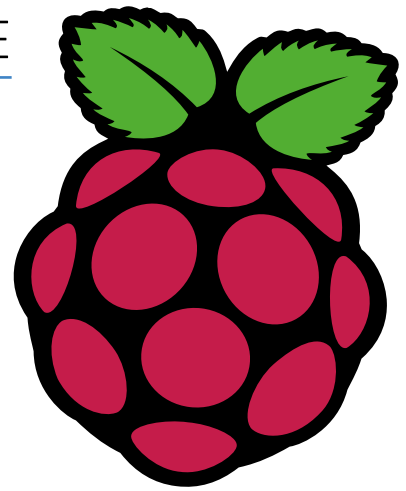




The MagPi



Issue 82 | June 2019 | magpi.cc The official Raspberry Pi magazine

BUILD A PI KEYRING

Keep your computer with you at all times

YURI 3 MARS ROVER

Recreating a space robot

LEARN TO CODE

WITH SCRATCH & PYTHON

PLUS!

- ▶ Build a MIDI sequencer
- ▶ Model railway projects
- ▶ Add NeoPixel lighting to a display

MAKE A MARAUDER'S MAP

Use beacons to locate people



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!



WELCOME

to *The MagPi* 82

Coding is a life-changing skill. One of my first memories is creating a platform game at school; even though I loved coding, I went on to study English and take up a career in journalism.

That was before Raspberry Pi and online training came along. A series of courses in Computational Thinking and Computer Science enabled me to pick up where I left off as a child. Coding had changed, but so had I: Scratch made it easy to visualise programs, and Python replaced BASIC. A greater focus on making fired my imagination. Computer science and coding also led me to Raspberry Pi, where I ended up editing this fine magazine.

The Raspberry Pi was created to increase the number of students learning code and computer science. Needless to say, it was a roaring success. Almost a year since winning the prestigious MacRobert award, Raspberry Pi now finds itself stamped into history by the Royal Mail, which is celebrating 50 years of British engineering. It joins the likes of the Harrier Jump Jet in a special set of postage stamps (page 6).

With the Pi's prestigious prizes and Cambridge University history, you might be forgiven for thinking that coding isn't for you – it's for engineers and students, and you've missed the coding boat. Nothing is further from the truth. Our feature (page 26) will get you started. Anybody can learn to code; everybody should learn to code.

Lucy Hattersley Editor



EDITOR Lucy Hattersley

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

magpi.cc

GET A
**RASPBERRY
ZERO W KIT**
WITH A SUBSCRIPTION!
PAGE 24



Contents

► Issue 82 ► June 2019

Cover Feature

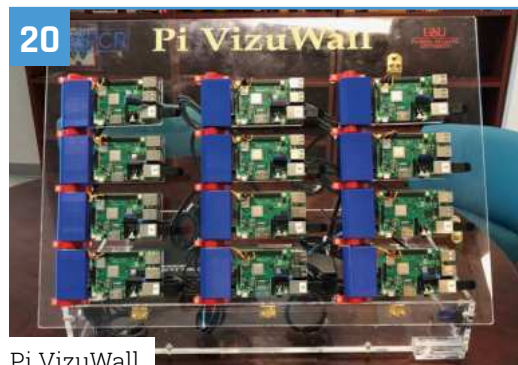
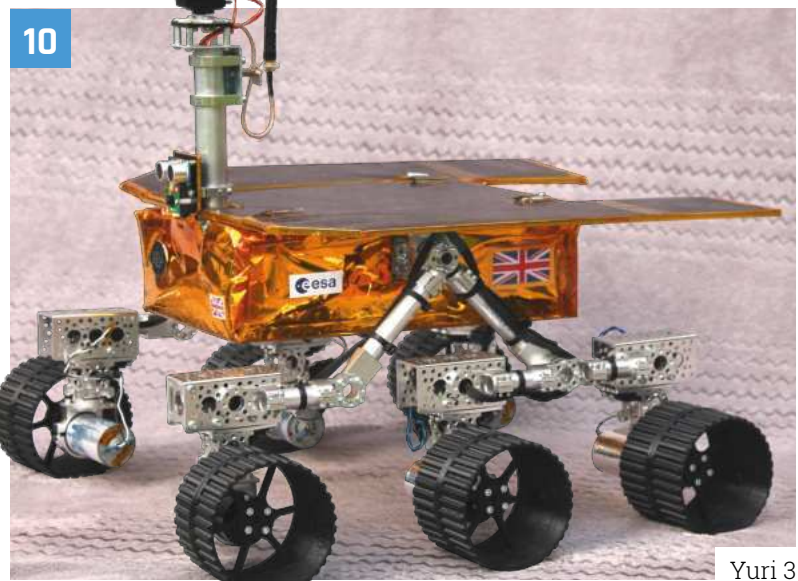
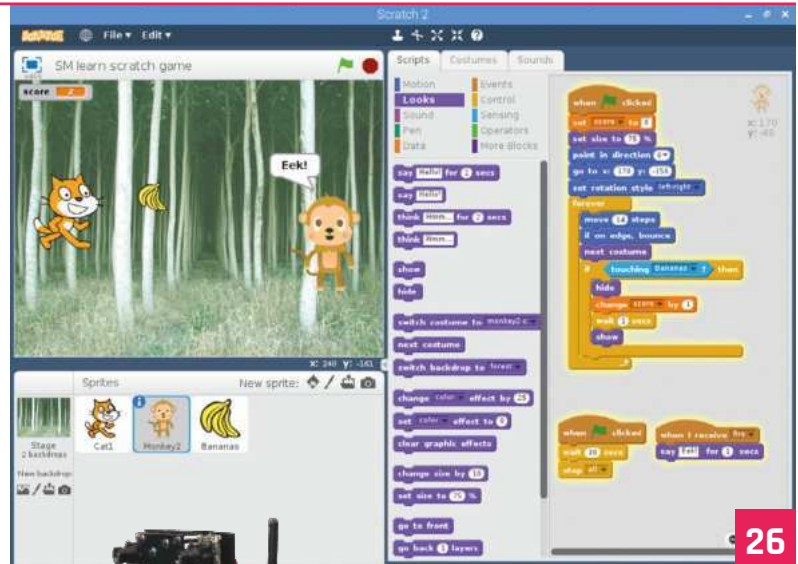
26 Learn to code

Regulars

- 06 The world of Pi
- 92 Your letters
- 97 Next month
- 98 Final word

Project Showcases

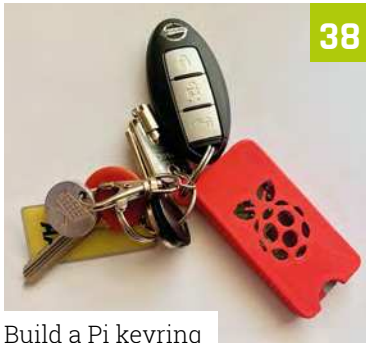
- 10 Yuri 3 Mars rover
- 14 Telepresence Hand
- 16 Cat shower
- 18 DIY Hardware Password Keeper
- 20 Pi Vizuwall
- 22 Hologram Machine



Pi Vizuwall

Yuri 3

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



38

Build a Pi keyring



46

Lego Boost part 3

Tutorials

- 38** Build a Pi keyring
- 42** Display cabinet lights
- 46** Lego Boost part 3 – Pi Bakery
- 54** Make a Marauder's Map
- 58** Buttons and labels with GTK



76

PiBug 2WD robot

The Big Feature



66

Model railway projects



84

Nicole Parrot interview

Reviews

- 76** PiBug 2WD robot
- 78** LibreELEC 9.0
- 80** Top 10 health projects
- 82** Learn AI resources

Community

- 84** Nicole Parrot interview
- 86** This month in Raspberry Pi
- 90** Events

WIN
ONE OF
TEN

JAM HATS
WITH MODMYPI

95

Raspberry Pi becomes first-class stamp

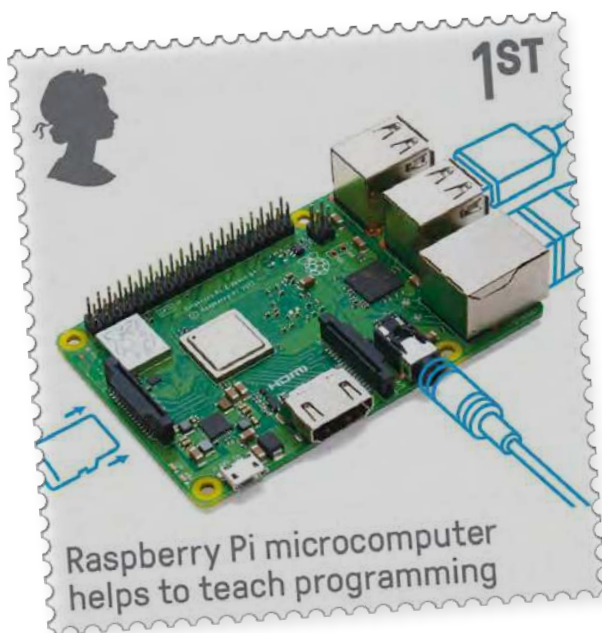
Royal Mail '50 Years of Engineering' series recognises role of ground-breaking mini computer. **Rosie Hattersley** reports on this prestigious recognition

The Raspberry Pi is a marvellous idea. We've been convinced of its brilliance ever since the idea of an ultra-affordable, incredibly adaptable, and very tiny computer for coding was first mooted. Now, our beloved Raspberry Pi has been honoured by appearing on one of six Royal Mail stamp designs celebrating 50 years of British engineering.

Liz Upton, Raspberry Pi's Executive Director of Communications, announced the incredible news that from 2 May, Raspberry Pi fans would be able to get their hands on a brand-new Pi



▲ The stamp series celebrates 50 years of British engineering

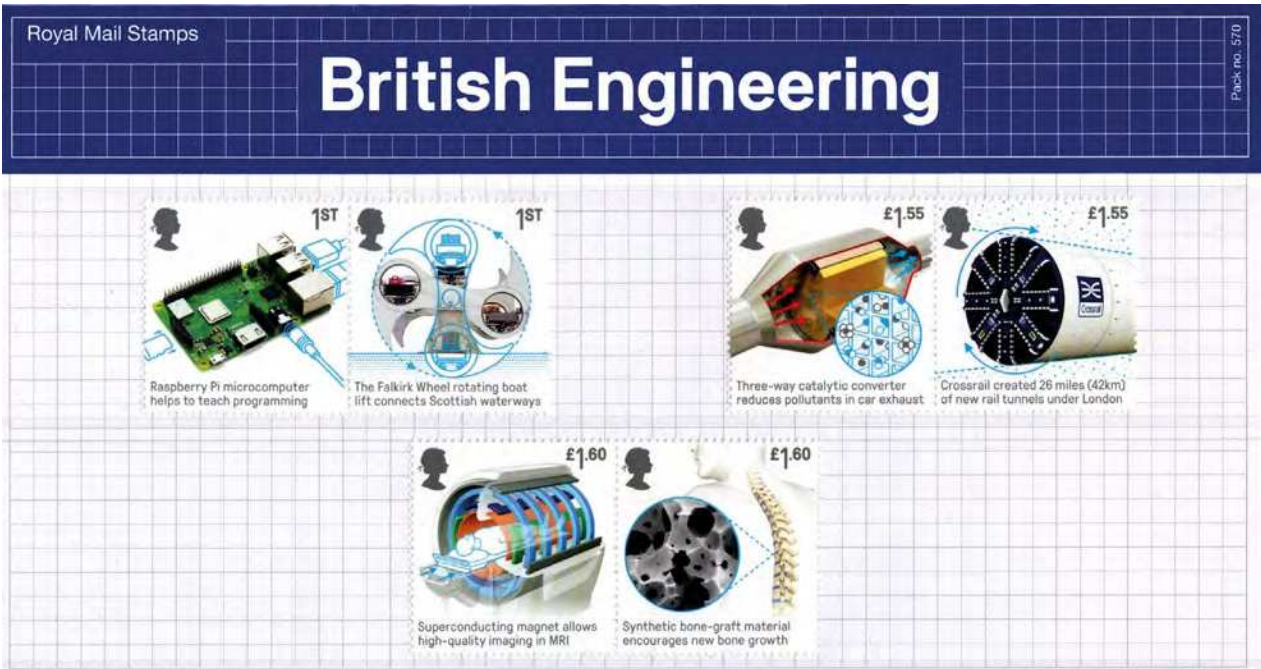


▶ Raspberry Pi can now be found on a first-class stamp

collectable – a special-edition stamp. Liz said, “We’re absolutely made up to be one of the engineering projects chosen: we’re in some exalted company.”

The Raspberry Pi stamp joins the Harrier Jump Jet, catalytic converter, Falkirk Wheel, synthetic bone-graft substitute, and superconducting magnets as world-class examples of British engineering.

Special first-day cover commemorative presentation packs, containing all six engineering-focused stamps, can be bought for £13 from the Royal Mail website (magpi.cc/zVBgUX) and at some Post Office counters. A sheet containing one of each of the



“ The honour of appearing on a stamp comes two years after the Raspberry Pi won the Royal Academy of Engineering MacRobert Award ”

▲ The Royal Mail engineering stamps presentation pack will delight first-day cover collectors

▶ Buy the complete set of engineering stamps as a sheet from Royal Mail

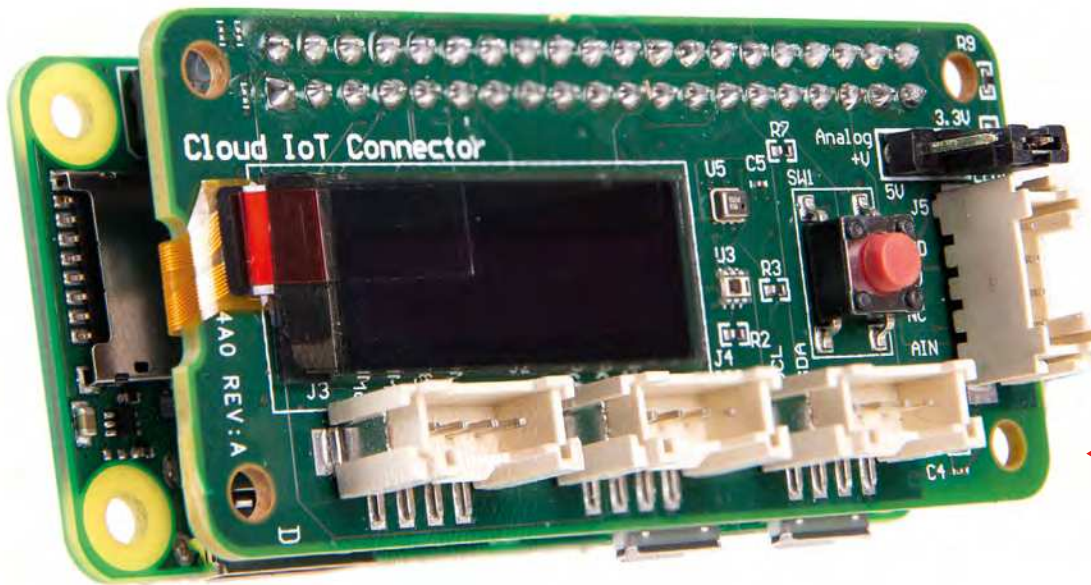
six engineering stamp designs costs £7.70. At larger Post Offices you can also buy a sheet of Raspberry Pi first-class stamps.

Leaping ahead

The engineering series of stamps has been published to coincide with the 50th anniversary of the Harrier Jump Jet. The honour of appearing on a stamp comes two years after the Raspberry Pi won the Royal Academy of Engineering MacRobert Award (magpi.cc/LzNeDT). The prize recognised the Pi’s role in reversing the downward trend in interest in computer science and programming. The stamps also honour 50 years of the MacRobert Award’s existence. **M**



Stamp designs © Royal Mail Group Ltd. Reproduced by kind permission of Royal Mail Group Ltd. All rights reserved.



◀ Packed with data-gathering sensors, the board contains a secure chip for connecting to Google Cloud IoT Core for analysing the information

Coral Environmental Sensor Board

Google launches new board for Pi with sensors and Google Cloud integration.

By **Lucy Hattersley**

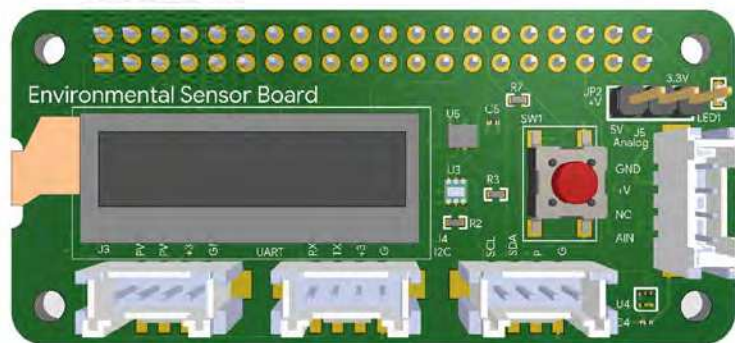
Google has released a new Raspberry Pi product under its Coral brand (previously AIY Projects). The Environmental Sensor Board adds a range of sensors and a 128×32 OLED display to the Pi.

More importantly, the board is designed to connect securely to Google’s Cloud IoT Core service (cloud.google.com/iot-core), enabling you to securely transmit data and perform detailed analysis on it (with Google’s range of ML tools).

The Environmental Sensor Board connects to Google IoT Cloud platform via a Microchip ECC608 crypto chip with Google keys.


On the board, you’ll find a HDC2010 humidity sensor, OPT3002 ambient light sensor, and BMP280 barometric pressure sensor, along with UART, I²C, and PWM connections. A full datasheet can be found here: magpi.cc/uxHKEE.

The new board also sports four Grove connectors (magpi.cc/eCbZbF), enabling you to quickly attach additional components for prototyping without needing to use jumper wires or a soldering iron.



▲ The Environmental Sensor Board packs four Grove connectors that make prototyping faster and easier

Google has documentation for a couple of Python classes, both of which we’re looking at closely for potential *The MagPi* tutorials. The first is `coral.enviro.board`, described as “an interface for all input and output modules”; the second is `coral.cloudiot.core`, which “manages a connection to Google Cloud IoT Core via MQTT, using a JWT for device authentication.” Both can be found at coral.withgoogle.com/docs.

The Environmental Sensor Board costs £20.62 / \$24.95, and is available now from Mouser.com (magpi.cc/fnebgR). 

3 ISSUES FOR £5



📞 Subscribe by phone: **01293 312193**

📧 Subscribe online: **magpi.cc/subscribe**

✉ Email: **magpi@subscriptionhelpline.co.uk**

Yuri 3 Mars rover

Airbus engineer John Chinner unveils the secrets

of the Yuri 3 Mars rover

at the Pi Wars

Wardle
11/6

In Mission Mode, Yuri can be set to live-stream its journey across Mars's terrain while young scientists control it remotely

MAKER

John Chinner

John is an engineer and STEM ambassador at Airbus UK. He worked on the Astro Pi project, testing the Raspberry Pi units sent into space.

magpi.cc/00maUK

In honour of the 50th anniversary of the Apollo moon landing, this year's Pi Wars was space-themed. Visitors to the two-day event – held at Cambridge University at the end of March – were lucky enough to witness a number of competitors and demonstration space-themed robots in action.

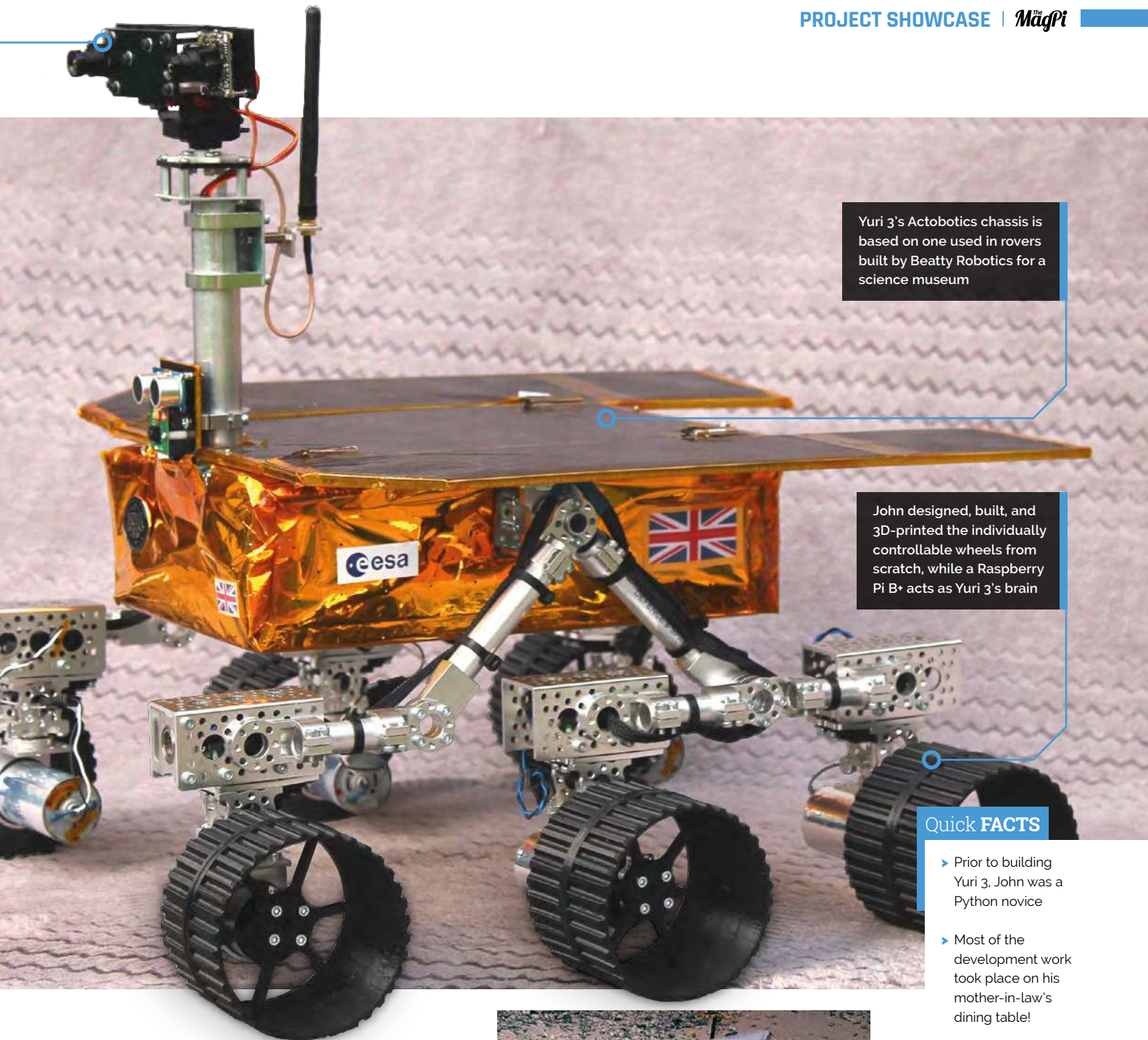
Among the most impressive was the Yuri 3 mini Mars rover which was designed, lovingly crafted, and operated by Airbus engineer John Chinner. Fascinated by its accuracy, we got John to give us the inside scoop.

