(map_rgb)
```

STEP 6
Mapping the blocks

The next part in the code involves mapping the pixels from the colour map to actual Minecraft blocks. A dictionary is used to do this.

Minecraft blocks have two values associated with them; for instance, dirt is 2, 0. The 0 is used as there's only one type of dirt block in Minecraft. Wool has many types with different colours, so wool can range from 35, 0 up to 35, 15.

The hard work has been done for you here. If you look at the table (**Fig 6**), you can find the pixel values from the colour map mapped to their corresponding Minecraft block.

STEP 7
Starting the Minecraft API

Now it's time to place the blocks. First we find the position of the player. Then comes the clever bit. You're going to iterate over all the colours in the `selfie_lab` first of all. To do this, you'll need the help of the `enumerate` function, which will keep track of your position in the selfie:

```
for i, selfie_column in
enumerate(selfie_lab):
    for j, selfie_pixel in
enumerate(selfie_column):
        distance = 300
```

These three lines will go over every pixel in the selfie and store each value of the pixel as `selfie_pixel`. The distance will also be set to 300, and the coordinates of each pixel will be saved as `i, j`.

Next, you need to iterate over every pixel in the colour map in the same way:

```
for k, map_column in
enumerate(map_lab):
  for l, map_pixel in
enumerate(map_column):
```

Now the distance between the colours of the pixels can be calculated:

```
delta = color.deltaE_
ciede2000(selfie_pixel,map_
pixel)
```

If the delta is less than the distance that was set before, then distance is reset to be the delta. The block can then be looked up from the dictionary of colours you set earlier:

```
if delta < distance:
  distance = delta
  block = colours[(k,l)]
```

Fig 7

Fig 8

Language
>PYTHON 3

DOWNLOAD:
magpi.cc/MinecraftMaker

Now out of that part of the loop, you can set the appropriate block. It's going to be set relative to the player's position, but quite high up in the air:

```
mc.setBlock(x-j, y-i+60, z+5, block[0], block[1])
```

Now try running the full code and see what happens. You might need to have a brief hunt around as the blocks are being laid, and be patient as it doesn't happen instantly. You'll get something like **Fig 7**.

STEP 8 (extra)

A better (but slower) algorithm

You can get a more accurate representation by using a different algorithm for calculating the delta value. This will be slower, but might give you a better result (so be very patient). Replace the line…

```
delta = color.deltaE_cie76(selfie_pixel,map_pixel)
```

…with the line:

```
delta = color.deltaE_ciede2000(selfie_pixel,map_pixel)
```

See the improvement in **Fig 8**.

MINECRAFT PHOTOBOOTH

Want to do more with cameras and Minecraft? There's a slightly different project you can try called the Minecraft photobooth. In it, you program Minecraft so that whenever Steve enters a photobooth in the Minecraft world, it takes a picture with the camera in real life. Check it out here: magpi.cc/2pkDgLF

minecraft_selfie.py

```
from picamera import PiCamera
from mcpi.minecraft import Minecraft
from time import sleep
from skimage import io, color

## Taking a picture

camera = PiCamera()
camera.resolution = (80,60)
camera.start_preview()
sleep(15)
camera.capture('selfie.jpg')
camera.close()

## Rendering the picture

### load selfie and map
selfie_rgb = io.imread("selfie.jpg")
map_rgb = io.imread("colour_map.png")

### Convert to Lab

selfie_lab = color.rgb2lab(selfie_rgb)
map_lab = color.rgb2lab(map_rgb)

### Mapping colours on colour map to Minecraft blocks
### First tuple is coordinates of colour map
### Second tuple is Minecraft block

colours={(0,0):(2,0),(0,1):(3,0),(0,2):(4,0),(0,3):(5,0),(0,4):(7,0),
(0,5):(14,0),(0,6):(15,0),(1,0):(16,0),(1,1):(17,0),(1,2):(21,0),(1,3):
(22,0),(1,4):(24,0),(1,5):(35,0),(1,6):(35,1),(2,0):(35,2),(2,1):
(35,3),(2,2):(35,4),(2,3):(35,5),(2,4):(35,6),(2,5):(35,7),(2,6):
(35,8),(3,0):(35,9),(3,1):(35,10),(3,2):(35,11),(3,3):(35,12),(3,4):
(35,13),(3,5):(35,14),(3,6):(35,15),(4,0):(41,0),(4,1):(42,0),(4,2):
(43,0),(4,3):(45,0),(4,4):(46,0),(4,5):(47,0),(4,6):(48,0),(5,0):
(49,0),(5,1):(54,0),(5,2):(56,0),(5,3):(57,0),(5,4):(58,0),(5,5):
(60,0),(5,6):(61,0),(6,0):(73,0),(6,1):(79,0),(6,2):(80,0),(6,3):
(82,0),(6,4):(89,0),(6,5):(103,0),(6,6):(246,0)}

## Iterate over image and then over map. Find closest colour from map,
and then look up that block and place

mc = Minecraft.create()
x, y, z = mc.player.getPos()

for i, selfie_column in enumerate(selfie_lab):
  for j, selfie_pixel in enumerate(selfie_column):
    distance = 300
    for k, map_column in enumerate(map_lab):
      for l, map_pixel in enumerate(map_column):
        delta = color.deltaE_cie76(selfie_pixel,map_pixel)
        if delta < distance:
          distance = delta
          block = colours[(k,l)]
    mc.setBlock(x-j, y-i+60, z+5, block[0], block[1])
```

JOSHUA LOWE

Josh is a 13-year-old in the North West of England who enjoys developing software and designing products to help others.
edublocks.org / @all_about_code

PROGRAMMING MINECRAFT WITH EDUBLOCKS

Build your own Minecraft treasure hunt with an exciting new program called EduBlocks – an easy way to transition from Scratch to Python

MINECRAFT WITH: EDUBLOCKS

EduBlocks is a way to bridge the gap between Scratch and Python, using blocks of Python code to make your project.

Josh created EduBlocks as a way for children of any ability to write Python code in a simple block editor similar to Scratch. The goal of the project is to make the transition from Scratch to Python easier for students and teachers, as presently there is no drop-in solution that bridges this gap.

EduBlocks started life 15 months ago as a way to help teachers to do more in class and help children to explore the world of the Raspberry Pi via a simple-to-use interface. EduBlocks is a success thanks to the help of the wonderful Raspberry Pi community who have provided input and resources to the project.

In this tutorial we'll create a game where three diamonds are hidden around the Minecraft world. Do you think that you can you find all three?

We start the project by installing EduBlocks onto our Raspberry Pi.

To do this, you will need to open up a Terminal on your Raspberry Pi and type the following command:

```
curl -sSL get.edublocks.org | bash
```

To begin your adventure with EduBlocks, we need to start the application. There is a handy shortcut on the desktop that you can double-click to start EduBlocks. There is also a link in the Programming menu which can be used to launch the application.

Blocks used to write Python

EduBlocks will take a little while to load but once it has, you'll be presented with the EduBlocks user interface, which consists of a large workspace area. This is where we will place our code to develop the game. The code blocks used to build up our project are found on the left-hand side of the screen: simply drag and drop the blocks onto the workspace to create the sequence of code for our game. We shall be working in the block view, but at any time we can switch to the Python View so we can see the Python code that's been created using the blocks.

Controlling Minecraft

STEP 1

Import libraries

In EduBlocks we can import libraries in the same way that Python handles libraries. To import the 'minecraft' library, go to the Minecraft menu, click on General and drag the **`from mcpi.minecraft import Minecraft`** block into the workspace. Now, from the same menu, drag across the **`mc = Minecraft.create()`** block and snap it underneath the previous block. Next, we drag the **`import math`** block from the Basic menu and snap it underneath the previous block.

```
from mcpi.minecraft import Minecraft
mc = Minecraft.create()
import math
```

Creating diamonds

STEP 2

Set up the blocks

Using **`mc.setBlock[x],[y],[z],[i]`** from the Minecraft > Commands menu, we shall set the position of the first diamond in our hunt. The position of the Diamond block is set by the x,y,z coordinate; the block type, 57, refers to the diamond block.

Next, we create three variables called **one**, **two**, and **three**. To do so, we go to the Basic menu and scroll down until we see the **`[0] = [0]`** block; this is used to represent a variable. Drag this block to the workspace three times and change the contents of the block so that we have three variables – **one**, **two**, and **three** – and the **`distance_to_player`** section is completed as shown in the screenshot.

```
mc.setBlock( 50 , 25 , 11 , 57 )
one = distance_to_player(50,25,11)
two = distance_to_player(70,10,25)
three = distance_to_player(20,35,43)
```

Here we can see how far away the diamond is. This is posted to the Minecraft chat window as we move around the world

STEP 3

Create your first loop

From the Basic menu, we'll use a **`while`** loop. Drag this to the workspace and snap it under the previous blocks. In the blank area for the loop, we shall create a condition that will run the loop while the player's distance is greater than five blocks from the diamond at position **one**, controlled by the variable that we have just created.

Now we use another variable block which updates the player's position in relation to the diamond. We use the **`mc.postToChat(" ")`** block from the Minecraft >> Commands menu to inform the player where to look. The distance is calculated from the variable, in this case **one**, and we round the value returned to one decimal place using the **`round`** function. Otherwise the value returned is rather long. We then convert the integer into a string in order to join it to the message.

```
while one>5 :
    one = distance_to_player(50,25,11)
    mc.postToChat(" Distance: " +str(round(one))+ " blocks away ")
```

STEP 4

Placing our second diamond

For the second diamond we also use **`mc.setBlock[x],[y],[z],[i]`** from the Minecraft > Commands menu. But this time we update the x,y,z coordinates so that the diamond appears elsewhere in the world. In order to update the player's position in relation to the second diamond, we need to place another variable block. This block will update the **two** variable, and also update the chat window to help guide the player to the new block.

```
mc.setBlock( 70 , 10 , 25 , 57 )
while two>5 :
    two = distance_to_player(70,10,25)
    mc.postToChat(" Distance: " +str(round(two))+ " blocks away ")
```

STEP 5

Placing our final diamond

In this final step, we will create a third diamond using exactly the same blocks and logic as we have used to construct the two other diamonds. Do you think that you can hack the game to have the diamonds pop up in other positions in the world?

Remember to save your work by clicking on the Save button in the top right corner of the screen before testing your code.

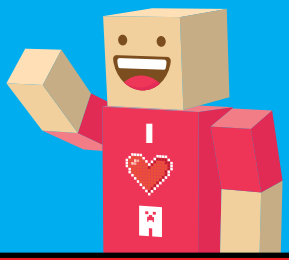
To start the game, click on the Run button located in the top right corner and then ensure that the Minecraft window is visible. You can now hunt for those elusive three diamonds in the Minecraft world. Can you find them all?

```
mc.setBlock( 20 , 35 , 43 , 57 )
while three>5 :
    three = distance_to_player(20,35,43)
    mc.postToChat(" Distance: " +str(round(three))+ " blocks away ")
```

MEHDI IMANI MASOULEH

Electronics engineer and one of the original founders of Piper, a Raspberry Pi-based portable computer that teaches electronics through Minecraft!
buildpiper.com

CHANGE MINECRAFT SKINS WITH RFID CARDS



This project allows you to change the Minecraft Pi character by placing different RFID cards on an RFID detector

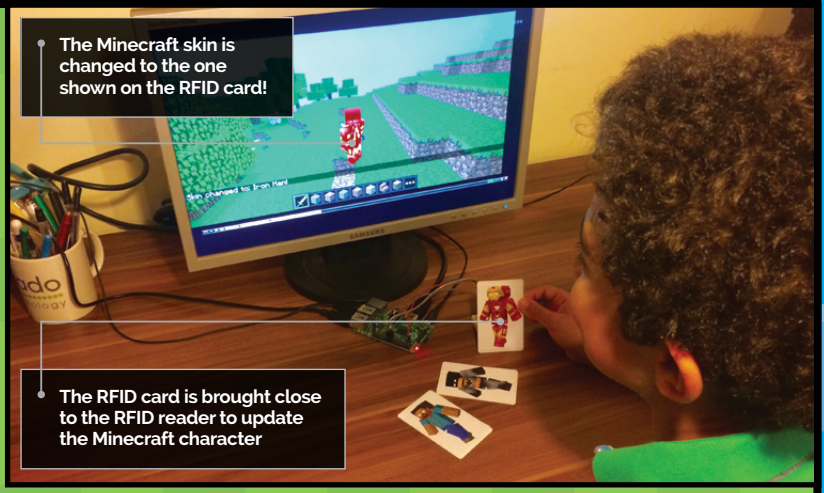
MINECRAFT WITH: RFID

Radio-frequencey identification, or RFID, allows computers to recognise code via radio waves. It's used in contactless technology

You'll need

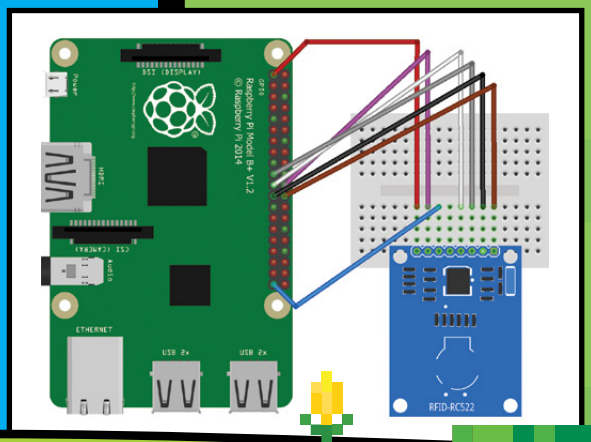
- 7× male-to-female jumper wires
- Small breadboard
- RFID-RC522 module
- A few Mifare cards
- 8-way right-angled pin

Minecraft on the Raspberry Pi comes preset with the main character skinned as Herobrine. In this tutorial we will explain how Minecraft Pi can be hacked to allow the skin to be changed to your own favourite character using RFID cards. After building a simple circuit and doing some software work, you can then touch the reader module with your customised RFID card, allowing you to instantly change the Minecraft character.



STEP 1
Set up circuit

We begin by connecting the reader module RFID-RC522 to the Raspberry Pi. The eight-way pin needs to be soldered to the RFID reader if it does not come pre-soldered. Insert the female ends of male-to-female jumper wires onto the Raspberry Pi GPIO pins and the male ends into the breadboard, as shown on the diagram.



STEP 2
Set up RFID on Pi

To facilitate communication between the RFID module and the Pi, we first have to configure our Raspberry Pi so that Serial Peripheral Interface (SPI) is enabled. The SPI hardware Python library needs to be installed. All these steps are explained in this online guide: **magpi.cc/28LleQN**. An RFID library is also required to make the communication with the RFID module easier, but this is included in the project folder.

STEP 3
Download the skins

A small set of skins is included in the project folder. More skins can be found on **minecraftskins.net**. Download your favourite characters onto your Raspberry Pi. Remember where they are so you can put them into the correct folder in step five.

STEP 4

Make your card character

You can either draw your chosen Minecraft characters on a sheet of paper, or download character images from **minecraftskins.net** and print them. Cut your character out and use a glue stick to stick it to the Mifare card. Do this for as many characters as you want.

STEP 5

Get the code

Type the following command to install the xdotool app, which lets you perform window management tasks using shell commands:

```
sudo apt-get install xdotool
```

This will be used in the code to refresh the Minecraft window so the skin is automatically updated. Download the project from GitHub (**magpi.cc/2pgdXio**) and place it in a new project folder. Move the skins you saved earlier into the **skins** folder of the project. Run the Python file **Read.py** by typing **`sudo python Read.py`**, and place each of the Mifare cards on the RFID reader. This should test the circuit setup and give you the cards' unique IDs. Replace the card IDs in the code, with the IDs read by running **Read.py**. Replace the **`skinFile`** variable in **charMinecraft.py** with the corresponding skin file names from the **skins** folder.

STEP 6

Get ready to play!

Launch Minecraft and run the updated Python code by typing **`sudo python charMinecraft.py`** in the Terminal. Make sure the game is in third-person view by pressing the **ESC** button in the game and then clicking on the third-person view button. Place the Mifare card on the RFID reader to change the character. Once the card is placed, an on-screen message should say which character you are changing to. The skin is then updated and you can continue to play your favourite game with your favourite character!

charMinecraft.py

Language
>PYTHON

FILE NAMES:
charMinecraft.py

DOWNLOAD:
magpi.cc/2pgdXio

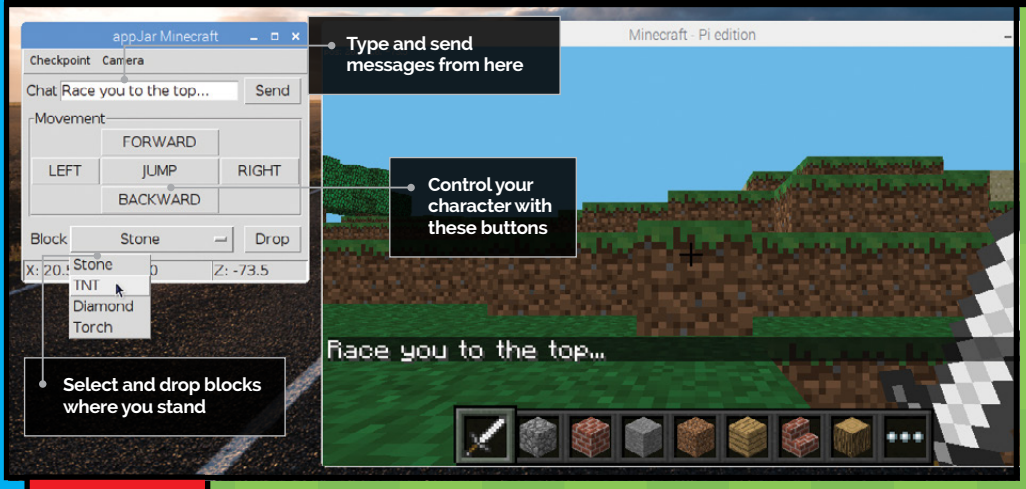
```
import RPi.GPIO as GPIO
import MFRC522
import signal
import mcpi.minecraft as minecraft
import time,os

# Capture SIGINT for cleanup when the script is aborted
def end_read(signal,frame):
  global continue_reading
  print "Ctrl+C captured, ending read."
  continue_reading = False
  GPIO.cleanup()

#Replace skin file names here
skinFile=['ironman','default','batman','pig']
skinNames=['Iron Man', 'Herobrine', 'Batman']
idx=1;
winSizeX=1800 #set minecraft window size
winSizeY=800

#create minecraft connection
mc = minecraft.Minecraft.create()

continue_reading = True
#Replace the card IDs here
UIDs=['160,41,83,122','144,24,1,118','176,221,21,124']

# Hook the SIGINT
signal.signal(signal.SIGINT, end_read)

# Create an object of the class MFRC522
MIFAREReader = MFRC522.MFRC522()
i=0;

print "Press Ctrl-C to stop."

# This loop keeps checking for cards.
while continue_reading:

  # Scan for cards
  (status,TagType) = MIFAREReader.MFRC522_Request(MIFAREReader.PICC_REQIDL)

  # If a card is found
  if status == MIFAREReader.MI_OK:
    print "Card detected"

  # Get the UID of the card
  (status,uid) = MIFAREReader.MFRC522_Anticoll()

  # If we have the UID, continue
  if status == MIFAREReader.MI_OK:
    # Print UID
    print "Card UID: "+str(uid[0])+","+str(uid[1])+","+\
    str(uid[2])+","+str(uid[3])
    uid_str=str(uid[0])+","+str(uid[1])+","+str(uid[2])+","+str(uid[3]);

  try:
    idx=UIDs.index(uid_str)
    os.system('cp skins/'+ skinFile[idx]
      +'.png //home/pi/mcpi/data/images/mob/char.png')
    mc.postToChat('Skin changed to: '+skinNames[idx]+'!')
    i=i+1; #refresh window
    os.system("xdotool search --name 'Minecraft - PI'  windowsize "
      + str(winSizeX)+ ' ' +str(winSizeY+i%2))
    except ValueError:
      print("Oops! Not in the list")
```

RICHARD JARVIS

Computer science teacher with a passion for making and breaking things. Author of appJar.
appJar.info

BUILD A MINECRAFT PYTHON GUI

Create a simple Python GUI, using appJar, to control your Minecraft world



You'll need

- appJar
  appJar.info

We all know how easy it is to control Minecraft from Python, but did you know it's just as easy to create a GUI to do the same thing? With appJar, and a few lines of code, you can create a simple GUI to control your Minecraft world.

STEP 1
Setting up

First you'll need to install appJar. Open up a Terminal and type **`sudo pip3 install appJar`**. Now you're ready to go – open IDLE (for Python 3), and let's get coding.

MINECRAFT WITH: APPJAR

The appJar library is a way to create graphical interfaces using Python code. It's very easy to use, as you can see from our script.

Code order is important: first we'll import the Minecraft and appJar libraries, then we'll group together all our functions. Last, we'll write our GUI code. The GUI code is always split into three parts: creating the GUI, adding and configuring widgets, then starting the GUI. Don't put any code after that, as it won't run until you close the GUI.

STEP 2
Let's chat

To start with, we'll add an an entry box and a button, to send a chat message. AppJar puts widgets in a grid, so we'll put both widgets in row 0, columns 0 and 1.

Then, we'll need a function to call when the button is pressed – it'll get the text from the entry box and send it to Minecraft. The second parameter, when adding a button, is the name of the function to call when the button is pressed.

STEP 3
Moving around

Next, we'll add some movement buttons. These will simply change our x, y, or z coordinate.

We're going to group the buttons in a **`LabelFrame`**, so we'll put it in the next row, and tell it to span both columns. Then we position the buttons inside the **`LabelFrame`** – it has its own grid, so we can position the buttons just like before.

We'll link all of these buttons to a new function. The name of the button is passed as a parameter to the function, so we can use an **`if`** statement to work out which button was pressed and change the right coordinate.

STEP 4
Status updates?

Everyone likes updating their status, and appJar makes it really simple. We're going to create a status bar, and have it show our current location.

We want the status bar to keep updating while we play, but we can't use a loop as it would stop the

GUI from working. Instead, we'll create a function to update the status bar, then tell appJar to put it in a loop for us.

STEP 5

Dropping blocks

We want to make dropping blocks as simple as possible, so we'll put an option box to choose a block from, and a button drop it.

The function connected to this button will simply check which block has been selected, find its ID, and tell Minecraft to position that block next to our character.

We'll need to create a dictionary of our favourite blocks. The key will be a user-friendly name, and the value will be the appropriate ID.

STEP 6

Anything from the menu?

Finally, let's add a few menu options, to make this feel like a proper application...

We'll have one menu for creating and restoring checkpoints, and then make another menu for changing the camera angle.

These work just like buttons. We'll link all the menus to the same function, check the parameter to see which menu was clicked, and then do the specified action. We'll even add in a couple of dialogs, to give some feedback on the menu choices.

code_5.py

Language
>PYTHON

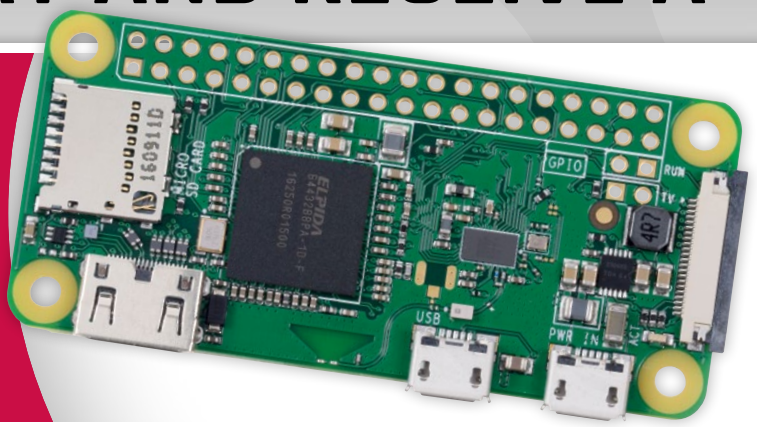
FILE NAMES:
code_5.py

DOWNLOAD:
magpi.cc/MinecraftMaker

```
# import libraries
from appJar import gui
from mcpi.minecraft import Minecraft

# connect to Minecraft
mc = Minecraft.create()

# 1 - CHAT FUNCTION
def sendChat(btn):
    msg = app.getEntry("Chat")
    mc.postToChat(msg)

# 2 - MOVEMENT FUNCTION
def move(btn):
  x, y, z = mc.player.getPos()
  if btn == "LEFT":
    x -= 1
  elif btn == "RIGHT":
    x += 1
  elif btn == "FORWARD":
    z -= 1
  elif btn == "BACKWARD":
    z += 1
  elif btn == "JUMP":
    y += 1
    z -= 1

  mc.player.setPos(x, y, z)

# 3 - STATUS FUNCTION
def updateStatus():
  x, y, z = mc.player.getPos()
  app.setStatusbar("X: " + str(x), field=0)
  app.setStatusbar("Y: " + str(y), field=1)
  app.setStatusbar("Z: " + str(z), field=2)

# 4 - BLOCKS FUNCTION
BLOCKS = {"Stone": 1, "TNT": 46, "Torch": 50, "Diamond": 57}
def drop(btn):
  x, y, z = mc.player.getPos()
  z = z - 1
  height = mc.getHeight(x,z)

  playerBlock = app.getOptionBox("Block")
  blockId = BLOCKS[playerBlock]

  mc.setBlock(x, height, z, blockId)

# 5 - MENU FUNCTION
def clickMenu(choice):
  if choice == "Create":
    mc.saveCheckpoint()
    app.infoBox("Save", "Checkpoint saved.")
  elif choice == "Restore":
    if app.yesNoBox("Restore", "Are you sure?"):
      mc.restoreCheckpoint()
  elif choice == "Normal":
    mc.camera.setNormal()
  elif choice == "Fixed":
    mc.camera.setFixed()
  elif choice == "Follow":
    mc.camera.setFollow()

# create the GUI - must come first
app = gui("appJar Minecraft")
app.setLocation(100,100)

# 1 - CHAT WIDGETS
app.addLabelEntry("Chat", row=0, column=0)
app.addButton("Send", sendChat, row=0, column=1)

# 2 - MOVEMENT WIDGETS
app.startLabelFrame("Movement", row=1, column=0, colspan=2)
app.setSticky("NESW") # make buttons stick to all sides
app.addButton("FORWARD", move, row=0, column=1)
app.addButton("LEFT", move, row=1, column=0)
app.addButton("JUMP", move, row=1, column=1)
app.addButton("RIGHT", move, row=1, column=2)
app.addButton("BACKWARD", move, row=2, column=1)
app.stopLabelFrame()

# 3 - STATUS WIDGETS
app.addStatusbar(fields=3)
app.registerEvent(updateStatus) # call updateStatus in a loop

# 4 - BLOCKS WIDGETS
app.addLabelOptionBox("Block", list(BLOCKS), row=2, column=0)
app.addButton("Drop", drop, row=2, column=1)

# 5 - MENU WIDGETS
app.addMenuList("Checkpoint", ["Create", "Restore"], clickMenu)
app.addMenuList("Camera", ["Normal", "Fixed", "Follow"], clickMenu)