Airbus ambassador

John is on the STEM ambassador team at Airbus and has previously demonstrated its prototype ExoMars rover, Bridget (you can drool over images of this here: magpi.cc/btQnEw), including at the BBC Stargazing Live event in Leicester. Realising

▼ Yuri 3 sitting alongside Featherstone and Rocky Rover at the Pi Wars 2019 event
Credit: Harry Brenton





Yuri 3's Actobotics chassis is based on one used in rovers built by Beatty Robotics for a science museum

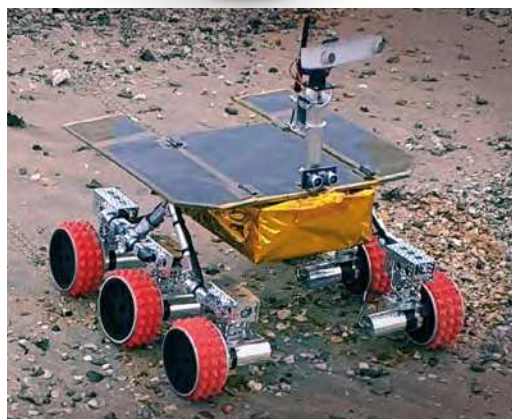
John designed, built, and 3D-printed the individually controllable wheels from scratch, while a Raspberry Pi B+ acts as Yuri 3's brain

Quick FACTS

- ▶ Prior to building Yuri 3, John was a Python novice
- ▶ Most of the development work took place on his mother-in-law's dining table!
- ▶ John was responsible for the shock, vibration, and EMC testing on the Astro Pi units
- ▶ Jim Henson's animatronic puppets were his original inspiration
- ▶ The gold blanket protects the rover in extreme conditions (as found on Mars)

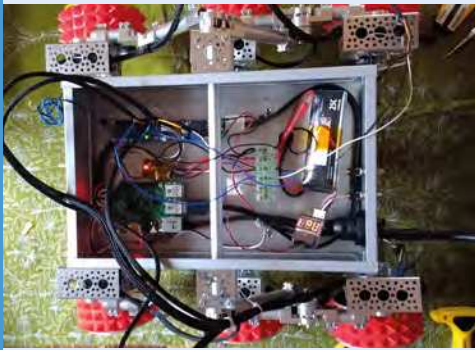
Yuri 3 usually runs in 'tank steer' mode. Cannily, the positioning of four of its six wheels at the corners means Yuri 3's wheels can each be turned so it spins on the spot. It can also 'crab' to the side due to its individually steerable wheels.

The part more challenging for home users is the 'gold thermal blanket'. The blanket ensures that the rover can maintain working temperature in the extreme conditions found on Mars. "I was very fortunate to have a bespoke blanket made by the team who make them for satellites," says John. "They used it as a training exercise for the apprentices."

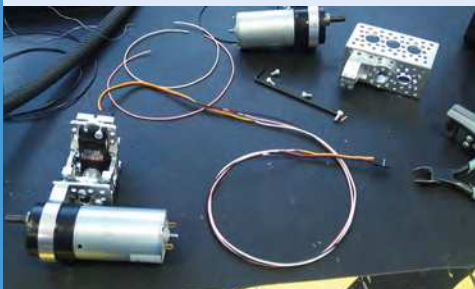


▲ Yuri 3 being tested on a beach to recreate challenging terrain navigation

Building a Mars rover



01 Yuri 3's chassis is based on a Beatty Robotics one, but the rover was designed and built by John. Some parts come from Actobotics, while the steering motors are standard RC-style servos. The gold blanket was bespoke-made for accuracy (but isn't required to function on Earth).



02 Yuri 3 has six separate servo motors, all controlled with Python. John operates the robot using a combination of Nintendo Wii Remote and Nunchuk controllers to manoeuvre the chassis and head respectively.



03 In 'mission mode' you get a rover-centric live-streamed view. Yuri follows commands entered by students in a classroom using the menu-driven Python script. Meanwhile, an audience watching in the school hall witnesses them attempt to avoid obstacles and seek a hidden alien.



▲ Yuri 3 explores the Martian landscape... only kidding - it's the local beach

John has made some bookmarks from the leftover thermal material which he gives away to schools to use as prizes.

Rover design

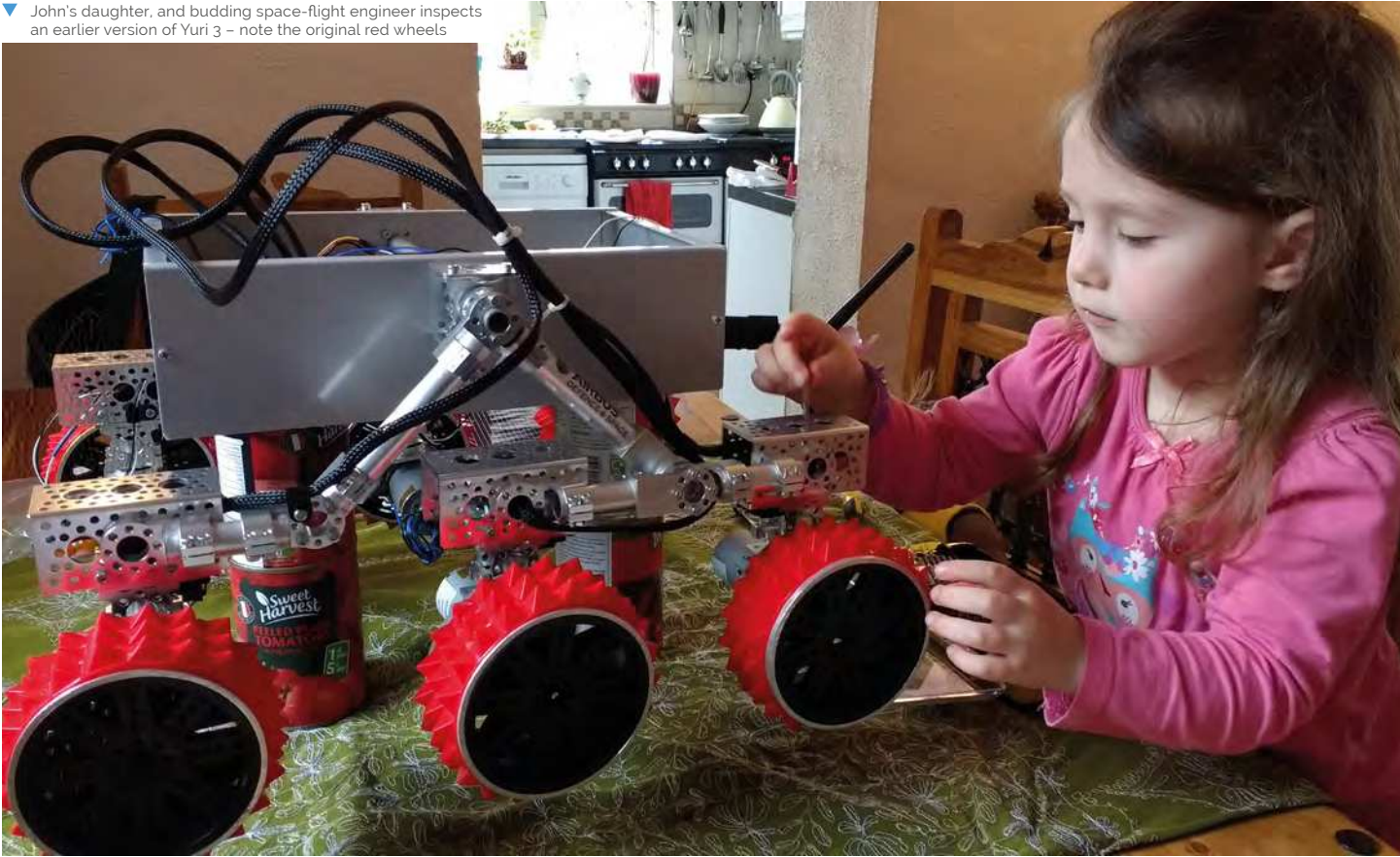
While designing Yuri 3, it probably helped that John was able to sneak peeks of Airbus's ExoMars prototypes being tested at the firm's Mars Yard. (He once snuck Yuri 3 onto the yard and gave it a test run, but that's supposed to be a secret!) Also, says John, "I get to see the actual flight rover in its interplanetary bio clean room".

His involvement with all things Raspberry Pi came about when he was part of the Astro Pi project, in which students send code to two Raspberry Pi devices on board the International Space Station. "I did the shock, vibration, and EMC testing on the actual Astro Pi units in Airbus, Portsmouth," John proudly tells us.

▶ With its six individually steerable wheels and rugged chassis, Yuri 3 can handle rough terrain



▼ John's daughter, and budding space-flight engineer inspects an earlier version of Yuri 3 – note the original red wheels



“ What a fantastic opportunity for exciting outreach ”

A very British rover

As part of the European Space Agency mission, ExoMars, Airbus is building and integrating the rover in Stevenage. “What a fantastic opportunity for exciting outreach,” says John. “After all the fun with Tim Peake’s Principia mission, why not make the next British astronaut a Mars rover? ... It is exciting to be able to go and visit Stevenage and see the prototype rovers testing on the Mars Yard.”

John also mentions that he’d love to see Yuri 3 put in an appearance at the Raspberry Pi store; in the meantime, drooling punters will have to build their own Mars rover from similar kit. Or, we’ll just enjoy John’s footage of Yuri 3 in action and perhaps ask very nicely if he’ll bring Yuri along for a demonstration at an event or school near us.

John blogged the first year of his experience building Yuri 3 on his blog (magpi.cc/oomaUK). And you can follow the adventures of Yuri 3 over on Twitter ([@Yuri_3_Rover](https://twitter.com/Yuri_3_Rover)). [M](#)



▲ The gold blanket is used to keep the Mars rover safe in extreme temperatures. Leftovers have been used to create bookmarks as rewards for students

Telepresence Hand for Hazardous Areas



Andrew Loeliger

A fourth-year student at the University of Strathclyde studying Electronic and Electrical Engineering.

Manipulate objects at a distance without using space wizard powers, thanks to this remote-controlled robo-arm. **Rob Zwetsloot** tries it on for size

There's a scene at the start of eighties classic *Short Circuit* where Steve Guttenberg is hiding away in a lab, programming a robot hand that is playing the piano. It's quite quaint by today's robotic standards (and was probably just a puppet at the time), which is only more apparent when viewing Andrew Loeliger's university project from the last year.

"I set out to provide a solution to the issue of first line responders operating in dangerous areas," Andrew explains. "Bomb disposal sites, biohazardous areas, and nuclear hot zones are all crucial areas where human intervention would be required but could be potentially life-threatening. The aim of the project was to develop an extremely low-cost robotic hand that can operate in hazardous areas and perform dextrous tasks while being controlled and viewed remotely. The visual feedback provided by the system allows the user to control every movement of the hand as if they were there."



▲ Here's the breakdown of how the system works

The user wears a glove that's connected to a 'base station', which also has a display. The display shows the pictures from a wirelessly connected camera, which is part of the remote robot hand system. The glove has a series of sensors to record how the fingers and hand are moved, and that is then relayed to the hand controller.

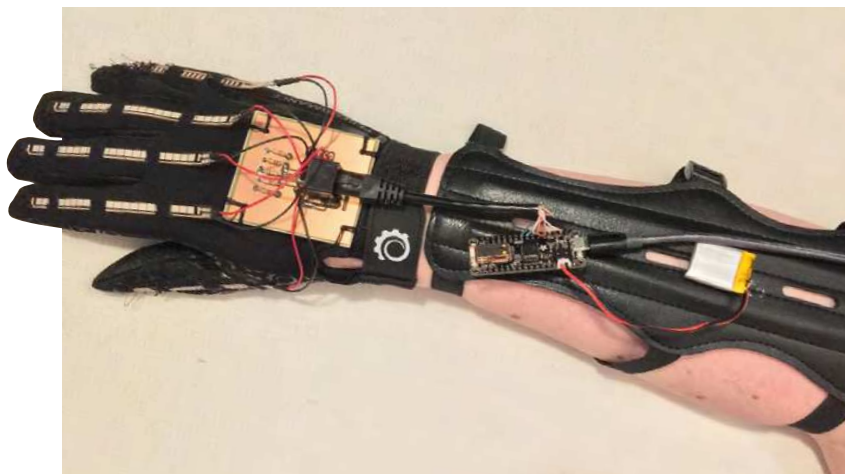
Hand print

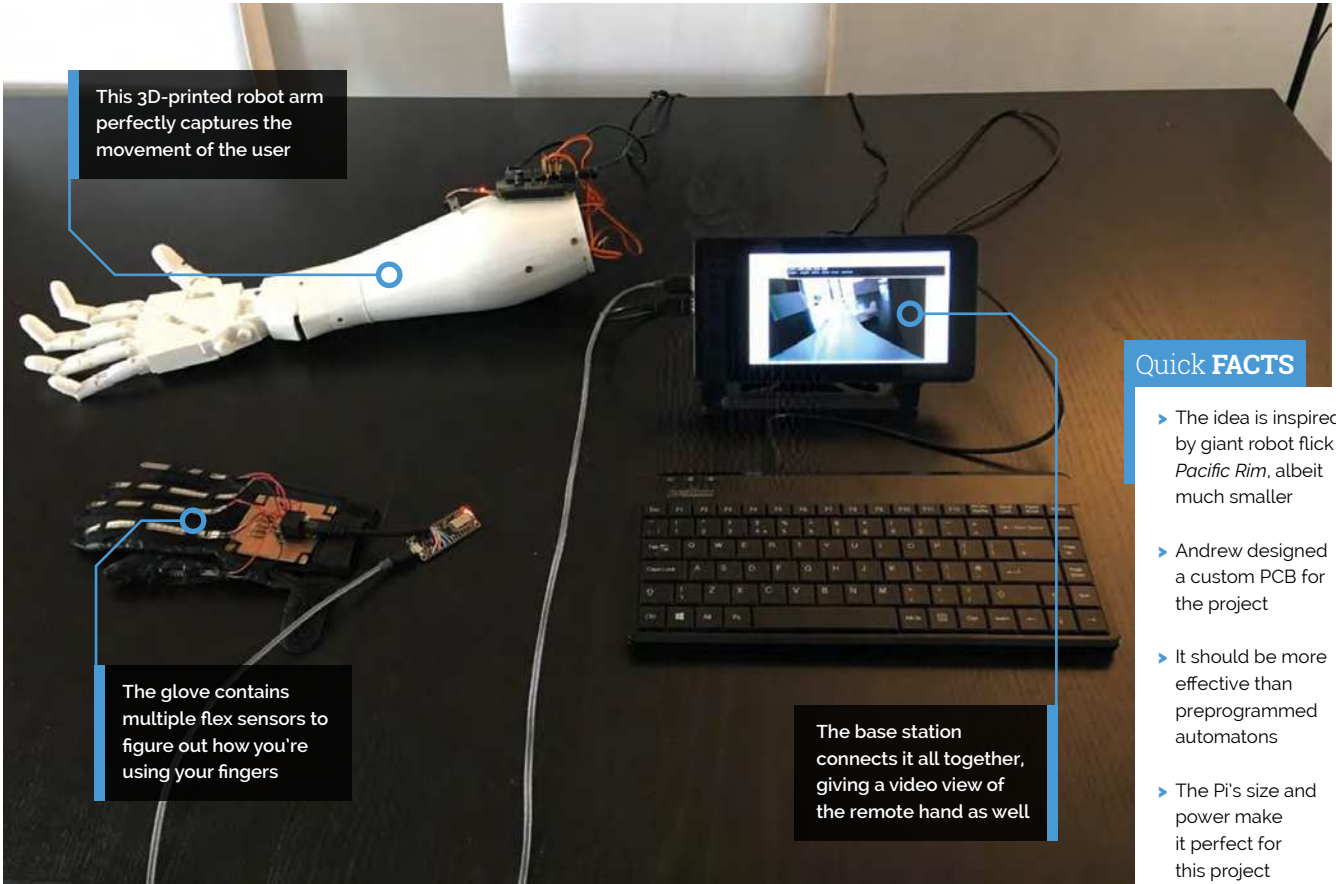
"The bulk of the robot hand comprises 3D-printed components," Andrew tells us. "The design files for the 3D-printed components were sourced from InMoov. The 3D-printed robot hand is designed to hold all of the servos needed to mimic user movements made in the glove remotely. To control the robot hand, there is the Raspberry Pi Zero W which takes the received values from the base station and sends them to the servo driver board to move the servos. There is one servo for each finger and each finger is moved via a braided fishing line pulley system within the finger."

Robotic mimicry

The hand works very well, according to Andrew: "The project worked as hoped and is simple to use, as the user simply needs to put the glove

▼ There's currently no wrist movement on the glove, but this can be easily added





This 3D-printed robot arm perfectly captures the movement of the user

The glove contains multiple flex sensors to figure out how you're using your fingers

The base station connects it all together, giving a video view of the remote hand as well

Quick FACTS

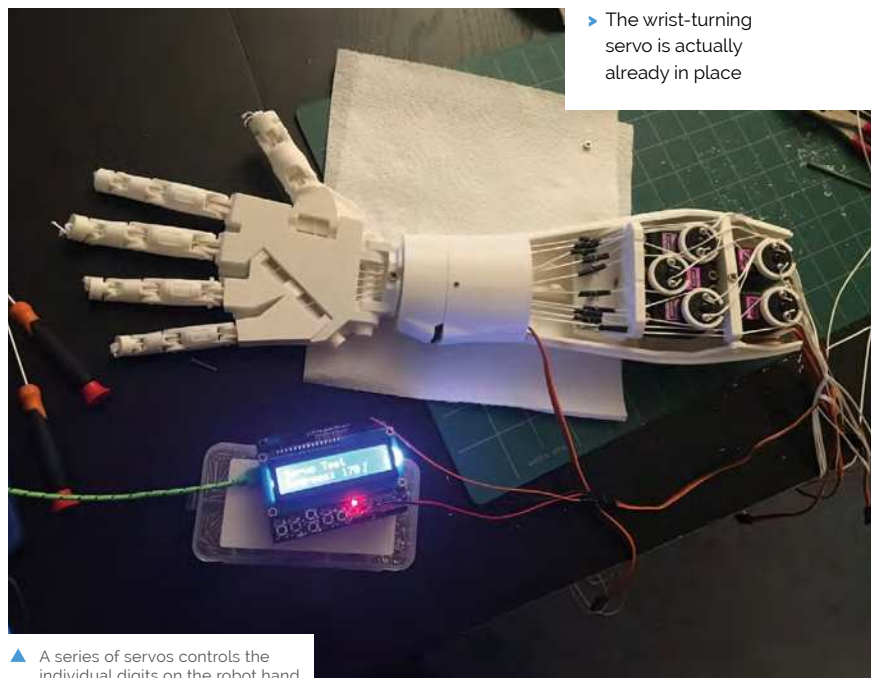
- ▶ The idea is inspired by giant robot flick *Pacific Rim*, albeit much smaller
- ▶ Andrew designed a custom PCB for the project
- ▶ It should be more effective than preprogrammed automatons
- ▶ The Pi's size and power make it perfect for this project

“ The user simply needs to put the glove on and move their hand to make the robot hand work ”

on and move their hand to make the robot hand work. One of the really nice features is that there is low latency between the glove and robot hand movements, which means the hand truly mimics the user's finger movements.”

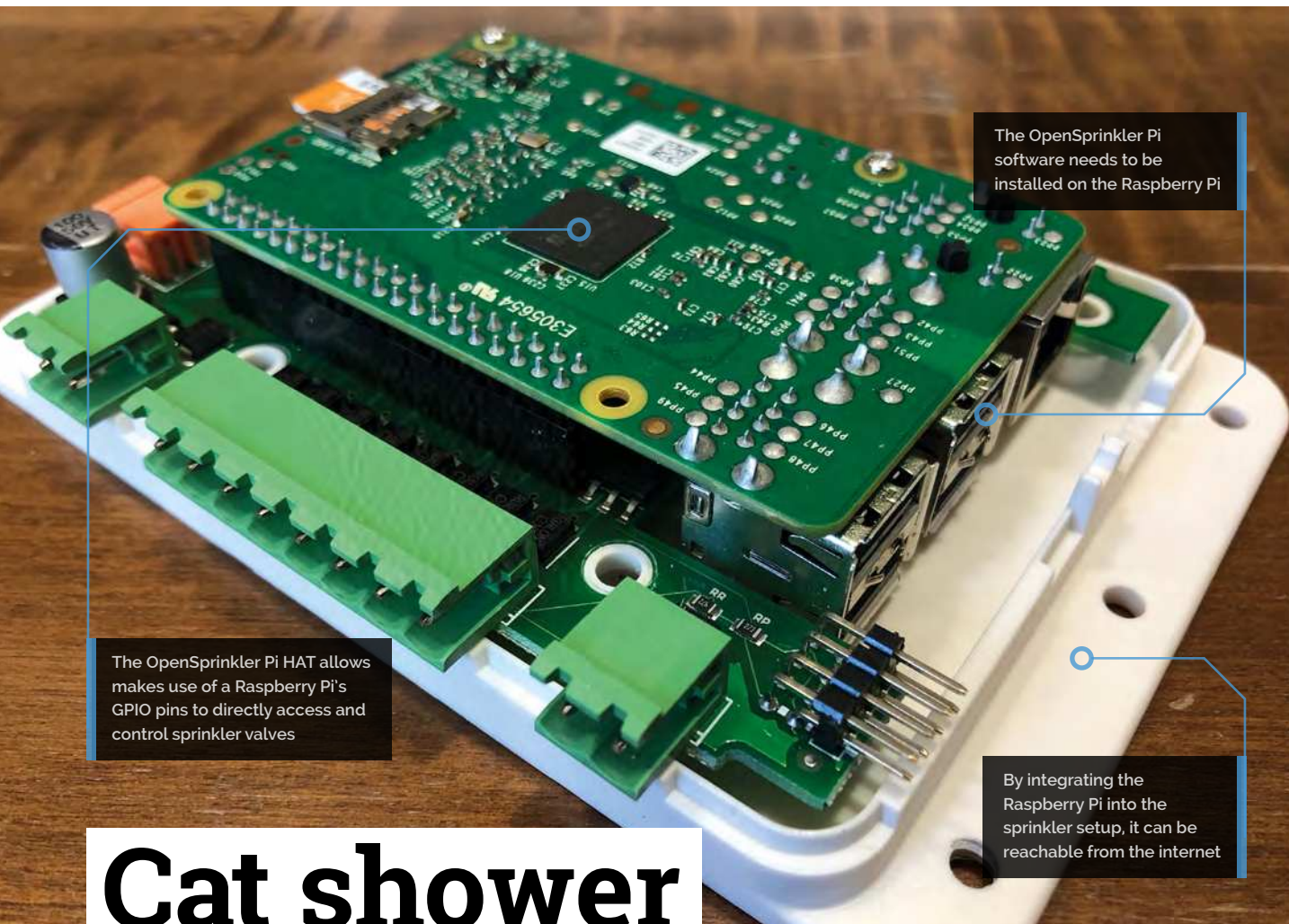
Andrew has plans to improve the design with a new version that makes use of Bluetooth Low Energy for the glow, along with sensors so the hand can turn at the wrist, and perhaps a haptic feedback system as well.

There is one final use case that Andrew has found: “Although I originally considered this project for hazardous areas, interest has been shown in it from a prosthetics point of view.”



▲ A series of servos controls the individual digits on the robot hand

▶ The wrist-turning servo is actually already in place



The OpenSprinkler Pi software needs to be installed on the Raspberry Pi

The OpenSprinkler Pi HAT allows makes use of a Raspberry Pi's GPIO pins to directly access and control sprinkler valves

By integrating the Raspberry Pi into the sprinkler setup, it can be reachable from the internet

Cat shower

Curiosity killed the cat, but water deterred them from pooping in Davide Magni's garden, as **David Crookes** explains



DAVIDE
Davide Magni

Davide is a telecommunications engineer who is addicted to art and design. He has been a programmer for 15 years.

tidal.it

The internet loves cats and they are indeed very cute. But when they're 'feline' in the mood for pooping, there's a good chance one will paw-se for a short spell in your garden and leave you to clear up the mess.

That happened to Davide Magni's cousin, who moved into the house next door to him in Italy. "She spent more than a week cleaning her garden of cat-droppings," he recalls. "All the cats in the area were using that small piece of land as their litter and I wondered how I could prevent it from happening in my own garden in the future."

As it happens, Davide was rearranging an irrigation system in his garden when he began to consider a solution. "I started to place sprinklers in key locations, looking for a way to equally wet the whole garden using trial-and-error and activating

them manually," he says, before realising they could serve a dual cat-chasing purpose.

Sprinkle of Pi

The first important thing was automating the system so that it could be used when Davide was away from home or if he forgot to activate it. "I decided to use a Raspberry Pi because I wanted something that would provide APIs and integrate with other Internet of Things devices I have at home," Davide tells us. "I certainly didn't want a closed, self-contained system that wouldn't interact with other devices."

Luckily, there is an open platform which allows irrigation systems to be controlled: called OpenSprinkler, it supports the Pi, has its own extension board, and is connected to the web for



◀ And it's off: the camera catches the moment a cat is squirted with water. It's soon over the gate

Quick FACTS

- ▶ OpenSprinkler Pi (\$75) is at the project's heart
- ▶ It makes use of an extensive sprinkler system
- ▶ Cats are detected by a smart camera
- ▶ Sprinklers are activated to chase them away
- ▶ No cats are harmed by the water

remote access. “It provides everything I need for good irrigation control such as daily programming, a web GUI, and weather-based prediction,” Davide continues. With that in place, it was time to turn his attention to those troublesome cats.

“ My cousin spent more than a week cleaning her garden of cat-droppings ”

Pi spy cats

Davide’s wife had bought him a Netatmo Presence outdoor smart camera as a Christmas gift. It detects movement, shines a bright light, and records what it sees; it can also distinguish between people, animals, and vehicles. As such, it was the perfect input device.

Initially, he pushed the Netatmo trigger through IFTTT to send a notification that activated the sprinklers, but it took so long that the cats would have done their business and already left. To improve performance, he created an ad-hoc application with Netatmo Connect which receives the camera trigger and sends it to a self-hosted site



that runs a small script. This decides if the shower needs activating and sends it to OpenSprinkler.

“I added a callback, a simple PHP script call, that is invoked directly from the camera every time it detects a movement,” says Davide. “The script verifies that the call is valid, checks the system status, sends the activation signal to OpenSprinkler, and logs the event.”

Now, when a cat strays into the garden, the sprinklers kick in within seconds and chase them away, but there are some pitfalls. “For future developments, I’m thinking about being able to keep a small history of events in order to avoid activating a shower if a person and an animal approaches – such as my mum passing with her dog,” he laughs. [M](#)

▲ The garden excavation for the laying of the water pipes and the electric cables to control the opening of the valves was the toughest part of the project

DIY Hardware Password Keeper

Do you use Password123 for all of your online services? **David Crookes** looks at a Pi Zero-based device that should help you come up with more secure logins



MAKER Eugene Dzhurynsky

Eugene is a Ukrainian software engineer living in Boston, Massachusetts and he is currently working as a data engineer and machine-learning specialist.

magpi.cc/SELOuy

Experts suggest that we should use a different password for every online service we access, but, as many users know, trying to remember them all can be rather difficult.

Better, then, to have a secure and convenient system that can store and recall them for you, which is why Eugene Dzhurynsky has created a hardware password storage device based around a Raspberry Pi Zero and radio frequency identification (RFID) technology.

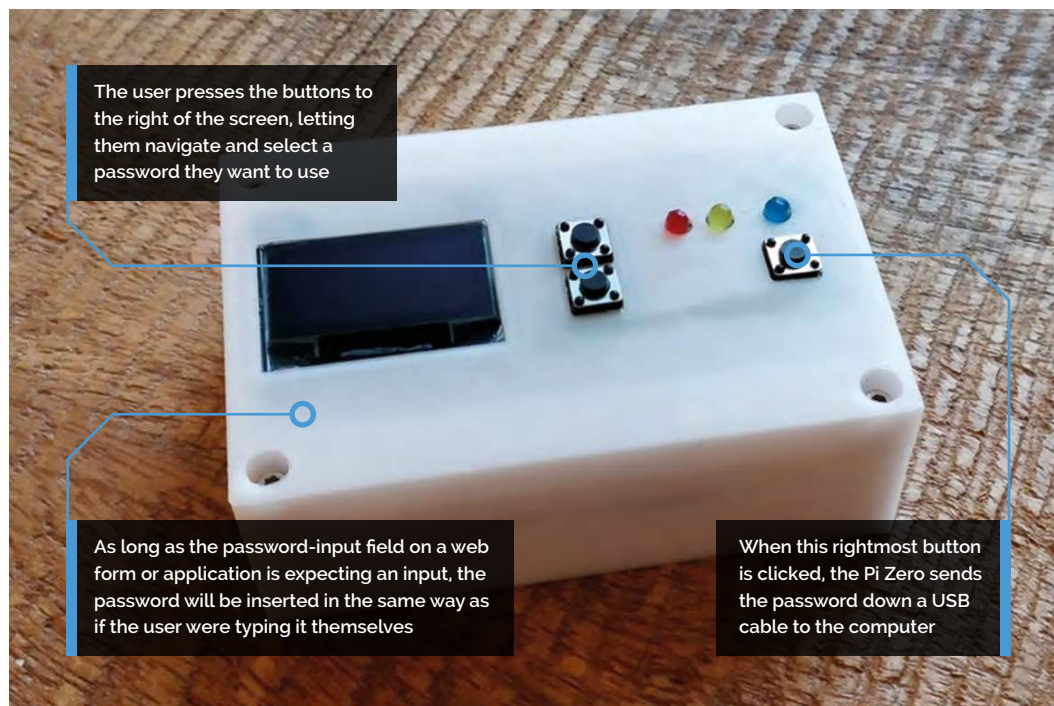
Inspired by a thread on Reddit which explained how the Pi Zero could function as a USB device, Eugene noted that the Pi Zero could control another computer and act as a keyboard. “It was a ‘wow’ moment and I was stunned by the possibilities,” he recalls.

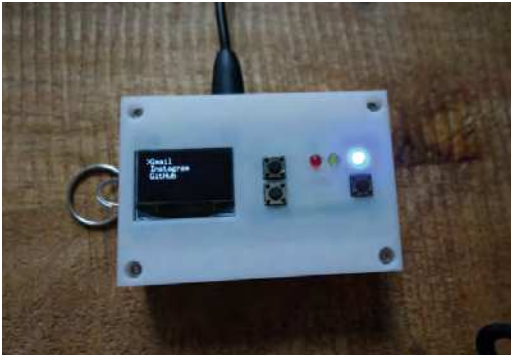
With this in mind, Eugene got to work. He wanted to create a device that couldn’t be hacked remotely over a network. What’s more, he didn’t want users to have to type in a password to unlock the device and he wanted it to be relatively small, with secure and encrypted storage.

Securing the system

Encryption, he says, was the easy part. “I could use any industry standard encryption,” he explains. The tricky aspect was the key – the component that would unlock the encrypted file containing a user’s passwords.

Eugene intended to store the passwords on the Pi Zero, so he figured that placing a private key on an RFID fob would work well. With the Pi-based





▲ When activated, the screen shows accounts that have stored passwords and allows them to be used on a linked computer

device wired to an RFID card reader and connected to a computer, he reckoned a user would only need to bring the fob close to unlock the device and allow the passwords to be accessed and shared. A user could then log in to whatever service they were trying to access.

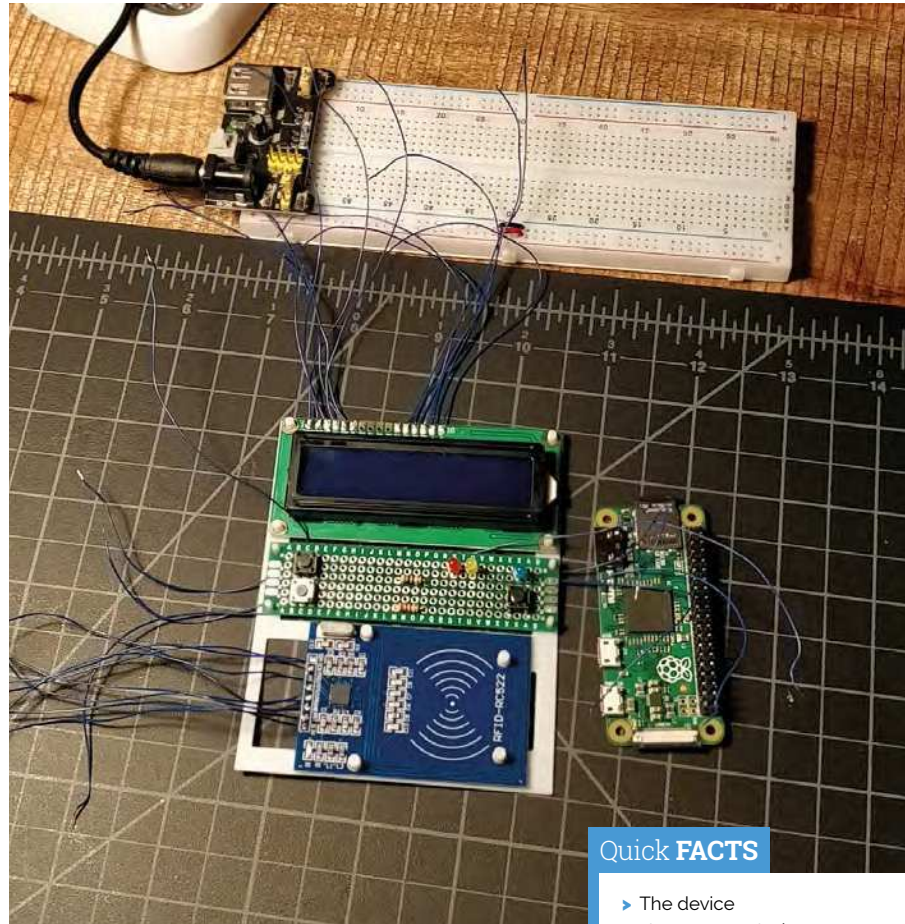
Certainly, the Pi Zero proved the perfect fit. “It has a Linux OS, it’s easy to manage, it’s developer friendly, and it has enough memory and network adapters,” Eugene says. He coded it using the programming language Go (golang.org) and he fitted the device with a small OLED SPI display before allowing it to be controlled via some buttons positioned on the front. All were placed in a self-made case.

“ I don't use it to keep my own passwords – it's just a pet project ”

Playing it safe

In order to add and manage the passwords, Eugene then made use of a web interface which was put together with the help of friend Maxim Vassilyev. “Pi Zero can present itself as network interface as well as a keyboard so by navigating to the local address <http://10.101.1.1>, the web interface for the password management will be accessible,” Eugene explains.

He says it only works in the presence of the fob: “So if someone will try to plug the Pi into a



▲ The RFID RC522 sensor is inexpensive and allows this system to become unlockable only when an RFID key fob is presented

computer, they won’t be able to access passwords.” And yet the system is not quite perfect. “The key can be compromised if it’s used with some cheap RFID cards,” Eugene admits. “But if you just keep the key in an RFID-protective cover, then it’s going to be safe.”

So could it be used as a professional tool in its current state? “Tough question,” replies Eugene. “I don’t use it to keep my own passwords, that’s the answer. But it’s just a pet project and there are many things to be added before it will become truly safe and secure.”

Indeed, he wants to add backup and restore options for the internal storage and RFID key, and he wishes to improve the web interface, add a help section, provide an easy way to add more keys and users, have the device generate passwords, and add a real-time clock for random seed generation. “People seem to like the approach I’ve been taking,” he says. [M](#)

Quick FACTS

- ▶ The device stores encrypted passwords
- ▶ The software was built using the language Go
- ▶ It makes use of an RFID key fob
- ▶ Passwords are added via a web interface
- ▶ Eugene wants to add voice recognition

Pi Vizuwall

This Raspberry Pi cluster has moving parts and a serious goal to help train programmers. **Phil King** finds out more



Matt Trask

Matt is a Researcher and Chief Engineer at the FAU Machine Perception and Cognitive Robotics Lab and is grateful to the FAU Office of Undergraduate Research and Inquiry for its financial support for this project.

magpi.cc/0TdyHN

Mounted on a clear acrylic plate, twelve Raspberry Pi boards suddenly spring into life, moving outwards, as if waving to the attendees at Maker Faire Miami. Not just a cool effect, the movement is proportional to each Pi's level of activity in a parallel computing cluster. This is Pi Vizuwall, a project created by long-time computer engineer Matt Trask during his degree course at Florida Atlantic University (FAU), while doing research into a new class of supercomputer systems.

"When I am successful (heh, nearly said 'if there...'), it will obsolete MPI [Message Passing Interface] as the main means of programming distributed compute clusters," explains Matt. "This means that my variant of the Beowulf architecture will function as a distributed symmetric multiprocessing system that appears to be a single unified system that is the sum of all RAM and all cores in the cluster: a virtual

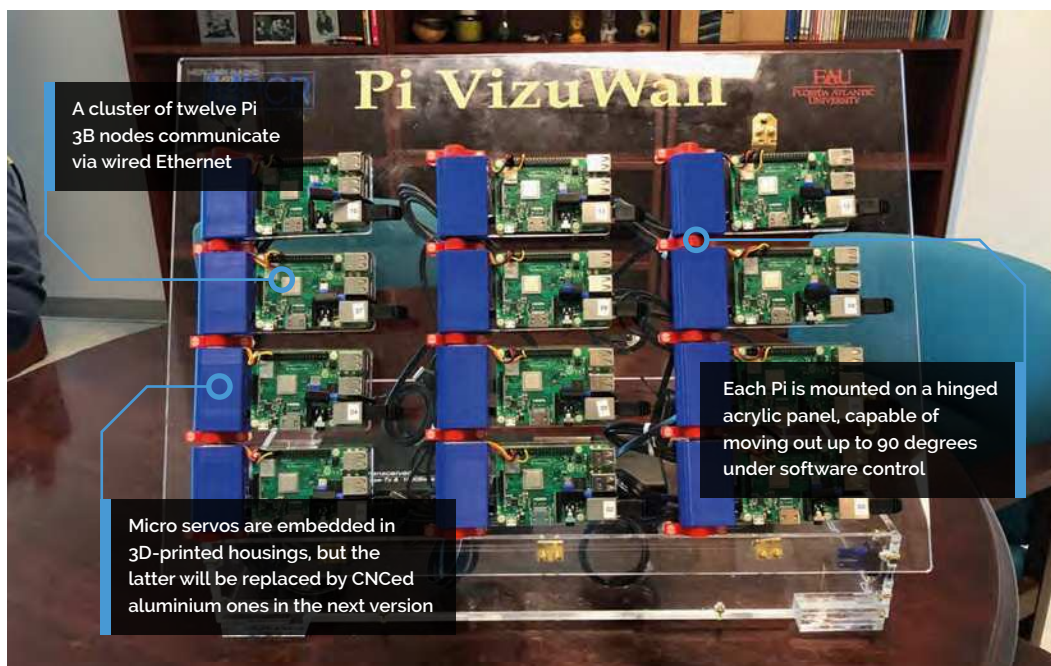
mainframe computer. Perhaps the solution to the so-called Ninja Gap?"

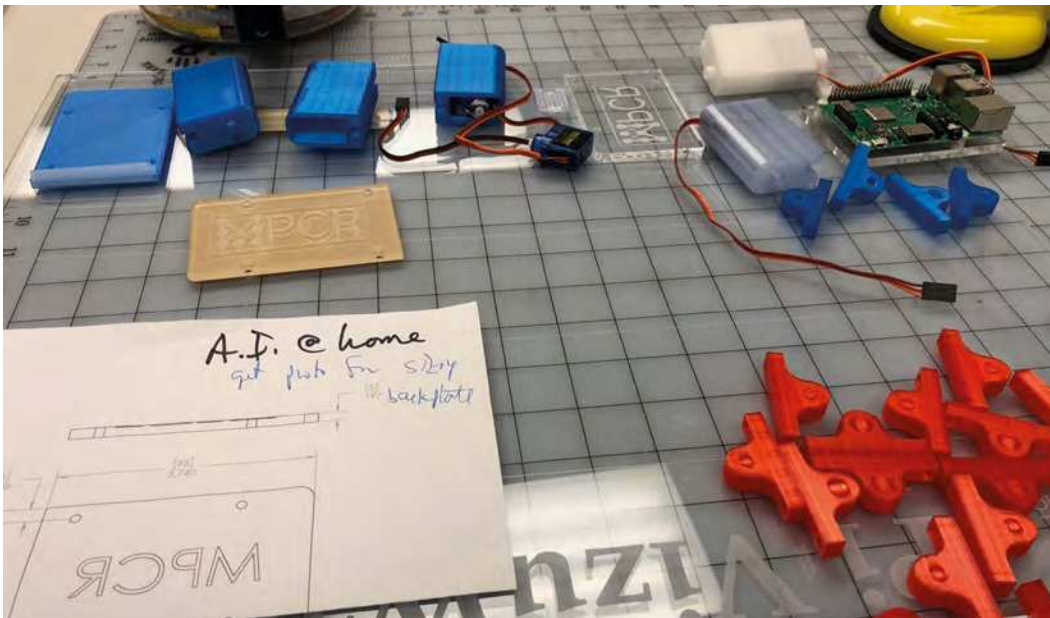
Matt is referring to the difficulty of enabling computer science students to obtain enough early experience programming parallel computing systems to become industry-proficient. Hence his motivation for building a low-cost cluster system with Raspberry Pi boards, in order to drive down the entry-level costs.

Moving parts

Matt reveals how Pi Vizuwall works: "Each node is capable of moving through about 90 degrees under software control because a small electric servo motor is embedded in the hinging mechanism. The acrylic parts are laser-cut, and the hinge parts have been 3D-printed for this prototype."

While the original concept was to also use LEDs to edge-light the acrylic plate and change the colours to indicate CPU usage, Matt says the idea





◀ Hinges and housings were 3D-printed, while the acrylic panels were laser-cut

Quick FACTS

- ▶ This prototype was created to justify the cost of building a much larger version
- ▶ PoE will be used to simply power distribution in the next iteration
- ▶ Matt has been a computer engineer for nearly 40 years
- ▶ He wrote the first virtual machine software on the 80386 processor in 1986
- ▶ He got his first Raspberry Pi in 2012

“ The physical motion helps student programmers understand their system utilisation ”

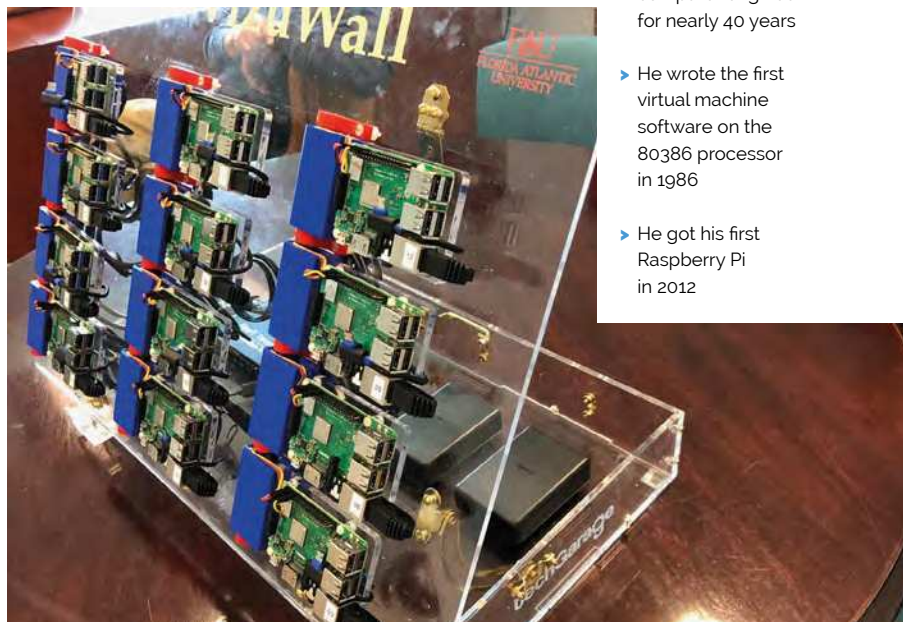
of the moving boards was always fundamental to the project: “I figured that the physical motion would help student programmers understand their system utilisation. And it looks cool.”