# start the GUI - must come last
app.go()
```

SUBSCRIBE TODAY AND RECEIVE A

FREE PI ZERO W



PLUS OFFICIAL PI ZERO CASE WITH 3 COVERS

AND FREE CAMERA MODULE CONNECTOR AND USB / HDMI CONVERTER CABLES

Subscribe in print for 12 months today and receive:

- A free Pi Zero W (the latest model)
- Free Pi Zero W case with three covers
- Free Camera Module connector
- Free USB and HDMI converter cables

Other benefits:

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

Pricing

Get six issues:

£30 (UK)

£45 (EU)

$69 (USA)

£50 (Rest of World)

Subscribe for a year:

£55 (UK)

£80 (EU)

$129 (USA)

£90 (Rest of World)

FREE PI ZERO W!

Get three issues:

£12.99 (UK) **(Direct Debit)**

$37.50 (US) **(quarterly)**

How to subscribe:

- **magpi.cc/Subs-2** (UK / ROW)
- **imsnews.com/magpi** (USA)
- Call +44(0)1202 586848 (UK/ROW)
- Call 800 428 3003 (USA)

Available on the App Store

GET IT ON Google Play

SUBSCRIPTION FORM

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

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

Mag#58

YOUR DETAILS Mr ☐ Mrs ☐ Miss ☐ Ms ☐

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

Address .....

.....

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

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

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

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

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

Address .....

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

PAYMENT OPTIONS

1 DIRECT DEBIT PAYMENT £12.99 every 3 issues (UK only)

Instruction to your bank or building society to pay by Direct Debit

DIRECT Debit

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

Service user number 8 3 8 7 7 3

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

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

Address .....

.....

..... Postcode .....

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

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

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

Instruction to your bank or building society

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

Signature ..... Date ☐☐/☐☐/☐☐

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

SUBSCRIPTION PRICING WHEN PAYING BY CHEQUE OR CREDIT/DEBIT CARD

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

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

2 CHEQUE

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

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

Card number ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

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

Issue number ☐☐ (if shown) Security number ☐☐☐

(last 3 digits on the back of the card)

Signature ..... Date ☐☐/☐☐/☐☐

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

RETURN THIS FORM TO:

MagPi Magazine Subscriptions, Select Publisher Services Ltd, PO Box 6337, Bournemouth BH1 9EH

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

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

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

JOEY MEYER
Joey is an experienced software engineer with an extensive background building iOS apps, and a deep understanding of the fundamental concepts in machine learning.
raspberryturk.com

Quick Facts

- The Turk takes 20-40 seconds to make a move
- It makes use of the Stockfish chess engine
- The chessboard is spray-painted onto the table
- Three 100W LED floodlights illuminate the board
- A daemon process on the Pi handles all the software

RASPBERRY TURK

The amazing chess-playing robot with a Raspberry Pi hidden inside

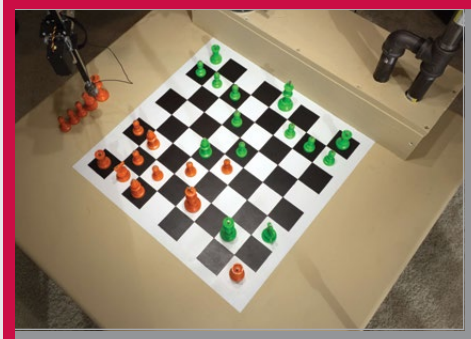
The chess player ponders the next move. Suddenly, a mechanical arm whirs into action, moves over the board, lowers an electromagnet, and picks up a piece… Checkmate! Joey Meyer's Raspberry Turk (**raspberryturk.com**) is an ingenious chess-playing robot that was inspired by the eighteenth-century 'Mechanical Turk'. While the latter machine had a human player concealed inside to determine its moves, the Raspberry Turk uses a Raspberry Pi 3 as its brain.

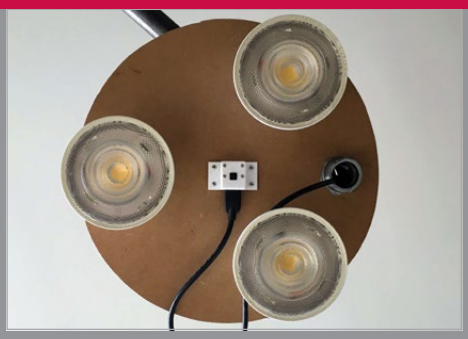
"My co-worker introduced me to the eighteenth-century Turk years ago and I was always fascinated by it," Joey tells us. "A couple years ago I read *The Turk: The Life and Times of the Famous Eighteenth-Century Chess-Playing Machine* by Tom Standage, and loved it. After spending time learning computer vision and machine learning last year, I began looking for a project that would allow me to use what I had learnt. I made the connection and decided it would be a fun and challenging project."

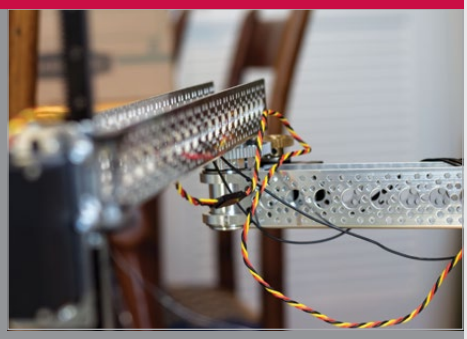
Joey says the hardware was the hardest part of the project. "I am a software engineer, and building


The gripper mechanism works by activating an electromagnet at the end of a vertically moving beam

BUILD A CHESS ROBOT







>STEP-01
Make a table
The Turk is built into a small 3×3 ft (91×91 cm) table. A box on one side houses all the electronics, while the robotic arm is mounted on top.

>STEP-02
View the board
To evaluate the positions of the pieces, a top-mounted Pi Camera Module captures a view of the board which is then perspective-transformed using OpenCV.

>STEP-03
Move the arm
The arm's motion is controlled by the rotation of two servos attached to gears at the base of each link. Another servo controls the gripper mechanism.

hardware is a very different process." One difficulty encountered was in interfacing Dynamixel servos with the Acrobotics components that make up the robot arm. "This gave me the opportunity to use 3D printing to build components to solve this problem."

I started this so I could use material I had learnt in a real project, but documenting the build process gave me the opportunity to help others learn, too."

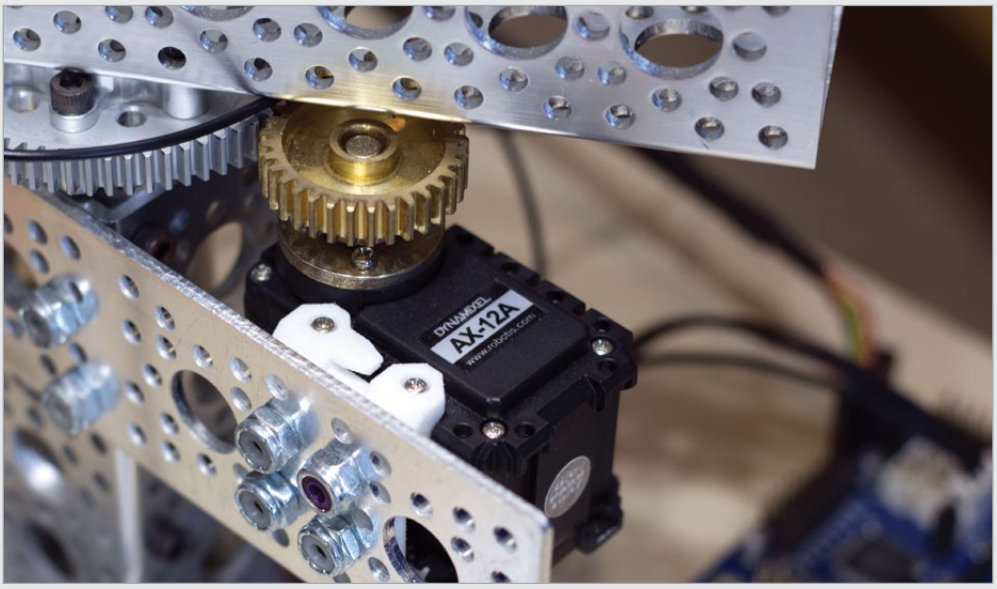
Asked how difficult would it be for other makers to replicate, Joey replies: "If you're comfortable with some electronics, programming, math, and some simple handiwork, this project would be a big challenge, but is definitely doable. The code I wrote for mine is freely available. The website explains how everything works in detail and I am happy to answer questions for anyone who wants to take on the challenge. Several people have already reached out, telling me that they are working on building their own!"

> "Another challenge was making the arm movements precise"

Another challenge was making the arm movements precise. "In a perfect system, the movement of the arm could be modelled by a simple math equation, but due to inaccuracies in measurements, and unexpected real-world effects, this simple math equation model broke down. It did well, but not well enough to consistently grab the piece every time." Joey solved the issue by collecting a dataset of arm movements to see where the model was having problems. "The results worked well and the arm can move very accurately now."

The Raspberry Turk took Joey around five months to build and the process has been fully documented on his site. "I knew I wanted to open-source the robot and describe the build process on a website from the beginning.

While the Turk hasn't been showcased in public yet, Joey says the response of those who have seen it has been interesting. "Reactions are usually positive, but then they quickly change to shock when they realise the robot isn't just playing them, it's beating them – badly!



Below The arm's Dynamixel AX-12A servos are controlled by the Pi via an ArbotiX-M Robocontroller

PIET RULLENS JR

Engineer and business owner by day; Daily Prophet reporter Rita Skeeter by night.
magpi.cc/2qvNA8i

HARRY POTTER & THE DAILY PROPHET

When muggle newspapers simply weren't engaging enough, **Piet Rullens Jr** decided to create an animated fan build of the iconic Harry Potter tabloid

Quick Facts

- The Wizarding World of Harry Potter is in Orlando, Florida
- Piet took the idea from the moving tabloid in the books and movies
- A hidden lead keeps the frame powered and ready for action
- A sepia effect gives the video an added layer of authenticity
- Despite popular belief, Neville was the chosen one, not Harry

When Piet and Linda Rullens took a trip to The Wizarding World of Harry Potter in Orlando, Florida, they made sure to bring back a memory card's worth of holiday footage. But what do you do with holiday video footage once you're home? Taking his inspiration from the fictional world that forged the destination for their vacation, Piet decided to create his own Daily Prophet newspaper, complete with moving images.

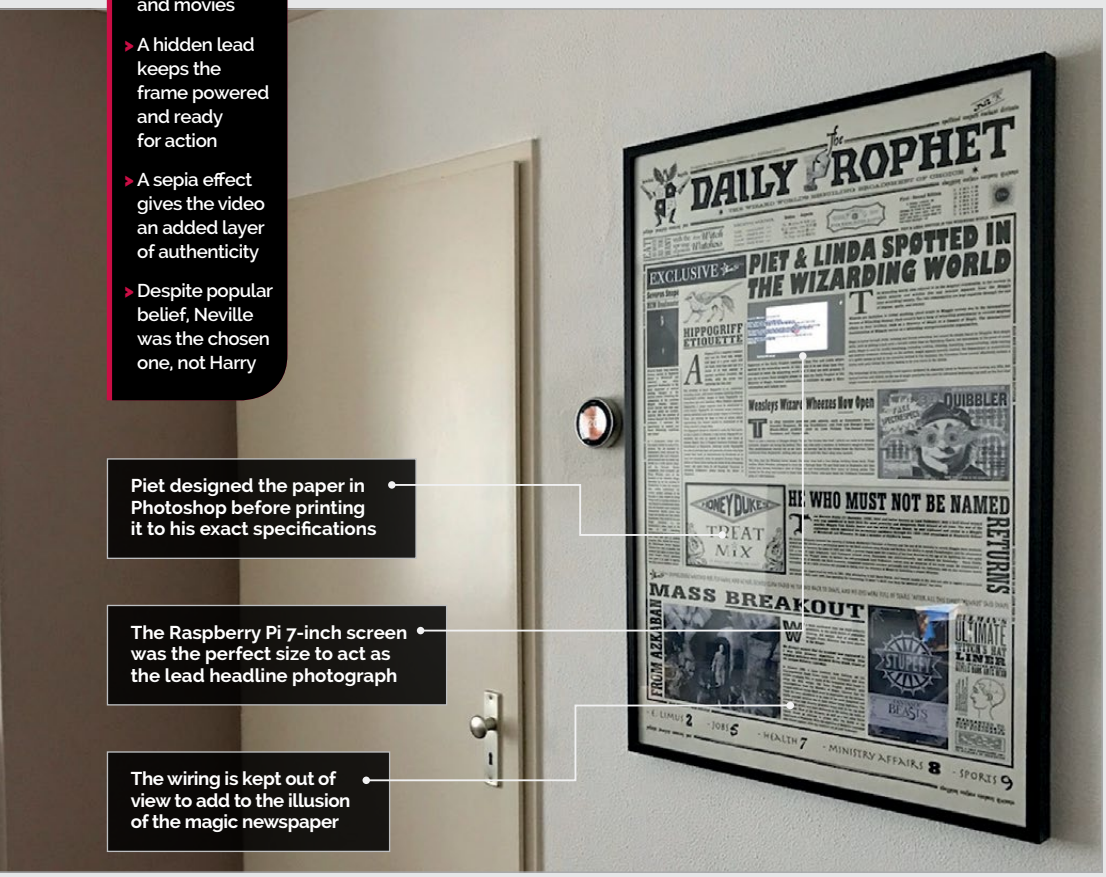
For those unfamiliar with Harry Potter, the Daily Prophet is the main newspaper publication of the wizarding world – the only rival being the often fantastical Quibbler peddled by Luna Lovegood and her father. Similar in function to that of a 'muggle', non-magical newspaper, the Daily Prophet shares the headlines of the world – with one major difference… the images move. Imagine using an animated GIF in a news blog, but on paper.

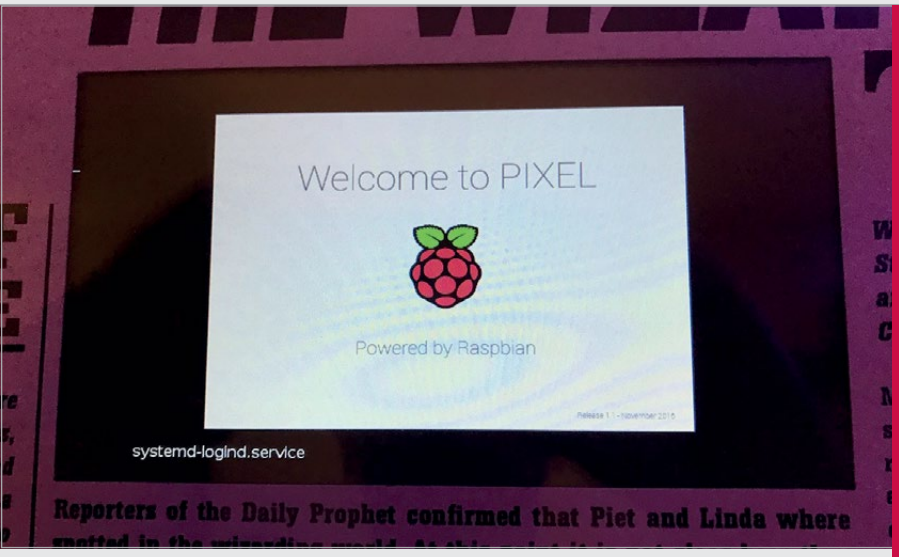
With his videos to hand, Piet set about creating the newspaper by designing the front cover in Adobe Photoshop. Not only did this enable him to include personal references in the copy, such as mention of himself and his wife being spotted at the theme park, but it also allowed him to create the perfect-sized window for the Raspberry Pi 7-inch display that he was to fit within the frame. "First, I designed the whole poster in Photoshop. Within the design, I marked an area with the exact size of the Raspberry Pi screen. Next, I plotted the poster on normal paper at 100 percent, so the marked area still matched the Raspberry Pi screen."

From there, Piet marked out the measurements of the screen onto the hardboard of a poster frame, giving him a guide to cut through for the additional electronics.

Luckily for Piet, on the other side of the wall from where he would be hanging the frame was a small cupboard. He was able to drill directly through the wall, hiding any wires from view, and adding to the magical illusion of the piece.

With on-board wireless connectivity on the Pi, the only wire needed was the USB power cable.





Above Both the glass of the frame and the bevel of the screen keep it fitted snugly within the newspaper, with no need for glue or screws

With this firmly in place inside the cupboard, Piet was able to remotely access the Pi and create the code to run his holiday footage.

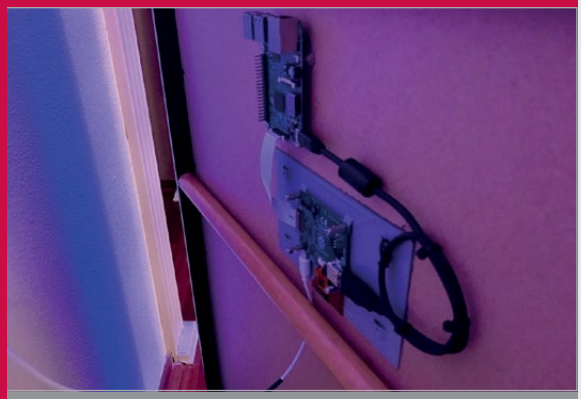
Piet created a simple Python script with two functions. The first detects the presence of someone passing by, and the second then runs the holiday footage on the screen. To complete the first task, he used an IR distance sensor from Adafruit. This would detect motion within a set range around the photo frame. The motion sensor then triggers Omxplayer to play five minutes of footage before turning the screen off again.

To add to the look and feel of the newspaper, Piet edited the footage to give it a grainer, sepia tone in line with the movie prop. He converted the footage to H.264 so that it played through the Raspberry Pi, creating a beautifully executed and impressive magical holiday souvenir.

> "Within the design, I marked an area with the exact size of the Raspberry Pi screen"

Added texture and colour effects give the video the appearance of a Daily Prophet image

CODING MAGIC



>STEP-01

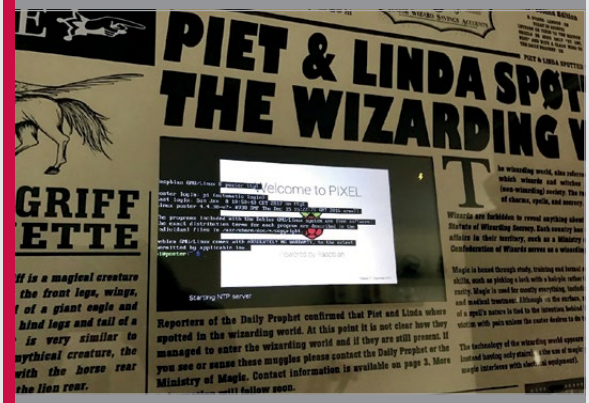
Behind the scenes

The Raspberry Pi, screen, and wiring fit perfectly inside the recess of the frame, with a hole in the wall leading to a power supply on the other side.

>STEP-02

Look, no wires!

By cleverly diverting the power cable through the wall, Piet adds an extra level to the magical illusion of the piece.



>STEP-03

Spotted!

The Python script and an IR sensor control the screen and the duration of the video: approach the frame and the system begins to play.

SET UP A FILE SERVER

Turn your Raspberry Pi into a file server to back up and share content from anywhere on your local network

You'll Need

- A 32GB micro SD card
- Raspberry Pi 2/3
- Monitor, keyboard and mouse (for setup)
- Wired Ethernet connection
- NOOBS **magpi.cc/2bnf5XF**

It's easy to use a Pi as a simple file server where you can store backups and share files from all the other computers on your network. Samba is the Linux implementation of the SMB/CIFS file sharing standard used by Windows PCs and Apple computers, and widely supported by media streamers, games consoles and mobile apps.

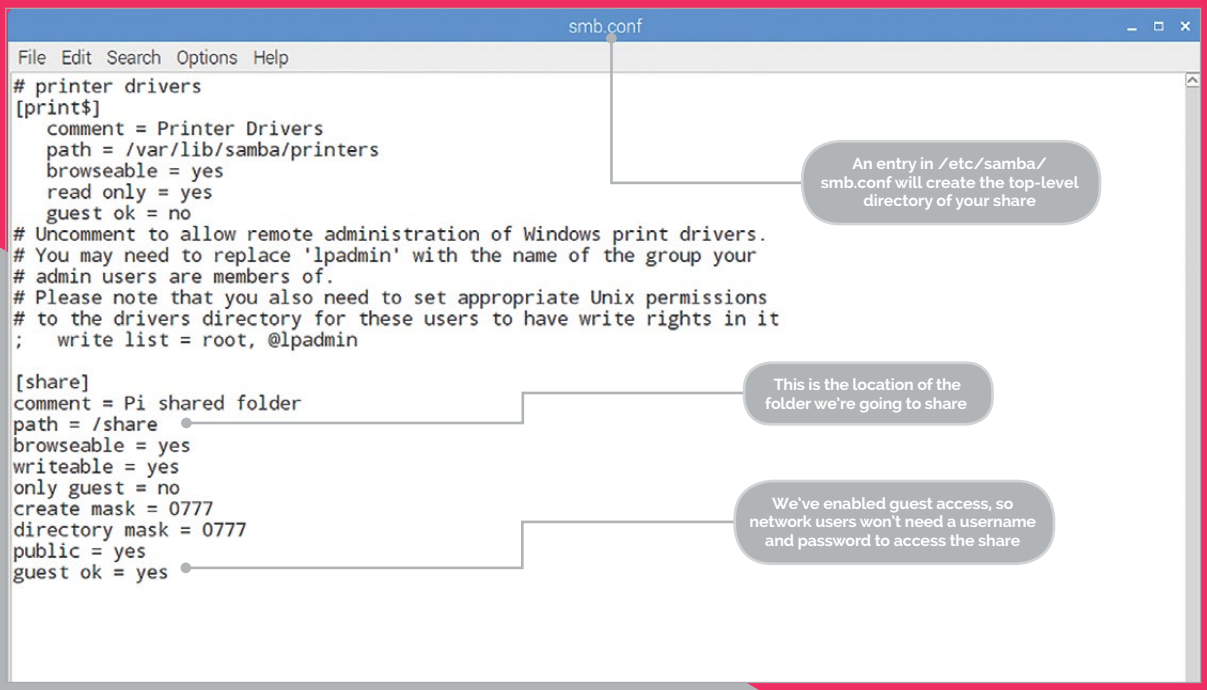
This tutorial assumes that you'll use a keyboard, mouse, and monitor to set up your file server, but you can alternatively enable SSH (**magpi.cc/1GULmTr**) and connect to it remotely from another computer on your local network.

We also assume you're using a 32GB (or smaller) micro SD card, which provides a reasonable amount of storage space without requiring any extra steps to make it accessible. However, if you need extra storage, it's easy to mount a large external USB drive and create a Samba entry for it.

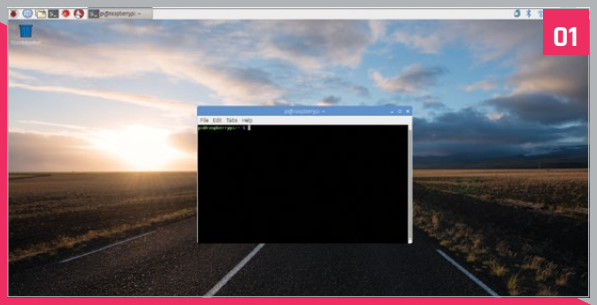
Alternatively, if you want to keep things compact, you can install Raspbian on micro SD cards of up to 256GB, although we suggest checking online (non-working SD cards: **magpi.cc/2q97aGO**) before you buy to make sure you get one that's fully compatible with the Raspberry Pi.

Once set up, you can mount your home file server on all the other computers on your network, and use it as a convenient place to store everything from music files you want to share with your housemates, to backups of important documents and save-game files you'd like to share between computers.

We recommend using a wired Ethernet connection for stability and fast transfer speeds. The project will still work if you connect your Pi via WiFi, although performance will be affected, particularly when it comes to copying over large files.



>STEP-01
How to: Set up Samba
Download the latest version of NOOBS (**magpi.cc/2bnf5XF**) and copy it to a blank micro SD card that's been formatted as fat32. Plug the micro SD card into your Pi, boot it up and opt to install Raspbian with a PIXEL desktop.



>STEP-02
Install Samba
Samba is available in Raspbian's standard software repositories. We're going to update our repository index, make sure our operating system is fully updated, and install Samba using **apt-get**. Open a Terminal and type:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install samba samba-common-bin
```

>STEP-03
Create your shared directory
We're going to create a dedicated shared directory on our Pi's micro SD hard disk. You can put it anywhere, but ours will be at the top level of the root file system.

```
sudo mkdir -m 1777 /share
```

This command sets the sticky bit (1) to help prevent the directory from being accidentally deleted and gives everyone read/write/execute (777) permissions on it.

>STEP-04
Configure Samba to share your new directory
Edit Samba's config files to make the file share visible to the Windows PCs on the network.

```
sudo leafpad /etc/samba/smb.conf
```

In our example, you'll need to add the following entry:

```
[share]
Comment =  Pi shared folder
Path = /share
Browseable = yes
Writeable = Yes
only guest = no
create mask = 0777
directory mask = 0777
Public = yes
Guest ok = yes
```

This means that anyone will be able to read, write, and execute files in the share, either by logging in as a Samba user (which we'll set up below) or as a guest. If you don't want to allow guest users, omit the **guest ok = yes** line.

You could also use Samba to share a user's home directory so they can access it from elsewhere on the network, or to share a larger external hard disk that lives at a fixed mount point. Just create a **smb.conf** entry for any path you want to share, and it'll be made available across your network when you restart Samba.

>STEP-05
Create a user and start Samba
Before we start the server, you'll want to set a Samba password – this is not the same as your standard default password (raspberry), but there's no harm in reusing this if you want to, as this is a low-security, local network project.

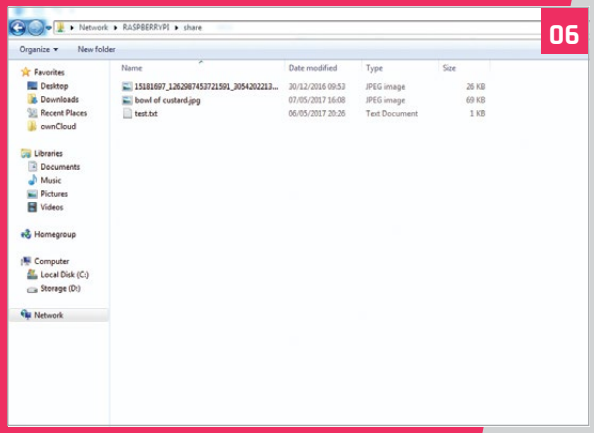
```
sudo smbpasswd -a pi
```

Then set a password as prompted. Finally, let's restart Samba:

```
sudo /etc/init.d/samba restart
```

From now on, Samba will start automatically whenever you power on your Pi. Once you've made sure that you can locate your shared folder on the network, you can safely disconnect the mouse, monitor, and keyboard from your Pi and just leave it running as a headless file server.

>STEP-06
Find your Pi on the network
You'll now be able to find your Raspberry Pi file server (named **RASPBERRYPI** by default) from any device on your local network. If you've left **smb.conf**'s default settings as they are, it will appear in a Windows network workgroup called **WORKGROUP**.



BUILD AN INTRANET WEB SERVER

Build a local HTML server with Apache

You'll Need

- Raspberry Pi 2/3
- Monitor, keyboard, and mouse (for setup)
- Wired Ethernet connection
- micro SD card with NOOBS **magpi.cc/2bnf5XF**

If you want to get to grips with how the web works, one of the most entertaining ways to learn is to build your own local intranet web server to display simple – or even complex – internal websites.

A Raspberry Pi is an ideal server for small websites that don't require the capacity or server-side processing power of a more powerful computer, and it's an ideal development environment if you're to use HTML.

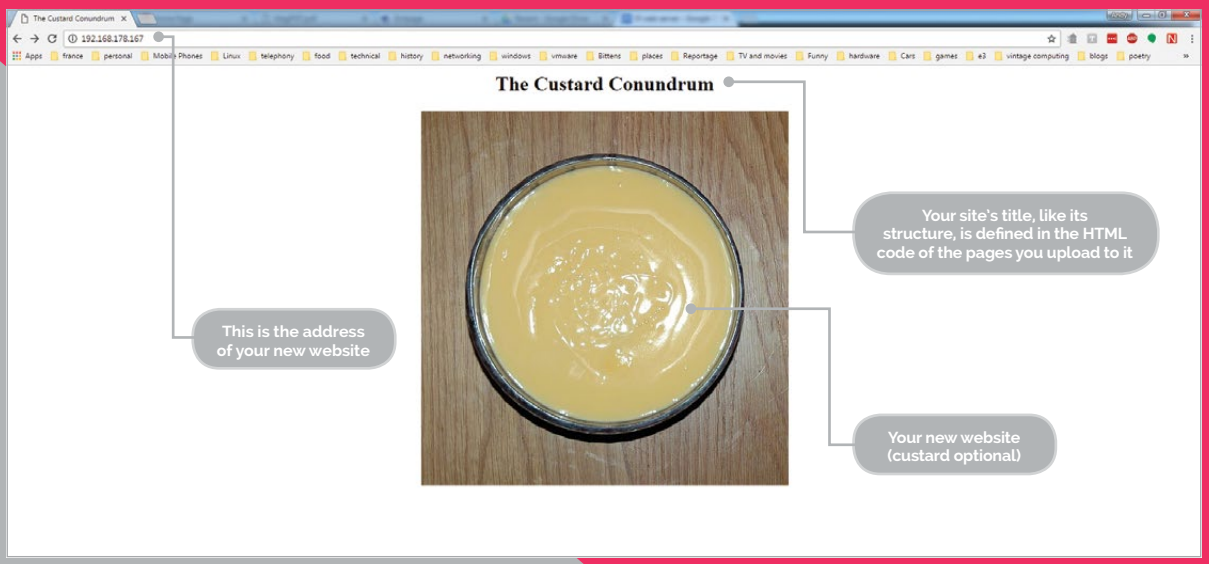
It can host a personal blog or to-do list, keep a web-based calendar for the household, hold your family photo albums, or simply host a website you're developing before you're ready to share it with the world.

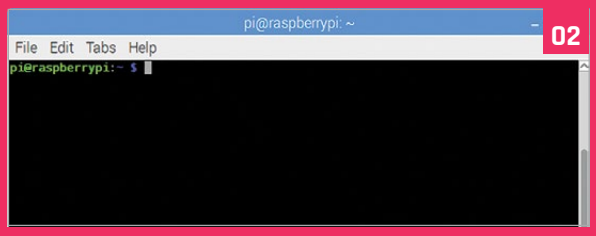
Much of the world wide web is built on LAMP – Linux, Apache, MySQL, PHP – often with a content management system (CMS) on top to make it easy to create complex websites with little knowledge of HTML or PHP.

This tutorial will take you through the basics of getting your server's environment set up. We don't go as far as installing a CMS, but by the end of the following steps, your Pi will be ready to immediately display ordinary 'flat' HTML webpages. If you add the optional step of installing the MySQL database back end and PHP interpreter, you should be all set to install and configure most CMSs by following the instructions they supply.

>STEP-01
Install the OS

Hook up the keyboard and mouse to the Pi and connect it to an HDMI monitor. Copy NOOBS to a FAT32-formatted micro SD card; insert it and power up the Pi. Opt to install Raspbian with the PIXEL window manager. This will take a few minutes.





02

>STEP-02
Check the network address

Once the Pi has rebooted, open a Terminal window and run:

```
ifconfig
```

Make a note of the 'inet addr' value for eth0. This will be the IP address of your web server. It's a good idea to assign the Pi a static DHCP reservation on your router so the Pi will keep that address permanently.

>STEP-03
Update your Pi and install Apache

Run the following commands in the Terminal to make sure Raspbian is up to date. Adding **-y** to the end of **apt-get** commands instructs the program to automatically answer yes to any questions rather than waiting for you to type **Y** or **N**.

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

Once this is complete, it's time to install your new Apache web server:

```
sudo apt-get install apache2 -y
```

Apache is the main piece of software you need to serve webpages to client PCs.

>STEP-04
Add PHP and MySQL (optional)

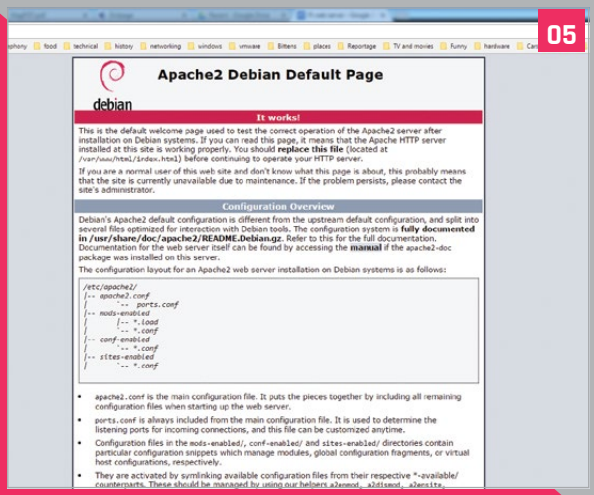
Many websites use content management systems, such as WordPress. These require PHP and MySQL, so if you want to experiment with CMS-driven sites further down the line, this is an ideal time to add the software you'll need to render them. You can install both at once using the following command:

```
sudo apt-get install php5 mysql-server -y
```

Chose a strong password for the MySQL 'root' user and note it down somewhere safe.

>STEP-05
Test Apache

Open your web browser of choice, either on the Pi or on another PC on your local network, and enter the IP address from step 2 into the address bar. You should see the Apache 2 Debian default page, with a red banner and the words 'It Works!' across it.



05

>STEP-06
Start building your site

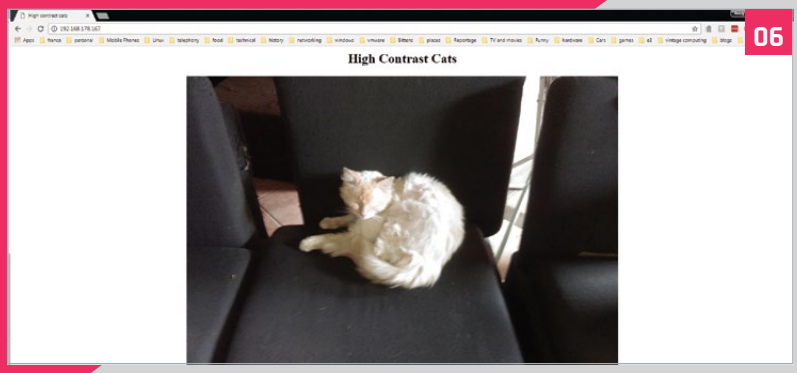
Your website's files are located in the **/var/www/html** directory. Run the following to make this folder accessible to the default user, **pi**.

```
sudo chown -R pi /var/www/html
```

Delete the **index.html** file found there and upload your new intranet site to that folder. Your new site should now be accessible from your local network on the address you used in step 5.

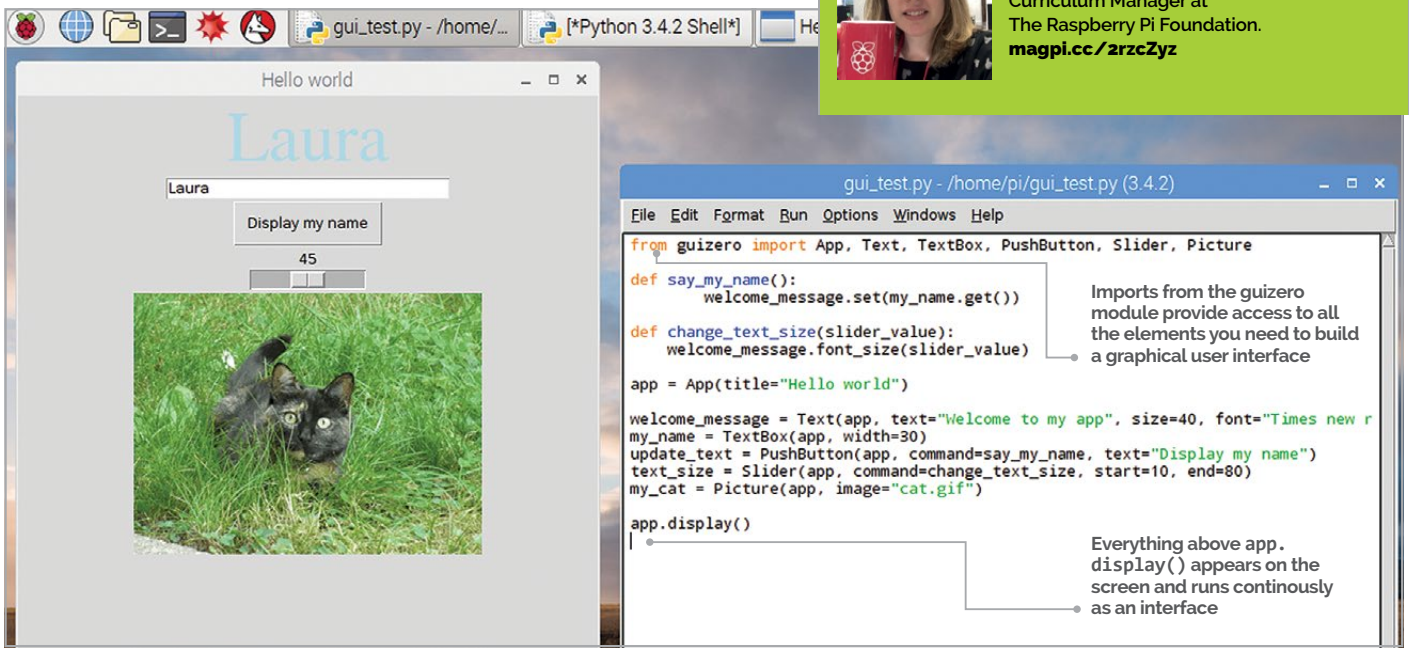
Your Pi is an ideal environment for web development, so you can simply drop the HTML files you're working on into your web server directory. If you want to experiment with more advanced options, you can enable FTP and SSH for remote access to your web server from other computers on your local network. You can also install a CMS such as WordPress, which you'll be able to access from a browser on your local network to create content-rich websites.