Although Matt came up with the project’s concept several years ago, he only started building it in late 2018. “I collaborated with Art Rozenbaum (FAU Mechanical Engineer) over the fall to develop the concept and submitted my research proposal in November,” he tells us. “Art and I worked through my original concept for mounting the servos behind the board and pivoted to his concept of embedding them in the hinge mechanism, a much cleaner solution.”

Currently, the Raspberry Pi boards have wired communication via a 14-port Ethernet switch, but Matt is looking into making it wireless. This will involve “evaluating whether the Pi’s wireless LAN capability is suitable for carrying the MPI message traffic, given that the wired Ethernet has greater bandwidth.”

Scale model

The original plan for Pi VizWall was to create a 4×8 ft (1.2×2.4 m) wall with 300 Raspberry Pi boards wired as a Beowulf cluster running the MPICH implementation of MPI. “When I proposed this project to my Lab Directors at the university, they



balked at the estimated cost of \$20–25,000 and suggested a scaled-down prototype first.”

Matt says some lessons have been learnt in the process of building it, including plans to replace the 3D-printed plastic motor housings – which suffered minor distortion due to heat from the servos – with CNCed aluminium. “This will [also] permit us to have finer resolution when creating the splines that engage with the shaft of the servo motor, solving the problem of occasional slippage under load that we have seen with this version.”

The ultimate goal is to “create a massive piece of kinetic art to embellish the entryway to our new Lab facility at the university.”

▲ Ethernet cables are tucked neatly between each Pi and its hinged panel, so as not to impede its movement

The Raspberry Pi Hologram Machine

Wondering what to do with a spare 17-inch monitor, Dan Aldred teamed it with an old IKEA table and an acrylic pyramid. The results are truly hologramtastic, as **Nicola King** discovers



Dan Aldred

Raspberry Pi enthusiast, NCCE facilitator, teacher, and coder who enjoys creating new projects and hacks to inspire others to start learning. Still trying to get a Kinect 360, Python, and Pi working with each other.

magpi.cc/JJSohM

Dan, a secondary school teacher, was chatting one day with his students about a hologram hack that some of them had developed using a mobile phone and an old CD case, as he explains: “The phone lies flat on a surface of the screen and a small, inverted pyramid sits on top of the screen. I asked one of the students if you could flip the pyramid and still create the hologram. They said, yes, if the screen is at the top rather than the bottom.”

So, armed with this snippet of inspiration, Dan decided to model his very own version: “I wanted to create a large standalone machine that would display holograms and let the users select them with buttons. I also knew that it would make a good centrepiece for open evenings!”

Building a pyramid

Simple in design, Dan’s idea developed quickly. His first issue concerned the dimensions of the pyramid, which he made himself out of acrylic using a laser cutter. “The main focus of the build was on the calculations for the pyramid, to ensure that it was the correct size for the screen,” says Dan. “I had a spare screen which I wanted to reuse, rather than buying a new one. I also had an old IKEA coffee table which I upcycled to hold the screen and house the pyramid. Basically, I cut the legs in half and used one half for the base legs, and the other for the monitor supports.”

So, how does this machine work exactly? Dan explains, “A screen is supported above a clear,

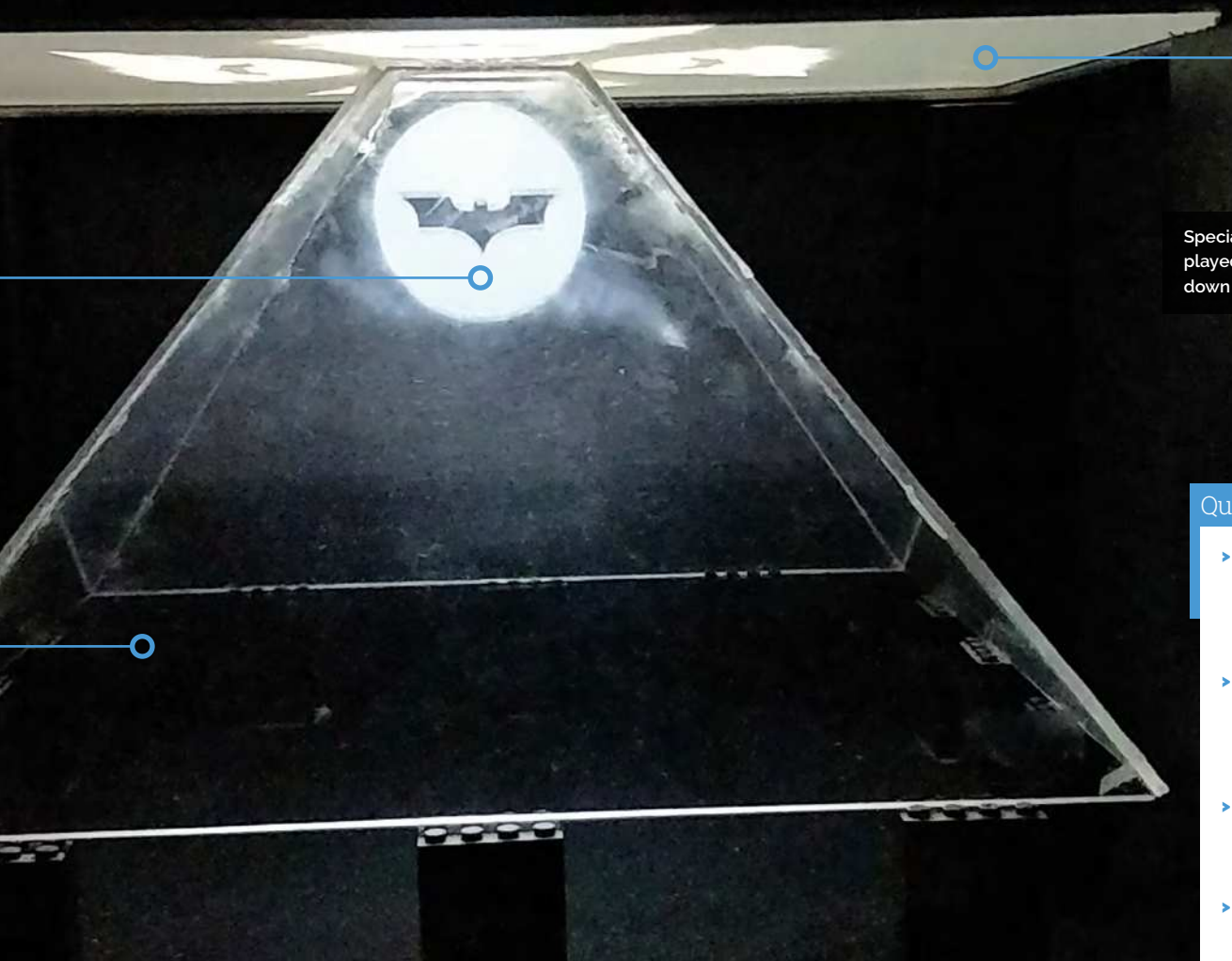
The four projected images combine into one, appearing as if inside the pyramid

Dan built the pyramid from laser-cut acrylic

four-sided pyramid. The Raspberry Pi A+ runs a Python program on bootup, which plays videos from a folder. These are ‘hologram’ videos which contain the same image orientated in four different angles: front, back, left, and right. The acrylic reflects back the image from the overhead screen, creating the illusion that a single solid object is floating in the middle of the pyramid. As you view the pyramid from a different side, you see a different perspective.” Dan has also added three buttons – coded with the GPIO Zero library – to pause a video, skip to the next hologram, and shut down the Raspberry Pi.

3D fun

As with many projects, a little fine-tuning was required to ensure it all worked as Dan had envisaged. For example, he had to be careful what videos he used in the machine: “I had to search for inverted videos as the most popular videos are designed for the phone version with the upside-down pyramid. This means that, when playing a video in my Hologram Machine, they were upside down.”



Special videos are played on an upside-down monitor

Quick FACTS

- ▶ Dan's most popular hologram is the incredibly realistic lightning
- ▶ Videos must be inverted to show correctly in the pyramid
- ▶ The video player is based on code by Les Pounder
- ▶ The sawn table legs were filled with expanding foam

“ I wanted to create a large standalone machine that would display holograms ”

One would guess that you could make the Hologram Machine any size, then, as long as your screen is big enough? “On posting the completed project on Twitter, one of the first responses was a retweet with the idea that I should use a projector as the screen and build a giant one-metre hologram! So yes, as long as the screen is big, you can increase the size of the pyramid and create a larger hologram,” reckons Dan.

As to the future, he says that he would like to develop the Hologram Machine with an HDMI monitor, so as to improve hologram quality. “I am also looking at an alternative to the acrylic as it is difficult to keep clean!” he adds. “You only have to look at it and it gets dirty – perhaps a glass pyramid.”

▼ The monitor is mounted on the sawn-off legs of an IKEA table, its top used as a base



▶ Dan is working on a real-time night-vision camera

SUBSCRIBE TODAY FROM ONLY £5

SAVE UP TO 35%



Subscriber Benefits

- ▶ **FREE Delivery**
Get it fast and for FREE
- ▶ **Exclusive Offers**
Great gifts, offers, and discounts
- ▶ **Great Savings**
Save up to 35% compared to stores

Rolling Monthly Subscription

- ▶ **Low Monthly Cost** (from £5)
- ▶ **Cancel at any time**
- ▶ **Free delivery to your door**
- ▶ **Available worldwide**

Subscribe for 12 Months

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

Free Pi Zero W Kit with 12 Month upfront subscription
only (no Pi Zero Kit with Rolling Monthly Subscription)

📞 Subscribe by phone: **01293 312193**

📧 Subscribe online: **magpi.cc/subscribe**

✉ Email: **magpi@subscriptionhelpline.co.uk**

JOIN FOR 12 MONTHS AND GET A
FREE Pi Zero W Starter Kit
 WITH YOUR SUBSCRIPTION

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

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

Offer subject to change or withdrawal at any time

WORTH £20



 Buy now: magpi.cc/subscribe

SUBSCRIBE
 on app stores

From **£2.29**



LEARN TO CODE

WITH **SCRATCH** & **PYTHON**

Anybody can learn to code! Programming a computer is much easier than you think. Come with us and we'll help you get started



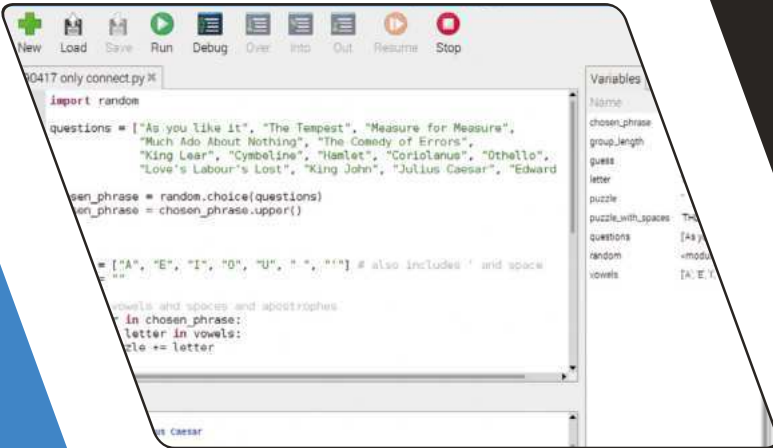
AUTHOR

**Sean
McManus**

Author/co-author of inspiring coding books including Mission Python, Cool Scratch Projects in Easy Steps, and Raspberry Pi For Dummies. Get free chapters at Sean's website.

sean.co.uk





```

import random

questions = ["As you like it", "The Tempest", "Measure for Measure",
             "Much Ado About Nothing", "The Comedy of Errors",
             "King Lear", "Cymbeline", "Hamlet", "Coriolanus", "Othello",
             "Love's Labour's Lost", "King John", "Julius Caesar", "Edward

chosen_phrase = random.choice(questions)
chosen_phrase = chosen_phrase.upper()

shift = 3

new_phrase = ""
for letter in chosen_phrase:
    if letter in vowels:
        new_phrase += letter
    else:
        new_phrase += chr(ord(letter) + shift)

print(new_phrase)

```

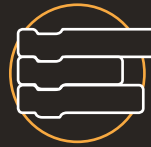
Learning to code can be one of the most profound skills you will ever develop. With code, you can control a computer. You can get it to do things for you, and also control gizmos and other computers. Kick back and let your computer do all the work.

Sure, that's cool. But coding is about more than that. It's about understanding how computers work, and getting a better understanding of how technology – and the modern world – works. It's about breaking down problems into little bits and solving them. It's an amazingly helpful life skill. That's why it's profound.

On a more practical level, knowing just a little code can lead to better job opportunities; a little more can open up well-paid and fun jobs. It's an impressive skill to put on your CV and anybody can do it. Anybody.

Coding is a lot easier than you think. And putting the power of computing and digital making into the hands of people is what Raspberry Pi is all about.

The Raspberry Pi is 'the little computer that could', and you're 'the person who can'. Don't worry: you've got this. We can help you get started.



p28 START CODING WITH SCRATCH



p31 CODE A QUIZ GAME WITH PYTHON



p34 MAKE AN LED TORCH WITH PYTHON



p35 BUILD AN ELECTRONIC GAME

“ The Raspberry Pi is ‘the little computer that could’, and you’re ‘the person who can’ ”

Start coding with Scratch

Beginners, arise! It's time to take your first steps with coding, as we introduce you to Scratch and Python, with a sprinkling of twinkling LEDs! By **Sean McManus**

Being able to write programs is like a superpower: it means you can get your computer to do whatever you want. Join us as we show you how to make your first programs using Scratch and Python. You'll also see how easy it is to build simple electronics projects.

A program is just a set of instructions. In Scratch, the instructions are written with visual blocks that lock together to make a sequence called a script.

The blocks are colour-coded to help you find them. To find the brown blocks, for example, click the brown Events button above the Blocks Palette.

Scratch makes coding easier, because you don't need to worry about the spelling of commands. And everything is laid out in front of you.

Understanding coordinates

The Scratch screen is divided into units called steps. When the cat moves 10 steps, it only makes one movement, but that stride shifts it 10 positions across the Stage. The middle of the Stage is at $x=0$, $y=0$. The x-axis (left to right) runs from -240 to +240, and the y-axis (bottom to top) runs from -180 to +180. The directions the sprite can move in are numbered 0 (up), 90 (right), 180 or -180 (down), and -90 (left). You can use numbers in between those numbers too, so -45 would be a north-west direction. Why not try starting a new project and joining some Motion blocks together to experiment? You can run a script or a block by clicking it, or use the **when flag clicked** block as we did in our program here.

You'll Need

- ▶ Scratch 2
- ▶ Raspbian with Desktop and Recommended Software
- ▶ A good sense of timing!

01 Start Scratch 2

Open Scratch by clicking on the Raspberry Pi Menu icon and choosing Programming > Scratch 2. You will see Scratch interface and a single character in the top-left, known as 'Scratch Cat'.

To control the Scratch Cat, we're going to drag blocks from the Blocks Palette into the Scripts Area and join them together.

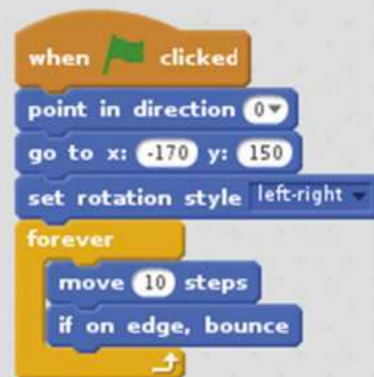
Start by clicking on Events and drag the **when clicked** block to the Scripts Area.

Now click on Motion and drag a **point in direction 90** block and connect it to the bottom of the **when clicked** block.

Click on the fields in the blocks to edit the numbers. Click on '90' and change it to '0'.

Now click and drag the blocks below, and edit their numbers, to build a script for Scratch Cat. This script runs when you click the green flag above the Stage. It sets the cat's movement direction to up, puts the cat in the top-left corner of the Stage, and sets it to always face left or right. Then, the movement blocks inside the **forever** bracket keep the cat moving all the time.

Click the green flag to run your script. Scratch Cat will move to the left side and bounce up and down.





- 01** **Blocks Palette:** Find instruction blocks here
- 02** **Scripts Area:** Drag and drop blocks here to build your script (program)
- 03** **Buttons:** Click to view different types of blocks in the Blocks Palette below
- 04** **Sprite List:** Select and manage sprites here

02 Send a broadcast

The moving objects in Scratch, including the cat, are called sprites. One sprite can send a message to all the other sprites using a broadcast. You can't hear it or see it on screen, but sprites can listen for it, and then start a script when they receive it. We'll use a broadcast to make the cat throw some bananas. Click the brown Events button, and add the two blocks below to the Scripts Area. This new script doesn't join to the existing script (from Step 1) – it sits on its own in the same Scripts Area.

You need to change 'message1' to 'fire'. Click the down arrow next to 'message1' in the broadcast block and choose New Message. Enter the message name 'fire' and click OK.

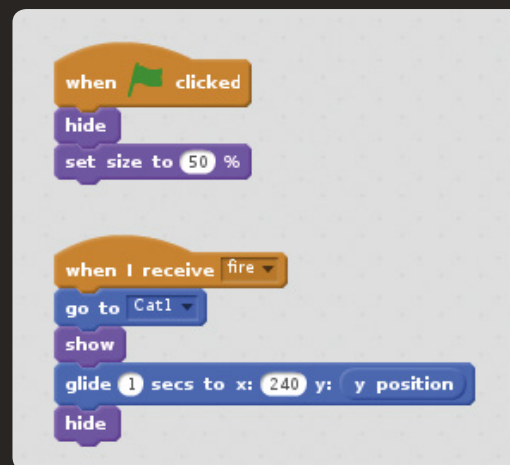
As the program is running, when you tap **SPACE**, the broadcast message is sent silently. Crafty!



03 Add aerodynamic bananas

Scratch enables you to get results fast because it includes its own images and sounds. Click the first New Sprite button above the Sprite List (it will display 'Choose sprite from library' as you hover over it).

Click the Bananas sprite to add it to the Sprite List. Notice that its Scripts Area is blank. We'll give Bananas two scripts. The first one sets the sprite's size and makes it invisible when you click the green flag. The second runs when the cat broadcasts its fire message. Click and drag the blocks below to the Scripts Area.



When adding the **glide** block, drag the **y position** block inside the round 'y:' field to replace the '10' default value.

The 'fire' broadcast makes the bananas jump to the cat (use the go to mouse-pointer block, and choose Cat1 in its menu). Then it makes the bananas visible, glides them across the screen, and hides them again.

Click the green flag, and tap **SPACE** to test. Scratch Cat now throws bananas.

Top Tip

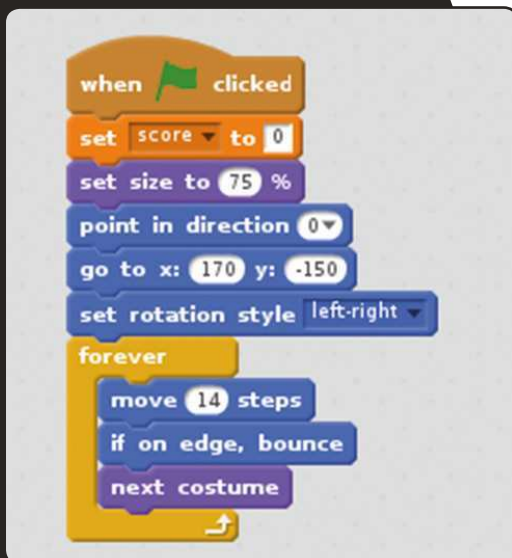
Get the right sprite

Make sure you're adding scripts to the correct sprite. You can select a sprite by clicking it in the Sprite List.

Top Tip

Blocks inside blocks

Some blocks can go inside other blocks. In Step 3, the **y position** block keeps the sprite's y position the same while its x position changes.



04 Add monkey magic

Variables are names used to remember information, such as a score, that might change. Click the Data button and click Make a Variable. Give it the name 'score', select For All Sprites, and click OK. Let's add a moving target for the cat to try to hit. Click the 'Choose sprite from library' icon and choose the Monkey2. Select Monkey2 in the Sprites area and give it the script above.

The monkey's script sets the new score variable to 0 when the game starts, then makes the monkey move up and down. Each sprite can have more than one image: the Next Costume block cycles through them, creating an animation.

05 Add collision detection

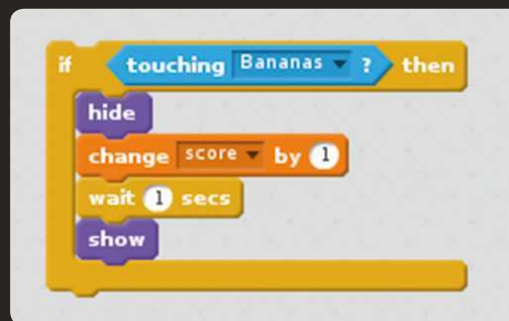
Our monkey will react when it's struck by a bunch of bananas. To do that, we use an **if** block; this checks whether

▶ These last two scripts add a timer and make the monkey react when it sees bananas



Where next?

We're huge fans of Scratch at *The MagPi*, so check out our past issues online for more Scratch tutorials. Issue 5 includes a memory game, like Simple Brian. Issue 34 has a multiple-choice quiz, and our 2018 Annual included an introduction to electronics and Scratch. See issue 76 for a roundup of resources to help you learn Scratch, and don't forget there's a Scratch book in *The MagPi's* own Essentials series (magpi.cc/learnscratch) and the *Code Club Book of Scratch* (magpi.cc/ccbook1).



something is true – in this case, whether the monkey is touching the bananas. If so, the blocks inside its bracket are run. Here, those blocks hide the monkey, add 1 to the score, and wait one second before showing the monkey again.

This whole code chunk goes inside the monkey's **forever** bracket – below the **next costume** block – so the program keeps checking whether the monkey has been hit.

06 Finishing touches

Let's add a simple timer to stop the sprites moving after 30 seconds, and make the monkey react when it sees incoming bananas. Add these two scripts to the monkey sprite. Now the game is complete, why not try experimenting with it? Can you make the monkey move erratically instead of disappearing when it's hit? Can you change the sprites' positions and directions to turn the game sideways, making it more like Space Invaders? What about adding more targets to hit? One of the best ways to learn to code is by experimenting with existing programs.

Code a quiz game with Python

Make your own text quiz game that mangles famous phrases using the Python language

Many people progress from Scratch to Python, a programming language that is powerful, easy to get started with, and much easier to read and write than other languages.

We're going to make a simple quiz question generator that strips the vowels and shuffles the spaces in a phrase. The player has to work out what that phrase is.

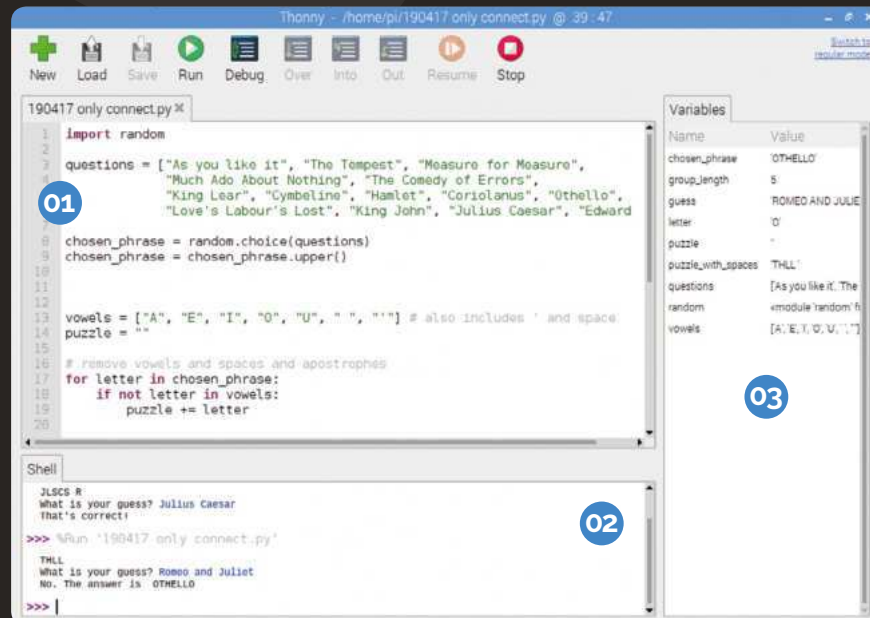
We'll be using Thonny, which provides a friendly single-screen environment for running and testing Python code. Like Scratch, the Thonny IDE (integrated development environment) comes pre-installed in the Raspbian with Desktop and Recommended Software operating system.

01 Create a list of questions

As well as variables, Python has lists, which can store multiple pieces of information. Our program creates a list called `questions`. Each item in the list is a piece of text, known as a string. In

Getting indentation right

Python uses indentation to show which instructions belong to a function, an if statement, or a repeating section. As you can see in Step 3 (overleaf), you can have multiple levels of indentation. The last line belongs to the if instruction, and that is repeated inside the for loop. The best way to get the indentation right is to remember the colon at the end of the previous line. Then, Thonny will add the indentation for you automatically. If you forget, use four spaces at the start of the line to insert the indentation. You'll still need to fix that missing colon, though!



Python, strings are surrounded by double quotes to show where they start and end. The whole list is enclosed in square brackets, and there are commas between the list items. Type in the code below, save your program, and then click Run. If it worked, you should see no error messages in the Shell window.

```
import random

questions = ["As You Like It",
             "The Tempest", "Measure for Measure",
             "Much Ado About Nothing",
             "The Comedy of Errors",
             "King Lear", "Cymbeline",
             "Hamlet", "Coriolanus", "Othello",
             "Love's Labour's Lost",
             "King John", "Julius Caesar",
             "Edward III"]
```

- 01 Type in and edit your program code here
- 02 Enter direct commands and see program input in the Shell here
- 03 Keep track of the data your program is processing here

You'll Need

- Raspbian with Desktop and Recommended Software
- Thonny

Top Tip

Pesky punctuation!

Take care to add the colons at the end of the `if` and `else` instructions. The code won't work without them.

02 Pick a random question

Python includes modules of prewritten code you can use, such as the `random` module we imported in Step 1. The first new instruction creates a new variable called `chosen_phrase` and puts a randomly chosen question into it. The second line converts the `chosen_phrase` to upper case. Run the program a few times and look at the value of `chosen_phrase` in the Variables pane. You should see different names come up, although names can also repeat.

Add a line of space between the code in Step 1 and add the following code:

```
chosen_phrase = random.choice(questions)
chosen_phrase = chosen_phrase.upper()
```

03 Strip the vowels and spaces

Let's create a new list of forbidden characters, chiefly the vowels, but also the space and the apostrophe. That last list item in the `vowels` list is an apostrophe inside double quotes. We create an empty string variable, called `puzzle`. We're going to go through each letter in the phrase, check whether it's in the `vowels` list, and if not, add it to the end of the `puzzle` string. The `for` instruction sets up a repeating piece of code, called a loop. The instructions that should be repeated are indented from left. Each time around the loop, the variable `letter` is set to contain the next character from the `chosen_phrase` string. The `if` instruction checks whether the letter is in the `vowels`

Where next?

You can find Python code to dissect in most issues of *The MagPi*. Issue 53 (magpi.cc/53) includes a more in-depth beginner's guide to Python, covering variables, looping with `while` and `for`, branching with `if`, and functions, which we'll cover in this issue shortly. Issue 54 (magpi.cc/54) introduces object-oriented programming in both Scratch and Python. Issue 73 (magpi.cc/73) includes a roundup of Python books and online resources. There is a book in our Essentials series too, called *Make Games with Python* (magpi.cc/gameswithpython).

list. If it is not, the `letter` is added to the end of `puzzle`. The `+=` means 'add at the end'. Run the program, then test it's working by looking at the contents of `puzzle` in the Variables panel. It should contain no vowels, spaces, or apostrophes.

Add the following code to the program:

```
vowels = ["A", "E", "I", "O", "U", " ", "'", '"']
puzzle = ""

for letter in chosen_phrase:
    if not letter in vowels:
        puzzle += letter
```

04 Insert random spaces

Each character in the string can be referred to by its position number, starting at 0. The number is called an index, and you put it in square brackets after the string. Try this in the Shell (click on the line starting with `>>>`). Instructions in the Shell are carried out immediately. Enter the following:

```
print("Hello"[1])
```

You get 'e' back (because the first character is number 0). You can get a chunk too (called a slice) by giving a start and end index, like this:

```
print("Hello"[1:4])
```

It gives you 'ell' because the last index position (4) is left out. We'll create a new list, called `puzzle_with_spaces`, by adding chunks of the `puzzle` string and a space until there's no `puzzle` string left. The `while` loop repeats the indented instructions below as long as the length of `puzzle` is more than 0. The `group_length` variable is given a random whole number (integer) from 1 to 5. Then that many letters are added to `puzzle_with_spaces` from the front of `puzzle`, plus a space. Those characters are

Debugging in Thonny

You can step through the program slowly to see what it's doing, which can help you to find errors. Click the Debug button in Thonny, then click the Over button to run through each instruction in turn. Watch the Variables pane on the right to see how the lists and strings change at each stage of the program. Thonny also helps you avoid errors by highlighting unclosed brackets and double quotes.

“ Python is **easy to get started with** and much easier to read and write than other languages ”

then cut off the front of puzzle. The slicing here only uses one number, so the other one is assumed to be the start or end of the string.

Add this code to your program:

```
puzzle_with_spaces = ""

while len(puzzle) > 0:
    group_length = random.randint(1,5)
    puzzle_with_spaces +=
puzzle[:group_length] + " "
    puzzle = puzzle[group_length:]
```

05 Add collision detection

It prints the puzzle_with_spaces. It then uses the input() function to ask you what your guess is. Your answer goes into the guess variable, and is then converted to upper case to make sure it matches the correct answer if it's right. The if instruction checks whether guess is the same as chosen_phrase. If so, it prints one message. Otherwise, the instruction indented under else runs, to tell you the right answer. In Python, one = is used to put a value into a variable, but two (==) are used to compare items in an if instruction.

Add this code to the end of the program:

```
print(puzzle_with_spaces)
guess = input("What is your guess? ")
guess = guess.upper()

if guess == chosen_phrase:
    print("That's correct!")
else:
    print("No. The answer is ", chosen_phrase)
```

Click the Run button and hopefully you'll see some letters in the Shell and 'What is your guess?' Enter an answer and you'll see 'That's correct!' or 'No. The answer is' and the correct response.

If you've typed the code out by hand, it's likely that you'll see an error message. Go through your code line-by-line and compare it to the full code in quiz_game.py.

quiz_game.py

DOWNLOAD
THE FULL CODE:

► Language: Python



magpi.cc/github82

```
001. import random
002.
003. questions = ["As You Like It", "The Tempest",
    "Measure for Measure", "Much Ado About Nothing",
    "The Comedy of Errors", "King Lear", "Cymbeline",
    "Hamlet", "Coriolanus", "Othello", "Love's Labour's Lost",
    "King John", "Julius Caesar", "Edward III"]
004.
005. chosen_phrase = random.choice(questions)
006. chosen_phrase = chosen_phrase.upper()
007.
008. vowels = ["A", "E", "I", "O", "U", " ", "'"]
009. puzzle = ""
010.
011.
012. for letter in chosen_phrase:
013.     if not letter in vowels:
014.         puzzle += letter
015.
016.
017. puzzle_with_spaces = ""
018.
019.
020. while len(puzzle) > 0:
021.     group_length = random.randint(1,5)
022.     puzzle_with_spaces += puzzle[:group_length] + " "
023.     puzzle = puzzle[group_length:]
024.
025.
026. print(puzzle_with_spaces)
027. guess = input("What is your guess? ")
028. guess = guess.upper()
029.
030.
031. if guess == chosen_phrase:
032.     print("That's correct!")
033. else:
034.     print("No. The answer is ", chosen_phrase)
```

Build an LED torch and electronic game

Discover how easy it can be to get lights blinking and buttons clicking using GPIO Zero and use your new-found skills to build an electronic game

One of the best things about the Raspberry Pi is that you can easily hook up your own electronics projects. Using some electronics components and the GPIO Zero library, you can program a puzzle game where you have to repeat a sequence of lights that gets longer each turn. You might remember a similar electronic game from your childhood, but we call ours Simple Brian. In issue 77 (magpi.cc/77) we showed you how to use Python code to play the game on screen. This issue, we'll show you how to make the electronic game itself, building on your new-found Python skills from Missing Vowels.

First, we're going to show you how to build a torch by lighting up LEDs. Let's get going.

01 Connect your first button

The torch circuit diagram (Figure 1) shows an LED light connected to the GPIO pins of a Raspberry Pi using a breadboard (see magpi.cc/breadboard for a primer on using this piece of equipment). Press the button into the board, then use jumper wires to form a circuit with your Pi, as shown by

the yellow wires in the diagram. The first button connects to the GPIO 2 pin on one side, and to the ground rail on the other side. We'll connect the latter to a ground pin on the Pi, so anything plugged in that row of holes connects to ground.

02 Connect your first LED

Always use a resistor when you connect an LED to your Pi, to prevent the LED drawing too much current and getting damaged. Both the LED and the resistor plug straight into your breadboard. The current flows from GPIO 18, through the resistor, through the LED (lighting it up), to the breadboard's ground rail. LEDs only work one way around: the short leg is the negative side, which you connect to ground. The LED won't light up yet.

03 Make an LED torch

You've made your first circuit! Let's test it by coding a torch. The `torch.py` code shows how to use an LED and a button. It imports the relevant parts of the GPIO Zero library, then sets up an LED called `light`, connected to GPIO pin 18. The button on pin 2 is set up with the name `button`. The `while True` loop checks whether the button is pressed forever. If so, the light is turned on. Otherwise, it's turned off. Pay attention to the capitalisation of LED and Button when setting them up.

04 Add the other buttons and LEDs

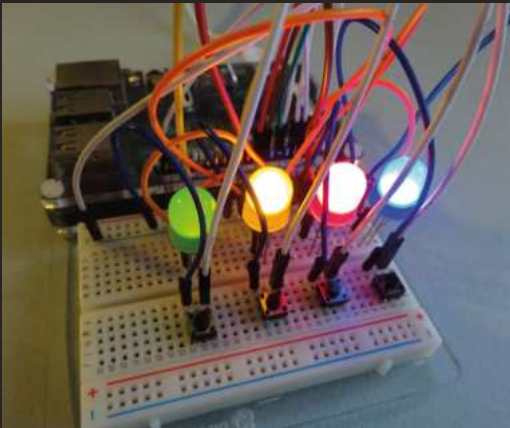
Take a look at Figure 2 (overleaf). It looks complex at first, but the other three buttons and LEDs are connected in the same way as

You'll Need

- ▶ 4 × LEDs (ideally different colours)
- ▶ 4 × 330 Ω resistors
- ▶ 4 × 6 mm Tactile momentary button switches
- ▶ 400-point breadboard
- ▶ 9 × Male-to-female jumper cables
- ▶ 8 × Male-to-male jumper cables
- ▶ PiBow case with Breadboard Base pimoroni.com

Where next?

The online documentation for GPIO Zero (magpi.cc/DPyuYc) provides more code examples, including a button-controlled camera, an LED bar graph, and a motion sensor. We surveyed useful resources for basic electronics in issue 77, and there's a book in our Essentials series called *Simple Electronics with GPIO Zero* (magpi.cc/gpio-zero).



▲ The finished game, with all the lights lit up for testing

the first ones, just using different pins on the Pi. All the buttons (yellow wires) connect to the Pi's inner row of pins, and the LEDs (blue wires) to the outermost row. We've separated the components in this diagram a bit so it's easier to see how to wire it up, but try to line up your LEDs and buttons on the breadboard so it's easier to play the game.

05 Test them all

Now you can test these LEDs and buttons too. Modify your torch code to use LED(23) and Button(3) and then run the program to test the next light switch works. Then check LED(24) and Button(4), and finally LED(25) and Button(17). The buttons should be next to the LED they illuminate.

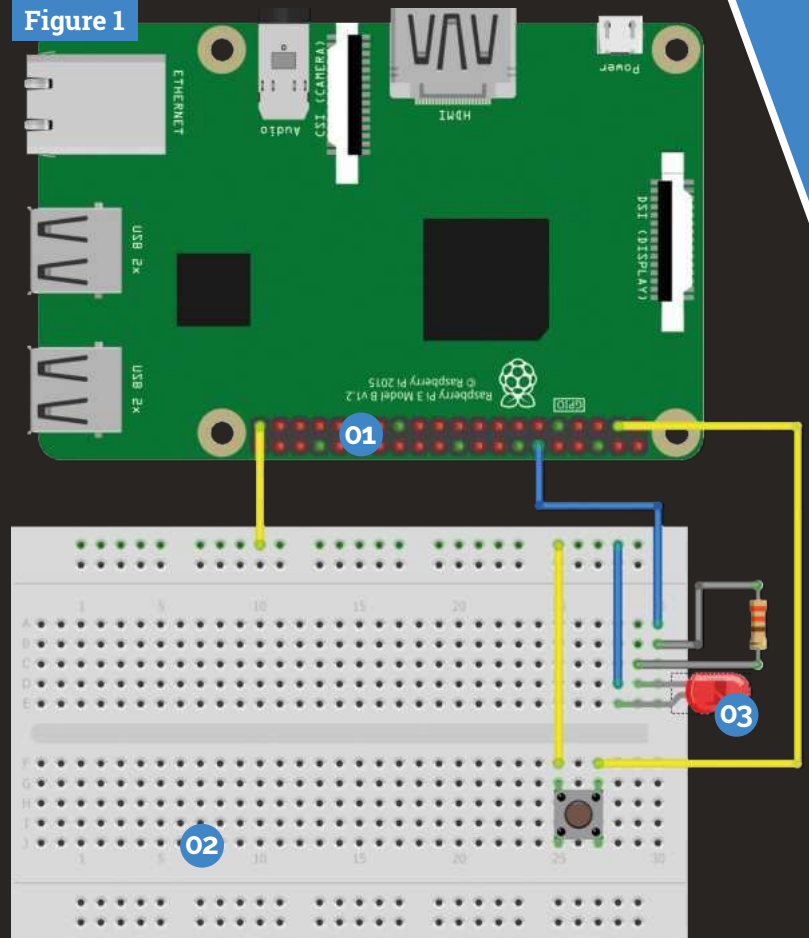
06 Build an electronic game

Now we're ready to start making the Simple Brian game (see `brian.py`). This starts by setting up a list of buttons, and a list of their associated LEDs. It also creates an empty list called `sequence`, which we'll use for the sequence of lights the player must repeat. With each turn, it'll get longer.

07 Add functions

Functions enable you to bundle up a set of instructions so you can reuse them. You have to define a function before you can use it. To define a function, you use `def`, followed by the function name, `()`, and a colon. The brackets are there to hold any info you're sending to the function, but we don't need to send any so they're empty. You can tell which instructions belong to a function, as they're indented. The `lights_on()` and `lights_off()` functions use a loop to go through all the items in

Figure 1



- 01 Count the pins to see where to connect your wires
- 02 The breadboard makes it easy to plug in components and quickly set up circuits
- 03 Using this simple circuit (with an LED and a button), you can make a push-button torch

torch.py

DOWNLOAD THE FULL CODE:

► Language: **Python**

magpi.cc/github82

```
001. # Torch demo
002. from gpiozero import Button, LED
003.
004. light = LED(18)
005. button = Button(2)
006.
007. while True:
008.     if button.is_pressed:
009.         light.on()
010.     else:
011.         light.off()
```

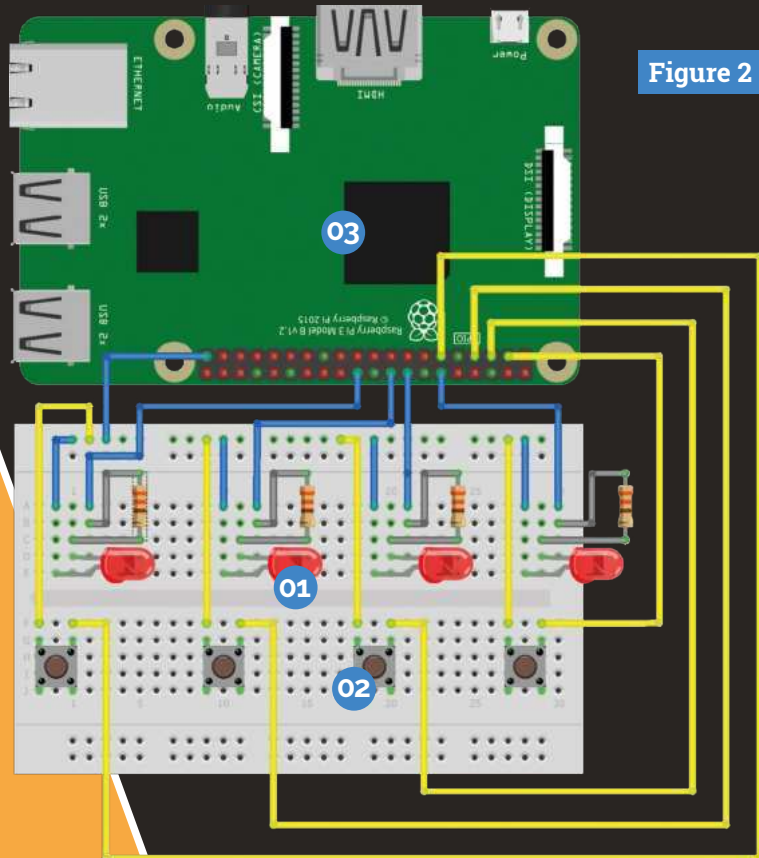
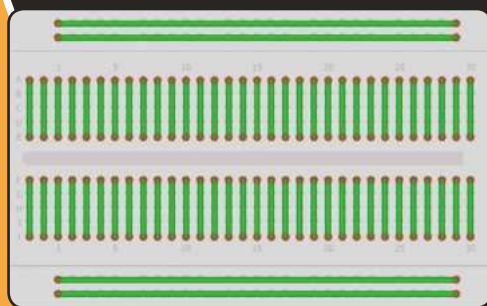


Figure 2

- 01 Just like its famous namesake, our Simple Brian game features four LED lights that flash up a sequence for you to memorise
- 02 The circuit also contains four buttons. We'll push these to repeat the sequence of lights that we've memorised
- 03 At the heart of our project is the Raspberry Pi. The buttons and lights are connected to our computer with wires. Code in the computer flashes the lights and keeps an eye on the buttons we're pushing



▲ On a breadboard, the rails along the edges are connected in a long line, and the short lines of dots in the middle are connected to each other

the `leds` list, putting them into `led`, then turning `led` on or off. The `flash_all()` function shows how to repeat a set number of times, in this case 3. The loop turns the lights on and off, with a 0.25 second pause after each change using `sleep(0.25)`.

08 Lights test flash all

After you've entered the functions (down to line 23), you can test the program by adding `flash_all()` as the last line and then running it. All the lights should flash together, three times. Delete that test line before you carry on. In line 25, the program runs the `lights_off()` function to ensure the lights are all off before the game begins.

09 Add to the sequence

Now we enter the main game loop, under `while True` (line 27). Everything from here on in is indented to show it belongs to that loop, repeating endlessly. The game sequence starts as an empty list, so the first thing we do is to add an LED. We pick a random LED using `random.choice()` and add it to the end of the sequence list using the `append()` list method. A list method is a built-in Python function that you can apply to a list. Other methods are available to insert and remove items, and sort the list, among other things.

10 Play the list sequence

The lights all flash three times using the `flash_all()` function before the sequence begins, to show this is the start of the sequence. Then a loop is used that takes each LED from the sequence list, and puts it into `light`, in turn. It's turned on, there's a short pause, then it's turned off. There's another short pause so it's obvious there are multiple flashes of the same light if it repeats in the sequence. In round one, there's only one light in the sequence list, but as the game progresses, this loop will get longer. You can run the program at this point to see the light sequence gradually extend, without the player getting a chance to guess.