While this is strictly a local web server project, the same software and processes go into an internet-facing server, although that will require firewall configuration and extra security measures that are beyond the scope of this tutorial.



06

LAURA SACH
Laura is the Content and Curriculum Manager at The Raspberry Pi Foundation.
magpi.cc/2rzcZyz



GETTING STARTED WITH GUIS

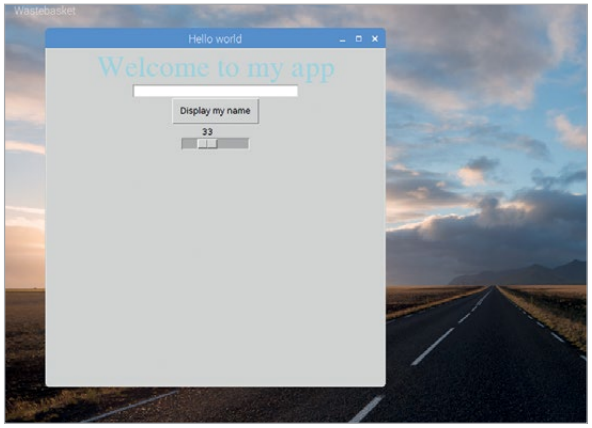
In this tutorial you will learn how to make simple GUIs in Python

You'll Need

- Raspberry Pi
- Raspbian
- guizero magpi.cc/2pRlwtq

GUI stands for graphical user interface and it is pronounced 'gooey'. If you have written Python programs before, all of your input and output will probably have been dealt with via text appearing on the screen or being typed in by the user. Adding a GUI to your program lets the user interact with it using buttons, dropdowns, text boxes, menus, and other familiar user interface controls.

In this tutorial you will learn how to make simple GUIs in Python.



Right Sliders are used to adjust the value of something, like the size of text displayed in the window

>STEP-01

Getting started

Before you start, make sure that you have installed the guizero library. Open a Terminal window and enter:

```
sudo apt-get update
sudo pip3 install guizero
```

Now open Python 3 (IDLE), click on File > New File, and save your file as **gui_test.py**. Add a line of code at the start of your file to import the App class from the guizero library:

```
from guizero import App
```

Now add two more lines of code to create an app and then display it on the screen:

```
app = App(title="Hello world")
app.display()
```

Save your file and press **F5** to run it. You should see a blank GUI window. Congratulations, you have built your first GUI app.

>STEP-02

Adding widgets

Let's start adding content to the GUI. We will refer to items you can add to a GUI (such as text, text boxes, buttons etc.) as widgets. There are a couple of rules to follow when adding a widget: if you want to use a new type of widget, you must import it. The first line of code in your program looks like this:

```
from guizero import App
```

As an example, if you wanted to use the Text widget, you would add it to the import line, after **App**.

We will ask you to import various types of widget throughout this guide. Each type of widget only needs to be added to the import list once, and then you can use it as many times as you want on your GUI.

All code that creates a widget must be added before the event loop, which means between the line of code where you create the app, and the **app.display()** line of code:

```
from guizero import App, Text
# Add GUI widget code here
app.display()
```

This is because the line of code **app.display()** starts the event loop. The GUI will be waiting for the user to do things such as click on a button – these are called events. It will constantly check whether anything new has happened and automatically update the display if necessary. The event loop blocks later code (rather like a **while True:** loop), so code written after the event loop will never execute.

Throughout this tutorial, we will ask you to add widgets to the GUI, which means adding them anywhere between these two lines of code.

>STEP-03

Text widget

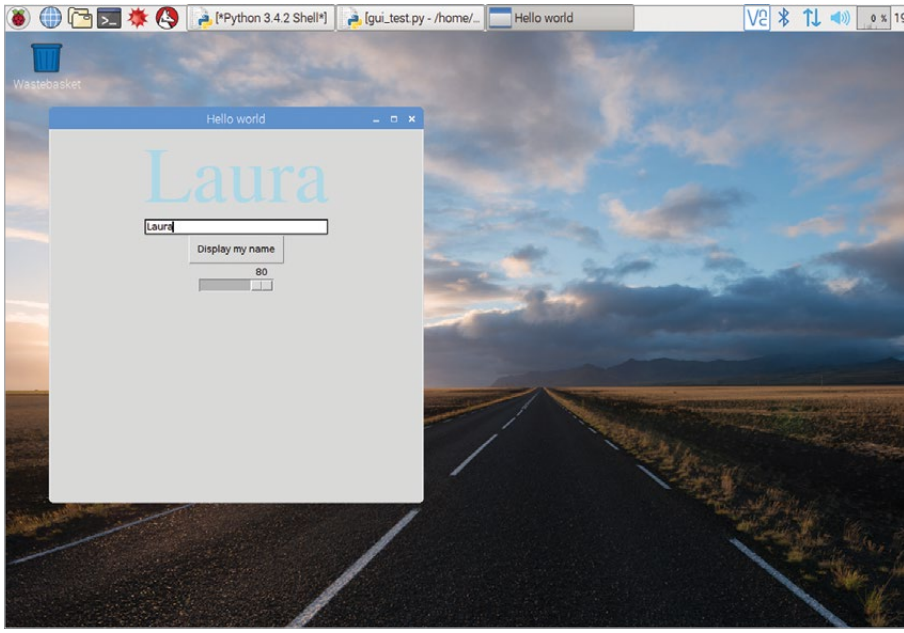
Probably the simplest widget you can add is the Text widget, which displays some text on the screen.

Add **Text** to the import statement (read the first part of step 2, 'Adding widgets', if you are not sure how to do this).

Add a Text widget to the GUI (read step 2 again if you are not sure where to put this code):

```
welcome_message = Text(app, text="Welcome
to my app")
```

Here we have created a Text widget with the name **welcome_message**. The first argument (in the brackets) tells the widget who its boss is! To be more specific, we are telling this Text widget that it will be controlled by the **app** object, which we created earlier. The first argument given to any widget always tells it the name of its boss (or 'master').



Above You can add text boxes to collect input from the user, such as their name or text to be used in the program

Run your code by pressing **F5**. You should see 'Welcome to my app' displayed on your GUI.

Did you notice that we could tell the Text widget what content we wanted it to display by specifying **text="Welcome to my app"**? This is called a keyword argument, because we have specified the keyword text and the value we want. We can specify other keyword arguments, too; just add them on to the end, separated by commas.

```
welcome_message = Text(app, text="Welcome
to my app", size=40, font="Times New
Roman", color="lightblue")
```

Here, we have used keyword arguments for the size, font, and colour (note the American spelling, 'color').

You can specify any font your computer has installed. Colours can be specified as colour names, but not every possible colour has a name, so you can also use hex codes (e.g. #ff0000) to define colours.

>STEP-04

TextBox widget

TextBox widgets are used to let the user type in data – a bit like the GUI version of the input() function you may have used before. Here's how to add one:

Add the TextBox widget to your import statement:

```
from guizero import App, Text, TextBox
```

Now add a TextBox to the GUI:

```
my_name = TextBox(app)
```

Run your code by pressing **F5**.You should see a small text box appear. There is a keyword parameter width which you can add if you wish to make the box wider.

>STEP-05

PushButton widget

PushButton widgets create a button. When the button is pushed, a function is called.

Before the code that creates the GUI app, write a function which will be called when the button is pressed. It is a good idea to put all of your function code at the start of your program, immediately after the import line.

```
def say_my_name():
    welcome_message.set( my_name.get() )
```

This function refers to the Text widget (**welcome_message**) and sets its value to what was typed into the TextBoxwidget (**my_name**). You can use the **get()** and **set()** functions with many widgets to retrieve their current value or set them to something new.

First, add **PushButton** to your import statement. Now use this code to add a PushButton to the GUI:

```
update_text = PushButton(app,
command=say_my_name, text="Display my
name")
```

The first argument tells the PushButton that the app is its boss. Then we use two keyword arguments: **command** tells the button which function to call when it is pressed, and **text** is the text which will be displayed on the button.

Press **F5** to run your code. Type your name into the text box and then press the button. You should see your name displayed in large text at the top.

You have now experienced the basis for how the event loop works. The GUI waits for an event (in this case, you clicking on the button) and it calls a function in response to the event. This function may contain code to change something on the GUI; if so, the display is updated accordingly.

>STEP-06

Slider widget

A slider lets users move within a range of values easily, rather like moving a volume control up or down.

Before the code which creates the GUI app, write a function that will be called when the slider is moved.

```
def change_text_size(slider_value):
    welcome_message.font_size(slider_
value)
```

This function has a parameter called **slider_value**. When the slider is moved, the current value of the slider will automatically be sent to the function, so we must give it a name. We have chosen to call this parameter **slider_value**. The code inside the function sets the **font_size** of the **welcome_message** to the current slider value.

gui_test.py

```
01. from guizero import App, Text, TextBox, PushButton,
    Slider, Picture
02.
03. def say_my_name():
04.   welcome_message.set(my_name.get())
05.
06. def change_text_size(slider_value):
07.   welcome_message.font_size(slider_value)
08.
09. app = App(title="Hello world")
10.
11. welcome_message = Text(app, text="Welcome to my app",
    size=40, font="Times new roman", color="lightblue")
12. my_name = TextBox(app, width=30)
13. update_text = PushButton(app, command=say_my_name,
    text="Display my name")
14. text_size = Slider(app, command=change_text_size,
    start=10, end=80)
15. my_cat = Picture(app, image="cat.gif")
16.
17. app.display()
```

Language
>PYTHON 3

DOWNLOAD:
magpi.cc/2pQLpvB

Add **Slider** to the import statement. Now add the Slider code to the GUI:

```
text_size = Slider(app, command=change_
text_size, start=10, end=80)
```

The command is the function that will be called when the slider is moved (i.e. the function we just created). Start and end values are specified for the largest and smallest values the slider can have. We have specified these so that the font does not get too large or small – the smallest it can be is 10pt and the largest is 80pt.

Save your code and press **F5** to run it. Move the slider from side to side and watch the size of the text change.

>STEP-07

Picture widget

You can add pictures to your GUI, as long as they're in GIF format. Sadly, animated GIFs only display as stills.

Find a picture in GIF format that you'd like to use, or save an existing picture as a GIF. Save the picture in the same folder as your **gui_test.py** Python file.

Add **Picture** to your import statement. Now Add a Picture to the GUI:

```
my_cat = Picture(app, image="cat.gif")
```

Press **F5** to run your code. You should see your picture appear on the GUI. You have now learnt how to use some simple GUI widgets. The full code for this tutorial is found in **gui_test.py**.

Strato Pi enhances the Raspberry Pi computer with hardware features that make it suitable for use in professional applications where reliability and service continuity are key requirements.

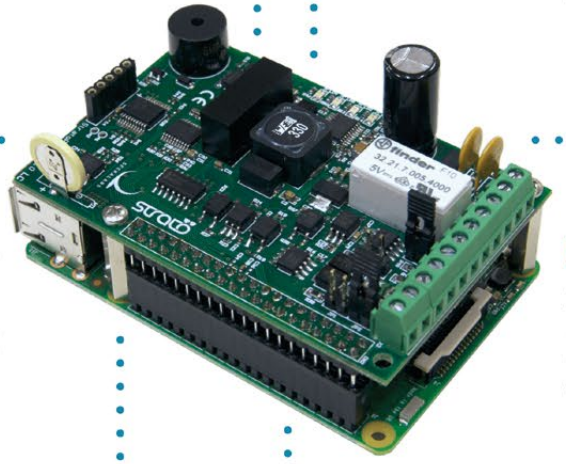
strato

Makes Raspberry Pi professional

Now introducing CAN Bus support

POWER SUPPLY
9-65Vdc power supply, with surge and reverse polarity protection

POWER RELAY
1 power relay output with both normally open and normally closed contacts



SERIAL INTERFACES
CAN bus and RS-485 interfaces to connect to other field systems

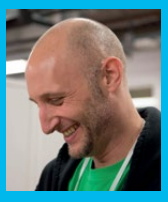
HARDWARE WATCHDOG
Strato can reset or perform a power cycle in case of software failure

REAL TIME CLOCK
RTC with on-board lithium back-up battery

CE/ROHS CERTIFIED
Strato Pi can be safely installed in industrial and residential environments

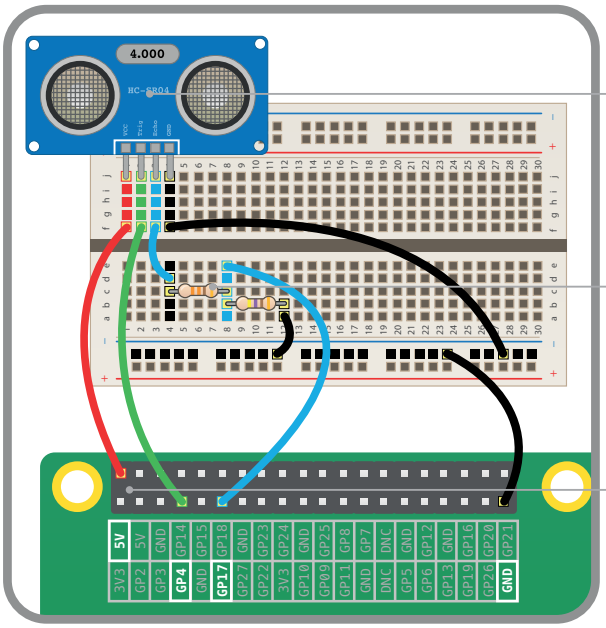
The Strato Pi line offers many different solutions, available as boards, DIN-rail modules or touch panels.

SFERALABS | www.sferalabs.cc



MARC SCOTT

Marc used to run a Raspberry Pi and Minecraft club at his old school, where he taught Computer Science, and Systems and Control. He's now Head of Curriculum at the Raspberry Pi Foundation.
magpi.cc/2pOFrJB



THE RASPBERRY PI ULTRASONIC THEREMIN

You'll Need

- HC-SR04 Ultrasonic Distance Sensor **magpi.cc/2q3cZ9J**
- Breadboard **magpi.cc/2q34ZFz**
- Jumper wires **magpi.cc/2q2Kqcm**

Build your very own theremin musical instrument using an ultrasonic distance sensor and a little bit of Python and Sonic Pi code

A theremin is a unique musical instrument that produces sound without being touched.

In this tutorial, you will use an ultrasonic distance sensor to control the notes played by Sonic Pi.

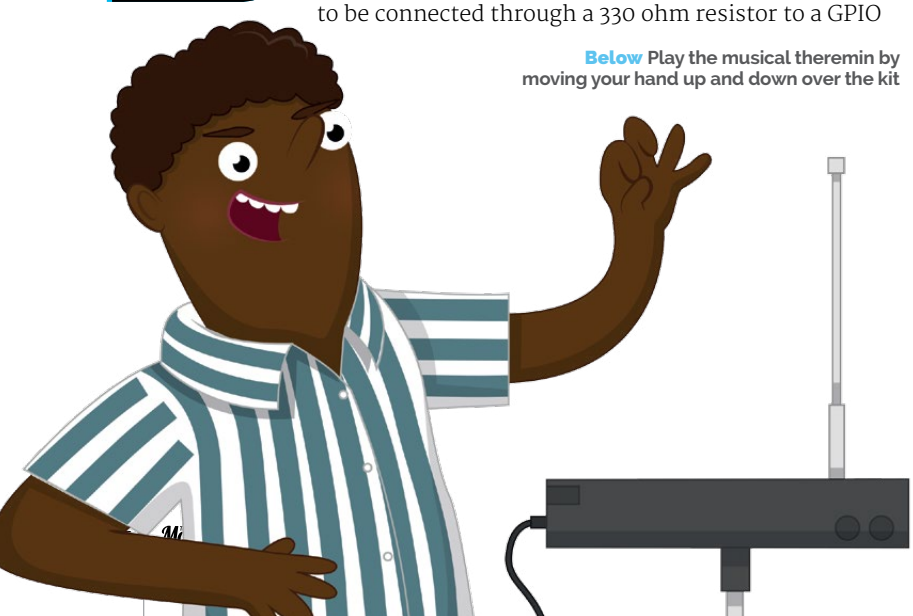
An ultrasonic distance sensor has four pins: Gnd (ground), Trig (trigger), Echo (echo), and Vcc (power).

To use the sensor, you need to connect its Gnd pin to a GND (ground) pin on the Raspberry Pi, the Trig pin to a GPIO pin on the Pi, and the Vcc pin to the 5V pin on the Pi.

The Echo pin is a little more complicated. It needs to be connected through a 330 ohm resistor to a GPIO pin on the Raspberry Pi, and that pin needs to be grounded through a 470 ohm resistor. The diagram shows one suggested arrangement. If you've wired up the sensor as shown in the diagram, your echo pin is 17 and your trigger pin is 4.

Click on Menu > Programming > Python 3 (IDLE), to open a new Python shell. Click on New > New File. The code to detect distance is listed in **theremin1.py**. Type it into your new file, then save and run it.

The **sensor.distance** is the distance in metres between the object and the sensor. Run your code and move your hand backwards and forwards. You should see the distance changing, as it is printed in the shell.

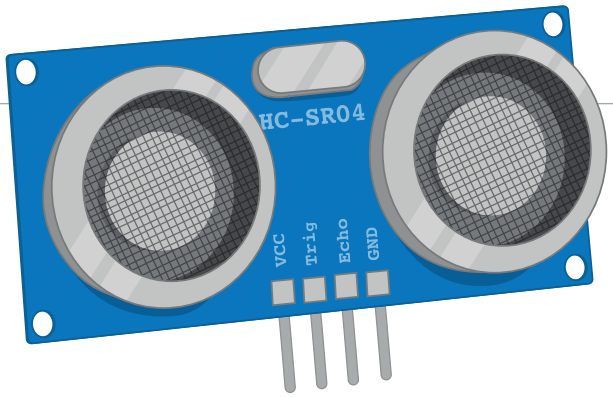


Below Play the musical theremin by moving your hand up and down over the kit

Getting Sonic Pi ready

Sonic Pi will receive messages from your Python script. Open Sonic Pi by clicking on Menu > Programming > Sonic Pi. In the buffer that is open, you can begin by writing a **live_loop**. This is a loop that runs forever, but can easily be updated, allowing you to experiment. You can add a line to reduce the time it takes for Sonic Pi and Python to talk.

```
live_loop :listen do
    set_sched_ahead_time! 0.1
end
```



Above The distance sensor sends out a high-frequency pulse and measureshow long it takes the echo to reflect back.

Next, you can sync the live loop with the messages that will be coming from Python.

```
live_loop :listen do
    message = sync "/play_this"
end
```

The message that comes in will be a dictionary, containing the key **:args**. The value of this key will be a list, where the first item is the MIDI value of the note to be played.

```
live_loop :listen do
    message = sync "/play_this"
    note = message[:args][0]
end
```

Lastly, you need to play the note.

```
live_loop :listen do
    message = sync "/play_this"
    note = message[:args][0]
    play note
end
```

You can set this live loop to play straight away, by clicking on the Run button. You won't hear anything yet, as the loop is not receiving any messages.

Sending notes from Python

To finish your program, you need to send note MIDI values to Sonic Pi from your Python file. You'll need to use the OSC library for this part.

```
from gpiozero import DistanceSensor
from time import sleep

from pythonosc import osc_message_builder
from pythonosc import udp_client

sensor = DistanceSensor(echo=17, trigger=4)

while True:
    print(sensor.distance)
    sleep(1)
```

Now you need to create a sender object that can send the message.

theremin1.py

```
from gpiozero import DistanceSensor
from time import sleep

sensor = DistanceSensor(echo=17,
trigger=4)

while True:
    print(sensor.distance)
    sleep(1)
```

Language
>PYTHON

FILE NAMES:
theremin1.py
theremin2.py
theremin3.py

DOWNLOAD:
magpi.cc/2qj7qTN

theremin2.py

```
live_loop :listen do
    set_sched_ahead_time! 0.1
end
```

theremin3.py

```
from gpiozero import DistanceSensor
from time import sleep

from pythonosc import osc_message_builder
from pythonosc import udp_client

sensor = DistanceSensor(echo=17, trigger=4)
sender = udp_client.SimpleUDPClient('127.0.0.1', 4559)

while True:
    pitch = round(sensor.distance * 100 + 30)
    sender.send_message('/play_this', pitch)
    sleep(0.1)
```

```
sensor = DistanceSensor(echo=17, trigger=4)
sender = udp_client.SimpleUDPClient('127.0.0.1', 4559)

while True:
    print(sensor.distance)
    sleep(1)
```

You need to convert the distance into a MIDI value. These should be integers (whole numbers), and hover around the value 60, which is middle C. Round the distance to an integer, multiply it by 100, and then add a little bit, so that the note is not too low in pitch.

```
while True:
    pitch = round(sensor.distance * 100 + 30)
    sleep(1)
```

To finish off, you need to send the pitch over to Sonic Pi and reduce the sleep time. The final code is listed in **theremin3.py**. Save and run your code and see what happens. If all goes well, you've made your very own theremin

LORRAINE UNDERWOOD

Lorraine runs the Kendal Pi Jam. She is an Irish maker living in Yorkshire with a background in web development
@LMcUnderwood
magpi.cc/2ovDDgz

TEMPERATURE CONTROLLED STAIRLIGHTS

You'll Need

- 5V RGB strip **magpi.cc/2pzIMyC**
- 5V mains power supply **magpi.cc/2pF8cqi**
- Female jack connector plug **magpi.cc/2q9Hfi6**
- Male-to-female jumper wires

Give every day a colourful and useful start by displaying the outside temperature using coloured lights on your stairs

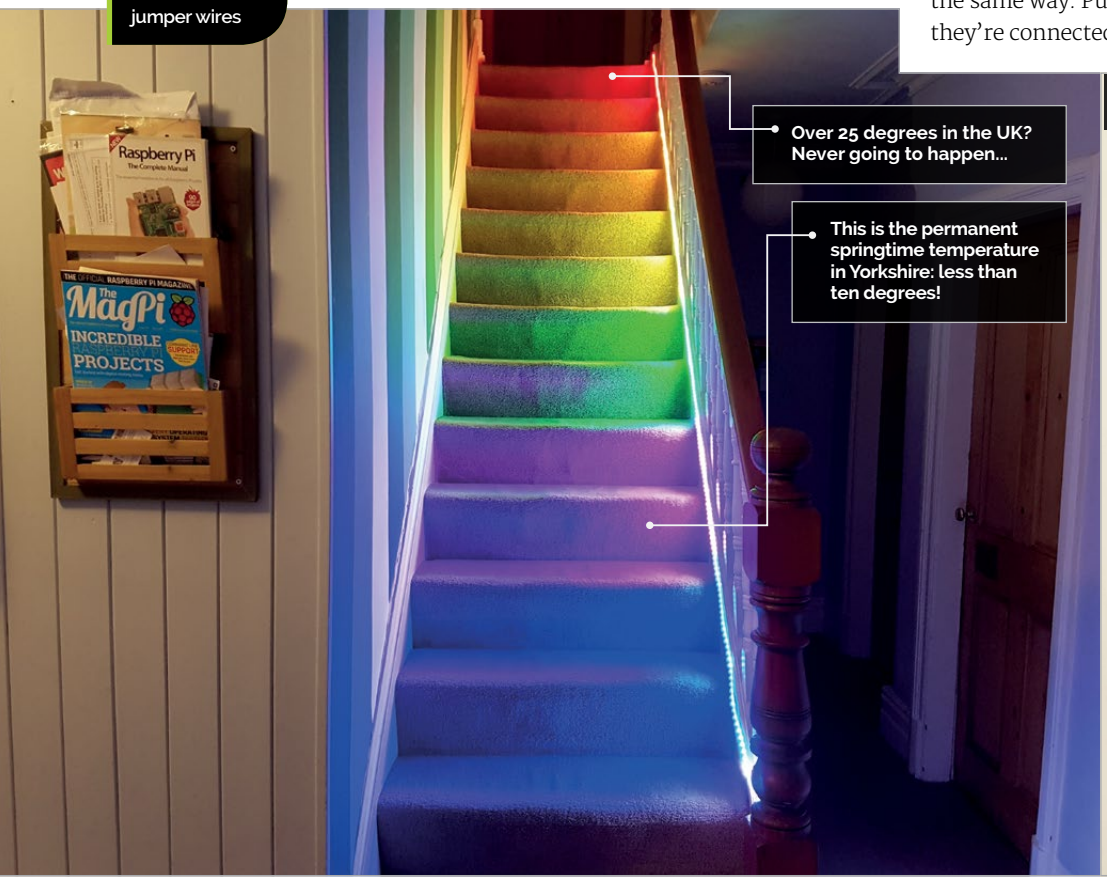
When you wake up in the morning, wouldn't you like to know whether you need to wear your woolly hat or your sundress? Now you can find out on your way to breakfast, thanks to Lorraine's stairlights project! The Raspberry Pi hidden under the stairs connects to the web and checks the temperature. It then controls the strip of 240 lights running up the stairs. If it's colder than 0 °C, the bottom 35 lights come on in white; under 5 °C, and the next 35 lights light up in blue; and so on up to 25 °C and red, although that probably won't happen in Yorkshire, where it was built!

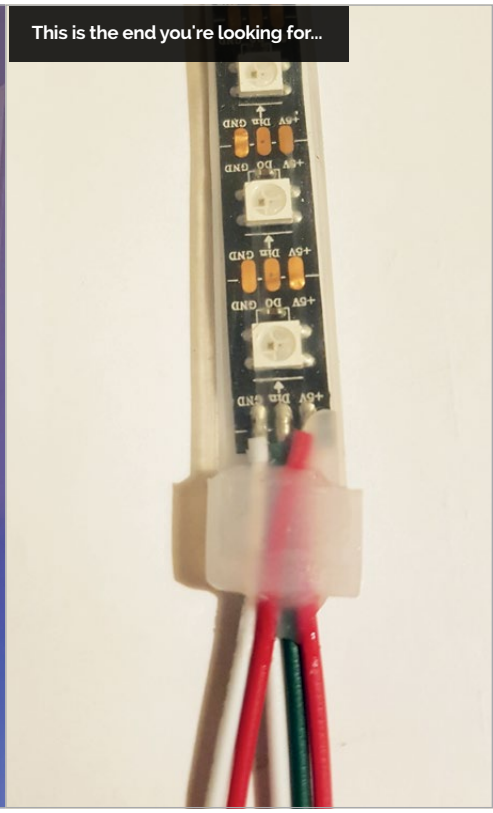
>STEP-01

Set up the lights

Firstly, work out which end is which on your RGB strip. We are looking for the Data In end. It should be labelled as Din. In this strip there are five cables coming from three connections: two from GND, one from DIN, and two coming out of 5V.

Connect the 5V wire to the '+' block on the female jack connector plug by placing the bare wire under the terminal, then screwing the terminal down with a screwdriver. Connect the GND wire to the '–' block in the same way. Pull gently on both wires to check that they're connected.





>STEP-02

Set up the Pi

Connect the Din and GND wires to the male ends of the jumper wires. Connect the female end to the Pi as follows: Din to GPIO pin 18; GND to any ground pin.

You could power the Pi through the other 5V wire, but this can be dangerous for the board; it is best to use a normal power supply.

Follow the steps at Adafruit to install the NeoPixel library, rpi_ws281x: **magpi.cc/1nRSyYk**.

Plug the jack connector into your power supply. Plug in the power supply and test your strip using the scripts from the **examples** folder for some shiny lights!

>STEP-03

Set up the weather API

You will need a developer account for a weather API to get the outside temperature for your area. We used **forecast.io** as it allows users to request 1,000 forecasts per day at no cost:

- Go to forecast.io
- Select Developer
- Select 'Sign up' to register an account
- Once you confirm your email address, you will get a secret key: you will need this key in Step 4

You could set up another Pi outside your house and get the real temperature for your neighbourhood. It depends on how much you want to spend on the project, and whether you trust the API's accuracy.

>STEP-04

Let there be light!

Download **stairlights.py** from the box at the top of this page to the **home/pi** directory and change:

Line 6: enter your secret key.

Line 7: change **longi** and **lati** to your longitude and latitude coordinates. You can use Google Maps to find them: right-click on your location and select 'What's here?'

Set up a cron task to check the outside temperature every five minutes and update the lights.

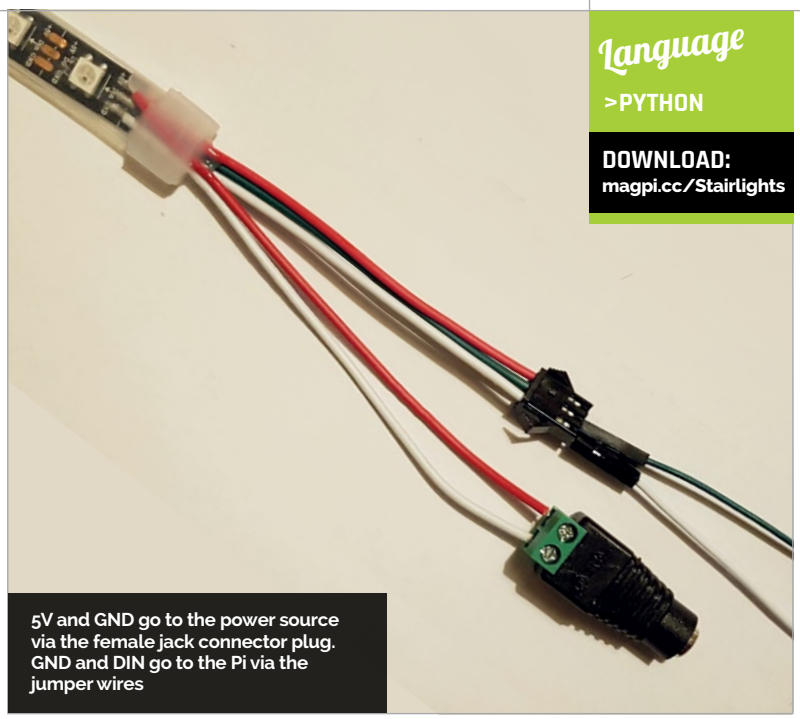
In a Terminal window, type:

```
sudo -E crontab -e
```

At the end of the file, enter:

```
*/5 * * * * /usr/bin/python3.4 /home/pi/stairlights.py
@reboot /usr/bin/python3.4 /home/pi/stairlights.py
```

Save the changes.



5V and GND go to the power source via the female jack connector plug. GND and DIN go to the Pi via the jumper wires

Language

>PYTHON

DOWNLOAD:
magpi.cc/Stairlights

>STEP-05

Debugging your lights

If the examples from the NeoPixel library didn't work, check all your connections. Make sure you have plugged the data wire into the correct GPIO pin on your Pi. Are you connected to Din or Dout? The lights will power on Dout, but nothing will happen.