11 Get the player's guess

Getting the player's guess uses a similar loop to the one that plays the lights sequence. It

brian.py

DOWNLOAD
THE FULL CODE:

► Language: Python

 magpi.cc/github82


```

001. from gpiozero import Button, LED
002. from time import sleep
003. import random
004.
005. buttons = [Button(2), Button(3), Button(4),
006.            Button(17)]
007.
008. leds = [LED(18), LED(23), LED(24), LED(25)]
009.
010. sequence = []
011.
012. def lights_on():
013.     for led in leds:
014.         led.on()
015.
016. def lights_off():
017.     for led in leds:
018.         led.off()
019.
020. def flash_all():
021.     for _ in range(3):
022.         lights_on()
023.         sleep(0.25)
024.         lights_off()
025.         sleep(0.25)
026.
027. lights_off()
028.
029. while True:
030.     # Add a new light to the end of the sequence
031.     new_light = random.choice(leds)
032.     sequence.append(new_light)
033.
034.
035.     # play the sequence
036.     for light in sequence:
037.         light.on()
038.         sleep(0.5)
039.         light.off()
040.         sleep(0.25)
041.
042.     # get the player's input
043.     for light in sequence:
044.         guess = None
045.         while guess == None:
046.             for button in buttons:
047.                 if button.is_pressed():
048.                     # convert button push to list
049.                     index number
050.                     guess = buttons.index(button)
051.
052.             if leds[guess] == light:
053.                 light.on()
054.                 sleep(0.5)
055.                 light.off()
056.                 sleep(0.25)
057.             else:
058.                 print("You failed at level ",
059.                       len(sequence))
060.                 for _ in range(10):
061.                     light.on()
062.                     sleep(0.15)
063.                     light.off()
064.                     sleep(0.15)
065.                 sequence = []
066.                 break

```

works its way through the sequence list, accepts a guess, and checks whether it matches the current item in the sequence. There are three loops inside each other here. The program sets the guess variable to None, a special value in Python. Then a while loop keeps repeating until the guess variable changes. Inside that, a loop goes through the buttons list, checking each one in turn to see whether it's pressed. If so, the guess variable is changed, ending the while loop. The program converts the button the player pressed into its index number in the list and puts that into the guess variable. That way, we can match the button to its LED, which will be at the same position in the leds list. You can find the position of an item in a list using `listname.index(item)`.

12 Check the player's guess

We're still inside the loop going through the sequence here, as the indentation of line 51 shows. Now we check whether the light the player guessed (`leds[guess]`) matches the current light in the sequence. If so, the light is turned on and then off again. If the two lights don't match, the player made a mistake. We can tell how long a list is using `len(listname)`. We use the length of the sequence list to tell the player which level they got to. The correct light is then flashed quickly ten times. The sequence list is emptied to start a new game, and the `break` instruction breaks out of the `for` loop that's getting player input. When the player has either guessed all the lights, or failed, the game repeats from line 28, adding a new light to the sequence. 

Top Tip



Underscoring repetition

The line `for _ in range(3)` repeats the indented instructions three times. The `_` shows we don't need to use the loop number. Often, you'd use a variable name instead.

The Pi Zero's size means it's a comfortable addition to a keyring

You can still access the microSD card so your Pi can be multi-purpose

Build a Pi keyring



PJ Evans

PJ is a writer, Jammer, and loves vintage computing. He is never more than five metres away from a Raspberry Pi.

@mrpjeans

Need a Raspberry Pi in your grab-bag? Here's how you can get a Pi up and running and networked just about anywhere

Although our beloved Pi is rightly praised for its diminutive nature, the need for a keyboard, mouse, and monitor can somewhat spoil the portability. If you've ever had a moment thinking, "If only I had a Raspberry Pi with me right now", this is the project for you. More than just a Pi-in-a-pocket, we're going to create a travel-ready Pi, free of peripherals but with full desktop support, that can join the local network either via its own WiFi hotspot, or a hidden feature of Raspbian: USB networking.

01 Prepare the Pi

Even though our portable Pi will not require a keyboard, monitor or mouse (well, sort of, you'll see), we want to get the most out of the device, so start by getting the full 'Raspbian Stretch with desktop and recommended software' image from the Raspberry Pi website (rpf.io/raspbian). Write

it to a microSD card using your favourite utility (we used balena Etcher – balena.io/etcher). However, just to show that it's possible to do all this without ever hooking the Pi up to a monitor, don't boot up just yet.

02 Set up for headless

Reinsert the microSD card into your computer. It will mount a drive called 'boot'. We're going to make some changes so the Pi starts with both WiFi and USB networking enabled. That's right, Raspbian has a feature that allows it to act like a network device to any computer connected by USB. Simply connect the Pi to your computer with a USB cable and a private network is set up between them, giving headless access without needing to get the Pi on the network. It's not enabled by default, so we'll make some changes to files in the **boot** directory to fix this.



▲ The pre-installed VNC server streams the Pi's desktop user interface to any other computer on the network

03 Enable WiFi and SSH

In the **boot** directory, create a file called **ssh**, with no extension. It doesn't need any content. On UNIX-style systems, you can enter:

```
touch ssh
```

Now, using a text editor, create a file in the same directory called **wpa_supplicant.conf** and enter the following (change country code if appropriate):

```
country=gb
update_config=1
ctrl_interface=/var/run/wpa_supplicant

network={
    scan_ssid=1
    ssid="Your network name"
    psk="Your network password"
}
```

Replace the values of **ssid** and **psk** to match your WiFi network name and password.

04 Size matters

When a Pi boots without a monitor attached (headless), the desktop still starts, but at a very small screen resolution. We're going to be using VNC to remotely access the desktop, so we need to make it a bit more usable. Using a plain-text editor, open the file **config.txt** in the **boot** directory. Be very careful making changes here as it can render your Pi unbootable. Scroll down the file until you see two lines as follows:

```
#framebuffer_width=1280
#framebuffer_height=720
```

“ We're going to create a travel-ready Pi, free of peripherals but with full desktop support ”

Note the '#' before them; this marks them as comments and so they are ignored. Remove the '#' so it looks like this:

```
framebuffer_width=1280
framebuffer_height=720
```

05 Enable USB networking

Scroll down to the bottom of the **config.txt** file and add the following line:

```
dtoverlay=dwc2
```

This instructs Raspbian to apply the USB networking module. Save and close **config.txt**. To make sure the module is made available, edit the file **cmdline.txt**, also in the **boot** directory. The contents of the file are on a single line. Find the end of the line and, making sure you do not add a new line in the process, add a single space followed by:

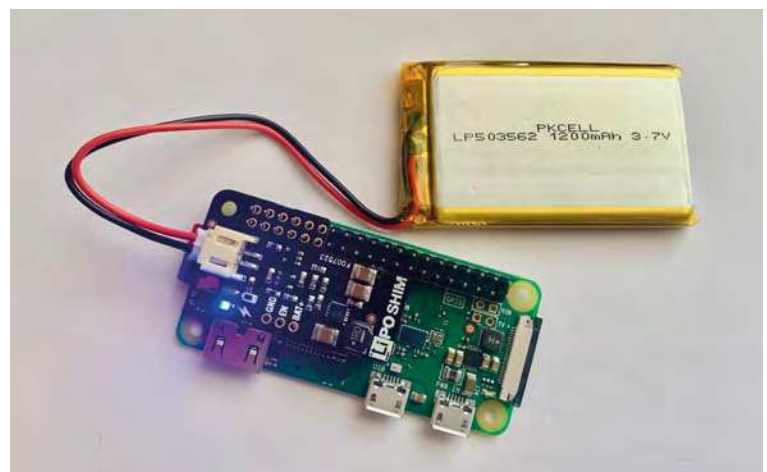
```
modules-load=dwc2,g_ether
```

Now save and close **cmdline.txt**, then eject the microSD card.

You'll Need

- ▶ micro USB to USB cable
- ▶ LiPo SHIM (optional) magpi.cc/zaMZab
- ▶ LiPo battery (optional) magpi.cc/F0uZmb
- ▶ Keyring case magpi.cc/hqjqdE

▼ Pimoroni's LiPo SHIM allows safe use of a LiPo battery without blocking GPIO ports



Top Tip

Easy WiFi

USB networking gives you a reliable method of setting up the Pi's WiFi connection if you want to give others access.

06 Connect via USB

It's time to wake up our Pi. Insert the microSD card into the Pi and connect a micro USB cable to the peripheral USB port on the Pi Zero, not the usual power USB port. If you're unsure, the peripheral port is the one nearest the mini-HDMI connector. We need full USB access for networking, and your computer will also supply enough power to keep the Zero happy. Connect the cable to your computer and wait a few minutes so Raspbian has time to resize the file system and set things up.

07 Log in

From the command line (users of Windows 8 or before may need to install PuTTY), run these commands:

```
ssh-keygen -R raspberrypi.local
ssh pi@raspberrypi.local
```

The first command cleans out any previous SSH keys you may have for **raspberrypi.local**. You do not need to run it again. Hopefully you'll now be

prompted for a password. Enter **raspberry** and you're in. If you find you can't connect to the Pi, re-examine the changes you've made to the boot directory or try booting with a monitor and keyboard. You may connect through WiFi rather than USB. Run **ifconfig** at the Pi's command line to see the different networks.

08 Configuration

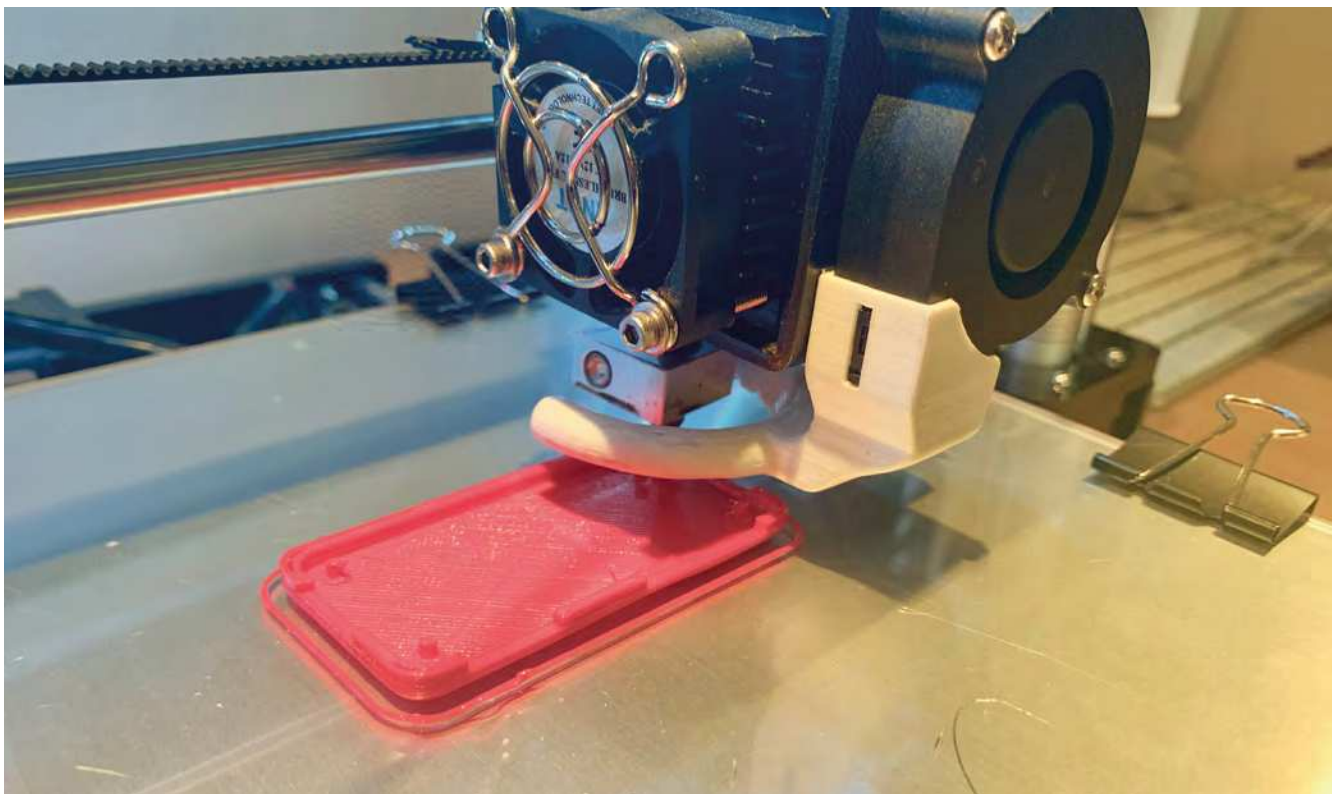
We've established we have headless access to the Pi Zero. While you are logged in, start with some housekeeping. Run this command:

```
sudo raspi-config
```

Firstly, change your password from **raspberry** to something harder to guess. Now go to 'Network Options' then 'Hostname'. Here you can rename your Pi to something else if you wish, reducing the risk of it clashing with another Pi on the network.

Finally, go to 'Interfacing Options' and enable the VNC server. This will allow you to stream the desktop to another device. Reboot the Pi now so the host name change takes effect.

▼ Using a 3D printer, you can customise and create your own case





▲ If you need GPIO access, the official case is perfect with its selection of covers. Make two small holes for a thread loop

09 Desktop access

To stream the Pi desktop, you'll need a client for your connected computer. A popular choice is VNC Viewer, available for a wide range of platforms from magpi.cc/FuGnye. Once installed, enter your Pi's new host name followed by '.local' as the server and press **ENTER**. In a few seconds, the desktop will appear on your screen.

You should see a 'Welcome to Raspberry Pi' dialog box. Now's the time to go through these menus and also update the current software when prompted. The Pi will need a working WiFi connection to do this.

10 Hotspot access

If you want a truly standalone experience, and internet connectivity isn't a concern, another connection option is to use the Zero's WiFi capability and create a hotspot. Then, no matter where you are, so long as you can get power to the Pi, any computer will be able to connect and SSH or VNC in.

Setting up a hotspot doesn't take long, but does involve several steps, including setting up a DHCP server and hostapd. Take a look at magpi.cc/BRdGKK for a comprehensive guide.



```
GNU nano 2.0.6      File: config.txt      Modified
#disable_overscan=1
# uncomment the following to adjust overscan. Use positive numbers if console
# goes off screen, and negative if there is too much border
#overscan_left=16
#overscan_right=16
#overscan_top=16
#overscan_bottom=16

# uncomment to force a console size. By default it will be display's size minus
# overscan.
framebuffer_width=1920
framebuffer_height=1280

# uncomment if hdmi display is not detected and composite is being output
#hdmi_force_hotplug=1

# uncomment to force a specific HDMI mode (this will force VGA)
#hdmi_group=1

Get Help  WriteOut  Read File  Prev Page  Cut Text  Cur Pos
Exit      Justify    Where Is  Next Page  UnCut Text To Spell
```

“ To stream the Pi desktop, you'll need a client for your connected computer. A popular choice is VNC Viewer ”

11 Batteries are included?

If you are taking the hotspot route, why not go the whole nine yards and add battery power? A Zero will run quite happily for hours from widely available USB power banks. If you're feeling fancy, why not add in a LiPo battery so you have a single self-contained unit? LiPo batteries come with safety concerns (see 'LiPo safety' box), so we need some circuitry to manage the battery and make sure it supplies the correct voltage. Pimoroni's LiPo SHIM does just this with the added bonus of not using up any of your valuable GPIO ports. Carefully solder it to the base of the GPIO pins and you've got power. Don't forget to add a switch.


▲ Always be careful editing the files in the boot folder. Take backups and use a text-only editor like nano

Top Tip

Add a NAT gateway

You can configure the connected computer to route internet traffic to and from the Pi using network address translation (NAT) routing software.

12 Print a case

Obviously, we can't leave our little Zero unprotected, so our final step is to provide a nice case. If you're wanting to use GPIO, the official case can be adapted to accept a keyring with a small drill bit and a steady hand. If you've added battery power, it may be time to get a CAD package out and design something that fits everything into one. We opted for a 3D-printed case (magpi.cc/hjqdE) from Thingiverse maker Haunt Freaks, which features a ready-to-go hoop to accept a keyring or lanyard clip. 

◀ A Pi Zero without headers means an extra slimline case with no pointy bits



**Rob
Zwetsloot**

Rob is amazing. He's also the Features Editor of *The MagPi*, a hobbyist maker, cosplayer, comic book writer, and extremely modest.

magpi.cc

MAKER

NeoPixel display cabinet lights

Light up a display cabinet with some NeoPixels, a Raspberry Pi, and a little Python code

We've covered NeoPixels in *The MagPi* before, with some wonderful cosplay lights (magpi.cc/45) and Christmas tree lights (magpi.cc/52), which we've used ourselves. It's been a long time since we've controlled any NeoPixels with a Raspberry Pi, though – so long that there's actually a newer and much easier method for doing it.

We thought it was high time to try this out, and get a display cabinet lit up fancily in the process.

You'll Need

- ▶ NeoPixel lights
- ▶ Circuit wire
- ▶ Push-button
- ▶ 470 Ω resistor
- ▶ 5V power source
- ▶ Soldering iron

01 Choose your NeoPixels

There are many configurations and types of NeoPixels that you can buy. For our display cabinet, we chose two quarter-circle stripes of lights that hold 15 LEDs a piece. This allowed us

to create a little semicircle in our cabinet for a bit more interesting coverage.

If you have a big cabinet and wish to line the entire thing, you can always get a long flexible strip of NeoPixels. You can even get single lights, or smaller circles of NeoPixels.

The only thing you need to make sure you do for all the NeoPixel types is correctly count the number of LEDs in your system. We'll tell you why in a few steps' time.

02 Choose a location

What do you want to light up? For our project, a shelf of figurines was all we wanted to illuminate, and so we decided to add the lights above the shelf – attached to the 'ceiling', as it were.

You'll need to take into consideration light coverage and camouflage in your display cabinet. Think about sight lines if you want to keep them hidden from specific angles, and look to see if your display cabinet has anything to aid in adding lights – the IKEA Detolf has a little plug on the top for wires, for example.

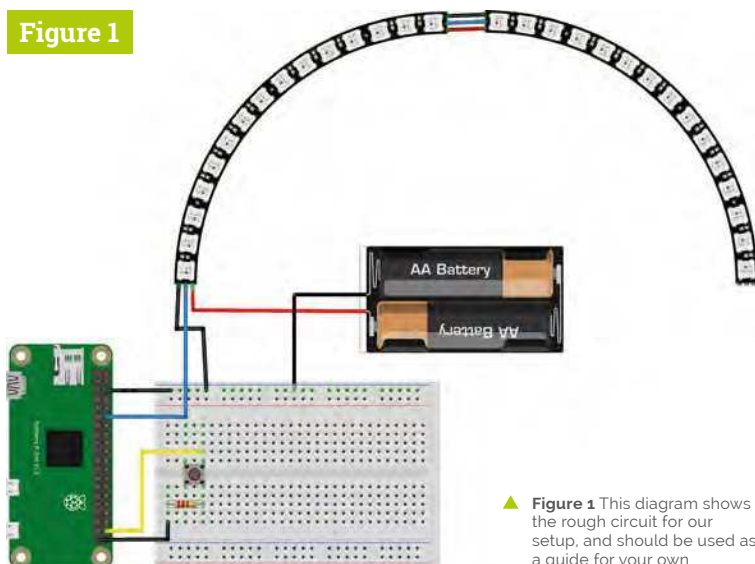
There also needs to be access for the Pi to control the lights, so keep that in mind.

03 Assemble your circuit

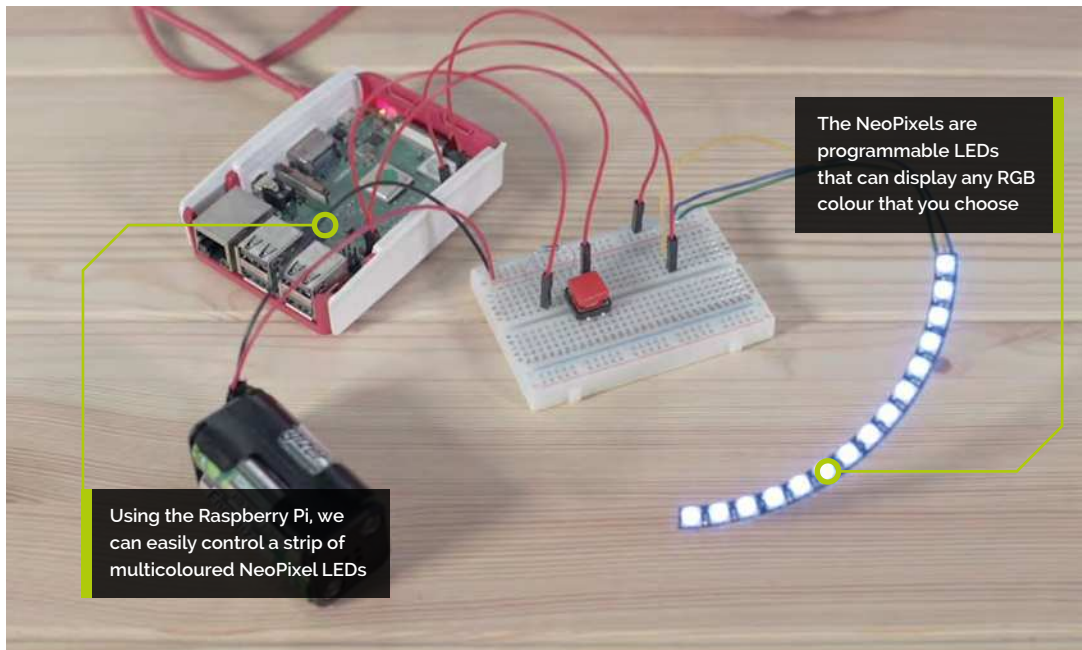
We've put together a handy circuit diagram ([Figure 1](#)) for you to follow along to. There are a few important things to note, though, to make sure you understand it.

The NeoPixel strips have three pads on them: one for 5V power, one for ground, and a 'data' port. The data needs to be connected to the GPIO

Figure 1



▲ **Figure 1** This diagram shows the rough circuit for our setup, and should be used as a guide for your own



Top Tip

RGB colour

Red, green, and blue lights make up a single NeoPixel. Using a value of each colour between 0 and 255, you can create an entire rainbow of colours.

pin we're sending the signals from. You also need to make sure to connect it to the Din (data in) pad. If you're chaining together strips like we've done, make sure you connect the Dout (data out) from the first strip to the Din of the next strip. It's also good practice for ground on the NeoPixels to go to the Raspberry Pi's ground, as well as the ground of the power source.

We've placed the button way down the GPIO pins to keep it clear. You need the resistor in the little button circuit to make sure it works properly and can be sensed by the Raspberry Pi.

04 Soldering the NeoPixels

This bit can be tricky, but you will have to solder some wires to your NeoPixel strips. Make sure your soldering iron is properly prepared if this is your first time using it (check this video: magpi.cc/GCUNyL). Also, if this is your first time soldering, check out the Raspberry Pi Foundation's video on it: magpi.cc/Ahvxdk.

We recommend putting a little solder on your wire (this is called tinning it), as well as a dab on the pads. That way, you just need to heat up the solder already on the wire and pad so that they fuse and connect.