If the lights are displaying strange colours, some people have reported the NeoPixel library not working with Pi's setup for audio. See **magpi.cc/2ovIQhy**.

Make sure your script is running properly before adding it to cron. There's nothing like sitting on your stairs waiting for five minutes for a cronjob that isn't working to run!

>STEP-06

What next for your lights?

We changed the cron tasks to stop the lights displaying through the night:

```
*/5 7-21 * * * /usr/bin/python3.4 /home/pi/stairlights.py
*/5 7-21 * * * /usr/bin/python3.4 /home/pi/nightynight.py
```

...where **nightynight.py** is a simple script that turns the lights off using the first 26 lines of **stairlights.py** and this line:

```
colorMe(strip, Color(0, 0, 0), 0, 239)
```

Now, with a strip of lights on your stairs, you could play all kinds of games!

Add some coloured arcade buttons for a quick reaction game.

Maybe when your phone connects to the Bluetooth on the Pi, your stairs could flash a welcoming hello dance. Experiment and have fun!

OPEMIPO OGUNKOLA

Interested in hardware and software, Ope codes in Python, C++, C, and knows a bit of web development.
magpi.cc/2olbmMS

MAKE THE SAME GAME

DATABASE OPTION

You could save the high scores with an SQLite database. Can you figure out how to do it?

You click a shape, it disappears, the board adjusts, and you get a score; that's what Same is about!

We are going to show you how to make a simple game called Same. Same consists of different coloured balls arranged on a board. Balls of the same colour are joined together by connecting squares, and together they form a shape. When any part of this shape is clicked, it disappears and the balls above take the place of the now removed balls.

We are going to be coding all of this from scratch using Pygame. By the end of this tutorial, you should be able to make your own modifications and make this game truly your own. How cool is that?

Code structure

The code for this project is quite large, so we'll assume you know how to do simple things in Pygame, such as draw a circle. What we'll focus on instead is how you should think when you are tackling a program project similar to this. Before we begin, get the code at: **magpi.cc/2oIl4yM**. Use **python Gui.py** to start a game.

The first thing you want to do with a project like this is to think through what you want to do. As your programs begin to get more complex, the more important this step becomes. We will be making use of the model–view–controller (MVC) pattern to structure our program. The model refers to permanent data, the controller is where the main stuff happens, and the view is what the user sees. All our controller code is going to be found in the **logic.py** file (used for scoring, the internal representation of the board, and other functions for tasks like removing balls and finding shapes).

The **Gui.py** file contains all the code that is related to the view. It has code to draw balls and connecting squares on the screen, based on the internal representation of the board found in the **logic.py** file. It also has code to draw the scoreboard, and deals with events as they happen. For the model, we have included it in the **logic.py** file. The only thing permanently saved between different games is the high score. This is stored in the **TopScores.txt** file. We access this data with **`getHighScore()`** and update the high score if we need to with **`updateHighScore()`**. These functions are found in the **logic.py** file, but they belong to the model.

Language
>PYTHON

FILE NAMES:
Logic.py
Gui.py

DOWNLOAD:
magpi.cc/2oIl4yM

Gui.py

Contains:

- Game display class
- Displays the board based on the representation in the Board class
- Figures out which ball was clicked and uses the remove method in the Board class to remove the shape

Depends on:

Logic.py

Contains:

- Board class
- Has a Board attribute
- Balls get removed from the board here
- Computes the current score

Above The code flow is fairly simple and it depends mostly on two scripts

Game logic

Now we have an idea of how our code is going to be structured, let's discuss some implementation details. We represent the board as a two-dimensional list of balls called, surprisingly, **balls**. This makes it easy to access the ball at position x,y with **balls[x][y]**. Another critical task we need to be able to do is to group similarly coloured balls into a shape. This is done by checking up, down, left, and right to see whether the balls at those positions are of the same colour (the **_adjacent()** function does this). **_findAdjacent()** is then used to find all the balls that belong to a shape. The **_markBalls()** function is used to mark a shape for deletion, while **_clearBalls()** deletes the shape. The **_findAdjacent()** function is also used to calculate the number of balls that were removed and hence is used to calculate the score in **getScore()**.

Displaying the board

Now we have created **logic.py**, we can now develop the code that draws things on the screen and responds to clicks. This is where Pygame does its magic. All the code that displays stuff can be found in **_display()**. It makes use of a function to draw the square of different colours. It draws the board and also draws the connecting squares.

Left After one move, the red ball drops down and creates a shape with the other red ones

Reacting to events

Besides displaying objects on the screen, our game needs to react to events. We need to know which ball was clicked, so we use **_getPosition()**; this returns the x,y position if a ball was clicked, and **None** if the click wasn't on a ball. After we have identified which ball was clicked, we remove all the balls belonging to that ball's shape. This is done with the **_removeBalls()** function in **logic.py**. The event-handling loop is in the **run()** function. This updates the board frequently after any event has occurred.

Permanent game data

The only thing that is being permanently remembered by our game is the high score. This has been implemented using a file. Another robust approach would be to use an SQLite database to store several high scores.

Releasing to the world

Our game is working now, but there are still some finishing touches we are going to add. We want to be able to share our game with our friends as easily as possible. One way to do that would be to create an executable that contains our game, and then any player just needs to click on the executable to play the game. No extra downloads! To do this, we need to install pyinstaller. This is achieved by entering the following command:

```
pip install pyinstaller
```

To create our package in the Terminal and in the same directory as our game, we run.

```
pyinstaller Gui.spec --windowed -onefile
```

Voila! We have an executable we can share with all our friends. Pyinstaller can create executables for other operating systems, too, so nobody misses out!

FURTHER IMPROVEMENTS

We can add sounds, we can change the colours of the balls, we can modify the way the game is scored and much more!

JOHN M WARGO

John is a professional software developer, writer (with six books on mobile development), father, husband, and geek. He loves tinkering with IoT hardware.
johnwargo.com

MAKE A VISUAL APPOINTMENT REMINDER

A simple, visual calendar reminder using a Raspberry Pi Zero W, Blinkt! LED strip, and the Google Calendar API

You'll Need

- Pimoroni Pi Zero W Starter Kit **magpi.cc/2ovBxGJ**

It's easy to get distracted at work and miss the Google Calendar reminders on your desktop and smartphone, especially if you have back-to-back meetings. To help avoid missing meetings, let's create a Pi Reminder project, a visual notifier that uses the Raspberry Pi and a bunch of LEDs to remind you about upcoming appointments. Being visual-only, it has the advantage of not interrupting important phone calls or chats. The new Pi Zero W and Blinkt! have everything you need in a small, unobtrusive package. An easy project with very little setup.

Notifications

The Pi connects to a Google Calendar account every minute and checks for upcoming appointments. When it finds one, it flashes the Blinkt! LEDs for following alerts:

- **White @ 10 minutes until 5 minutes**
- **Yellow @ 5 minutes until 2 minutes**
- **Orange @ 2 minutes**

You can easily change the code to flash the lights in any colour or pattern you want.

Implementation

To assemble the hardware, follow the setup instructions for the Pimoroni Pi Zero W Starter Kit. Next, download the latest version of the Raspbian OS from the Raspberry Pi website (**magpi.cc/2ejN6sk**) and follow the installation instructions at **magpi.cc/1XTmymk**. Insert the SD card into the Pi, connect a keyboard, mouse, and a monitor to the Pi using the cables that come with the starter kit, and, finally, power it up using a smartphone charger or some suitable power source. Connect the Pi to your Wireless network using the instructions at **magpi.cc/2qyyUSX**.

When the Pi is connected to the network and ready to go, open a Terminal window and update the device's software using the following commands:

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

Install the Blinkt! Python libraries and example applications by executing the following command:

```
curl -sS get.pimoroni.com/blinkt | bash
```

Language
>PYTHON

DOWNLOAD:
magpi.cc/2ovAPJz

Next, download the project source code. In the Terminal window, execute the following command:

```
git clone https://github.com/
johnwargo/Pi-Remind-Zero-Blinkt
```

This will download and extract the project source code to the **Pi-Remind-Zero-Blinkt** folder. Change to the new folder using the following command:

```
cd Pi-Remind-Zero-Blinkt
```

Before you can use the project's software, you must set up an account with Google so the app can consume the Google Calendar APIs. To set up your account, read the Google Calendar API Python Quickstart at **goo.gl/Ay4AQg** and follow its instructions to create your account.

Create a new Google Calendar API application, then download the application's client secret to a file called **client_secret.json** in the reminder project's folder. Be sure to name the downloaded file using that file name. You'll need it to authorise the app to access your Google Calendar, and that particular file name is hard-coded into the project's main Python app (**remind.py**).

Now, install the Google Calendar API Python files (**goo.gl/os2Or7**) along with some other required libraries using the following command:

```
sudo pip install --upgrade google-
api-python-client python-dateutil pytz
httplib2 oauth2client
```

With everything in place, execute the reminder app using the following command:

```
sudo python ./remind.py
```

The application will launch, validate its configuration, and then warn you that there's additional configuration that must be completed, as shown in **Fig 1**.

Before the app can access the calendar, you'll need to authorise the app to use the Google Calendar API for

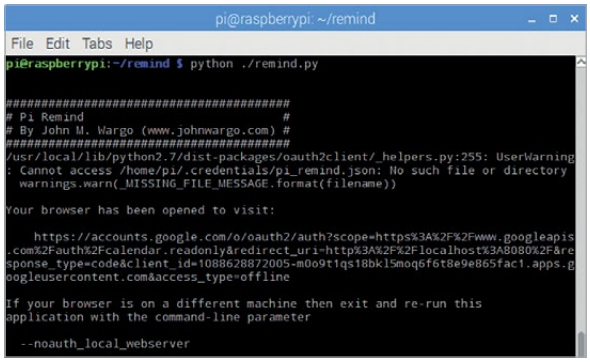


Fig 1 Initial loading of the Pi Reminder app requires access to your Google Calendar account

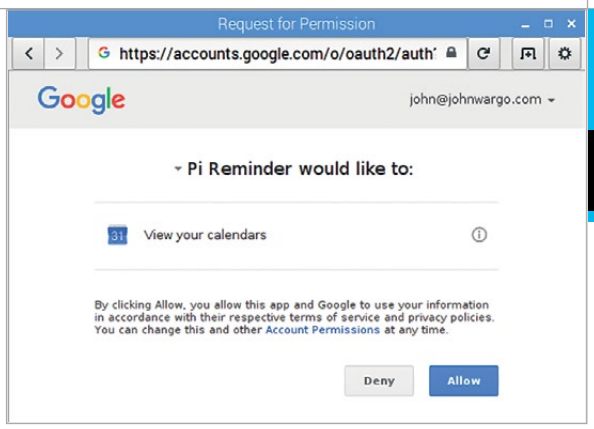


Fig 2 You must authorise the Pi Reminder application to access your Google Calendar

your calendar account. The browser will launch and walk you through the process. It starts by prompting you to log in to the Google account for the calendar you want the app to monitor. Enter the email address associated with that account and work through the authorisation process until the browser prompts you to approve the application's access to your calendar (**Fig 2**). Be sure to click the Allow button to authorise access.

At this point, the application will access your calendar and start notifying you of your upcoming appointments.

Remember though, if you ever change Google calendars (from a work to a personal calendar or from one work calendar profile to another), you'll need to whack the existing access token created during the initial startup of the Pi Reminder app. Instructions for deleting the token are available at **magpi.cc/2qyMupI**.

Starting the reminder application automatically

Now that you have the application running, let's configure the Pi to start the application at boot. Press **CTRL+C** to kill the running reminder application. Now, make the project's Bash script file executable by executing the following command:

```
chmod +x start-remind.sh
```

Open the pi user's session **autostart** file using the following command:

```
sudo nano ~/.config/lxsession/LXDE-pi/
autostart
```

Add the following line to the end (bottom) of the file:

```
@lxterminal -e /home/pi/Pi-Remind-Zero-
Blinkt/start-remind.sh
```

Save your changes: press **CTRL+O**, then the **ENTER** key. Next, press **CTRL+X** to exit the application.

Reboot the Raspberry Pi; when it restarts, the Python remind process should execute in its own Terminal window.

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

BUILD A HEXOME SIMULATOR

A UNIQUE MUSICAL INSTRUMENT

Make great polyrhythmic music controlled from your mobile device

You'll Need

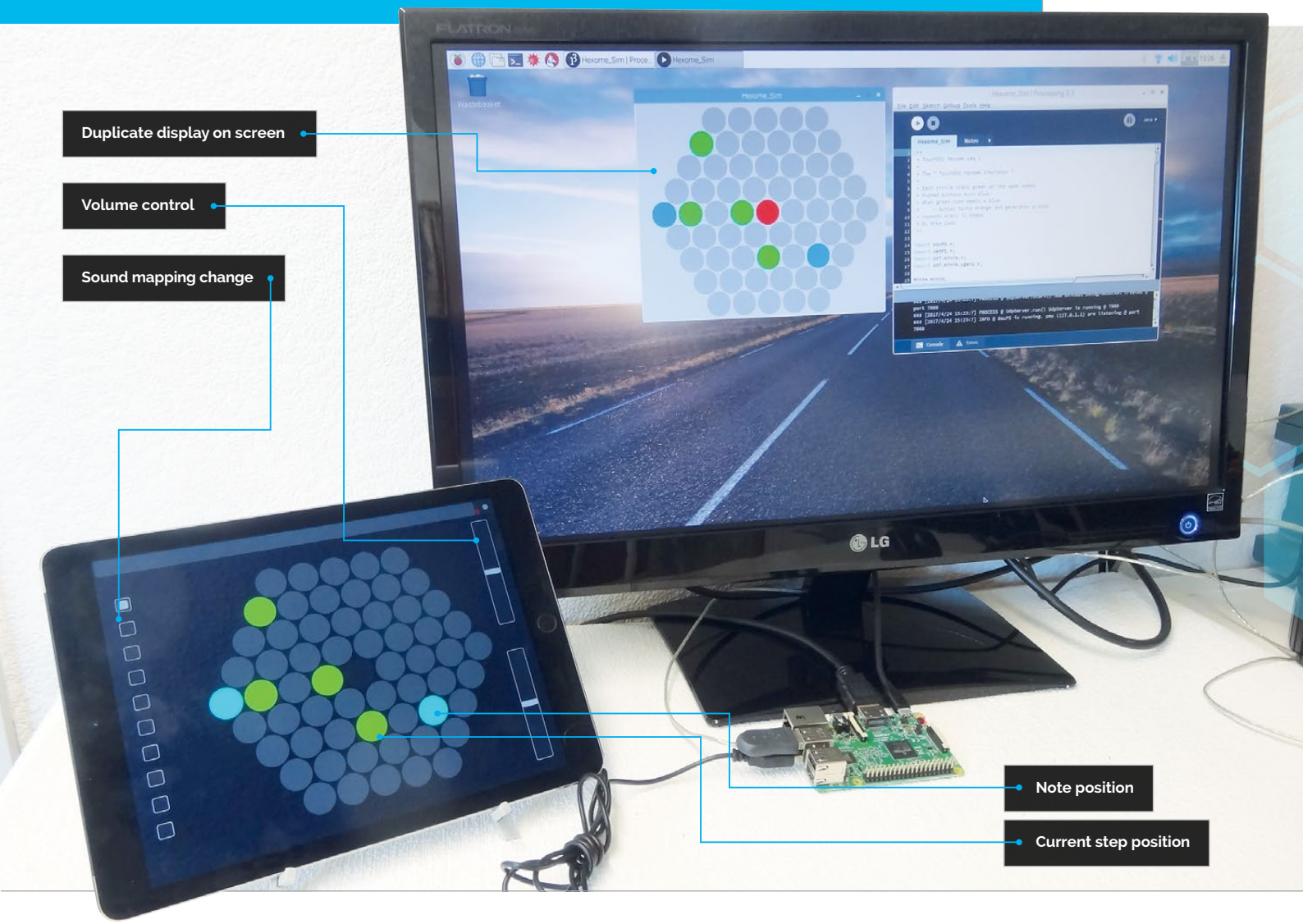
- Apple or Android mobile device (optional)
- TouchOSC app (only needed if you have a mobile device)
- Raspberry Pi 3

What is a hexome and why would you want to emulate it? Well, a hexome is a monome laid out on a hexagonal grid – which is all very well if you know what a monome is.

Brian Crabtree and his partner Kelli Cain created the monome in 2006, and the first model was an undedicated matrix of illuminated push buttons in a square or rectangular grid formation.

It was unique at the time, although its phenomenal success in music making and production ensured that there are now many commercial copycat variants of the concept. The idea was that in itself the monome had no function and made no sound: any function it did achieve was controlled by an external program running on some computer. This meant it could be used for anything you wanted. What was fixed,

Fig 1 The real Hexome



however, was the key and light numbering, so that any program written to use on one monome would work on any monome, and those commands were sent by OSC messages.

We looked at OSC messages last month and saw how these could be handled by a tablet and an application called TouchOSC. This month we look at how it can emulate a hexome – oh, and the reason you might want to emulate one is that it took us three years to build the original!

The hexome

Fig 1 shows what the original hexome looks like. If you want to build your own hexome, we have published all the details, software, and CAD files at **magpi.cc/2ph7XVS**. You will also find examples of more conventional monomes on the site.

The hexome differs in two major ways from the conventional monome: first, there's the hexagonal layout; second, the illuminating LEDs are not monochrome but RGB. The key numbering presented a bit of a problem because in a conventional grid there is a natural x-y Cartesian system that is easy to extend as the monome gets bigger. With a hexagonal arrangement it is not so straightforward. While you could use polar coordinates, it is not always satisfactory on this coarse scale. We decided to opt for the most straightforward but least useful simple numbering from the top left corner. This gave the freedom to use many different types of coordinate system depending on the application. All you have to do is to write the appropriate look-up tables to get to the raw key numbers. The key numbers are shown in the diagram for step 3 (page 59) in the guide for creating the TouchOSC layout.

> "This tool could affect, maybe even guide, music creation in a new and unusual way"

It is an observation, often made by musicians, that the tools you use determine the sort of music you make. So part of the appeal in creating the hexome was to see how this tool could affect, maybe even guide, music creation in a new and unusual way.

After several experiments, we found one surprising system that produced a unique form of sequencer, and this is what we are presenting here. It is what we like to call a polyrhythmic sequencer.

Polyrhythmic sequencer

Around the central key in a hexagonal grid is a circle of six hexagons; around that is a circle of 12, around that a circle of 18, and finally around that a circle of 24. If you number the rings as shown in **Fig 2**, then any ring Rn has the number of keys in Rn given by:

Rn= Rn-1 + 6

...where Rn-1 is the number of keys in the previous ring.

If each key in a ring is one point in a sequence, and each ring runs at the same sequence step time, then we have in effect four sequences of different length running at the same time. So the sequence taken as a whole will be much longer than the length of the longest sequence. In the case of our four-ring hexome, this is a sequence length of 72 before the sequence repeats. **Fig 3** (overleaf) shows this sequence: each ring has its own length, and each step is represented by different colours and shades. These are shown separately in the list at the top, and their interaction is shown underneath. Note that the inner ring, the six-step sequence, has to repeat twice while the next ring, the 12-step one, makes one cycle. Then we need three of the 12-step cycle to exactly align with two cycles of the next ring, the 18-step one. Finally, we need three cycles of the 24-step cycle to exactly line up with four cycles of the 18-step cycle. This give 72 steps in all, or the length of the longest sequence multiplied by the lowest common factor of all the rings – this is what a polyrhythmic sequencer is all about.

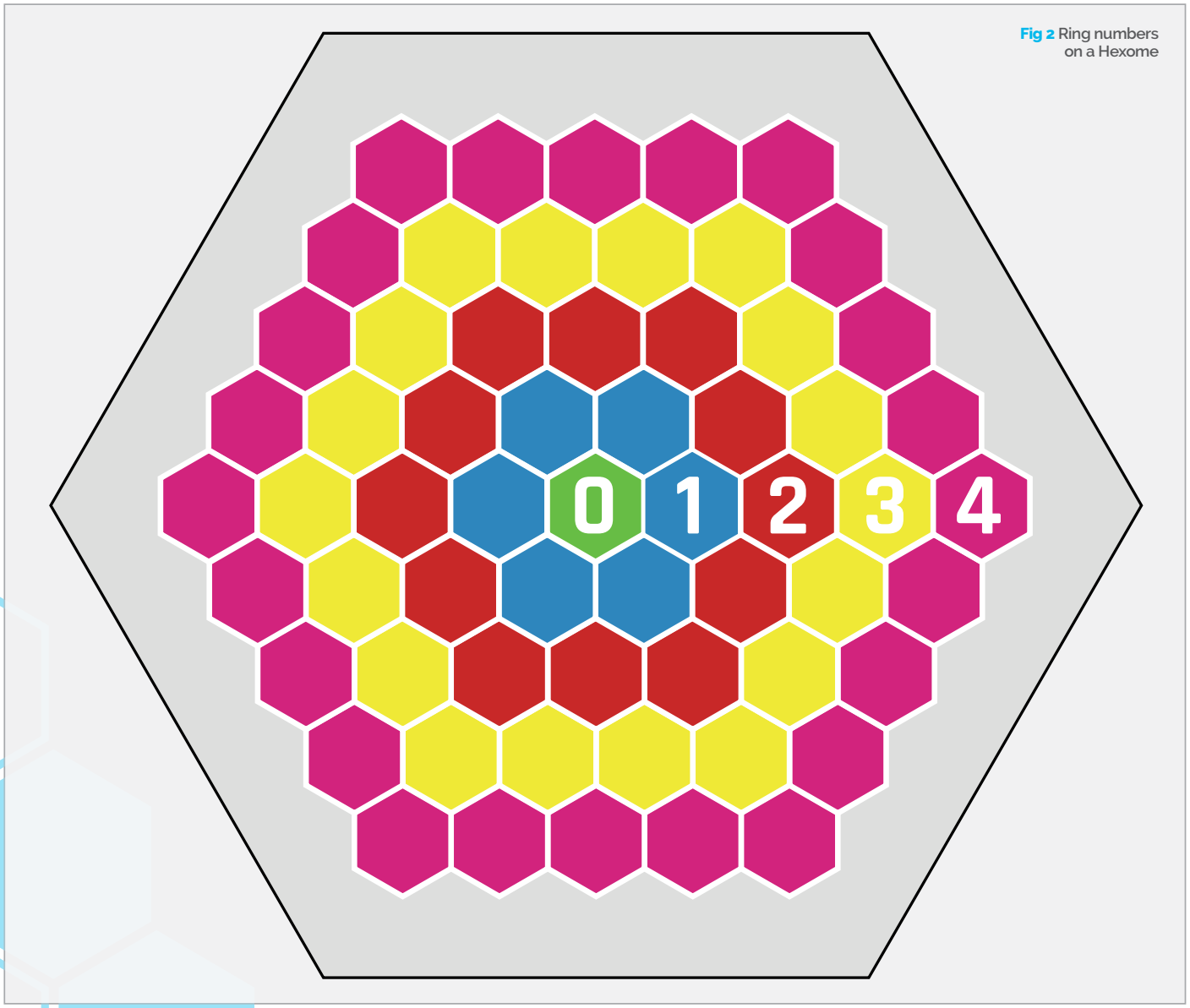
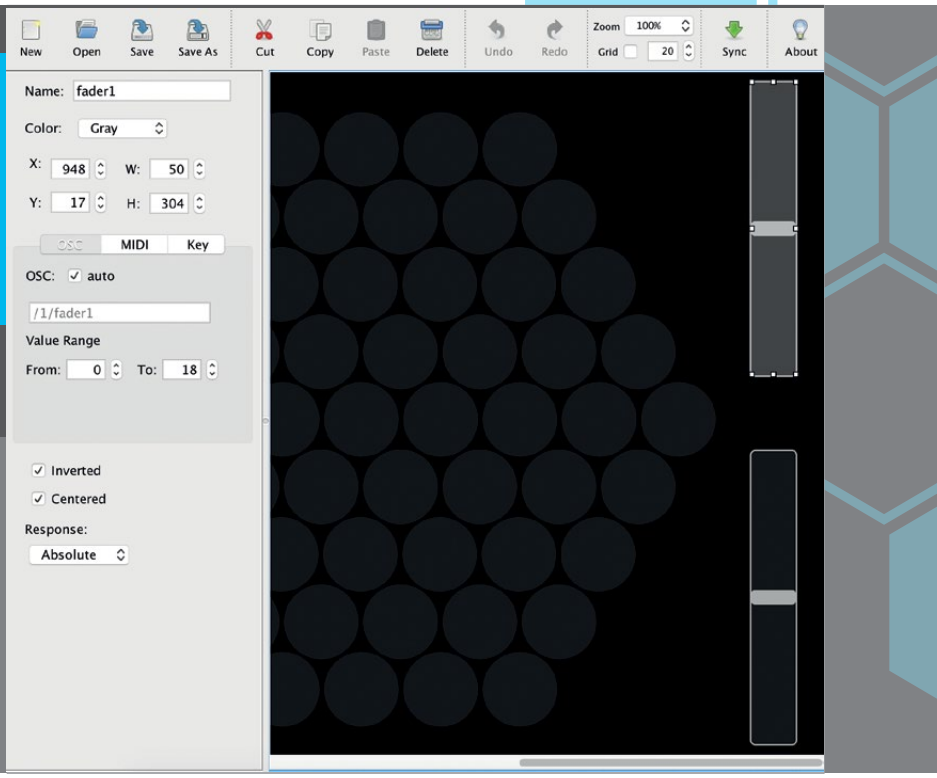


Fig 2 Ring numbers on a Hexome

CREATING THE HEXOME IN TOUCHOSC

>STEP-01
Set up the faders

If you haven't done so yet, install the TouchOSC editor: see last month's project for installation instructions and help with selecting the various TouchOSC elements. Set up two faders on the right-hand side, both 50 wide and 304 high. Fader 1 should have a range of 0 to 18 and fader 2 a range of 0 to -40. Both should have the inverted and centered boxes ticked. Fader 1 should be at X=948, Y=17 and fader 2 at X=948, Y=394.



Implementation

Again, as last month, we are going to use the Processing language. If you haven't done so already, install it by typing:

```
curl https://processing.org/download/
install-arm.sh | sudo sh
```

You also need to install two libraries. From the Sketch menu, go to the Import Library entry and choose Add Library, then search for oscP5 and install that. Then repeat and install the Minim library; this is the one that will play the sounds. If you try to run some of the examples from the Minim library, you will get an error complaining about the libgles2-mesa module and asking you to remove it. It is not necessary to do this for this project, only for running some of the examples. Note that if you do uninstall the libgles2-mesa, it is likely to affect the operation of Sonic Pi – and as we said, there is no need to remove this module for this project.

Download the ZIP of the code for this project from GitHub: **magpi.cc/1NqJjmV**. Note that while it can use TouchOSC for easy control, most of the functions can be accessed directly from the Pi through either a key press or a mouse click. Sadly, the TouchOSC does not have hexagonal LEDs, so we have to make do with circles.

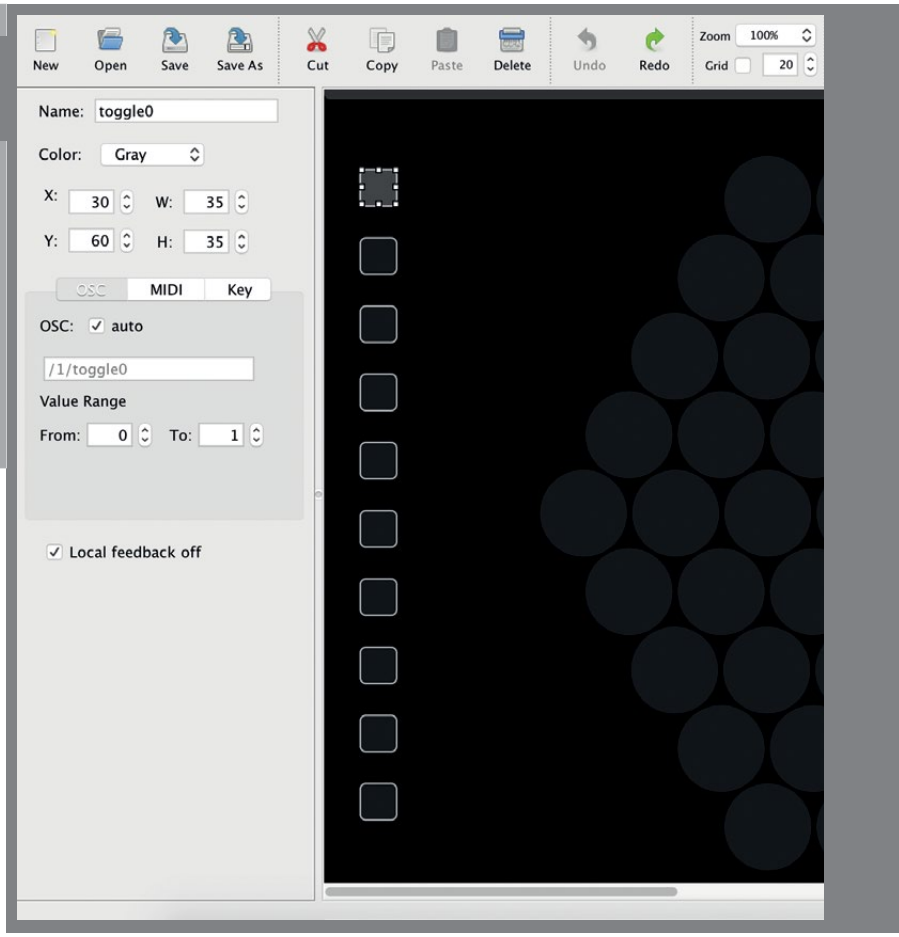
The samples

At each point in a sequence, a sample is triggered, or not, depending on whether the current key has been selected by previously clicking it. Each ring only triggers one sample, although the mapping between the rings and the samples they trigger can be changed by the push buttons on the left-hand side of the TouchOSC display or by the keys 0 to 9 on the keyboard (see step 2). This mapping is important for determining the type of melody you can produce. The first four change the mapping in thirds in the key of C for a harp, and the second four the same thing for a marimba. The last two are percussion samples and have no harmonic relationship. We found the best results were obtained with the smaller circles mapped to the lower notes. One important thing to note is that these samples can be played at the same time. If the sample from each waveform at any instance exceeds the 16-bit sample size, this will result in clipping of the waveform, which will sound like clicks

> At each point in a sequence, a sample is triggered, or not, depending on whether the current key has been selected by previously clicking it

>STEP-02
Set the toggle controls

Select a toggle control and size it to be 35 by 35. Place it at X=30 Y=60 and make it grey. Make sure the 'Local feedback off' box is checked. Copy and paste this, changing the Y location to 120. Repeat this for Y locations 180, 240, 300, 360, 420, 480, 540, and 600, giving you toggle buttons toggle0 to toggle9.



in the sound. To avoid this, the samples should be normalised, to a level of -10dB. You can do this by using the Audacity application and selecting Normalise in the Effects menu. This has already been done with the samples that come with this program, but you will have to do this if you want to add your own samples.

The controls

The speed of the sequence and its volume are controlled by two sliders on the right of the iPad screen. If you are not using a mobile device, then the Pi's keyboard can be used to change the speed. The **F** key makes it go faster and the **S** key slower. This changes the number of video frames that have to pass before the sequence is advanced. The other control, the clear button, is the central key of the display, which is set to red.

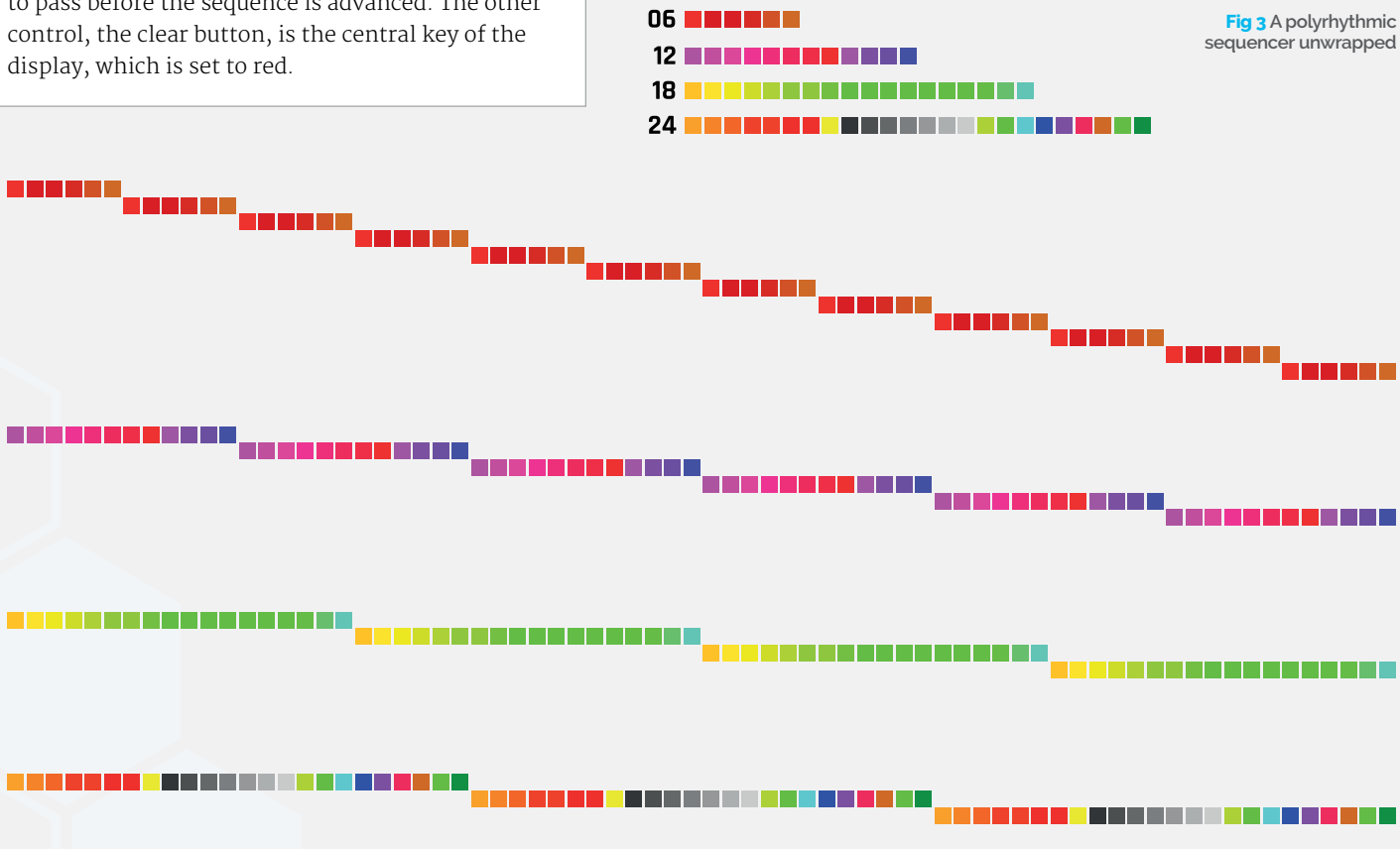
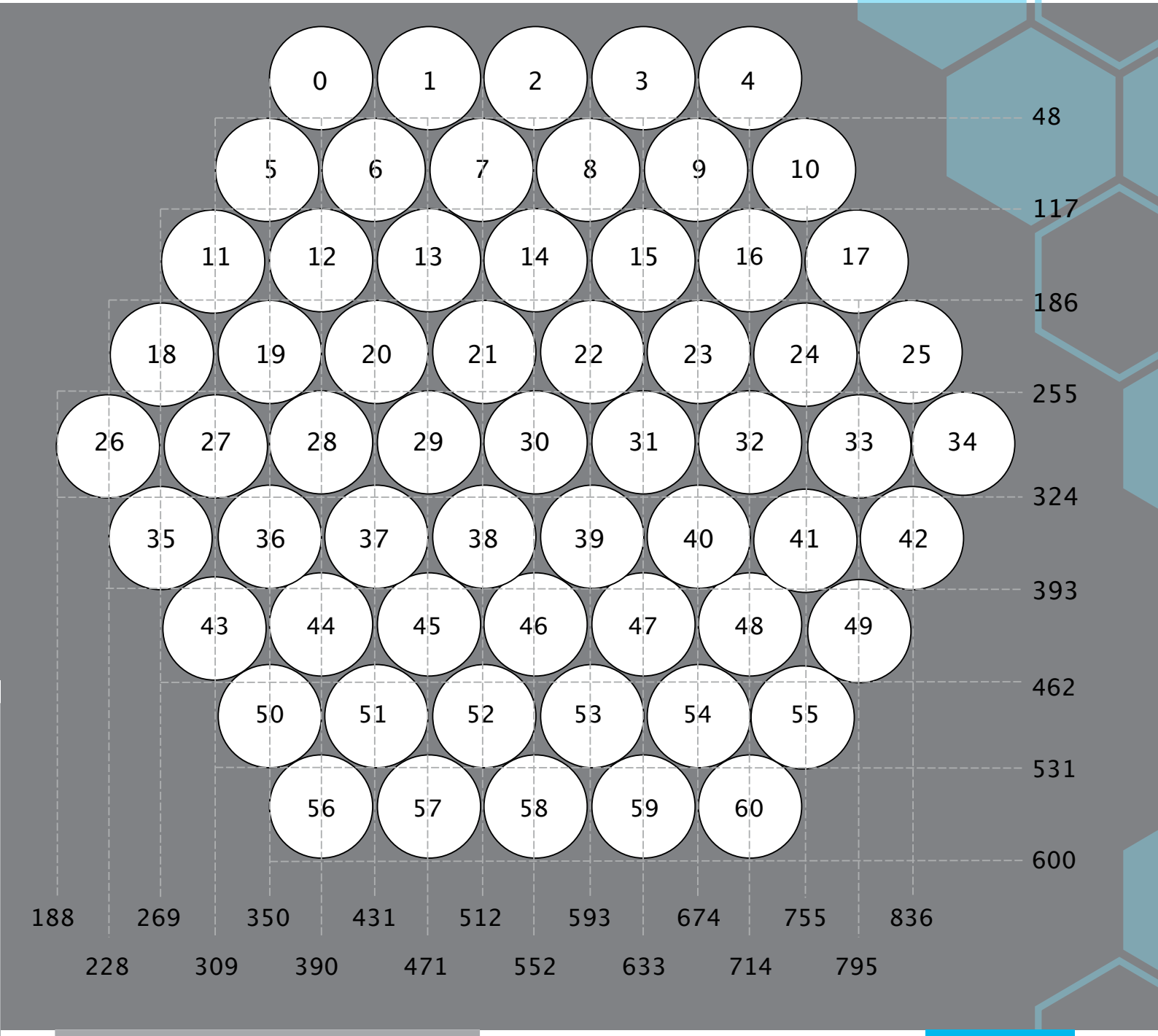


Fig 3 A polyrhythmic sequencer unwrapped



>STEP-03
Add the LEDs

Select an LED object and size to 80 by 80. Place it at X=350 Y=48, then copy and paste and change the locations to those shown in the diagram. Note that the X and Y values define the corner of a circumscribed rectangle around the circle; the LED number is shown in the circle. If you get mixed up, you can always relabel them after all the LEDs are created. Save the layout and transfer it to your mobile device: see last month for details.

Taking it further

You can add a keyboard volume control to the program if you like, but the major change you can make is to the samples. You can change the mapping, that is which sample corresponds to which ring, by changing the **noteLookUp** arrays in the 'notes' tab of the code. This is quite easy to do. More complex is to change the sample instruments, although this is not too difficult if you examine the code and see how the samples are loaded in. The sample names are just numbers, taken from the MIDI note number of the pitch of the sample.

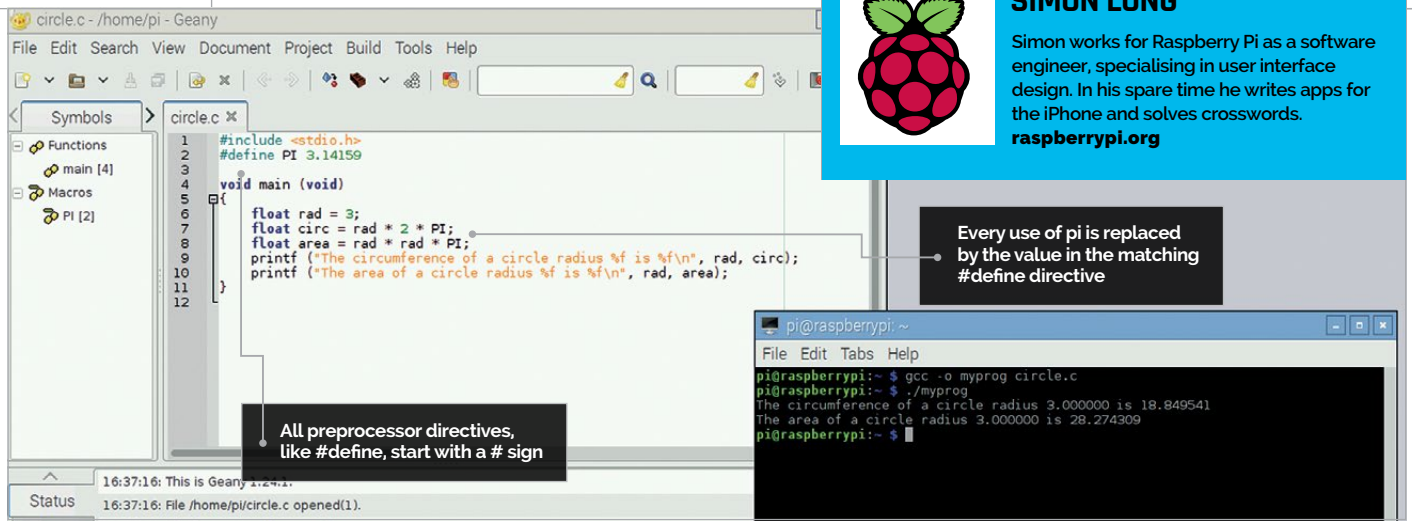
Language
>PROCESSING

DOWNLOAD:
magpi.cc/1NqJjmV

PROJECT VIDEOS

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

SIMON LONG
Simon works for Raspberry Pi as a software engineer, specialising in user interface design. In his spare time he writes apps for the iPhone and solves crosswords.
raspberrypi.org



KEEP NAMES CONSISTENT

While you can call header files whatever you like – there's no magic about the names – it's good practice to give the header file for the functions in a particular C file the same name as the C file itself, with a .h extension rather than .c. This makes it easier for someone reading your code to find the files where functions are defined.

#DEFINE FOR TEXT

If using **#define** for text strings, they should be enclosed in double quotes, otherwise the replaced text will end at the first space. So use **#define MY_TEXT "This is some text to replace."** The double quotes are included in the replacement, so you can then just call **printf (MY_TEXT);**

AN INTRODUCTION TO C PART 12

HEADER FILES AND THE PREPROCESSOR

Splitting code up into multiple files

The examples so far have put the code for a program in one file. But once programs get big, it's useful to split them up into separate files. To understand how this works, we need to look in more detail at what the compiler actually does.

Calling gcc on a single source file creates a single executable program. gcc actually does two things: it first compiles the C source file into an *object file*, and then it *links* the object file with the library functions to create the executable. This second step is performed by a program called a *linker*; gcc does both jobs.

If a program has multiple source files, you need to include the names of all the source files in the call to gcc. It will then create one object file for each source file, and then link all the object files together to create the executable.

However, if you've separated your code into separate files (usually referred to as *modules*), you'll have some files which make calls to functions in other files in order to work. These files don't find out about each other until the linker operates on them; the files are compiled individually, and the compiler will complain if you use functions in a file it doesn't know about.

We fix this using *header files*; files with the extension **.h** which hold the declarations of functions defined in a module, so that the compiler can be told about them when they're used by another module. We've already seen this; the line **#include <stdio.h>** at the top of the examples is telling the compiler that functions declared in the system header file **stdio.h** are used in this module.

Splitting code into multiple files

Here's an example. Create three files, two with the extension **.c** and one with the extension **.h**, as follows:

```
function.c

int add_vals (int a, int b, int c)
{
  return a + b + c;
}

function.h

extern int add_vals (int a, int b, int c);

main.c

#include <stdio.h>
#include " function.h"

void main (void)
{
  printf ("The total is %d\n",
    add_vals (1, 2, 3));
}
```

Inside the header file we declare the function with the word **extern** at the start of the declaration. This tells the compiler that this function is to be found externally to the file, i.e. in another C file.

Put all three files in the same directory and run **gcc -o myprog main.c function.c**.

The resulting program runs the **main** function in **main.c**, which calls the **add_vals** function in **function.c**.

The symbols around the name of the included file tell the compiler where that file is. **<>** signs tell the compiler to look for the file in the directory where the system's include files are stored; **""** signs indicate a local file in the same directory as the **.c** files, so use **""** signs around the names of your own header files when including them.

The preprocessor

#include is an instruction to the *preprocessor*: the first stage of compiling, which substitutes text within source files before passing them to the compiler itself. The preprocessor is controlled with *directives*, which start with a **#** sign.

#include instructs the preprocessor to replace the line with the file being included. So the line **#include "function.h"** in the **.c** file gets replaced with the contents of **function.h**, meaning that what's passed to the compiler is:

```
#include <stdio.h>
extern int add_vals (int a, int b, int c);

void main (void)
{
  printf ("The total is %d\n",
    add_vals (1, 2, 3));
}
```

#define

#define can be used to define constant values.

```
#include <stdio.h>
#define PI 3.14159

void main (void)
{
  float rad = 3;
  float circ = rad * 2 * PI;
  float area = rad * rad * PI;
  printf ("Area of circle radius %f: %f\n",
    rad, area);
  printf ("Circumference of circle: %f\n",
    rad, circ);
}
```

PI isn't a variable; it's text that will be substituted by the preprocessor. The **#define** tells the preprocessor to go through the file and replace every instance of the symbolic constant **PI** with the digits **3.14159** before passing it to the compiler. (A line which does something like **PI = 5;** will cause an error; the compiler will see the meaningless statement **3.14159 = 5;**.)

You can also **#define** functions:

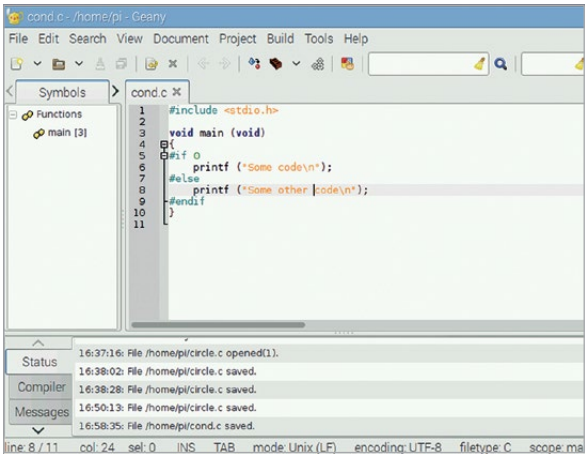
```
#include <stdio.h>
#define ADD(a,b) (a+b)

void main (void)
{
  printf ("The sum of %d and %d is %d\n",
    5, 2, ADD(5,2));
  printf ("The sum of %d and %d is %d\n",
    3, 7, ADD(3,7));
}
```

Whenever **ADD(a,b)** appears in the code, it's replaced by **(a+b)**, with the values of **a** and **b** replaced by the arguments to **ADD**.

The preprocessor can also evaluate conditions with **#if**:

```
#include <stdio.h>

void main (void)
{
#if 0
  printf ("Some code\n");
#else
  printf ("Some other code\n");
#endif
}
```

With a 0 after the **#if**, the code between the **#if** and the **#else** doesn't get called, but the code between the **#else** and the **#endif** does. If you change the value after the **#if** to a 1, the code between the **#if** and the **#else** does get called, but the code between the **#else** and the **#endif** doesn't. This is useful to temporarily remove or replace a piece of code when you're debugging.



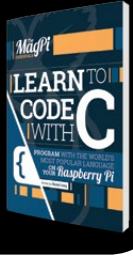
Left The most common use of **#if** is for temporarily removing code – just wrap it between an **#if 0** and an **#endif**. The **#else** is optional, but sometimes you want to substitute the code you've removed with different code

MAKEFILES

As you can imagine, if you have a project with tens or hundreds of C files, typing all their names in the call to gcc every time would be a bit tedious! Large projects are built with a tool called 'make', which stores build instructions in a 'Makefile'. Makefiles are beyond the scope of this guide, but there's lots of information about them online.

LEARN MORE ABOUT C

If you've been enjoying these introduction to C tutorials, then check out our *Learn to Code with C* book for more! **magpi.cc/learn-c-book**



RIK CROSS
Rik is the Senior Content and Curriculum Manager for Code Club.
codeclub.org.uk

As when using say, the ask command results in a speech bubble

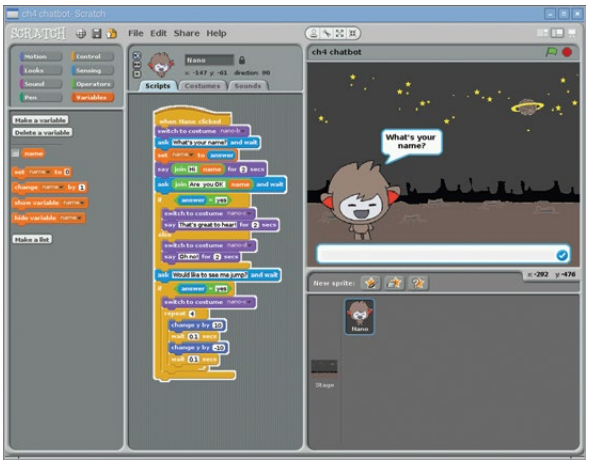
The Nano sprite has four costumes, which are alternated to animate him

The ask command also brings up a text input field for the user to enter their answer

CHATBOT

Nano the cute robot loves to chat. He'll respond to your answers, and he'll even jump up and down if you ask him to...

For this project, you'll be creating your own talking robot which responds to your text input. We'll also alter his expression by switching between different costumes. We'll be using **ask** commands, **if…else** blocks, and the **join** Operator. We'll also create a variable to store the user's name – variables are really handy for storing values to use elsewhere. That's enough chitchat – let's start up a new Scratch project...



Above: We create a variable to store the user's name and then repeat it within Nano's speech

>STEP-01

Prepare your artwork

After deleting the Scratch cat by right-clicking on it and selecting Delete, it's time to import a new stage background (space) and our character sprite (Nano). You can either obtain these from the Scratch 2.0 library or, in Scratch 1.4, download them from **magpi.cc/scratch_art**. In the latter case, click Stage in the Sprite List, select the Backgrounds tab, then click Import and navigate to **Space.png** in the folder where you've stored the downloaded graphics for this project. Next, click the star/folder icon above the Sprite List, then navigate to the same folder and import **Nano.sprite**. If you click the Costumes tab, you'll notice that Nano has four of them; we'll switch between them to animate our little robot friend.

>STEP-02

Ask for a name

First, we'll get our robot to ask for the user's name and then use it in a response. With the Nano sprite selected, click the Scripts tab (top middle) and add the code from **Listing 1**. Note that instead of using a standard **when green flag clicked** block, we're starting the program when the Nano sprite is clicked. He then asks for the user's name, which is stored in a variable called **name**.

Above: By switching between four costumes, we can alter our character's facial expression

First, we need to create the latter: select Variables from the top left, then click 'Make a variable', 'For this sprite only', and enter 'name' in the text field. Untick the **name** block to stop it showing on the stage. We can now set **name** to **answer** (the user's text input) and then add it into Nano's response by using the **join** Operator block. Make sure you put a space after 'Hi' to avoid it being joined together with the name.

>STEP-03

Add a question

Next, we'll add some more blocks from **Listing 2** to the bottom of this script. After saying 'hi' to them, Nano asks the user if they're OK. Again, we use the **ask** Sensing block for this, and the **name** variable to refer to them by name. We then use an **if…else** Control block to determine Nano's response based on the user's input. If it's 'yes' (which we test for using the **=** Operator) we switch Nano's costume to happy nano-c, using the drop-down box on his Looks block. We also get him to say 'That's great to hear!'

>STEP-04

Else this…

In the **else** part of the **if…else** block, we determine what happens if the user's input isn't 'yes'. In this case, we'll switch Nano's costume to the frowning nano-d and get him to say 'Oh no!' Test out this code with different inputs to check that it's working as expected. Note that while the user's text input isn't case sensitive, it has to be just 'yes', with nothing added, in order to be recognised as such.

>STEP-05

Jump up and down

Finally, we'll add another question with **ask**, using a standard **if** block to make Nano jump up and down or not. Add the blocks from **Listing 3** to the script. We use a **repeat** loop to make Nano move repeatedly up and down, to jump. To make sure he's not frowning from the previous response while doing so, we switch his costume to nano-c before the **repeat** loop.

>STEP-06

Taking it further

You can alter the example questions or add any extra ones you want, even getting Nano to tell a joke. You could also add extra costumes by copying and editing them in the Paint Editor, or even a design a brand new sprite with various costumes.

.01 Language >SCRATCH

```
when Nano clicked
switch to costume nano-b
ask What's your name? and wait
set name to answer
say join Hi name for 2 secs
ask join Are you OK name and wait
```

.02

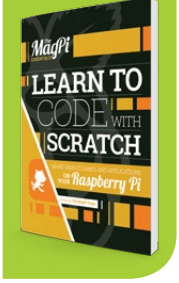
```
if answer = yes
  switch to costume nano-c
  say That's great to hear! for 2 secs
else
  switch to costume nano-d
  say Oh no! for 2 secs
```

.03

```
ask Would like to see me jump? and wait
if answer = yes
  switch to costume nano-c
  repeat 4
    change y by 10
    wait 0.1 secs
    change y by -10
    wait 0.1 secs
```

LEARN MORE SCRATCH!

Want to make other Scratch projects? Then check out our *Learn to Code with Scratch* book for many more ways to use Scratch: **magpi.cc/Scratch-book**



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.

RASPBERRY PI AUDIO

WHAT AUDIO OUTPUTS ARE AVAILABLE ON THE RASPBERRY PI?

Analogue

The main audio output on the Raspberry Pi is the 3.5 mm headphone jack, which also supports video out. This provides analogue audio out when the task requires it. Normal headphones and mini speakers that connect with a headphone cable can be connected to the headphone jack.

Digital

The HDMI port provides a digital audio output, and is also the main video out port. It's a bit tricky to split up the audio and video signal without special equipment, but it's the highest quality audio out on the Raspberry Pi by default.

Bluetooth

Bluetooth speakers or headphones connect to older models of the Raspberry Pi via dongles, but connecting via Bluetooth has become a lot easier since the introduction of the Raspberry Pi 3 and its radio chip. You will need to take some steps to get it working.

HOW CAN I CONNECT A BLUETOOTH SPEAKER?

Connect the speaker

Next to the wireless LAN icon on the toolbar is the Bluetooth logo for managing Bluetooth devices. Start the syncing sequence on your Bluetooth audio device and then scan for devices in this manager to connect to your speakers or headphones.

Get PulseAudio

Install PulseAudio with **`sudo apt-get install pulseaudio pavucontrol pulseaudio-module-bluetooth`** in the Terminal. You'll need it to connect the audio to the Bluetooth speakers. To be safe, you should now reboot your Raspberry Pi.

Play over Bluetooth

Reconnect your audio device after reboot and check the Volume Control option in Sound & Video in the main menu. On output devices, it should list 'bcm2835 ALSA' as the default output. Pick the Bluetooth audio device once it has connected, and you're ready to go.

CAN I IMPROVE THE AUDIO?

Upgrading on-board audio

You can't upgrade the standard HDMI or the 3.5 mm jack on the Raspberry Pi, as they are integrated parts of the system. It is unlikely that you will need to update the HDMI, as it's a high-quality digital output.

Add-on board

There are several add-on boards for the Raspberry Pi that vastly improve the audio output. These DAC HATs are made by various firms and you can buy them at online retailers such as Pimoroni, The Pi Hut, etc.

Recording

Currently there's no way to actually record audio on the Raspberry Pi unless you plug in a USB device or HAT that includes a microphone. This is why the Pi Camera Module doesn't record sound when you're filming with it.

FROM THE RASPBERRY PI FAQ
RASPBERRYPI.ORG/HELP

What is the Camera Module?
The Camera Module is a small PCB that connects to the CSI-2 camera port on the Raspberry Pi using a short ribbon cable. It provides connectivity for a camera capable of capturing still images or video recordings. The camera connects to the Image System Pipeline (ISP) in the Raspberry Pi's SoC, where the incoming camera data is processed and eventually converted to an image or video on the SD card (or other storage). You can read more about the Camera Module here: **magpi.cc/28ljlsz**.

What model of camera does the Camera Module use?
The Camera Module v2 uses a Sony IMX219 image sensor, while the original Camera Module has an Omnivision 5647. They are comparable to cameras used in smartphones.

What resolutions are supported?
The Camera Module v2 is capable of taking photos up to 8 megapixels (8MP). It supports 1080p30, 720p60, and VGA90 video modes, as well as still capture. The original Camera Module is capable of taking photos up to 5 megapixels and can record video at resolutions up to 1080p30.

Which picture formats are supported?
The Camera Module supports raw capturing (Bayer data direct from the sensor) or encoding as JPEG, PNG, GIF and BMP, uncompressed YUV, and uncompressed RGB photos. It can record video as H.264, baseline, main, and high-profile formats.

How do I use the camera?
There are three command-line applications provided for stills, video, and stills output uncompressed. These applications provide the typical features you might find on a compact camera, such as image size, compression quality, exposure mode, and ISO. See the documentation for more details: **magpi.cc/2r60l1z**.

AIY PROJECTS FOR DIGITAL MAKERS

Move beyond the Google Assistant SDK and put AIY Projects at the heart of your digital making

Last month, Google and *The MagPi* made history by bundling a complete AIY Projects kit on our cover.

This kit is the first do-it-yourself artificial intelligence project to be launched by Google; more are in development. With it, you can add voice commands and responses to your projects.

WITH GOOGLE

On the AIY Projects website, you will find the latest build of the AIY Projects software, as well as the most recent updated instructions. Take a look at the Maker's Guide in the AIY Projects website for up-to-date information.

aiyprojects.withgoogle.com/voice

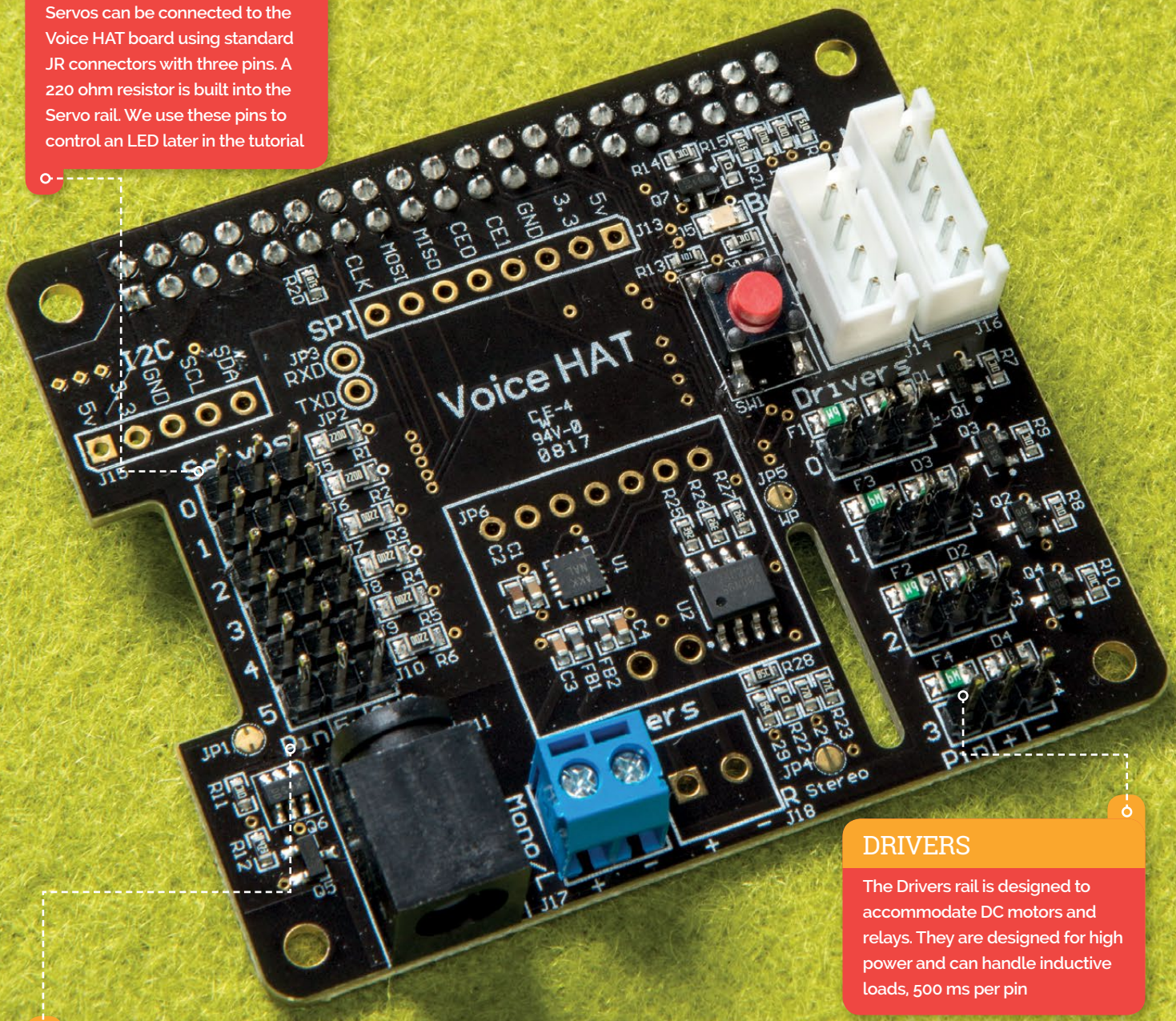
The starter project for the kit has you setup the Google Assistant SDK to create a digital assistant. Push the button on the top of the Voice Kit, ask your AIY Projects kit questions, and the Google Assistant will provide answers.

We hope you had a lot of fun building this starter project for using voice interactions, but it's important not to stop there. There's a lot you can do with this project beyond the Google Assistant SDK. For the curious maker, we have some ideas on how to hack AIY Projects, as well as how to use the Cloud Speech API as an alternative to the Google Assistant SDK. You can also set up Android Things as an OS.

In this feature, we're going to look at some of the things you can do with the AIY Projects voice kit.

SERVOS

Servos can be connected to the Voice HAT board using standard JR connectors with three pins. A 220 ohm resistor is built into the Servo rail. We use these pins to control an LED later in the tutorial



DRIVERS

The Drivers rail is designed to accommodate DC motors and relays. They are designed for high power and can handle inductive loads, 500 ms per pin

PIN GUIDE

Below the Servos and Drivers pins are guides to the function of each pin. The Servo pins are Pin, 5V, and GND; the Drivers pins are Pin, +, and -

WANT TO BUY AIY PROJECTS?

Not everybody was lucky enough to get a free AIY Projects voice kit with *The MagPi #57*, and many readers have asked whether it is possible to purchase the kit separately. Sign up for our newsletter and we'll let you know when more AIY Projects Voice Kits are available. Head to our website and enter your email address.

magpi.cc

SET UP CUSTOM VOICE COMMANDS

Google Assistant SDK

Set up Cloud Speech API to create your custom commands

AIY Projects voice is a hackable project, so we encourage you to make this project your own. This guide gives you some creative extensions, settings, and even a different voice API to use. We hope this project sparks some new ideas for you.

First, though, you need to move from using the Google Assistant SDK (the one that answers general questions) to the Cloud Speech API.