05 Powering the system

You need to power two parts of your light display: the Raspberry Pi and the NeoPixels. The

“ You need to power two parts of your light display: the Raspberry Pi and the NeoPixels ”

NeoPixels require a 5V input, and also draw a lot of power at the same time; so, to be safe, you really shouldn't power more than two or three NeoPixels with the 5V pin of the Raspberry Pi. In our circuit, we've used four 1.5V AA rechargeable batteries, as they're technically closer to about 1.2V each – that means we get roughly 5V out of them.

A more convenient way is to get a 5V power supply and use a power terminal with two screw ports to attach the positive and negative ends. Please make sure the power supply is unplugged while you do this, and be very careful when plugging it in.

You can technically run a Pi using the GPIO pins; however, we have elected to run a micro USB cable up to our Pi.

06 Basic code

We've written the code, `rollcall.py`, to work with our specific system. Download it from magpi.cc/DisplayLights and we'll run through it.

First, we import the generic stuff: `time`, `gpiozero`, as well as the `neopixel` library and its related `board` library. We've also used the `numpy` library so that we can create RGB values for the LEDs, which enables them to fade between colours.



► We used Blu Tack to attach the lights to the top of our cabinet

Our system uses 30 LEDs, and we've connected it to GPIO 18 on the board. We've then defined six colours that represent the figurines in the display for our added colour cycle effect. After that, we tell the code about the strip of NeoPixels.

We then define how to calculate the values for colour fading, and the colours to transition between, before turning the strip of LEDs to pure white. Finally, we set the main `rollcall_cycle` function to loop, so we can press the button whenever we want.

Top Tip

Roll call

The six colours chosen represent the main team of figurines in the cabinet – it's a tradition in Japanese superhero TV shows to run down your name and colour in a specific order.

07 Alter your code

The main parts of the code to pay attention to are the `LED_COUNT`, `LED_PIN`, and `button` values. Your number of LEDs will likely differ from ours, and you may have connected the strip and button to different GPIO pins.

As for the colour cycle, play around with that to your heart's content, or remove it completely! You can even change the main colour value for the standard lights, using RGB values from 0 to 255.

08 Testing your lights

Before attaching everything to your cabinet, we highly recommend testing your LEDs. Run the code from within your preferred Python IDE, and make sure that not only are you getting the correct

colours (you may have a GRB set of NeoPixels instead of RGB, for example), but that also the button works.

09 Final Pi prep

The Raspberry Pi in our setup is going to be headless, which means we want the Python code to load up after the Pi turns on. Our preferred method of doing this is adding a line to `/etc/profile` – it makes it a lot easier. Open up a Terminal window and type:

```
sudo nano /etc/profile
```

Use the arrow key to go to the bottom, then add:

```
sudo python rollcall.py
```

If you've saved the Python script to a specific folder other than the home directory, make sure to include the path to it as well. Save and exit the file. You can also turn off 'boot to desktop' in the Raspberry Pi configuration settings, so that the entire system loads up faster.

10 Mounting the lights

Depending on your NeoPixels, you can attach them in several different ways. We've used Blu Tack to stick ours to the ceiling of our display

rollcall.py

> Language: Python

DOWNLOAD
THE FULL CODE: magpi.cc/DisplayLights

```

001. #!/usr/bin/env python
002.
003. import time
004.
005. from gpiozero import Button
006.
007. import board
008. import neopixel
009. import numpy as np
010.
011. button = Button(21)
012.
013. # LED strip configuration:
014. LED_COUNT = 30 # Number of LED pixels.
015. LED_PIN = board.D18 # GPIO pin
016. LED_BRIGHTNESS = 0.2 # LED brightness
017. LED_ORDER = neopixel.GRB # order of LED colours.
    May also be RGB, GRBW, or RGBW
018.
019. # The colour selection selected for this
    project: red, blue, yellow, green, pink, and
    silver respectively
020.
021. gokai_colours = [(255,0,0),(0,0,255),(255,255,0)
    ,(0,255,0),(255,105,180),(192,192,192)]
022.
023. # Create NeoPixel object with appropriate
    configuration.
024. strip = neopixel.NeoPixel(LED_PIN, LED_COUNT,
    brightness = LED_BRIGHTNESS, auto_write=False,
    pixel_order = LED_ORDER)
025.
026. # Create a way to fade/transition between
    colours using numpy arrays
027.
028. def fade(colour1, colour2, percent):
029.     colour1 = np.array(colour1)
030.     colour2 = np.array(colour2)
031.     vector = colour2-colour1
032.     newcolour = (int((colour1 + vector*percent)
    [0]), int((colour1 + vector * percent)[1]),
    int((colour1 + vector * percent)[2]))
033.     return newcolour
034.
035. # Create a function that will cycle through the
    colours selected above
036.
037. def rollcall_cycle(wait):
038.     for j in range(len(gokai_colours)):
039.         for i in range(10):
040.             color1 = gokai_colours[j]
041.             if j == 5:
042.                 color2 = (255,255,255)
043.             else:
044.                 color2 = gokai_colours[(j+1)]
045.             percent = i*0.1 # 0.1*100 so 10%
    increments between colours
046.             strip.fill((fade(colour1,colour2,
    percent)))
047.             strip.show()
048.             time.sleep(wait)
049.
050. strip.fill((255,255,255))
051. strip.show()
052.
053. # Main function loop
054.
055. while True:
056.
057.     time.sleep(1)
058.
059.     button.wait_for_press()
060.     rollcall_cycle(0.2) # 0.2 seconds between
    colour updates

```

case; however, you could also use glue. For the long strips, you can always nail them in with staples so that you don't actually have to go through the strip.


Just make sure any exposed PCB is not touching anything conductive.

11 Permanent circuit tips

You don't really want exposed wires everywhere just to light up your display case. Creating a 3D case to house the Pi is a good first step, and using heat-shrink tubing to encase all the

cables makes the whole thing a lot neater. You can also cover any soldered joints with a bit of hot glue.

12 Have fun and experiment!

This basic setup can very easily be expanded upon. As the Pi is internet-connected, you'll be able to use some IoT stuff like Twitter triggers or noise activation or temperature-dependent colouration. You can even add more strips to the Pi so that you can have several layers of lighting effects. We hope this really improves your display cabinets. 

Hack Lego Boost with Raspberry Pi



Mike Cook

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

magpi.cc/TPaUfT

Make a MIDI sequence generator that creates sequences from our small mobile rover: Si-clops

Last month we looked at three example projects where we explored the finer points of control that are normally hidden from you by the standard Lego software. In this final part (at least for now), we will look at making a MIDI sequence generator that creates sequences by letting a small mobile rover loose over any flat surface, as we introduce you to Si-clops.

Figure 1 shows Simon; he's always asking people about colours. However, this month the one-eyed Simon rides his colour-sensing Si-clops to generate music sequences from what it sees. This can be by roaming over existing artwork or scanning a set of coloured tiles placed in a tile holder. The sound is generated from a MIDI sound module attached to the Raspberry Pi with a USB MIDI lead. Once playing, the sequencer turns each

colour into a note of settable pitch, velocity, and instrument at a rate determined by the tempo parameter. The sound from each colour can also be individually muted.

01 The theory

The Boost system can recognise six different colours and if we have a sequence of eight steps in length, we get an enormous number of possible sequences. In fact, the number of possible sequences is six (the number of different colours we have in each place) raised to the power of eight (the number of steps that make up our sequence). Which, in terms of hard numbers, is 6^8 , giving a total of 1,679,616 different possible combinations. As we

You'll Need

- ▶ Lego 17101 Boost Creative Toolbox magpi.cc/JtUiDe
- ▶ USB MIDI lead magpi.cc/fZiGvF
- ▶ MIDI sound generator
- ▶ Optional cardboard and paint



Figure 1

▲ Figure 1 Meet Simon, or Si for short

Si-clops.py

> Language: Python 3

DOWNLOAD
THE FULL CODE:



magpi.cc/dhaAam

```

001. #!/usr/bin/env python3
002. # coding=utf-8
003. # Siclops Colour Sequencer
004. # By Mike Cook April 2019
005.
006. import time, pygame, os
007. from pygubst import *
008. from pygubst.movehub import MoveHub, COLORS
009. from pygubst.peripherals import EncodedMotor
010. import rtmidi
011.
012. midiout = rtmidi.MidiOut()
013. pygame.init()
014. pygame.display.set_caption(
    "Si-clops -> Colour Sequencer")
015. os.environ['SDL_VIDEO_WINDOW_POS'] = 'center'
016. pygame.event.set_allowed(None)
017. pygame.event.set_allowed([pygame.KEYDOWN, pygame.
    QUIT, pygame.MOUSEBUTTONDOWN, pygame.MOUSEBUTTONUP])
018. screenWidth = 430 ; screenHeight = 400
019. cp = screenWidth // 2
020. screen = pygame.display.set_mode([screenWidth,
    screenHeight],0,32)
021. textHeight=18 ; font = pygame.font.Font(None,
    textHeight)
022.
023. samples = 8 ; step = 0 ; midiStep = 0 ; newScan =
    False ; motorUpdate = True
024. backCol = (160,160,160) ; lineCol = (128,128,0) ;
    hiCol = (0,255,255)
025. tileColours = [ (0,0,0), backCol,(0,0,0),
    (0,0,255), (128,128,255), (0,255,255),
026. (0,255,0), (255,255,0),
    (255,128,0), (255,0,0), (255,255,255)]
027.
028. scannedBlock = [3,10,9,0,7,6,9,10]
029. colourToTrack = [5,-1,-1,0,-1,-1,1,2,-1,3,4]
030. chan = [1]*6 ; velocity = [64]*6 ; mute =
    [False]*6
031. note = [56+i for i in range(0,6)]
032. lastNote = [0] * samples ; lastChan =
    [-1] * samples
033. shutDown = False ; update = False
034. nextStep = time.time() ; tempo = 200 # BPM
035. running = False # MIDI sequencer
036. lastColour = 255 ; updateColour = 0
037. tileDistance = 0.0 ; currentMotor = 0
038.
039. noteNames = ["C", "C#", "D", "D#", "E", "F", "F#", "G",
    "G#", "A", "A#", "B"]
040. instNames = ["Piano", "Harpichord", "Glockenspiel",
    "Marimba", "Bells", "Organ", "Guitar",
041. "Bass", "Timpani", "Percussion", "Alto Sax",
    "Clarinet", "Rain", "Goblins",
042. "Banjo", "Tinkle Bell"]
043. insNums = [0,6,9,12,14,19,27,32,47,52,65,71,96,
    101,105,112] # to match above
044.
045. def main():
046.     global update,movehub, shutDown, step,
    currentMotor,conn
047.     print("Si-clops - Colour sensor sequencer")
048.     print("press the green button now")
049.     conn=get_connection_auto()
050.     print("Looking for Hub")
051.     movehub = MoveHub(conn)
052.     print("Hub connected")
053.     init()
054.     initMIDI()
055.     print(
    "Initialised - Press green button to scan")
056.     drawScreen()
057.     while not shutDown:
058.         checkForEvent()
059.         if newScan:
060.             sensorScan()
061.             if time.time() > nextStep:
062.                 doMIDIstep()
063.
064.     def doMIDIstep():
065.         global nextStep, midiStep, lastNote, lastChan
066.         if not running:
067.             time.sleep(0.1) # yield for other programs
068.             return
069.         tileNumber = colourToTrack[
    scannedBlock[midiStep]]
070.         #turn off the note you are about to play
071.         if lastChan[midiStep] != -1:
072.             midiout.send_message([0x90 | lastChan[
    midiStep],lastNote[midiStep],0])
073.
074.         if not mute[tileNumber]:
075.             midiout.send_message([0x90 +
    chan[tileNumber]-1,note[tileNumber],velocity[
    tileNumber]])
076.             lastNote[midiStep]= note[tileNumber]
077.             lastChan[midiStep]= chan[tileNumber]-1
078.             nextStep = time.time() + stepTime
079.             updateSamples(midiStep)
080.             midiStep += 1
081.             if midiStep >= samples:
082.                 midiStep =0
083.
084.     def alloff():
085.         for i in range(0,16):
086.             midiout.send_message([0xB0 | i,120,0])
087.         # normally only one is needed
088.         midiout.send_message([0xB0 | i,123,0])
089.     def sensorScan():
090.         global step,currentMotor, newScan
091.         alloff() # turn off any sound

```



Figure 2

▲ Figure 2 How to tidy up the colour sensor's wire

found last month, the optimum distance for working with the colour sensor is 10 mm, so the Si-clops is designed to get as close as possible to this distance.

02 The practice

The construction of the Si-clops is simple enough, and could be guessed from the pictures we show here. However, a step-by-step guide to building Si-clops is on the GitHub page (magpi.cc/dhaAam). All the parts are from the Boost set, so you'll have them to hand. However, the Studio modelling program does not handle flexible parts, so **Figure 2** shows you how to wrap the cable from the colour sensor to Port C of the Boost Hub. It's best to plug the colour sensor's wire into the Hub and then temporarily remove the Lego brackets and push the cable behind and then replace them.

03 What the code needs to do

We need the Python controlling code to instruct the Boost's Hub to move the two built-in motors through eight short steps, stopping at the end of each step to send back a colour sample. This sampling can be controlled through an on-screen button or by pushing the Hub's green button. Then this sequence is displayed and a click of the Run button starts things going. The exact note pitches, velocities, and channel/instrument are controlled by mouse clicks, as are the tempo and the channel mute. The code is written under the Pygame framework and is shown in the **Si-clops.py** listing.

04 Icons

The program needs a directory called **icons**; in it, there needs to be two 15×15 pixel images on a

transparent background, called **0.png** and **1.png**. The **0.png** image is a green plus sign, and the **1.png** image is a red minus sign; both can be made with any paint package, such as GIMP. **Figure 3** shows the layout of the screen; the whole user interface is based on Pygame's rectangle data structure, **rect**. Rectangles are defined for each user control, and the mouse down and up events change the appropriate numbers. This results in easy-to-modify, efficient code.

05 Code in depth

The **insNums** list holds the instrument number associated with each MIDI channel, with the **instNames** list showing the names. You can change these if you would like to use a different set of instruments. Each instrument potentially sounds for the full length of the sequence, the note-off message only being sent just before the note-on message. If a colour appears more than once in the sequence, then the note-off message is sent whenever that colour is encountered. A brown square in the middle of a sample colour indicates that it is the one currently being played.

06 Customising the code

All GUI colours are defined by variables, so they are easy to change. The motor moves by the number of degrees defined by the **inc** variable in the **incMotor** function for each step. The original code is set so that Si-clops moves 20 mm per step; you can change this to whatever you want if you

Top Tip



Aligning Si-clops

We drew a white square round the end of the sample tile holder to allow us to set Si-clops in exactly the right place. If you shade external light with your hand, you can see the illuminated area the colour sensor uses to detect the colour.

▶ Figure 3 The sequencer control panel

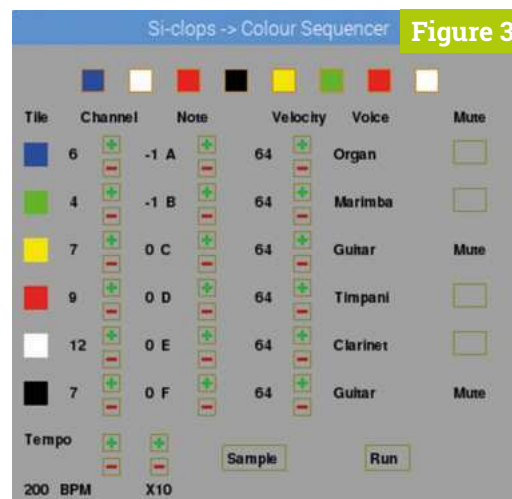


Figure 3

Si-clops.py (continued)

```

092.     for i in range(0,samples):
093.         scannedBlock[i] = 1 # background colour
094.         incMotor() # to move before first sample
095.         step = 0
096.         while step < samples:
097.             checkForEvent()
098.             getCol()
099.             movehub.motor_AB.angled(-currentMotor, .4) #
back to start
100.         newScan = False
101.         updateValues()
102.         step = 0 ; currentMotor = 0
103.
104. def getCol():
105.     global lastColour, updateColour, motorUpdate,
scannedBlock, shutDown
106.     if updateColour >= 2 and motorUpdate:
107.         # put in the index of the colour
108.         scannedBlock[step] = correctColour(lastColour)
109.         updateSamples(-1)
110.         updateColour = 0
111.         motorUpdate = True
112.         incMotor()
113.     else:
114.         updateColour = 0
115.
116. def initMIDI():
117.     global midiout
118.     available_ports = midiout.get_ports()
119.     print("MIDI ports available:-")
120.     for i in range(0,len(available_ports)):
121.         print(i,available_ports[i])
122.     if available_ports:
123.         try:
124.             midiout.open_port(1)
125.         except:
126.             print(
"No MIDI port found opening virtual port")
127.             midiout.open_port(0)
128.     for ch in range(0,16): # set up channels
129.         if ch != 9:
130.             midiout.send_message([0xC0|ch, insNums[ch]])
# set instrument
131.     midiout.send_message([0xB0 | ch,0x07,127])
# set volume
132.     setTime()
133.
134. def setTime(): # calculates the step time from the
tempo (BPM)
135.     global stepTime, tempo
136.     stepTime = 1/(tempo / 60)
137.     updateValues()
138.
139. def init():
140.     global motor, markerRect, shutDown, driveRect,
incRect, decRect, icon, decRect, muteRect,
voiceRect, sampleRect, rsRect
141.     motor = None
142.     movehub.button.subscribe(call_button)
143.     movehub.color_distance_sensor.subscribe(
callback_colour, granularity=0)
144.     icon = [pygame.image.load(
"icons/"+str(i)+".png").convert_alpha()
145.             for i in range(0,2) ]
146.     incRect = [pygame.Rect((0,0),(15,15))] *20
147.     decRect = [pygame.Rect((0,0),(15,15))] *20
148.     for j in range(0,3):
149.         for i in range(0,6):
150.             incRect[i+j*6] = pygame.Rect((
76 + j*80,76+i*40),(15,15))
151.             decRect[i+j*6] = pygame.Rect((
76 + j*80,96+i*40),(15,15))
152.             incRect[18] = pygame.Rect((76,326),(15,15))
153.             decRect[18] = pygame.Rect((76,346),(15,15))
154.             incRect[19] = pygame.Rect((116,326),(15,15))
155.             decRect[19] = pygame.Rect((116,346),(15,15))
156.             driveRect = pygame.Rect((176,335),(54,22))
157.             rsRect = pygame.Rect((296,335),(38,22))
158.             muteRect = [pygame.Rect((0,0),(15,15))] *6
159.             voiceRect = [pygame.Rect((0,0),(15,15))] *6
160.             for i in range(0,6):
161.                 muteRect[i] = pygame.Rect((370,80+i*40),
(28,20))
162.                 voiceRect[i] =pygame.Rect((268,85+i*40),
(100,20))
163.             sampleRect = [pygame.Rect((0,0),(15,15))] *8
164.             for i in range(0,samples):
165.                 sampleRect[i] = pygame.Rect((58+i*40,20),
(20,20))
166.             markerRect = [pygame.Rect((0,0),(15,15))] *8
167.             for i in range(0,samples):
168.                 markerRect[i] = pygame.Rect((63+i*40,25),
(10,10))
169.             setTime() # calculate MIDI time interval
170.
171. def drawScreen():
172.     screen.fill(backCol)
173.     for i in range(0,20): # increment / decrement
icons
174.         screen.blit(icon[0],(
incRect[i].left,incRect[i].top))
175.         pygame.draw.rect(screen,lineCol,incRect[i],1)
176.         screen.blit(icon[1],(
decRect[i].left,decRect[i].top))
177.         pygame.draw.rect(screen,lineCol,decRect[i],1)
178.
179.     # draw all tiles
180.     playingToLego = [3,6,7,9,10,0]
181.     for i in range(0,6):
182.         pygame.draw.rect(screen,
tileColours[playingToLego[i]], (10,82+40*i,20,20),0)
183.         drawWords("Tile",10,54,(0,0,0), backCol)
184.         drawWords("Channel",58,54,(0,0,0), backCol)
185.         drawWords("Note",138,54,(0,0,0), backCol)

```

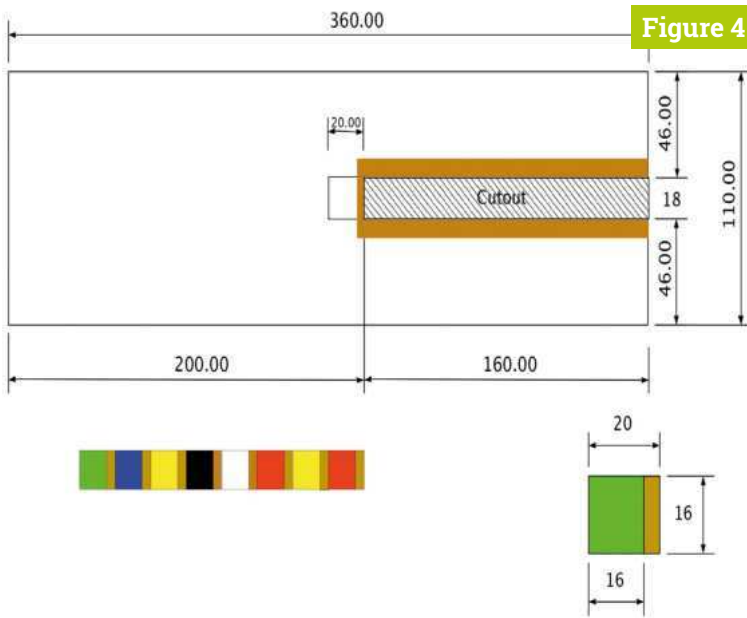


Figure 4

▲ Figure 4 Sample holder and tiles – dimensions in mm

want to roam over artwork. Si-clops also returns to its starting position after the sample scan, but you could change that also. By moving the two Hub motors by different angles each step, you can get Si-clops to move in a curve as opposed to a straight line.