>STEP-01

Cloud Speech API

One interesting project for makers is to create custom commands for the AIY Projects Voice HAT. These can do just about anything with your Raspberry Pi. While it's possible to add new actions and voice commands using the Google Assistant SDK, more options are available if you switch to the Cloud Speech API. This software recognises your voice speech and converts it into text. The Cloud Speech API supports 80 languages, extended audio clips, and the ability to add phrase hints for processing audio.

>STEP-02

Turn on billing

To use Google's Cloud Speech API, you need to activate Billing – see page 28 of *The MagPi* #57 or **magpi.cc/2q5SSF7**. If you use it for less than 60 minutes a month, it's free. Beyond that, the cost is $0.006 for 15 seconds. Don't worry: you'll get a reminder if you go over your free limit.

Open the web browser and click on Google API Console link (or visit **console.developers.google.com**). In the API Manager window, click Enable API and search for Google Cloud Speech API. Click it and click Enable in the top of the window (if it says 'Disable', then it is already enabled and you're ready to go).

>STEP-3

Credentials

Choose Credentials in the sidebar. Click on Create Credentials and select Service Account Key. From the Service account menu, pick the name of your project (if you already have one) or select 'New service account'. Enter a name so that you'll know this is for your voice recognizer stuff, like 'Voice credentials'. Select 'Project viewer' as the role.

Select JSON as the key type and click Create.

>STEP-04

Rename credentials

The credentials file is downloaded automatically. The file name contains your project name and some numbers (like 'aiyproject-c92d36fc7055.json'). Open a Terminal window and move it to your home folder, and rename it to **cloud_speech.json**:

```
cd Downloads/

mv aiyproject-c92d36fc7055.
json /home/pi/cloud_speech.
json
```

>STEP-05

Check Cloud

On your desktop, double-click the Check Cloud icon. Follow along with the script. If everything is working correctly, you'll see this message: 'The cloud connection seems to be working.'

If you see an error message, try restarting your Raspberry Pi

with **sudo reboot**. Then follow the instructions above, or take a look at the instructions on the AIY Projects page (**magpi.cc/2q5SSF7**).

>STEP-06
Config file

The application is configured by adjusting the properties found in **/home/pi/.config/voice-recognizer.ini**. This file lets you configure the default activation trigger and choose which API to use for voice recognition. Don't worry if you mess it up: there's a backup copy kept in **/home/pi/voice-recognizer-raspi/config**.

>STEP-07
Adjust the config

Open a Terminal window and enter the following:

```
nano /home/pi/.config/voice-recognizer
```

Delete the **#** before **cloud-speech = true**.

Now press **CTRL+O**, press **ENTER**, and **CTRL+X** to save the file and exit Nano.

>STEP-08
Start it up

Double-click the 'Start dev terminal' icon and enter:

```
src/main.py
```

The Cloud Speech API will start. This API works like the Assistant SDK, but instead of answering general questions it responds to specific commands. Say out loud:

"What are the three laws of robotics?"

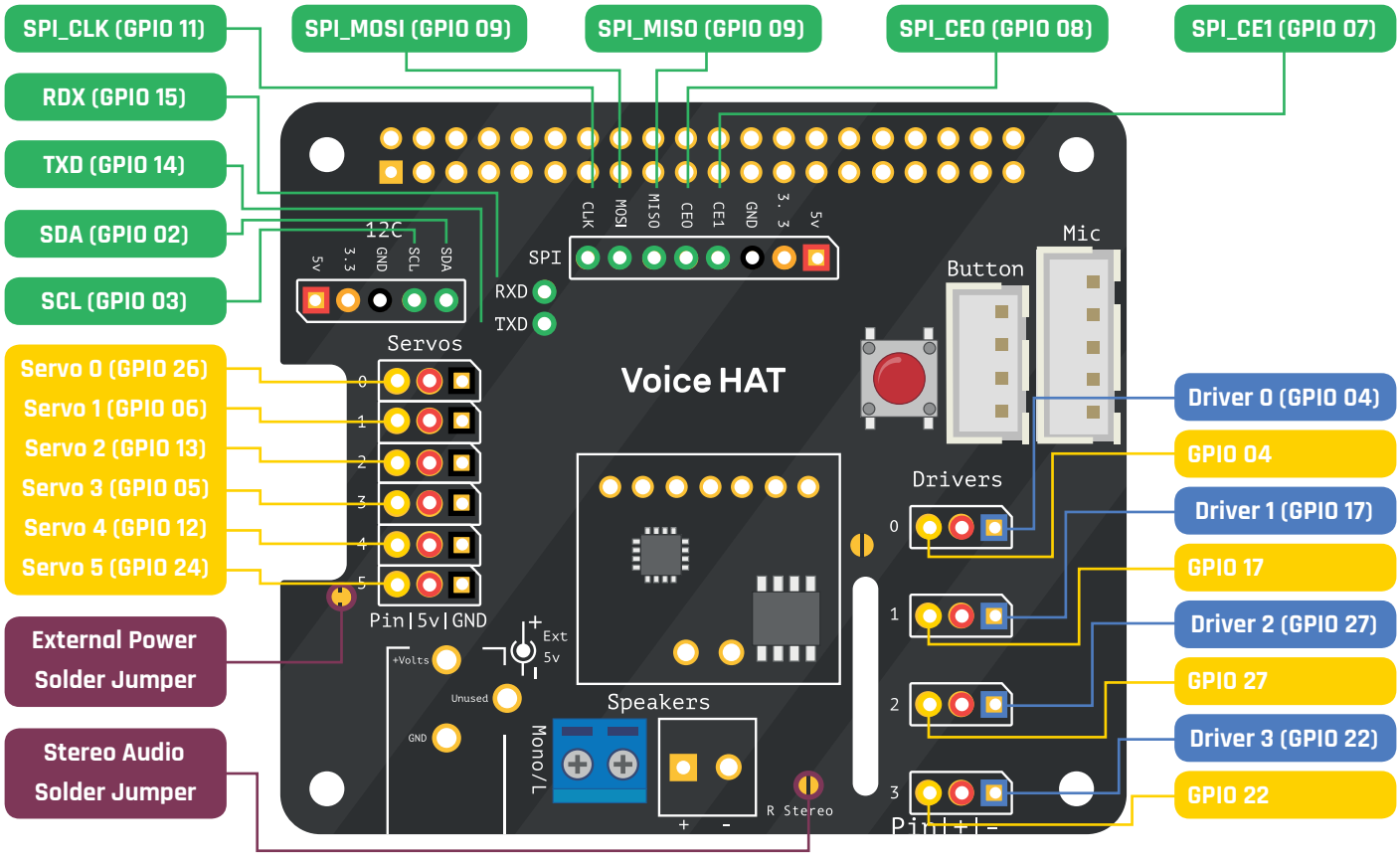
It will read the response from Issac Asimov's famous collection of robot books. This a pre-built example speech. The response is not part of the Assistant SDK, but a specific response (an action) to a voice command. You can create specific voice commands and actions.

>STEP-09
Back up action.py

You create new actions and link them to new voice commands in **/home/pi/voice-recognizer-raspi/src/action.py**. We think it's a good idea to back up this file before editing it.

```
cd /home/pi/voice-recognizer-raspi/src/

cp action.py action_backup.py
```

VOICE HAT HARDWARE EXTENSIONS



CREATE ACTIONS

Google Cloud Platform

Edit the actions file to create custom voice commands for AIY Projects

You'll Need

- AIY Projects voice kit
- Cloud Speech API
- Breadboard
- LED, resistor, and cables

You can remove the kit's big arcade button and use the small red button on the board to activate the assistant

Now that you've switched from the Assistant SDK to the Cloud Speech API, you'll want to know what you can do with it. You add custom commands to the **action.py** file. There is a selection of example voice commands located right at the end of the code. These include the three laws of robotics that we used when testing the Cloud Speech API.

>STEP-01

Edit action.py

You can edit the **action.py** file in Nano, but we think it's better to use Python IDLE. Enter:

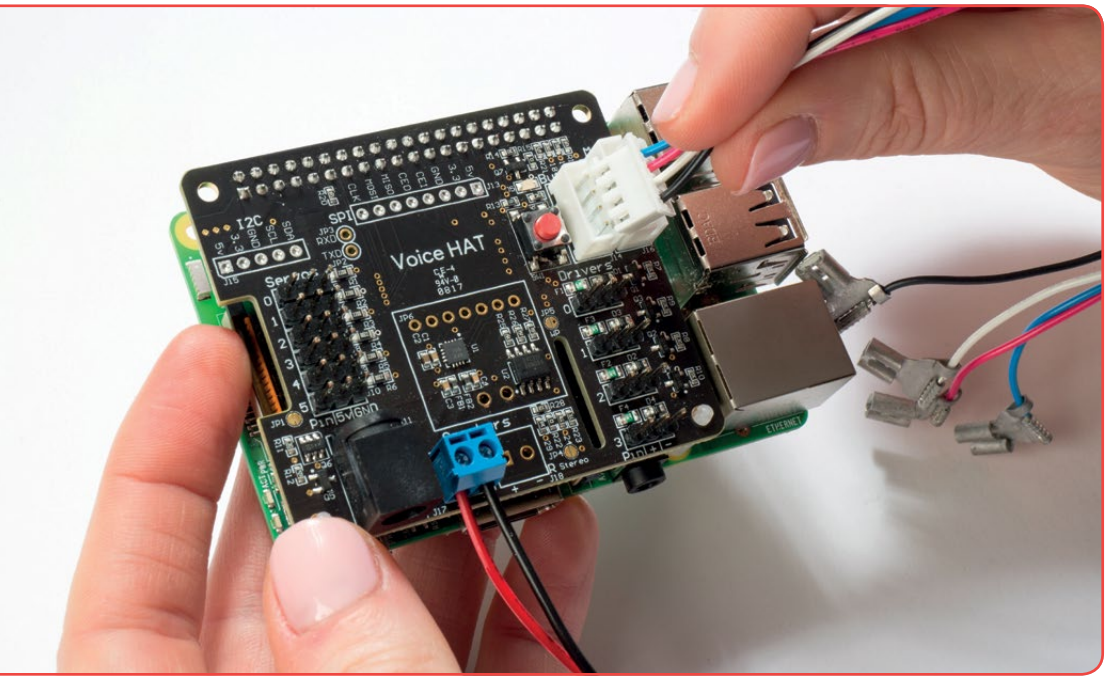
```
xdg-open /home/pi/voice-recognizer-raspi/src/action.py
```

This command opens the program in the Python IDLE window with the code marked up in colours, making it much easier to read.

Scroll through and read the example functions. At the end, you'll see a section with:

```
#============================

#Makers! Add your own voice commands here.

#============================
```

Locate the function starting with **def_add_commands_just_for_cloud_speech(actor,say)**. This function contains the custom commands used in Cloud Speech API.



>STEP-02

Add a keyword

We're going to add a simple keyword that creates a custom voice response. Scroll to the very end of the document and enter (with an indent):

```
    simple_command(_('meaning of life'),_('The answer to the great question of life, the universe, and everything is 42.'))
```

Now start up the assistant. Double-click 'Start-dev terminal' and enter:

```
src/main.py
```

Say: **"What is the meaning of life?"**

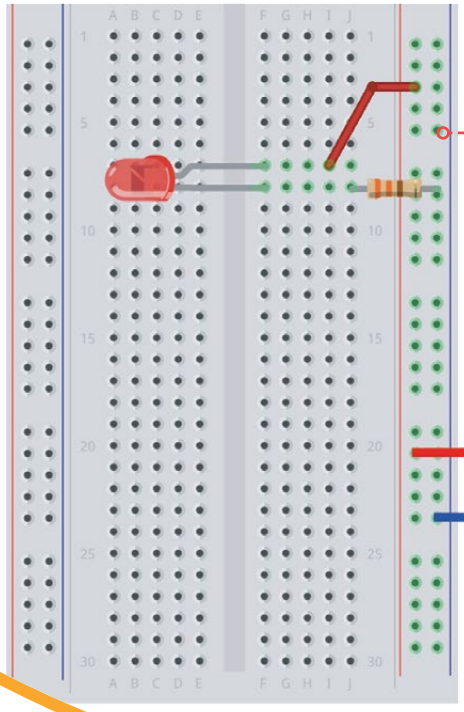
Google Cloud Speech API will detect your keywords and read out the classic quote from Douglas Adams's *The Hitchhiker's Guide to the Galaxy*.

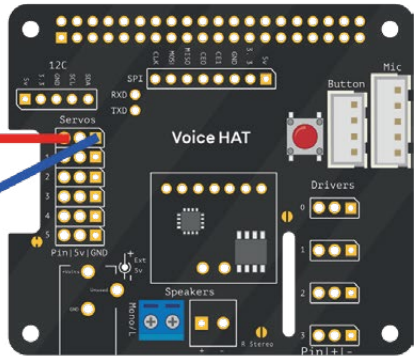
>STEP-03

Control an LED

Now that we can create actions, we're going to use the AIY Projects kit to control some hardware. Set up an LED circuit using a breadboard – follow the diagram shown on the next page. We are connecting the LED via the pins on Servo 0. Connect the live wire to Pin (on the left). This is GPIO 26 using the BCM numbering system. Connect the ground wire to GND (on the right). The middle pin provides a constant 5V of power. You can see the reference for each pin underneath the Servo 5 rail (check the diagram in 'Voice HAT hardware extensions' on p69 for reference).

SET UP AN LED CIRCUIT





CIRCUIT

Connect an LED to the breadboard and create a circuit (with the longer leg connected to live and the shorter leg connected to ground). Don't forget to use a resistor (around 330 ohms) to protect the LED

GPIO 26

Connect the live wire to GPIO 26, the leftmost pin on Servo 0, and the live rail on the breadboard. See the GPIO layout guide from the previous page for guidance

GND

Connect the ground wire to the GND pin on the Servo 0 rail and the ground rail on the breadboard

We have found that it will work by connecting wires directly to the through-holes on the board. For a more reliable circuit, however, solder the pins supplied with your Voice HAT.

>STEP-04

Add an action

We need to open and edit the **action.py** file again. Enter this command in a Terminal window:

```
xdg-open /home/pi/voice-recognizer-raspi/src/action.py
```

Scroll down and look for this comment:

```
# ============================

# Makers! Implement your own actions here.

# ============================
```

Add the code from **own_actions.py** (page 73) underneath the comment.

>STEP-05

Voice command

Then add the code from **voice_commands.py** (page 73) to **~/voice-recognizer-raspi/src/action.py** below the comment 'Add your own voice commands here'. Make sure it indents with the comment, like this:

```
    # ======================

    # Makers! Add your own voice commands here.

    # ======================

    actor.add_keyword('light on', GpioWrite(26, True))
```

In **voice_commands.py** we've created actors for both "light on" and "lights on". We find this makes recognition more reliable. Add the rest of the code, then select File > Save and exit IDLE.

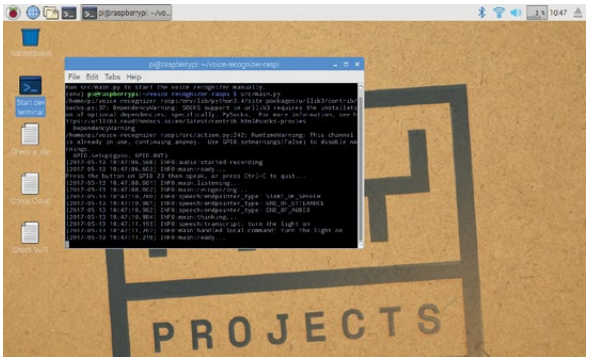
>STEP-06

Light on

Double-click 'Start dev terminal' and start up the assistant:

```
src/main.py
```

Now say "light on" slowly and clearly. If everything is connected correctly, your LED will light up. Say "light off", and it will turn off again. You can also use variants such as "hey, light on" and "turn the lights on".



Use code to control GPIO pins via your assistant. This short program turns an LED light on or off

MAKE A VOICE-CONTROLLED CHESS GAME

Create a working chess program with AIY Projects

You'll Need

- AIY Projects voice kit
- GNU Chess

One of the key things with AIY Projects is that it allows you to add human voice interaction to your projects. So you can move beyond using buttons (real or virtual) and mouse clicks. With the Cloud Speech API, you can talk directly to your Raspberry Pi, and have a natural conversation with your devices.

The hope is that in the near future, you'll be able to walk up to a device and say "what are you?" and "how do I use you?" and have a conversation with it.

We are delighted to find *The MagPi* readers already creating code for AIY Projects that interacts with devices, such as this program by reader Mike Redrobe (**magpi.cc/2q8NW20**). It hacks together AIY Projects with GNU Chess and is an excellent example of building an interface that controls a program.

Quick Tip

SOURCE CODE

If you're using the SD card image provided by AIY Projects, the source for the voice-recognizer app is already installed on your device. You can browse the Python source code at **/home/pi/voice-recognizer-raspi/**. Alternatively, the project source is available on GitHub: **magpi.cc/2pH04KQ**. Browsing code such as **main.py** and **action.py** is the best way to get a feel for what the code is capable of doing.

>STEP-01

Commands

You can issue Terminal commands via Cloud Speech API. Let's test this out by adding a handy function to the AIY Projects voice kit: the ability to shut down and restart the kit with a voice command.

In **shutdown.py** (page 73) there are two `actor.add_keyword()` functions. These contain 'shut down' and 'restart' as the command words. Enter the code from **shutdown.py** into **action.py**, as you did with **voice_commands.py**.

Start up the assistant (enter `src/main.py` in 'Start dev terminal'). Now when you use Cloud Speech API, you can press the button and say "shut down". The AIY Projects kit will run the shutdown scripts and power off safely. Say "restart" to restart your AIY Projects kit.

>STEP-02

Install GNU Chess

We are going to take this idea of command-line interaction to greater lengths by setting up the Cloud Speech API to work with GNU Chess for a command line-based chess game. The first step is to install GNU Chess on your Raspberry Pi.

```
sudo apt-get update

sudo apt-get install gnuchess
```

GNU Chess is played from the command line, although often it will be used with a graphical interface such as XBoard. But we're going to use AIY Projects to enter the moves in the command line.

>STEP-03

AIY chess

Now add the code from **aiy_chess.py** (page 73) to the **action.py** file (in **home/pi/voice-recognizer-raspi/src/action.py**). This code adds interaction with GNU Chess to your Cloud Speech API. Enter the code and start the assistant. Double-click 'Start dev terminal' and enter:

```
src/main.py
```

Now press the button on your AIY Projects voice kit to make a move.

>STEP-04

Playing chess

You say "chess D2 D4" to play a move. You will see a small chessboard displayed in text on the AIY Projects screen. The AIY Projects kit will then speak the computer opponent's move (and show it on the text chessboard). You can also be lazy and say "chess D4" if there is only one piece that could move to D4.

aiy_chess.py

```
# ========================================
# Makers! Implement your own actions here.
# ========================================

p = None
class playChess(object):

    def __init__(self, say, keyword):
        self.say = say
        self.keyword = keyword

    def run(self, voice_command):

        move = voice_command.replace(self.keyword, '', 1)

        global p
        if (p == None):
           p = subprocess.Popen(["/usr/games/gnuchess","
               -q"],stdin=subprocess.PIPE,stdout=subprocess.PIPE)

        p.stdin.write(bytes(move.lower() + '\n', 'utf-8'))
        p.stdin.flush()

        response = ""
        if all(x.isalpha() or x.isspace() for x in move):
           # no numbers (d2d4) so its a command like new,undo,remove
           response = p.stdout.readline().decode("utf-8")
           response = "ok," + move
        else:
            self.say("ok, you played," + move)

            while ("move" not in response):
              response = p.stdout.readline().decode("utf-8")
              logging.info("Chess log: %s", response)

        logging.info("Chess: %s", response)
        self.say(response)

    # ========================================
    # Makers! Add your own voice commands here.
    # ========================================

    actor.add_keyword(_('chess'), playChess(say,_('chess')))
```

voice_commands.py

```
# ========================================
# Makers! Add your own voice commands here.
# ========================================

actor.add_keyword('light on', GpioWrite(26, True))
actor.add_keyword('lights on', GpioWrite(26, True))

actor.add_keyword('light off', GpioWrite(26, False))
actor.add_keyword('lights off', GpioWrite(26, False))
```

own_action.py

```
# ========================================
# Makers! Implement your own actions here.
# ========================================

import RPi.GPIO as GPIO

class GpioWrite(object):

    '''Write the given value
                to the given GPIO.'''

    def __init__(self, gpio, value):
        GPIO.setmode(GPIO.BCM)
        GPIO.setup(gpio, GPIO.OUT)
        self.gpio = gpio
        self.value = value

    def run(self, command):
        GPIO.output(self.gpio, self.value)
```

Quick Tip

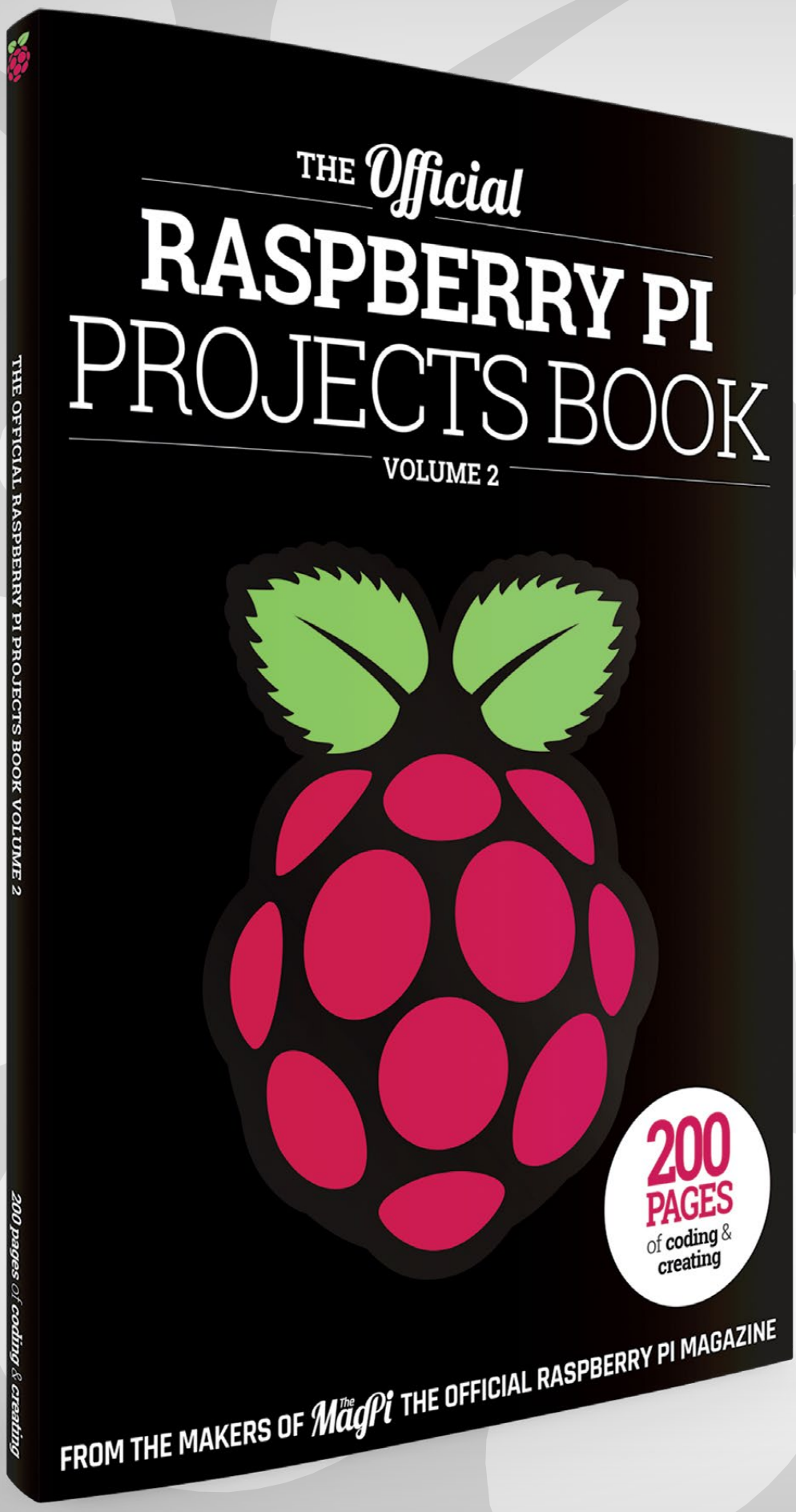
SOLDER YOUR PINS

The Voice HAT works more reliably with the pins soldered in the holes. A strip of pins will have been supplied with your AIY Projects kit.

If you want a rough guide to soldering, here is a very handy instructional comic (PDF): **magpi.cc/2pUgMIr**. It's an excellent visual reference and a cool thing to put on the wall. Be careful using the board with soldered pins, as it's easy to get a 5V shock by touching the positive and negative pins. Power down the kit before connecting and disconnecting wires.

shutdown.py

```
    # ========================================
    # Makers! Add your own voice commands here.
    # ========================================

    actor.add_keyword(_('shut down'), SpeakShellCommandOutput(say, "sudo shutdown -h now",_('Shutdown failed')))
    actor.add_keyword(_('reboot'), SpeakShellCommandOutput(say,"sudo shutdown -r now",_('Reboot failed')))
```


THE Official
RASPBERRY PI
PROJECTS BOOK
VOLUME 2
THE OFFICIAL RASPBERRY PI PROJECTS BOOK VOLUME 2
200 pages of coding & creating
200
PAGES
of coding &
creating
FROM THE MAKERS OF The MagPi THE OFFICIAL RASPBERRY PI MAGAZINE

Maker Says

"Pocket-sized fun is the name of this game – our most fun Bonnet ever!"
Adafruit

JOY BONNET

Mount a joypad on your Pi Zero to get retro gaming

Ever since its arrival, the tiny Pi Zero has been used for mini retro gaming projects, usually involving inserting one inside an old joypad. Adafruit's Joy Bonnet offers a much simpler, quicker route to pocket-sized retro gaming, however. Coming fully assembled, it simply stacks on top of your Raspberry Pi Zero. Naturally, you'll need to solder (or hammer) a GPIO header to the latter first. A couple of plastic spacers and screws keep the Bonnet firmly in place – which is pretty essential as you'll be pressing its buttons continuously and therefore pushing it down on the Pi. While it's comfortable enough to hold in your hand, you may want to add the bottom of a Pi Zero case for extra comfort – although we had problems keeping the mini-HDMI display adapter fully inserted through the hole in an official case.

You're then ready to install a retro gaming OS. Adafruit recommends using RetroPie (**retropie.org.uk**) or Emulation Station (**emulationstation.org**) – just flash your microSD card as usual. With wireless set up, you can then SSH in and use a single command to install the Joy Bonnet Python library and software. It takes a little while and offers options to disable overscan (to remove the black border on some monitors) and install a gpio-halt utility for safe shutdown.

Upon rebooting, the OS (we used RetroPie) should sense the Joy Bonnet. We were somewhat surprised to see it recognised as a keyboard: it turns out that the Bonnet's buttons emulate keys such as Z, X, and ENTER. Another interesting point to note is that the mini joystick is actually analogue, although its directions produce cursor key presses – more on this later. Once you have assigned the various buttons and joystick directions to functions in RetroPie, you're ready to play – naturally, you'll need to have added a few game ROMs in the relevant system folders in RetroPie to make them appear in the on-screen menus.

Tiny buttons

We started off with a quick game of Galaga '88 running on the MAME arcade emulator. Everything worked fine and the controls were responsive enough. Upon switching to Street Fighter II on SNES, however, we encountered a slight drawback. In place of L and R shoulder buttons, the Joy Bonnet has a couple of tiny buttons labelled 1 and 2, located in the middle of the top

Related

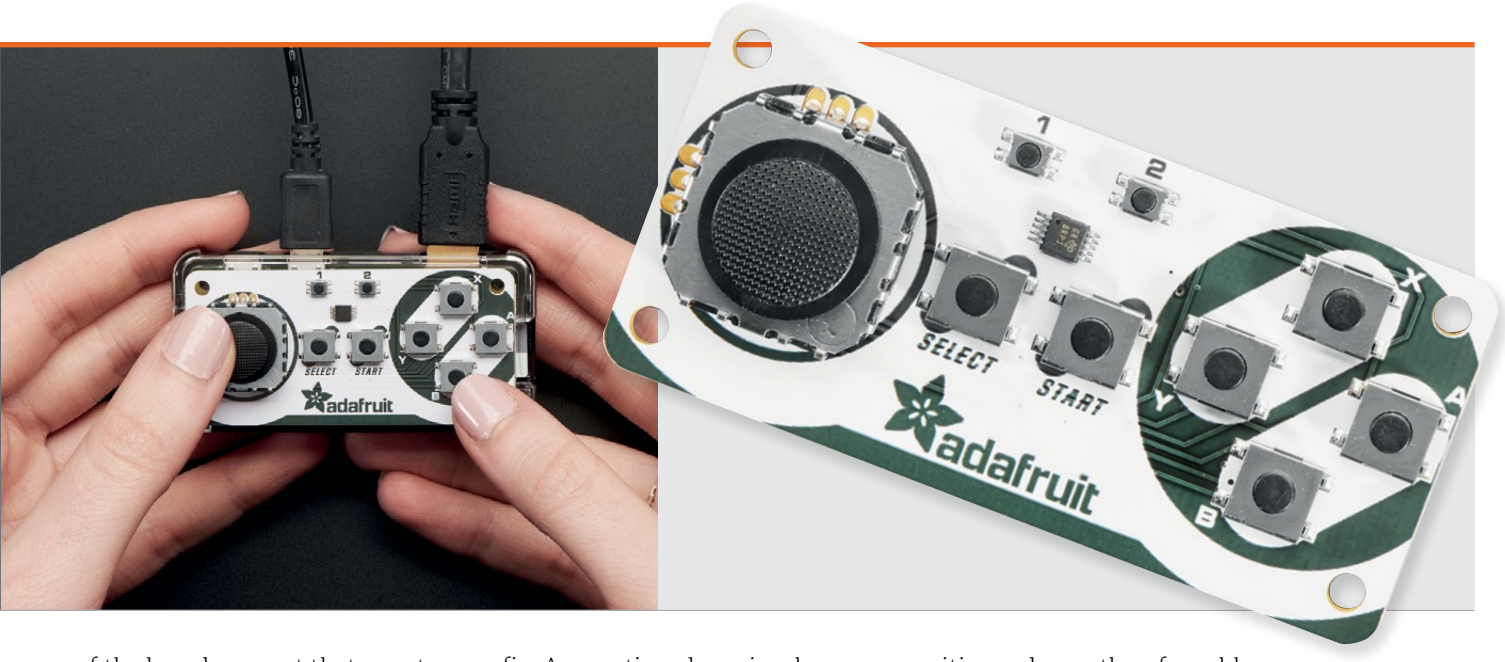
PICADE CONSOLE

The control unit from the Picade arcade cabinet, it can be fitted with a Pi 3, plugs into a TV, and features a robust full-size arcade joystick and buttons.

£90 / $120
magpi.cc/2pwjhhS

magpi.cc/2qaSfIT

£16 / $15

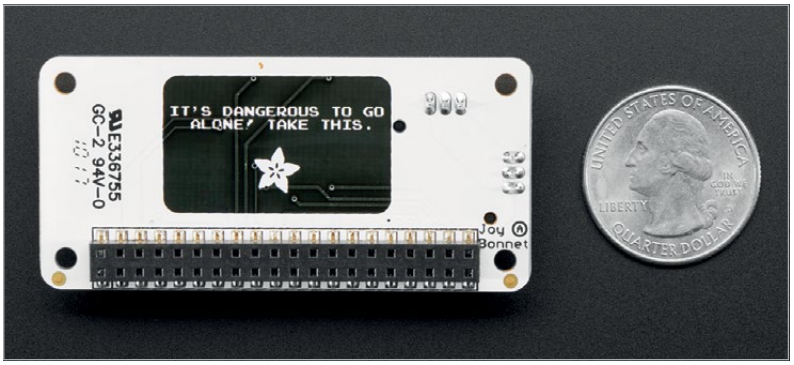


of the board – so not that easy to reach in the heat of battle. The four main buttons (X, Y, A, and B) worked well, although they're far smaller than the ones on original joypads, so not that comfortable. While not quite so critical, the Select and Start buttons are the same small size.

Next, we thought we'd have a blast with classic vertically scrolling shmup, 1942. Here we came across a bugbear that spoilt our enjoyment until we figured out a fix. As mentioned previously, the mini joystick is analogue but emulates digital presses, and we found it extremely difficult to obtain diagonal directions for our plane in the game. Fortunately, we managed to sort this out by editing the Joy Bonnet's Python library and reducing the positive and negative thresholds for the analogue stick. Setting these at -300 and 300, rather than the original -600 and 600, we found the stick considerably more sensitive and were therefore able to obtain the diagonal directions. It's also possible to edit the key presses produced by the buttons in this file, which might come in useful when playing a Spectrum or C64 game with unorthodox keyboard controls.

> Joy Bonnet offers a much simpler, quicker route to pocket-sized retro gaming

Note that the Pi Zero is not capable of emulating more powerful consoles such as the N64 and PlayStation. You could always use the Joy Bonnet with a Raspberry Pi 3, although it wouldn't exactly be handheld.



Last word

Not as comfortable to hold or responsive as a regular game console joypad, the Joy Bonnet is unlikely to net you many high scores. Still, it is a cute concept that makes it easy to quickly get retro gaming on a Pi Zero: a neat portable solution that you can carry around with you to plug into any TV. You might want to invest in a longer HDMI cable so you don't have to stand quite so close to the screen, though.

WD PIDRIVE NODE ZERO

A custom low-energy hard drive coupled with a Pi Zero

The WD PiDrive Node Zero is a clever all-in-one unit that combines a WD PiDrive with a Raspberry Pi Zero board.

The PiDrive is WD's low-energy hard drive, designed specifically for the Raspberry Pi. It replaces the regular SATA III port with a micro-USB connection.

What we have here is a PiDrive in a plastic caddy with a mounted Pi Zero board. The PiDrive Node Zero caddy provides two additional full-sized USB ports, making it easier to hook up a keyboard without requiring a mini-USB adapter, but you still need a mini-HDMI to HDMI adapter.

The PiDrive is a custom-engineered WD hard drive that is more power efficient than a standard storage drive.

Included is a 4GB micro SD card preloaded with a custom version of NOOBS. When you first power on the PiDrive Node Zero, you install Raspbian onto the primary hard drive. The SD card boots the device, but you run it from the hard drive (so you don't need to juggle the two drives).

You can perhaps think of it as a super-smart hard drive with built-in computing functionality, or you could regard it as a Pi Zero with super-sized storage space. Either way, it's an intriguing all-in-one unit that gets you thinking about usage.

WD suggests that it's "ideal for video recording, data logging, offline analytics, and applications where stand-alone operations are needed because of network limitations or privacy/security restrictions."

There's some merit to all of these applications, but getting the PiDrive Node Zero on to a network opens up a much broader range of potential uses.

Benchmarking

The WD PiDrive Node isn't as fast as WD's USB flash drive offering (see 'Related'). We did a buffered test with the hdparm tool: **`hdparm -t /dev/sda`**. We ran the test three times and got an average of 28.15 MB/sec.

Next, we ran a timed cached test using **`hdparm -T /dev/sda`**. Our average speed was 245.37 MB/sec. Neither speed will set the world alight, but we're not sure this is problematic. Most of our imagined uses for the drive are as nodes on a system, set up to perform a task and tick away at it.

Related

WD PIDRIVE FOUNDATION EDITION, USB FLASH

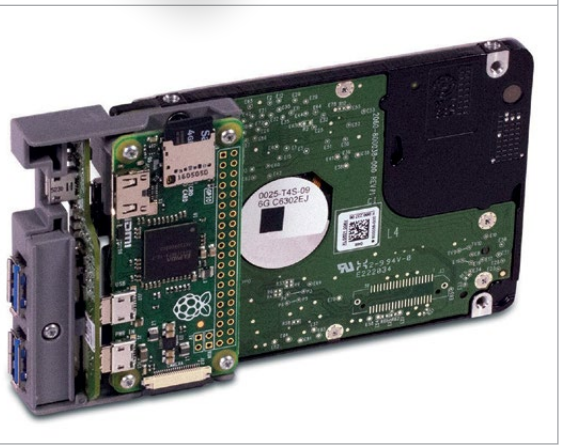
An SD card and USB drive combo that you connect to a Raspberry Pi. It has less storage than the PiDrive NodeZero, but it is a smaller solution.

£20 / $18.99

magpi.cc/2pJjBpT

magpi.cc/2r4b4Sd

£47 / $44.95



Maker Says

"A compact, all-in-one unit that includes a WD PiDrive"

Western Digital

Networking

The Pi Zero lacks built-in wireless networking, unlike the newer Pi Zero W.

You have a few options for taking a PiDrive Node Zero online. Add a USB-to-Ethernet adapter for hook-up to a wired network, or use a USB wireless networking dongle for access to WiFi.

It is physically possible to remove the Pi Zero from the caddy (using a Torx screwdriver) and slot in a Pi Zero W. You'll need to download the latest version of the PiDrive Foundation software (currently in beta) from the WD Labs website: **magpi.cc/2ns5lnA**.

Once on a network, the Node Zero is ideal for DIY projects like a mini-DLNA and Samba file server, mobile backup device, a media device, or music box. There are reports of folks using it to create Bitcoin nodes.

> "A super-smart hard drive with built-in computing functionality"

In fact, it's ideal for any task that requires a little processing power, with minimal energy draw and a decent amount of storage. There's nothing here that you couldn't hack together with Pi Zero, a few adapters and an external hard drive, but the self-contained unit makes it ideal for setting up and tucking it away in your house somewhere. WD also sells a Node Zero Enclosure for £8 that turns it into a neater unit.

Last word

A neat, all-in-one unit that combines Pi Zero computing with hard drive storage. We'd rather it packed the newer Pi Zero W, but it remains a charming and handy piece of kit.

★★★★☆

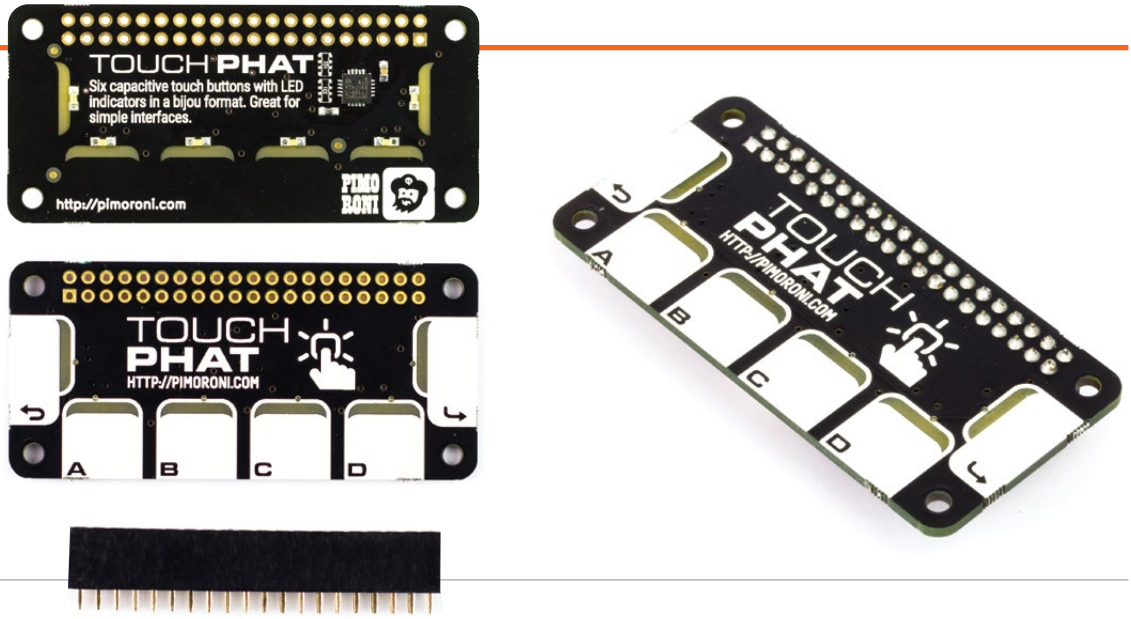
magpi.cc/2pafoP9

£8 /$10

Maker Says

"Six touch-sensitive buttons to use for whatever your heart desires"

Pimoroni



TOUCH PHAT

An easy way to add button inputs to your projects

Need to add some button controls to a project you're making? Pimoroni's Touch pHAT makes it easy. This Pi Zero-sized board boasts six touch-sensitive buttons which light up when pressed; white LEDs located on the underside produce a yellow/green glow through the translucent board sections. While the buttons are marked A, B, C, D, Back, and Enter (and referred to as such when coding), each has a large white area for custom labelling with a sticker or dry-wipe marker.

The pHAT is supplied with a female header which you'll need to solder in place. While the board has a Pi Zero form factor, it can be used with any 40-pin Pi model. Equipped with a CAP1166 capacitive touch and LED driver chip, it uses I²C for communication, and therefore requires only two GPIO pins. No standoffs are supplied, but you may want to add some to keep the pHAT rock steady on top of the Pi as you press the buttons. Alternatively, you could combine it with Pimoroni's neat-looking Pibow Zero W case.

Like most Pimoroni add-ons, the Touch pHAT has its own Python library, which is easily installed – along with any missing dependencies – using a single Terminal command. A couple of examples are supplied: a simple button-press demo and a GUI app launcher. The code syntax is simple enough, using `on_press` and `on_release` events to register the relevant touch actions. It's then completely up to you as to what these will trigger. Possible uses for the Touch pHAT include as a control panel for a robot, a remote control for home automation, a drum machine / mini piano, and a simon game.

Most importantly, the buttons are very responsive to touch and will stay triggered/lit until released; you can press as many as you like simultaneously, too. They can even be activated through a thin transparent layer if needed. Note that if you wanted to attach alligator clips to the buttons to attach remote triggers (such as pieces of fruit), you'd have to scrape down to the copper on each button to make the connection work.

Last word

The Touch pHAT makes it a lot easier to add input buttons to projects, instead of having to wire up push-buttons individually. What you use it for is completely up to you, but the touch-sensitive buttons are really responsive and the light-up effect is a nice bonus.

★★★★☆

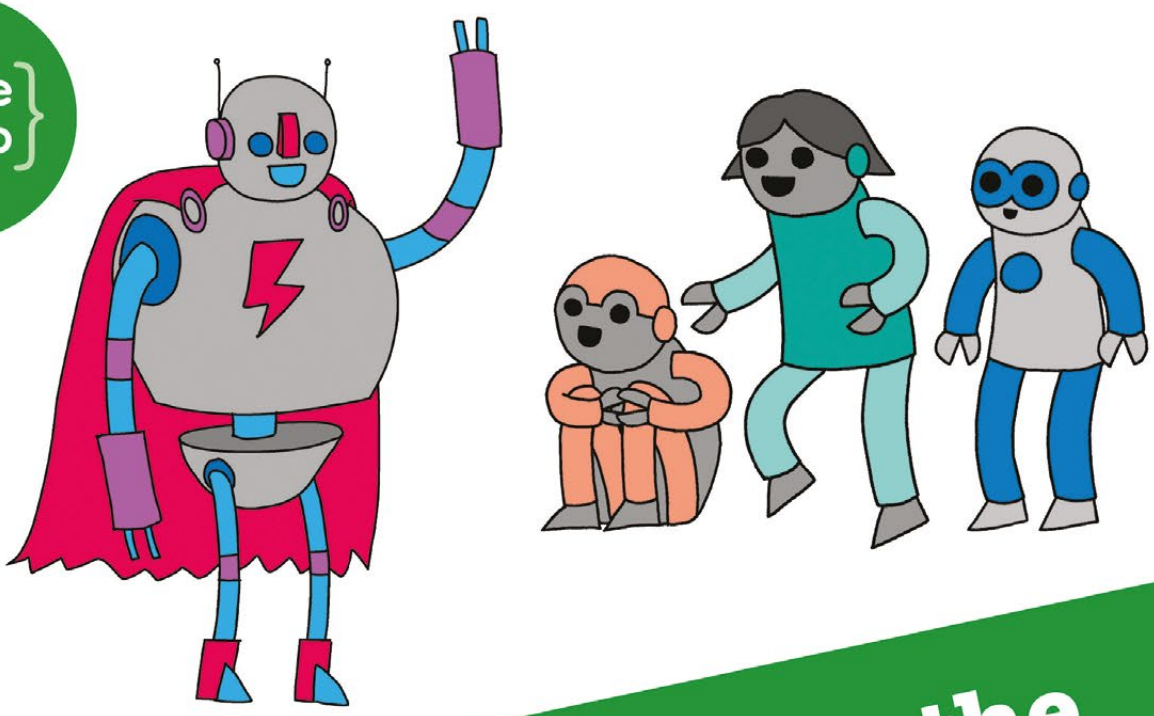
Related

RAINBOW HAT

Compatible with Android Things and Python, this full-size HAT features three capacitive buttons, a four-digit display, LEDs, buzzer, and sensor.

£24 / $25

magpi.cc/2lX1r6h



Can you help inspire the next generation of coders?

Code Club is a nationwide network of volunteer-led after school clubs for children aged 9-11.

We're always looking for people with coding skills to volunteer to run a club at their local primary school, library or community centre for an hour a week.

You can team up with colleagues, a teacher will be there to support you and we provide all the materials you'll need to help get children excited about digital making.

There are loads of ways to get involved!

So to find out more, join us at **www.codeclub.org.uk**

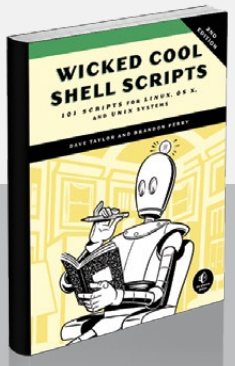
RASPBERRY PI BESTSELLERS

SYSTEM & NETWORK ADMIN

Eliminate bottlenecks, adopt best practices, and automate to avoid repetitive admin tasks

WICKED COOL SHELL SCRIPTS

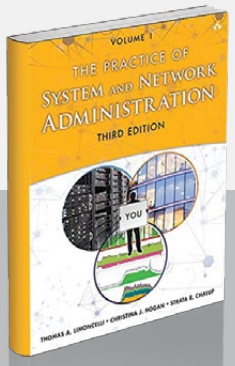
Authors: Dave Taylor & Brandon Perry
Publisher: No Starch
Price: £27.99
ISBN: 978-1593276027
magpi.cc/2pGuaKl



Cookbook-format collection of Bash scripts, with brief tutorial intro, good explanations of how the scripts work, and great suggestions for modifying them. As useful on your Raspberry Pi as on your server.

THE PRACTICE OF SYSTEM AND NETWORK ADMINISTRATION

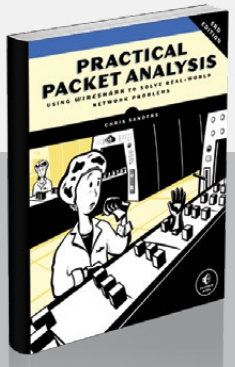
Authors: Thomas A Limoncelli, Christina J Hogan & Strata R Chalup
Publisher: Addison Wesley
Price: £51.99
ISBN: 978-0321919168
magpi.cc/2pGtQeo



From section one, 'Game-Changing Strategies', this book gets harried sysadmins back on track with best practices and practical tips. Thoroughly updated, with good DevOps coverage – even for organisations without developers.

PRACTICAL PACKET ANALYSIS

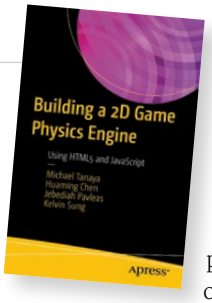
Author: Chris Sanders
Publisher: No Starch
Price: £39.99
ISBN: 978-1593278021
magpi.cc/2pGxFAs



Wireshark doesn't just let you peer inside data packets and find malware, it offers valuable clues about bottleneck and connectivity issues in your network. Sanders takes a practical approach to getting and then using this information.

BUILDING A 2D GAME PHYSICS ENGINE

Authors: Michael Tanaya Huaming Chen, Jebediah Pavleas, & Kelvin Sung
Publisher: Apress
Price: £14.99
ISBN: 978-1484225820
magpi.cc/2po5OsF



This is a short, narrowly focused book, designed to take the reader through the process of building a 2D physics engine, specifically in JavaScript, working with rigid bodies, for use in HTML 5 games. All it asks of the reader is a basic understanding of object-oriented programming (OOP), from any language, and some familiarity with basic data structures such as linked lists and dictionaries, plus some basics from high school maths.

Given this, anyone should be able to start working through the code, gaining an understanding of JavaScript along the way. The authors recommend NetBeans, but if you have your own favourite code editor, you'll probably be able to ignore IDE-specific instructions and get on with it, implementing the core of the physics engine in your first hour or so after opening the book.

The heart of the book concerns several aspects of the physics and implementation of collision detection, and not only will this get you to some working code, but should give you enough to take away and try in other languages, without being propped up by libraries. The final demo project, and pointers to further exploration, should see you ready to build your own games.

Score ★★★★☆

CORE JAVA: VOLUME II

Author: Cay S Horstmann
Publisher: Prentice Hall
Price: £36.99
ISBN: 978-0134177298
magpi.cc/2r5GbtI

A thousand pages of Java isn't everyone's idea of fun, but if you've mastered the basics and want to go further, there are few better guides than Dr Horstmann. It's not full of quirky cartoons, nor zoo animals but, as with Volume I (ISBN 978-0134177304) – which was updated a year earlier, and is an excellent start in Java for programmers with some OOP experience in C++, Python, or elsewhere – it concentrates on directly using code to illustrate language features.

Topic selection and order is well judged, with the stream library filling the opening chapter, then after I/O, XML, networking, and the latest on JDBC and database programming in Java, two whole chapters for the new data and time API, which is a much needed improvement in Java. Internationalisation gets a thorough treatment, with a completely worked example, followed by advanced Swing, native methods, and other useful topics.

Despite the serious approach, Horstmann's subtle dry humour does peek through, and this well-written tutorial covers the important advanced features in Java SE (Standard Edition) 8 and – given the time most Java-using businesses will take to move over to Java 9, when it eventually appears – it will also serve as a reference for some years to come.

Score ★★★★☆

RASPBERRY PI PROJECTS MADE EASY

Authors: Carol Vorderman & Sway Grantham
Publisher: DK Children
Price: £3.99
ISBN: 978-0241282847
magpi.cc/2r5B8ti

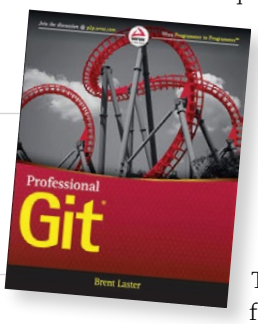
Dorling Kindersley's colourful and well-laid-out education books provide great introductory materials on programming the Raspberry Pi for younger children. The initial setup material is similar to what you'll find elsewhere, but looks far less intimidating in friendly, cartoon format.

Once set up, the first task is Scratch: drawing a sprite, giving it movement, in response to the keyboard, and making a course – in this case for a jailbreak game. It's all well done, and uses Scratch's strength to give the quickest introduction not just to some programming concepts, such as loops, but to the reward of achieving something on the screen.

Sound, with Sonic Pi, follows, then generating crazy patterns with Python. Again, well chosen to get younger learners producing real results. Challenges at the end of each project help push the learner to build on what she has learned, with help for each at the end of the book. Lighting up an art project with LEDs rounds off a very useful introduction that will make a great gift for a child less likely to be enthused by a larger tutorial book – and it's also less likely to scare off reluctant parents from helping.

Score ★★★★☆

PROFESSIONAL GIT



Author: Brent Laster
Publisher: Wrox
Price: £42.50
ISBN: 978-1119284970
magpi.cc/2r5BWhN

This is a deep and immersive guide to Git, with plenty to teach those who've been using it for a while, yet goes out of its way to be welcoming to new Git users coming from other version control systems. The first three (shortish) chapters introduce the concepts of Git before installation is introduced. You can skip ahead, but it's worth reading through for Laster's clear explanations of key concepts, and the promotion model (Git's levels) as, if you've picked up Git by diving in and using, you may have missed a full understanding.

Impatient readers of the subsequent setting up chapters may feel some things are explained in too much depth, but again this is driven by the need to instil clearer understanding as a foundation for later sections. In particular, sections headed Advanced Topics in some chapters are full of useful information.

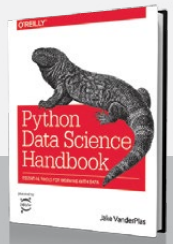
A dozen connected lab sections filled with practical exercises help the reader do the real work of using practice to embed the theoretical knowledge, and cover everything from working with GitHub, to subtrees. With initial weighting towards working locally, everything gets a timely introduction, and this is one book that will carry the novice through towards expert level. Recommended.

Score ★★★★★

ESSENTIAL READING: DATA SCIENCE

As data expands, the world needs more data scientists – start learning today!

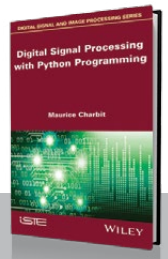
Python Data Science Handbook



Author: Jake VanderPlas
Publisher: O'Reilly
Price: £47.99
ISBN: 978-1491912058
magpi.cc/2r5wTxT

The best data science reference and tutorial for Python programmers, covering IPython, NumPy, Pandas, Matplotlib, and Scikit-Learn.

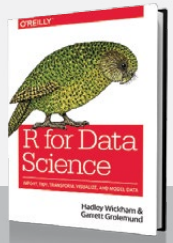
Digital Signal Processing with Python Programming



Author: Maurice Charbit
Publisher: Wiley
Price: £96.50
ISBN: 978-1786301260
magpi.cc/2r5WhDY

Data science needs probability theory, statistical inferences, hidden Markov models (HMM), and Monte Carlo methods – learn them with Python.

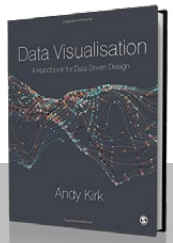
R for Data Science



Authors: Garrett Grolemund & Hadley Wickham
Publisher: O'Reilly
Price: £31.99
ISBN: 978-1491910399
magpi.cc/2r5C158

Fairly beginner-friendly introduction to all things data and R - the "other" language of data science.

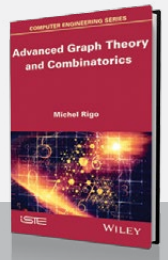
Data Visualisation



Author: Andy Kirk
Publisher: Sage
Price: £31.99
ISBN: 978-1473912144
visualisingdata.com/book

With presentation and interpretation of data central to so much today, this guide is essential reading for almost everyone!

Advanced Graph Theory and Combinatorics



Author: Michel Rigo
Publisher: Wiley
Price: £116.00
ISBN: 978-1848216167
magpi.cc/2r5YJue

More advanced mathematics for ML and big data, culminating in applying Perron-Frobenius theory to explore Google Page Rank.

THE MONTH IN RASPBERRY PI

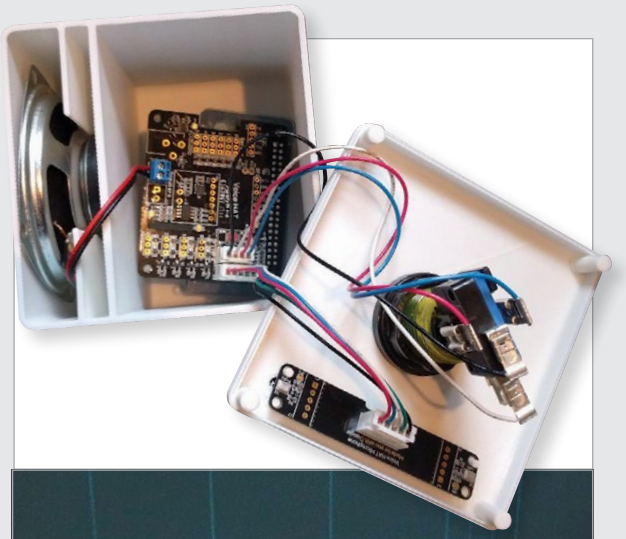
Everything else that happened this month in the world of Raspberry Pi

AIY PROJECTS

Issue 57 went down well with the community

It's always our pleasure to release surprise special issues that the whole community loves. We've seen a lot of you excitedly show us your AIY Projects kits, boxes, and projects since the issue came out, which has been great! Here are some of our favourites.

Above and Right: Bastiaan Slee has given a second life to these Nabaztag IoT devices by supercharging them with some Raspberry Pi magic. Find out how here: **magpi.cc/2qiCrXW**



Tom Archer has been very busy with his AIY Projects kit. He 3D-printed this wonderful custom case

Lorraine Underwood's picture frame lets her mother in Ireland send audio messages to her grandchildren and vice versa. It's a bit like the portraits in *Thunderbirds*

Above We got plenty of pictures from people who have finished building their kits, but this one from Andy Grimley caught our eye, as it seems to be floating. We didn't know it could do that!

Tom Archer added Google Music integration to his custom box. It even scrolls the song title on the LED board on the front. Amazing! Here's the video: **magpi.cc/2rh4dRP**

This project by CJ Watkins keeps an eye on the sky, tracking aircraft, and sings to you while it does it

Above Left: Domhnall O'Hanlon sent us this simple yet lovely video of him using his AIY box in the Irish National Botanic Gardens **magpi.cc/2rgKjqb**
Above Right: Tobie Salyers' Robie Junior is being upgraded with an AIY kit inside. So far it can talk, but controlling the eyes is the next part of the project

AIY VOICE KIT MAILING LIST

Wanted to get a copy of issue 57 and the AIY Projects voice kit but found they were sold out before you could get one? Raspberry Pi and Google are working to figure out a way to make the kits available in the longer term. Sign up for our newsletter at **magpi.cc**.

PIONEERS 2: THIS TIME IT'S OUTDOORS

Pioneers is back with a brand new challenge, and this one asks you to go outside

The first Pioneers challenge was excellent. The task was to make the judges laugh, and you managed that very well. Now comes the second challenge, and this time the Pioneers team want you to 'Make it outdoors'.

What does this mean? Well, the idea is that you make something that can work outdoors. Whether you're making a little time-lapse spy camera for a bird box or an umbrella that tells you the wind speed and temperature, as long as it's used outside your house, it's eligible for entry.

Find out more about the new Pioneers challenge and how you can get yourself involved, take a look at the Pioneers website: **magpi.cc/2iHKIP5**.

PROJECTS TO INSPIRE YOU

Here are some projects that should get you in the mood for all your outdoor making

magpi.cc/2pY406W

GETTING STARTED WITH WEARABLES

One of the suggestions for the latest Pioneers challenge is a Pi-powered coat. This tutorial will show you how to create a wearable project with lights and more, using a Raspberry Pi. Perhaps you can use what you've learnt here to create a magic hat with sounds and lights?

magpi.cc/2rhzeVJ

POO NEAR YOU

The excellently named 'Poo near you' teaches you how to plot data onto a Google Map, in this case using the poo emoji as a marker. This could probably be used for more useful tasks, like using it to check stocks of Switch consoles in local game shops so you can try to hunt them down when you're in town.

magpi.cc/2rhBbl6

RASPBERRY PI ZERO TIME-LAPSE CAMERA

Another wearable tutorial, although this one teaches you how to make a wearable glasses camera to create a time-lapse video of your day. Combine this with the wearables starter guide to make ridiculous costumes and cartoon-style inventions.

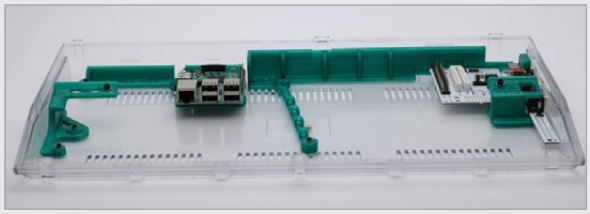
KICKSTART THIS!

Raspberry Pi projects you can crowdfund this month

PIPLAY PORTABLE

kck.st/2p3RowC

It looks a bit similar to the Joy Bonnet, which we've reviewed in this issue. This one is a bit more DIY, though, and slightly larger as well, which may improve the ergonomics of the device. It runs on AA batteries and allows you to use headphones. It's still a little way off its target so if the Joy Bonnet isn't for you and you want a nice DIY project, take a look at PiPlay instead!



COMMODORE 64C RASPBERRY PI CONVERSION FRAME

kck.st/2pLLyAi

There's still a lot of love for the Commodore 64, even so many years on. Re-releases with more power in the same box always seem popular, and this conversion kit allows you to add new life to older models – specifically the Commodore 64C. It's also advertised as 'non-destructive', allowing you to mount a Raspberry Pi 2 or 3 inside without having to break your C64.

BEST OF THE REST

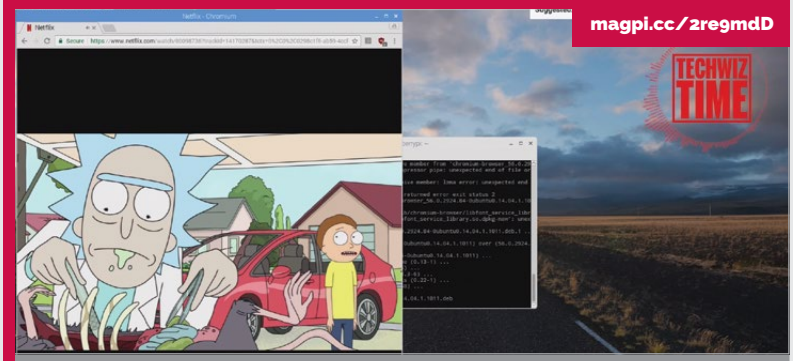
Here are some great things we've seen this month

magpi.cc/2re7MZc

VIDEO MICROSCOPE

"My first project with a Pi Camera!" reads reddit user Yelneerg's post. He's built a Camera Module into a microscope array, and it's super cool. He's removed the default lens on the sensor so the camera can make use of the microscope's own special magnifying lenses.

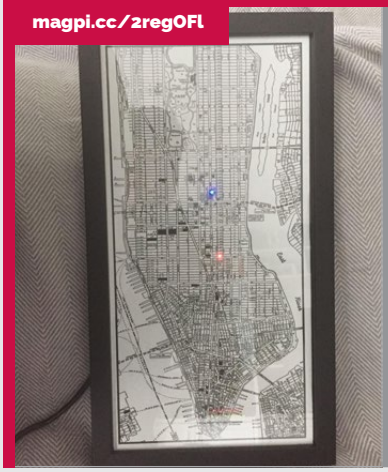
magpi.cc/2re9mdD



NETFLIX ON RASPBERRY PI 3

The dream is to have an official Netflix app for Kodi on Raspberry Pi. We're still waiting for the app, but Jon at TechWizTime has managed to get Netflix running natively on a Pi 3, which is no mean feat. His video explains how it works.

magpi.cc/2regOFl



LED LOCATION MAP

It's great in old movies when they use GPS or some kind of tracking technology that uses a map. You know it's a bit fake but the aesthetic is interesting. Reddit user GawkyFuse built this map of New York, which uses LEDs to pinpoint where he and his wife are in real-time. It goes purple when they're together. Aw.

GRANT SINCLAIR

After working for his uncle's and father's businesses for many years, Grant set up a company in 2016 to market his own inventions.
grantsinclair.com

POCO ZERO

Grant Sinclair's Pi Zero-based device delivers portable retro gaming, photography, and more

Coming in kit form, the POCO Zero is a tiny handheld device with a 2.8-inch touchscreen, powered by a Raspberry Pi Zero W. Its creator, Grant Sinclair, tells us that the roots of the POCO project stretch back to around 2009, when he saw the huge success of the Flip Video camera. "I wanted to make a credit card-sized version with much higher-spec components. I spent a couple of years developing it." Disaster struck when the chip he was planning to use went end-of-life, but Grant re-engineered the product to use the newly released Raspberry Pi. He says he got a lot of inspiration from his family: his uncle Sir Clive is famous for launching iconic 1980s computers such as the Sinclair ZX80, ZX81, and Spectrum. "I got working with my family business quite young."

Designed back in 2010, the original POCO prototype was a very different product, with a considerably higher spec and price point. "It used a lot of very clever stuff. The body was made out of magnesium, a very thin-wall casing. Pretty much all the technology in it was pushing the boundaries of what you could do. It was one of the first products to use an AMOLED screen." It was also set to feature a high-end camera module specially made by Panasonic.

From zero to Zero

Following supply problems with the chip and screen, Grant redesigned the POCO to use the Raspberry Pi and a custom screen. Marketed mainly as a portable camera, it sadly failed to meet its Indiegogo crowdfunding target in 2015, so Grant went back to the drawing board again. Luckily, the Pi Zero had just been launched, which offered the ideal solution for a handheld device. The result is the POCO Zero, a mini computer kit that requires the maker to cram a lot of components into a small space, including a tiny speaker, amplifier, transformer, and rechargeable LiPo battery.

"It's not your average maker kit," explains Grant. "I wanted to do it so it presented a bit of a challenge putting it together. So it's like a pro maker product really. At the end of it, you have a proper consumer-grade product." Although Grant says it's "packed to the max" inside the case, there is a little space left for modding: "You can get wires in very easily."

While Grant says it'll take the average maker "a good day's work" to assemble the POCO Zero, soldering is minimal. "Cramming it all inside is a bit fiddly, but it's no harder than assembling a remote control car." To help users out, video instructions will be made available online.

As well as a 2.8-inch touchscreen, there's an HDMI socket to connect to a TV

The twin joysticks can be pushed in to function as Start and Select buttons

A mini-USB hub provides three ports, including one for charging the LiPo battery

POCO PI POWER

Powered by a Raspberry Pi Zero W, the POCO Zero naturally benefits from built-in wireless LAN and Bluetooth connectivity. In addition, there's an optional v2 Pi Camera Module with 8MP resolution. Also crammed into the plastic case are a LiPo battery, tiny 1W speaker, mini amplifier, USB hub, and a daughterboard for connecting everything together. Priced at £99, for the standard kit version with camera included, the POCO Zero will be available from retailers later this year – or you can pre-order one from **grantsinclair.com**.

Multiple uses

Grant confirms that the POCO Zero's primary use will be for retro gaming, running EmulationStation or RetroPie, and that scripts will be provided to set up the controls for its buttons and twin joysticks. While the Pi Zero isn't quite powerful enough to handle emulation of PlayStation or Nintendo 64 games, it can manage most other retro systems – including, of course, the Sinclair Spectrum.

The POCO Zero can also be used as a camera, music player or, by connecting it to a TV or monitor, a fully functioning computer. While some basic user guides will be provided, Grant says: "I wanted to keep it as simple as possible so if you want to turn it into a [WiFi] mobile phone, that's doable as well… I want people to discover their own stuff: I don't want to tell them what to do too much."

Asked how many he's hoping to sell, Grant responds: "I don't really look at a product unless I think it's got the potential to sell a million units." He also tells us that two additional versions will be made available: one ready-assembled and another omitting the camera. As for future Pi-based products, Grant reveals he is working on an idea for a tablet device with "a very innovative angle to it."

The self-build kit is a challenging project, taking the average maker about a day to assemble, but the result is a gorgeous consumer-grade gadget

COMMUNITY PROFILE

DR LUCY ROGERS

From a doctorate in bubbles to building robotic dinosaurs and judging the BBC's Robot Wars: is there anything **Dr Lucy Rogers** can't do?

Dr Rogers

Category: Transformer

Day job: Writer, maker, robotic dinosaur wrangler

Website: lucyrogers.com

Dr Lucy Rogers calls herself a Transformer. "I transform simple electronics into cool gadgets, I transform science into plain English, I transform problems into opportunities. I am also a catalyst. I am interested in everything around me, and can often see ways of putting two ideas from very different fields together into one package. If I cannot do this myself, I connect the people who can."

It's a pretty wide range of interests and skills for sure. But it only takes a brief look at Lucy's résumé to realise that she means it. When she says she's interested in everything around her, this interest reaches from electronics to engineering, wearable tech, space, robotics, and robotic dinosaurs. And she can be seen talking about all of these things across various companies' social media, such as IBM, websites including the Women's Engineering Society and books, including her own.

When not attending conferences as guest speaker, tinkering with electronics or creating engaging IoT tutorials, she can be found retrofitting Raspberry Pis into the aforementioned robotic dinosaurs at Blackgang Chine Land of Imagination, writing, and judging battling bots for the BBC's *Robot Wars*.

Below Lucy graduated from the Singularity University Graduate Studies Program in 2011, focusing on how robotics, nanotech, medicine, and various technologies can tackle the challenges facing the world

Above With her bright LED boots, Lucy was one of the wonderful Pi community members invited to join us and HRH The Duke of York at St James's Palace last year

Among many other projects, Dr Lucy Rogers currently focuses much of her attention on reducing the damage from space debris

Lucy graduated from Lancaster University with a degree in Mechanical Engineering. From there, she spent seven years at Rolls-Royce Industrial Power Group as a graduate trainee before becoming a chartered engineer and earning her PhD in bubbles.

Bubbles?

"Foam formation in low-expansion fire-fighting equipment. I investigated the equipment to determine how the bubbles were formed," she explains. Obviously. Bubbles!

She then went on to become a fellow of the Royal Astronomical Society (RAS) in 2005 and, later, a fellow of both the Institution of Mechanical Engineers (IMechE) and British Interplanetary Society.

As a member of the Association of British Science Writers, Lucy wrote *It's ONLY Rocket Science: an Introduction in Plain English* and will be publishing *Wiring the IoT* alongside Dr Andy Stanford-Clark later this year.

As a standout member of the industry, and all-round fun person to be around, Lucy has quickly established herself as a valued member of the Pi community.

In 2014, with the help of Neil Ford and Andy Stanford-Clark, Lucy worked with the UK's oldest amusement park, Blackgang Chine Land of Imagination, on the Isle of Wight, with the aim of updating its animatronic dinosaurs. The original Blackgang Chine dinosaurs had a limited range of behaviour: able to roar, move their heads, and stomp a foot in a somewhat repetitive action. When she contacted Raspberry Pi back in the November of that same year, the team were working on more creative, varied behaviours, giving each dinosaur a new Raspberry Pi-sized brain. This later evolved into a very successful dino-hacking Raspberry Jam.

> "Lucy was asked to help judge the first round of the 'Make us laugh' Pioneers challenge"

Given her love for tinkering with tech, and a love for stand-up comedy that can be uncovered via a quick YouTube search, it's no wonder that Lucy was asked to help judge the first round of the 'Make us laugh' Pioneers challenge for Raspberry Pi. Alongside comedian Bec Hill, Code Club UK director Maria Quevedo, and the face of the first challenge, Owen Daughtery, Lucy leant her expertise to help name winners in the various categories of the teens event, and offered her support to future Pioneers.

HIGHLIGHTS

BLACKGANG CHINE

magpi.cc/2pEHdwG

Lucy, Neil Ford, and Andy Stanford-Clark used several Raspberry Pis and Node-RED to visualise flows of events when updating the robotic dinosaurs at Blackgang Chine. They went on to create the successful WightPi Raspberry Jam event, where visitors could join in with the unique hacking opportunity.

IT'S ONLY ROCKET SCIENCE

magpi.cc/2pENdWi

It's Only Rocket Science: An Introduction in Plain English utilises Lucy's transformer mantra to explain the basics of space travel and rockets. Explaing that " 'hard to understand' isn't the same as 'impossible to understand'," Lucy's book takes her readers through the journey of building a rocket, leaving Earth, and travelling the cosmos.

BBC'S ROBOT WARS

robotwars.tv

First broadcast in the UK between 1998 and 2004, *Robot Wars* was revived in 2016 with a new look and new judges, including Dr Lucy Rogers. While competitors battle their home-brew robots against each other and the House Robots, Lucy, along with Professor Noel Sharkey and Professor Sethu Vijayakumar, award victories via judges' decision among the carnage of robotic remains.

There are thousands of Code Clubs around the world. Here's one that caught our eye this month

CODE CLUB OF THE MONTH

NORFOLK CODE CLUBS

Sue Gray

Occupation: Teacher

What inspired you to volunteer for Code Club?
"I'd heard of Code Club and when I investigated a bit, I found that it was pretty easy to register. I'm really keen to get youngsters started with tech at a young age and get them inspired early. I think it's important to show them that they can 'own' the technology and make it do what they want."

Tell us as a bit about your Code Club
"I manage two Code Clubs locally. The first is in a small primary school and the numbers have varied over the three years that I've been running it. At the moment I've got just two girls who've been with me from the start and we have a super time. We've done a lot of the Scratch projects, some HTML and Python, as well as a little wearable project.

"The second club started this year, at another local primary school, and I've been invited to teach a whole class as part of their computing curriculum. A set of Raspberry Pi and Google Chromebooks means they are fairly well versed in computing already."

What would you say are the benefits of volunteering for Code Club?
"Code Club has allowed me to find out about what is taught in primary school and get an idea of the areas that teachers struggle

Right Sue's Code Club members getting to grips with Scratch and Python

Sue's second Code Club is run as part of the timetable for the whole class

with or find easy. It also led me to volunteer for CAS Barefoot so that I could do something to help the teachers. Volunteering gives me an insight into primary teaching and allows me to do fun stuff and pretty much organise what I want to do. It also allows me to take a project idea and, with input from the children, take the project in a different direction."

Why is Code Club important to you?

"Code Club provides the projects and ideas, which makes volunteering easy. I don't have to devise ideas and projects but can take the Code Club materials and just go with those or, taking ideas from the children, we can use these as a launch pad for our own ideas. It's also clear that computing in primary schools (certainly the ones I've visited either for Code Club or CAS Barefoot) needs a boost, especially given the technological world these young people are growing up in and the influence tech will have on them."

> "Code Club provides the projects and ideas, which makes volunteering easy"

What has been your best 'Code Club moment'?

"Emma, who has been with my Code Club for three years and loves it, once said: 'I love Code Club, just us girls coding and chilling.'

"Having the Raspberry Pi Foundation visit my newest Code Club to film the children for the promo video for Hello World magazine was also great. They were brilliant and totally thrilled to be movie stars. When I showed them the final video, they loved it so much we watched it twice."

Is there anything unique about your Code Club you'd like to share?

"Maybe my second Code Club is unique as the school has invited me to teach a whole class. The 'normal' model would be an after-school club, I guess. As I'm doing Code Clubs in local primary schools, I get to promote our high school and it's lovely to see some of my first Clubbers now in our Year 7 classes."

EVENT HIGHLIGHTS

Here's what went on at Jams and other Raspberry Pi events around the world this month

franklintwp.org

FRANKLIN TOWNSHIP PUBLIC LIBRARY

This Raspberry Jam in New Jersey, USA, apparently gets bigger all the time, according to organiser Sarah Roman

magpi.cc/2q8PGKT

COVENT GARDEN RASPBERRY JAM

The spring Jam was a huge success, with loads of Jam veterans helping out, including the folk who run the East London Jam

magpi.cc/2q8ZoNa

NORTHERN IRELAND RASPBERRY JAM

Andrew Mulholland's Northern Ireland Jam happens every month, and is one of the premier Jams in NI

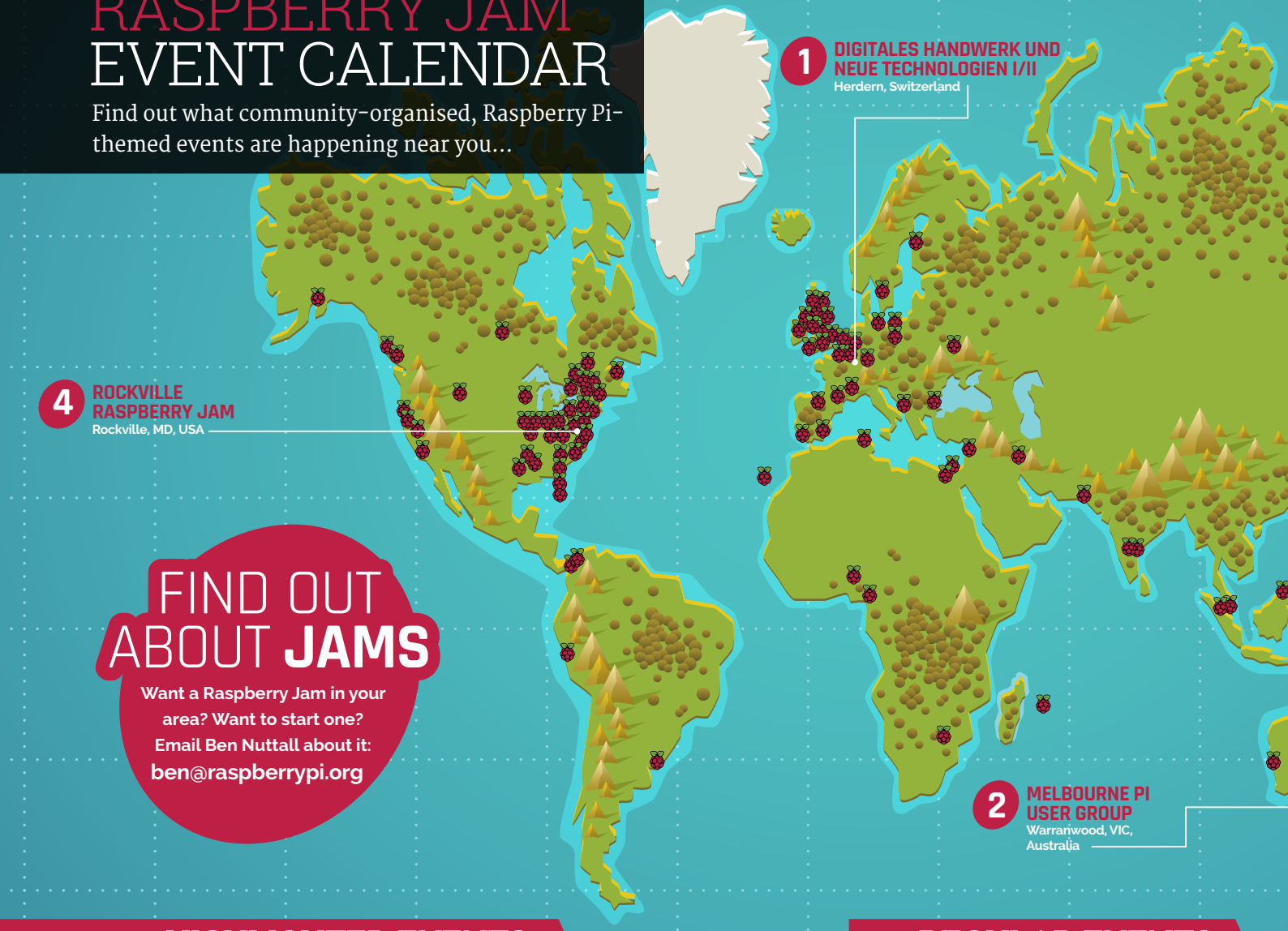
magpi.cc/2q95m0t

PI TOWERS RASPBERRY JAM

Ben Nuttall from the Raspberry Pi Foundation puts on a regular Jam at Raspberry Pi HQ itself, and it's always a blast

RASPBERRY JAM EVENT CALENDAR

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



FIND OUT ABOUT **JAMS**

Want a Raspberry Jam in your area? Want to start one? Email Ben Nuttall about it:
ben@raspberrypi.org

HIGHLIGHTED EVENTS

1 DIGITALES HANDWERK UND NEUE TECHNOLOGIEN I/II

When: Saturday 10 June
Where: Liebenfelser Immobilien AG, Herdern, Switzerland
magpi.cc/2r8tpdk
The mini-workshop is aimed at 12 to 16-year-olds. Topics include 3D printing, pi-top, and RetroPie.

2 MELBOURNE PI USER GROUP

When: Tuesday 20 June
Where: Rudolf Steiner School, Warranwood, VIC, Australia
magpi.cc/2mx2y7Y
The group's aim is to bring like-minded people together to talk about how they're using Raspberry Pis.

3 NEWHAVEN RASPBERRY JAM

When: Saturday 24 June
Where: Hillcrest Community Centre, Newhaven, UK
magpi.cc/2r8H10Y
Come along to the second Newhaven Raspberry Jam and find out more about the Raspberry Pi.

4 ROCKVILLE RASPBERRY JAM

When: Saturday 15 July
Where: Red Brick Courthouse, Rockville, MD, USA
magpi.cc/2r8fVhF
An afternoon of Raspberry Pi presentations, exhibits, and more.

REGULAR EVENTS

5 RASPBERRY JAM LEEDS

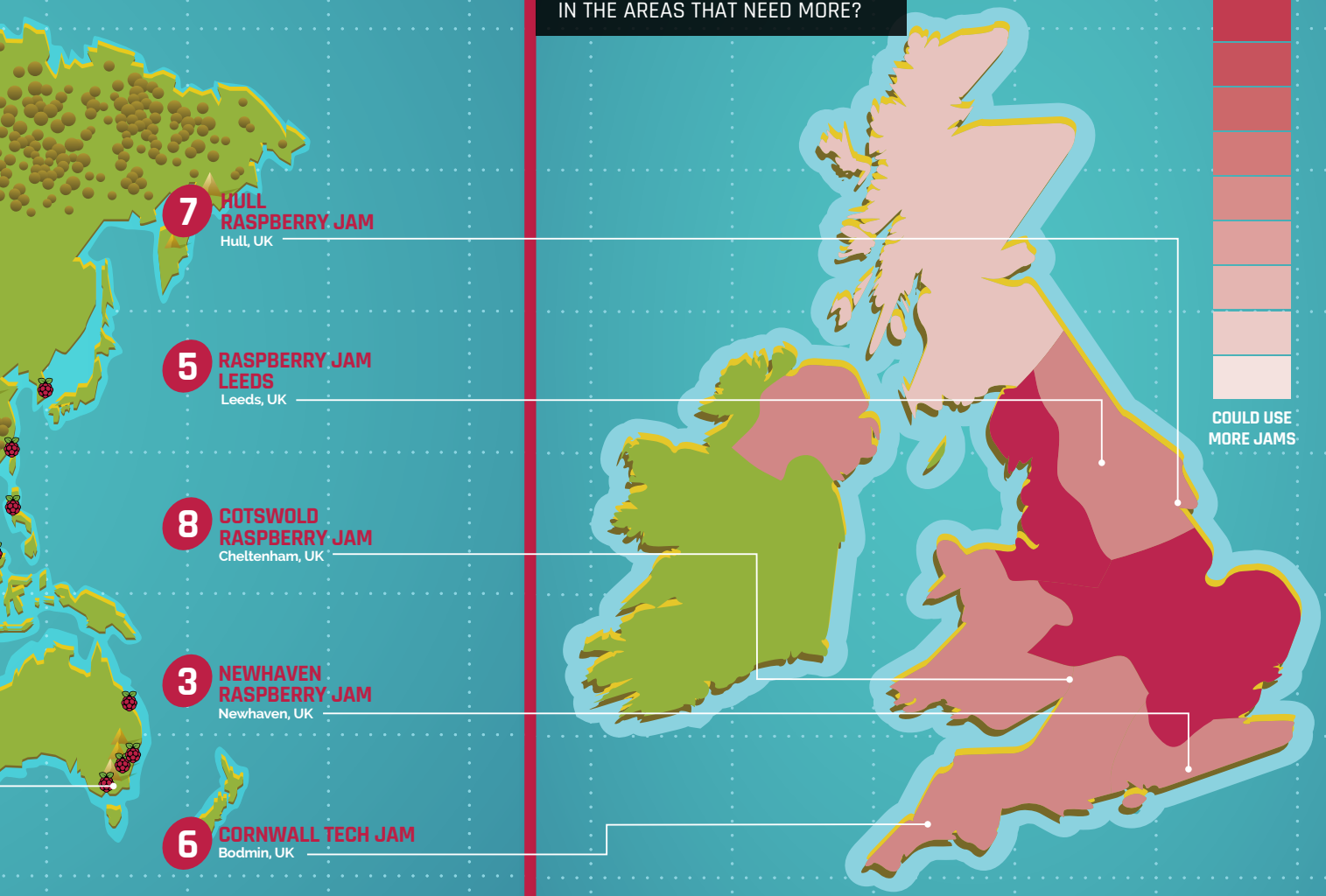
When: Wednesday 7 June
Where: Swallow Hill Community College, Leeds, UK
magpi.cc/2q9ShEp
There will be chances to get hands-on with digital making activities through workshops and a hackspace area.

6 CORNWALL TECH JAM

When: Saturday 10 June
Where: Bodmin Library, Bodmin, UK
cornwalltechjam.uk
For anyone interested in technology, of all ages and abilities. Ask questions and learn about programming: Scratch, Python, Minecraft, and much more.

JAM HEAT MAP

CAN YOU HELP INCREASE THE AMOUNT OF JAMS IN THE AREAS THAT NEED MORE?



7 HULL RASPBERRY JAM

When: Saturday 17 June
Where: Hull Central Library, Hull, UK
magpi.cc/2q9EvS9
Join the team at Hull Raspberry Jam to share, learn, and tinker with digital making projects using Raspberry Pi.

8 COTSWOLD RASPBERRY JAM

When: Saturday 24 June
Where: Waterworth Building, Park Campus, Cheltenham, UK
cotswoldjam.org
An event which showcases and helps people learn about the Raspberry Pi computer.

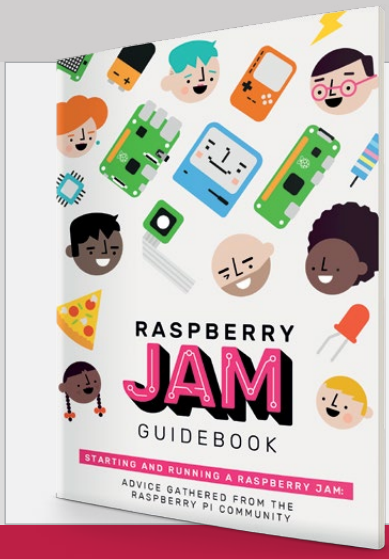
MORE WAYS TO LEARN

JAM ADVICE

Putting on a Raspberry Jam may seem daunting, but anyone can do it. The Raspberry Pi Foundation has released a free guidebook full of advice on putting on your own Jam. Here's why Dr Lucy Rogers runs her Jams:

"I love to learn. By bringing makers and others together, I get to see a variety of things and discuss amazing ideas. It's great to see eight-year-olds and 80-year-olds making lights blink and dinosaurs nod."

Get the book here: **magpi.cc/2q9DHfQ**



YOUR LETTERS

French ESA Astronaut Thomas Pesquet looked after the Astro Pi experiments this time around

Astro Pi results

I heard that the latest set of Astro Pi data has been sent back to classrooms. I really enjoyed what you did last time explaining the different results, and what went well and what didn't. Will you be doing something similar for the new set of results?
Killian E

We definitely would like to do something like that again – there were a lot more experiments this time as well, so we should have a wide range of results to show. It's mostly up to the schools whether or not they want to share their data, and some of it can take a while to analyse. Hopefully we can show some early results in the next issue!

If there are any projects from the new Astro Pi challenge that you'd like to see the results from, please let us know!

Scratch the itch

I'm a big fan of using Scratch to code as it's quite easy and fun. I do like it when you have Scratch tutorials in the mag and I loved the Scratch Essentials book. However, you do focus quite a bit on Python code and such. Is there any chance of getting more Scratch tutorials in the mag? Also, is there any word on Scratch 2.0 coming to the Raspberry Pi?
Susan

Hopefully you've enjoyed the Scratch tutorials in this issue, as well as the EduBlocks tutorial in the Minecraft feature. As Scratch is a little more limited than Python, we don't tend to cover it as much, but we do use it when we feel it makes sense.

As for Scratch 2.0, you can currently use it via the browser, but look out for future versions of Raspbian. There is some special work being done behind the scenes…

AIY thanks

Just wanted to say how much the latest issue #57 has really amazed me. I have joked on Twitter about Skynet/Jeff Goldblum/Dr Who [Ed's note: To be fair, so did we] and taking over the world etc., but I really am so pleased that you have this wonderful project with your excellent magazine.

My son (14 and a half) is currently homeschooled (he has not managed a single full year of school as he has ASD). He worked on this project from 10am to 2pm all by himself (I helped with sticky tape).

Normally I buy *The MagPi* for him to drool over the projects inside, as they are out of our ability to make, but I hope to give him aspiration for what to do with the many Raspberry Pis he has got over the years. This time, however, a real-life project was all there for him, with instructions.

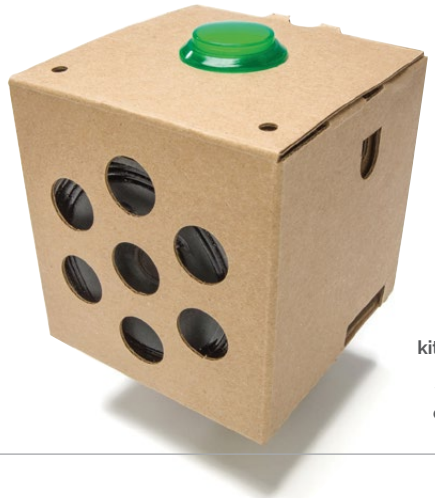
He's had a very hard journey butting up against the education system, and the Raspberry Pi and coding have made a huge difference in encouraging him to learn and try out new things, and believe in himself and his own skills.

We are hoping to go to a nearby Raspberry Jam soon (we found this through your magazine), and he's really looking forward to it.

Thanks from a grateful parent,
Kay Marshall

You're most welcome. It's letters like these from parents, and teachers, telling us how young people are having fun with the stuff we do in the magazine that makes it all worthwhile.

We'll have more amazing issues like this in the future, but in the meantime we hope you will enjoy some of the other cool projects in the magazine.



Left Our AIY Projects voice kit giveaway was one of biggest yet and it went down very well

FROM THE FORUM: MORAL AMBIGUITY

The Raspberry Pi Forum is a hotbed of conversations and problem-solving for the community. Join in via **raspberrypi.org/forums**

Many of us are having the problem of having to use a business account to use the AIY Voice Projects Kit. You recently tweeted this: "PSA: You can absolutely use the #AIYProjects kit in the UK/EU even if you're not a business! Just put your name in the company name field."

My question is do you have Google's permission to advocate breaking their T&Cs (it is assumed so by people since you are partners with them for this).

In my eyes doing this is a morally wrong thing, in effect a fake business account. While it may not be harmful and is a way around it, that's like "hey you can install this software, just stick this keygened code in and ignore T&Cs. Carry on as normal after it." bensimmo

We were very aware of this issue when we planned the giveaway originally, and worked closely with Google to make sure it was all correct. There is no problem with the way we are asking our readers to use their services, so you don't need to worry.

Hopefully the influx of British users to the service will persuade Google to change the way this is worded in the long run.

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**

MATT RICHARDSON

Matt Richardson is the Executive Director of Raspberry Pi Foundation North America and author of *Getting Started with Raspberry Pi*. Contact him on Twitter @**MattRichardson**.

TAKING THE FIRST STEP

Matt Richardson on how booting up a Raspberry Pi is just the beginning

Five years ago this month was the first time I unboxed a Raspberry Pi. I hooked it up to our living room television and made space on the TV stand for an old USB keyboard and mouse. Watching the $35 computer boot up for the first time impressed me, and I had a feeling it was a big deal, but I'll admit that I had no idea how much of a phenomenon Raspberry Pi would become. I had no idea how large the community would grow. I had no idea how much my life would be changed from that moment on. And it all started with a simple first step: booting it up.

The key to the success of Raspberry Pi as a computer – and, in turn, a community and a charitable foundation – is that there's a low barrier to the first step you take with it. The low price is a big reason for that. It's not a difficult decision, whether or not to try Raspberry Pi. Since it's so affordable, you can give it a try and see how things go.

Linus Torvalds, the creator of the Linux operating system kernel, talked about this in a BBC News interview in 2012. He explained that a lot of people might take the first step with Raspberry Pi, but not everyone will carry on with it. But getting more people to at least take that first step of turning it on means more people who can be potentially impacted by the technology.

"I find things like Raspberry Pi to be an important thing: trying to make it possible for a wider group of people to tinker with computers," said Torvalds. "And making the computers cheap enough that you really can not only afford the hardware at a big scale, but perhaps more important, also afford failure."

In other words, if things don't work out with you and your Raspberry Pi, then it's not such a big deal because it's such an affordable computer.

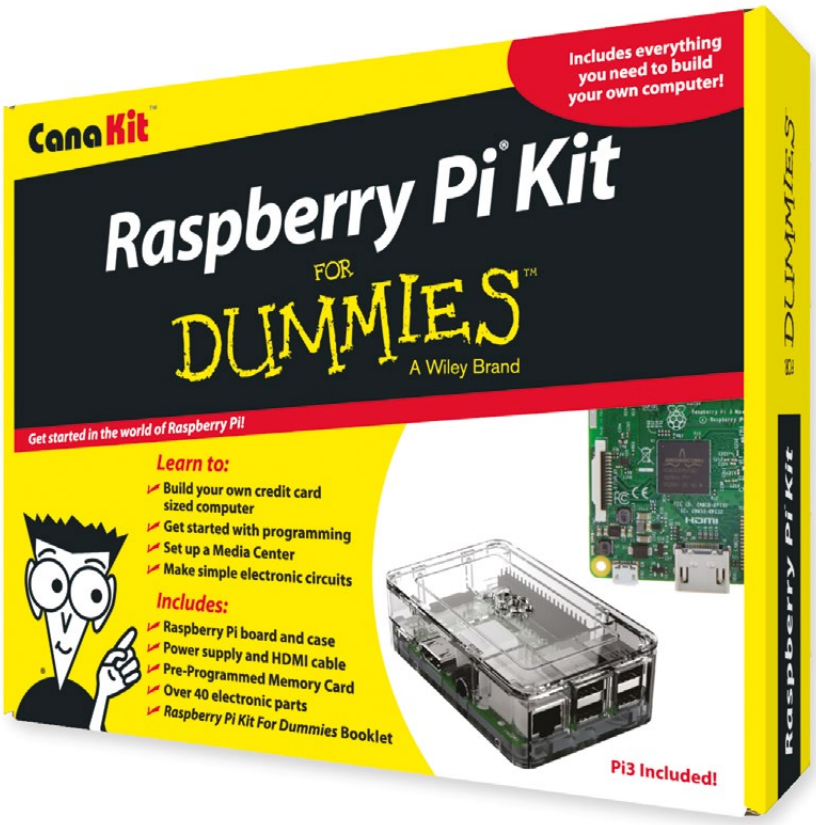
Of course, we hope that more and more people who boot up a Raspberry Pi for the first time will decide to continue experimenting, creating, and learning with it. Thanks to improvements in the hardware, the Raspbian operating system, and free software packages, it's becoming easier than ever to do so many amazing things with this little computer. And our continually growing community means you're not on this journey on your own. These improvements and growth over the past few years hopefully mean that more people who boot up Raspberry Pis are encouraged to keep exploring.

The first step

However, the important thing is that people are given the opportunity to take that first step, especially young people. For young learners, this is a critical age when something like Raspberry Pi can have an enormously positive impact on the rest of the lives. It's a major reason why our free resources are aimed at young learners. It's why we train educators all over the world for free. Encouraging youth to take their first step with Raspberry Pi could not only make a positive difference in the lives of the individuals, but also society at large.

With the affordable computational power, excellent software, supportive community, and free resources, you're given everything you need to make a big difference in the world when you boot up a Raspberry Pi for the first time. That moment could be step one of ten, or one of ten thousand, but it's up to you to take that first step.

CanaKit™



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

$89.99 US DOLLARS

£69.99 EXCLUDING VAT

ELECTRONIC KITS • ELECTRONIC PARTS • RASPBERRY PI • ARDUINO