07 Tile sample holder

While running Si-clops over artwork can be fun, it can also be a bit hit and miss. If you want to explore sequences with absolute control, you need a colour sequence you can set up as you want. So we made a sample holder, which is basically just a piece of 1 mm (or thicker) mounting board, cut out in the dimensions shown in **Figure 4**. A slot is cut out at

▶ Figure 5 Si-clops and a loaded sample holder

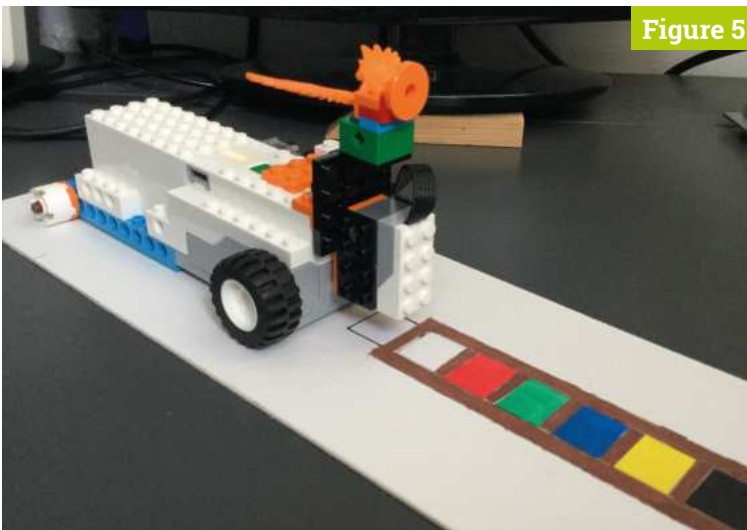


Figure 5

Top Tip



Installing rtmidi

If you haven't got the Python MIDI library installed, open up a Terminal and enter:

```
sudo apt-get install libjack0
sudo apt-get install libjack-dev
sudo apt-get install build-essential
sudo apt-get install libasound2-dev
sudo pip3 install python-rtmidi
```

one side, and we painted round the slot with brown paint. We cut out small pieces of card that will slide along the slot and painted them different colours. Fixing the sample holder to the desk with very small pieces of Blu Tack stopped it moving.

08 Painting

We used low-cost acrylic paints from an art shop to paint our tiles; there is a 4 mm strip down the edge of the tile and we painted it brown. We found that brown was a good neutral colour that did not affect the detection of adjacent colours. Then we painted the main colour in the rest of the tile. For a full set of tiles you need eight of each colour, but six is enough in practice. Also, the mounting board we used had a black back, so there is no need to make any black tiles – just simply turn any tile over. **Figure 5** shows the finished assembly.

In conclusion

Well, that wraps it up for our experiments with the Lego Boost set. We hope you have enjoyed it. But we have hardly scratched the surface – we certainly could write a whole book on the topic. Hopefully, we have inspired you to make your own creations and we would love to see what you have made with this set. We might revisit this set sometime in the future, though, for some more fun. [M](#)

Disclaimer

LEGO® is a trademark of the LEGO Group of companies, which does not sponsor, authorise, or endorse this article.

Si-clops.py (continued)

```

186.     drawWords("Velocity",218,54,(0,0,0), backCol)
187.     drawWords("Voice",285,54,(0,0,0), backCol)
188.     drawWords("Mute",370,54,(0,0,0), backCol)
189.     drawWords("Tempo",10,324,(0,0,0), backCol)
190.     drawWords("X10",112,366,(0,0,0), backCol)
191.     drawWords("BPM",40,366,(0,0,0), backCol)
192.     drawWords("Sample",180,340,(0,0,0), backCol)
193.     pygame.draw.rect(screen, lineCol,driveRect,1)
194.     updateValues()
195.     updateSamples(-1)
196.
197. def updateSamples(point):
198.     for i in range(0,samples): # draw sample
sequence
199.         pygame.draw.rect(screen, tileColours
[scannedBlock[i]], sampleRect[i],0)
200.         pygame.draw.rect(screen, (200,128,0),
sampleRect[i],1) # outline
201.         if point != -1:
202.             pygame.draw.rect(screen, (200,128,0),
markerRect[point],0)
203.         pygame.display.update()
204.
205. def nameNote(index):
206.     noteNumber = note[index]
207.     octave = (noteNumber // 12) - 5
208.     noteInOct = noteNumber % 12
209.     return str(octave)+" "+noteNames[noteInOct]+" "
210.
211. def updateValues():
212.     for i in range(0,6):
213.         drawWords(str(chan[i])+" ",48,85+i*40,
(0,0,0), backCol)
214.         if chan[i] != 10:
215.             drawWords(nameNote(i),112,85+i*40,(0,0,0),
backCol)
216.         else:
217.             drawWords(" "+str(note[i])+"
",112,85+i*40,(0,0,0), backCol)
218.             drawWords(str(velocity[i])+"
",204,85+i*40,(0,0,0), backCol)
219.             pygame.draw.rect(screen,backCol,
voiceRect[i],0)
220.             drawWords(str(instNames[
chan[i]-1]),270,85+i*40,(0,0,0), backCol)
221.             pygame.draw.rect(screen,backCol,muteRect[i],0)
222.             pygame.draw.rect(screen,lineCol,muteRect[i],1)
223.             if mute[i]:
224.                 pygame.draw.rect(screen,backCol,muteRect
[i],0)
225.                 drawWords("Mute",370,85+i*40,(0,0,0),
backCol)
226.                 drawWords(str(tempo)+" ",10,366,(0,0,0),
backCol)
227.                 pygame.draw.rect(screen,backCol,rsRect,0)
228.                 if running:
229.                     drawWords("Stop",300,340,(0,0,0), backCol)
230.
231.     else:
232.         drawWords("Run ",300,340,(0,0,0), backCol)
233.     pygame.draw.rect(screen,lineCol,rsRect,1)
234.     if not newScan:
235.         pygame.draw.rect(screen, backCol,driveRect,0)
236.         pygame.draw.rect(screen, lineCol,driveRect,1)
237.         drawWords("Sample",180,340,(0,0,0), backCol)
238.         pygame.display.update()
239.
240. def drawWords(words,x,y,col,backCol) :
241.     textSurface = font.render(words, True, col,
backCol)
242.     textRect = textSurface.get_rect()
243.     textRect.left = x # right for align right
244.     textRect.top = y
245.     screen.blit(textSurface, textRect)
246.     return textRect
247.
248. def callback_colour(colour, distance):
249.     global lastColour, updateColour, tileDistance
250.     lastColour = colour
251.     tileDistance = distance
252.     updateColour += 1
253.
254. def correctColour(colour):
255.     if colour > 10: # to correct for not a colour
256.         colour = 0
257.     correctColour = colour
258.     if colour == 5: # to correct for green error in
sensor
259.         correctColour = 6
260.     return correctColour
261.
262. def incMotor():
263.     global currentMotor, motorUpdate, updateColour,
step
264.     inc = 76.39
265.     currentMotor += inc
266.     movehub.motor_AB.angled(inc, .4)
267.     step += 1 # number of steps taken
268.
269. def handleMouse(pos): # look at mouse down
270.     global pramClick, pramInc
271.     #print(pos)
272.     if driveRect.collidepoint(pos):
273.         pygame.draw.rect(screen,hiCol,driveRect,0)
274.         pygame.display.update()
275.     if rsRect.collidepoint(pos):
276.         pygame.draw.rect(screen,hiCol,rsRect,0)
277.         pygame.display.update()
278.     for i in range(0,6):
279.         if muteRect[i].collidepoint(pos):
280.             pygame.draw.rect(screen,hiCol,muteRect[i],0)
281.             pygame.display.update()
282.
283.     pramClick = -1
284.     pramInc = 0

```


Si-clops (continued)

```

284.     for i in range(0,20):
285.         if incRect[i].collidepoint(pos):
286.             pramClick = i
287.             pramInc = 1
288.             pygame.draw.rect(screen,hiCol,
incRect[pramClick],1)
289.             pygame.display.update()
290.     for i in range(0,20):
291.         if decRect[i].collidepoint(pos):
292.             pramClick = i
293.             pramInc = -1
294.             pygame.draw.rect(screen,hiCol,
decRect[pramClick],1)
295.             pygame.display.update()
296.         if pramClick == 19: # for tempo X10
297.             pramInc = pramInc * 10
298.
299.     def handleMouseUp(pos): # look at mouse up
300.         global mute, velocity, note, chan, tempo,
running, newScan, midiStep, lastChan
301.         if rsRect.collidepoint(pos):
302.             running = not running
303.             if not running:
304.                 allOff()
305.                 midiStep = 0
306.             else:
307.                 for i in range(0, samples):
308.                     lastChan[i]=-1
309.                 updateValues()
310.                 updateSamples(-1)
311.             if driveRect.collidepoint(pos):
312.                 newScan = True
313.                 updateValues()
314.
315.             if pramClick != -1:
316.                 if pramClick < 6:
317.                     chan[pramClick] += pramInc
318.                     chan[pramClick] =
constrain(chan[pramClick],1,16)
319.                 elif pramClick < 12:
320.                     note[pramClick-6] += pramInc
321.                     note[pramClick-6] = constrain(note[
pramClick-6],15,127)
322.                 elif pramClick < 18:
323.                     velocity[pramClick-12] += pramInc
324.                     velocity[pramClick-12] = constrain(velocity[
pramClick-12],0,127)
325.                 elif pramClick >= 18:
326.                     tempo += pramInc
327.                     tempo = constrain(tempo,30,800)
328.                     setTime()
329.             if pramInc !=0:
330.                 if pramInc < 0:
331.                     screen.blit(icon[1],(
decRect[pramClick].left,decRect[pramClick].top))
332.                     pygame.draw.rect(screen,lineCol,
decRect[pramClick],1)
333.                 else:
334.                     screen.blit(icon[0],(incRect[
pramClick].left,incRect[pramClick].top))
335.                     pygame.draw.rect(screen,lineCol,
incRect [pramClick],1)
336.                     updateValues()
337.             for i in range(0,6):
338.                 if muteRect[i].collidepoint(pos):
339.                     mute[i] = not mute[i]
340.                     updateValues()
341.
342.     def constrain(val, min_val, max_val):
343.         return min(max_val, max(min_val, val))
344.
345.     def call_button(is_pressed): # scan on hub
button press
346.         global newScan
347.         if not is_pressed :
348.             newScan = True
349.             pygame.draw.rect(screen,hiCol,driveRect,0)
350.             pygame.display.update()
351.             time.sleep(1.0)
352.
353.
354.     def terminate(): # close down the program
355.         global midiout
356.         print ("Closing down")
357.         allOff()
358.         movehub.button.unsubscribe(call_button)
359.         movehub.color_distance_sensor.unsubscribe(
callback_colour)
360.         conn.disconnect()
361.         del midiout
362.         pygame.quit() # close pygame
363.         os._exit(1)
364.
365.     def checkForEvent(): # see if we need to quit
366.         global shutDown, newScan
367.         event = pygame.event.poll()
368.         if event.type == pygame.QUIT :
369.             terminate()
370.         if event.type == pygame.KEYDOWN :
371.             if event.key == pygame.K_ESCAPE :
372.                 terminate()
373.             if event.key == pygame.K_SPACE :
374.                 newScan = True
375.         if event.type == pygame.MOUSEBUTTONDOWN :
376.             handleMouse(pygame.mouse.get_pos())
377.         if event.type == pygame.MOUSEBUTTONUP :
378.             handleMouseUp(pygame.mouse.get_pos())
379.
380. if __name__ == '__main__':
381.     main()

```

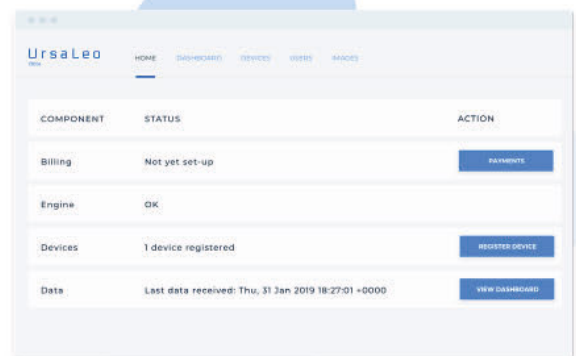
Use your Pi to collect sensor data to the Google Cloud



Ursa Leo

Download our Raspbian package to turn your Pi into a Google cloud gateway. Display data on dashboards, store and download it, use it to drive emails, texts and other alerts.

Enter code **magpi1** at account creation to receive a free Pi debug board*



- LEDs indicate BLE, internet and cloud connectivity
- Console interface
- Safe power down switch

ursaleo.com/raspbian

*Offer available for North America and Europe only



Make a Marauder's Map



PJ Evans

PJ is a writer, Raspberry Jammer, and developer. He solemnly swears he is up to no good.

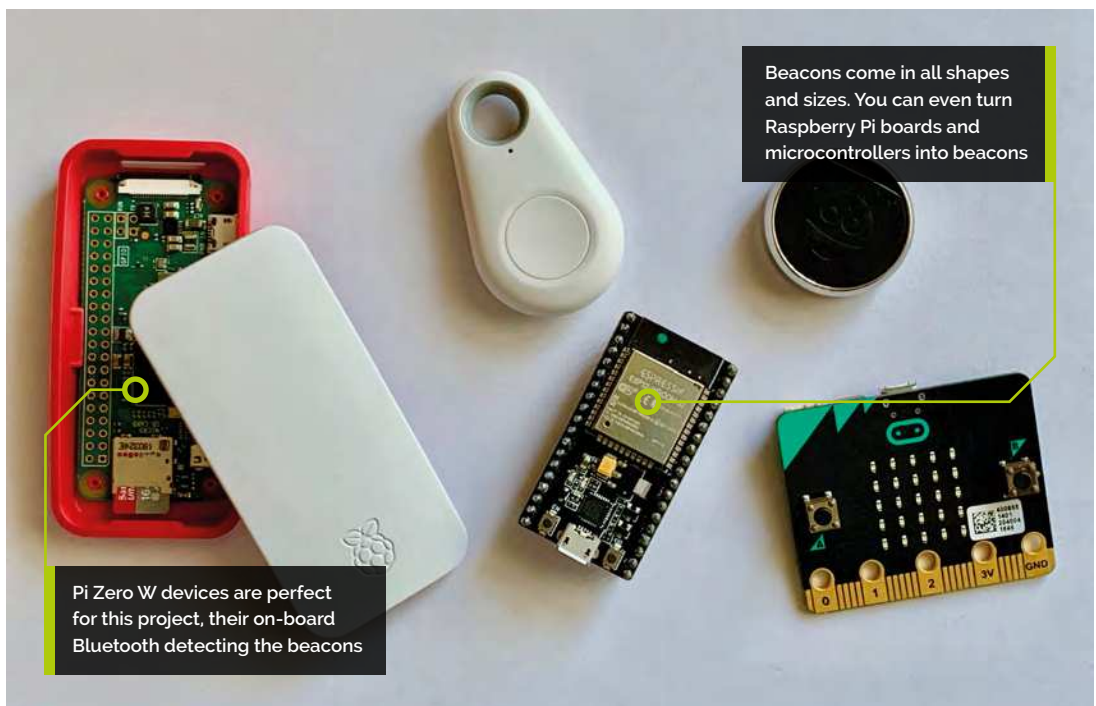
@mrpjevens

Make your own Marauder's Map and track your family, pets, and friends (or enemies?) using Bluetooth beacons

In the Harry Potter series, the Marauder's Map showed you the location of every person in Hogwarts. That map worked with magic, but ours will work with Raspberry Pi and beacons. Bluetooth beacons are low-energy devices that constantly ping out a signal that can be read by any device. Typically, they are used by museums or supermarkets in conjunction with a smartphone app to detect where visitors are and offer relevant information. We're going to flip this and have people carry the beacons with them. Raspberry Pi devices in each room will detect someone's presence and update a web-based map.

01 Get some beacons

Bluetooth Low Energy (BLE) beacons are simple devices that send out a constant signal in the form of a unique code or URL. Bluetooth 4.0 capable devices can detect this signal without needing to pair. There are a few different standards and our project is going to support the two most popular: iBeacon (Apple) and Eddystone (Google). These types of beacons are easily found online and tend to be small button-sized devices capable of running for up to a year on a single battery. Alternatively, software is available so you can use Raspberry Pi boards and microcontrollers (e.g. ESP32) so they act as beacons.



Beacons come in all shapes and sizes. You can even turn Raspberry Pi boards and microcontrollers into beacons

You'll Need

- > At least two rooms
- > One Pi Zero W per room
- > One or more willing participants
- > One beacon per participant, e.g. magpi.cc/iGmnAa

Pi Zero W devices are perfect for this project, their on-board Bluetooth detecting the beacons

02 Prepare your Pi devices

Each person is going to carry a beacon with them. As they move from room to room, a Pi Zero W in each room will detect the presence of the beacon and report it back to a designated server (which can be one of the Pi devices). Set up each Pi with Raspbian Stretch Lite, get them on the WiFi network, then update the software:

```
sudo apt update && sudo apt -y upgrade
```

We need to install a few libraries:

```
sudo apt install python3-pip libbluetooth-dev
sudo pip3 install beacontools[scan]
```

Now we can scan for beacons in Python.

“ Each scanner is going to report its findings back to a central server ”

03 Get the beacon IDs

Each beacon has either a unique ID code (iBeacon) or broadcasts a web address (Eddystone). If you're using iBeacons and the vendor hasn't supplied the ID, you'll need to discover it by scanning. Create a file containing the code from the **test.py** listing. Save it as **test.py** and run as follows:

```
sudo python3 test.py
```

All being well, you'll see your beacon's transmissions. Make a note of the 32-character string just after 'uuid'. Repeat for each one.

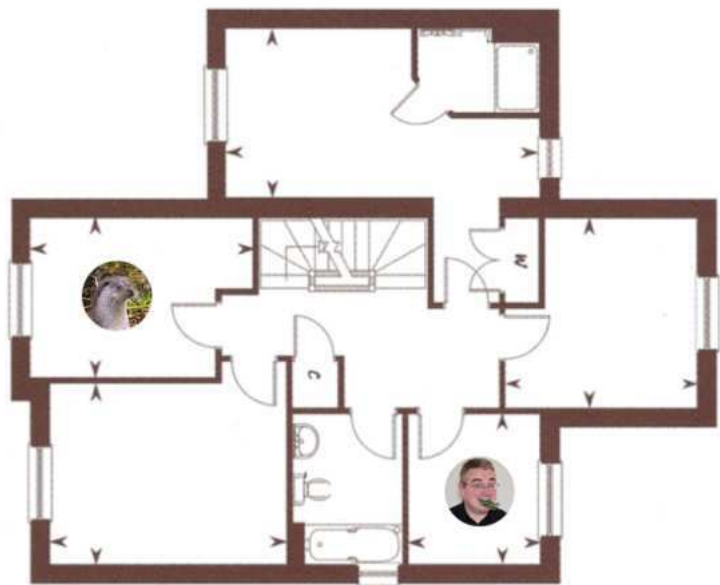
For Eddystones, instructions will be provided on how to set the web address. Set each one to **http://example.org/name**, where 'name' is the person's first name.

04 Install the server software

Each scanner is going to report its findings back to a central server. Pick one of the Pi devices for this role, then download the code from **magpi.cc/Hjhtwi** to a directory in your home folder called **beaconmap**. First, install Flask:

```
pip3 install flask
```

The PI-RAUDER'S MAP



Then test the server by running:

```
python3 ~/beaconmap/server/server.py
```

If you're on the same machine, open **http://127.0.0.1:5000** in a browser; otherwise you'll need to replace 127.0.0.1 with the IP address or host name of the server. You'll see a very boring webpage asking 'Where is everyone?'. **CTRL+C** will stop the server.

▲ Our final web app updates as people move around the house. There's lots of scope for getting creative here

05 Install the scanners

On each Raspberry Pi, create a file called **scanner.py** and enter the code from the listing here. If you don't fancy typing, install the software package from **magpi.cc/Hjhtwi** and you'll find it in the **scanner** directory.

The code uses Citruz's BeaconTools library to scan the area for beacon signals in bursts of ten seconds. When this happens, the code notes the ID/URL provided and gives it a score of one, with subsequent transmissions incrementing the score. After ten seconds, the scores are sent

Top Tip

Signal strengths

To avoid all the scanners seeing your beacon, put it on a low power setting if available.



▲ With its small size and smart casing, the Raspberry Pi Zero W makes a tasteful addition to any room

to the server. The server can compare scores for different scanners to work out where someone is, eliminating overlapping areas.

06 Configure and test the server

IDs aren't very useful, so configure the server by editing the **beacons** dictionary. Replace each key with a beacon ID and set the value to `{'name': 'name'}`. (There are more instructions in the code.) Add as many as you like. When the reports come in from the scanners, we will now know who they are!

You should have something like this:

```
beacons = {
    'b63cc056-6f3a-4a9b-80bf-11ff1c6ff724': {
        'name': 'PJ Evans'
    },
    '144dd069-e22e-418f-b940-c622d64b7252': {
        'name': 'Jazz The Cat'
    }
}
```

Test the server by starting it as before. If you've made any typos in the beacon list, you'll find out now. Leave it running.

07 Configure and start your scanners

Edit **scanner.py** and replace the value of `serverUrl` with the address of your server. Make sure `/'readings'` remains at the end. Then edit `room` to be the Pi's room or location name.

On each scanning Raspberry Pi, enter the following command:

```
sudo python3 ~/beaconmap/scanner/scanner.py
```

Each scanner will scan for ten seconds and then report scores back to the server. Check that traffic is flowing correctly. The server will echo incoming data on-screen. If things are not working, check if the server is running a firewall; it needs to allow traffic in through port 5000.

08 Get tracking

With **scanner.py** running on each Pi and the server up, take one of your beacons and place it next to one of the scanning Raspberry Pi devices. After ten seconds, refresh the webpage. Your beacon should have been matched and the location displayed. Repeat with another Pi. Check the page again. Did it move?

If you get confusing results, the Raspberry Pi boards may be too close to each other (Bluetooth goes through walls!), causing an overlap. If you can set the beacon's power output, it should be as low as possible; this increases battery life and accuracy.

09 Get creative

So far, so good, but it's all a bit boring on the webpage. Grab your crayons and design a map. We're not using any kind of positioning technology, so you can have fun and not worry about accuracy. Make sure it's nice and big as an image (try around 1000×1000 pixels) and save as **beaconmaps/servermap/static/rooms.png**.

You'll also need some avatars, so grab a selfie or screenshot your favourite meme, and create a square image 75×75 for each person and save in the same directory in the format **name.jpg** so it matches the names in the server configuration.

10 Run the advanced server

We've provided a fancier web server using Flask, to make the map a bit more fun. Stop the standard server (**CTRL+C**), configure the **server.py** file in **servermap** as before, then run the following command:

```
python3 beaconmap/servermap/server.py
```

Top Tip

Beacon types

There are many different beacon standards; make sure you're using iBeacon or Eddystone for this project.

Give the scanners time to report in and have a look at the page. It will probably be a bit of a mess, but you can have a look at the code for instructions on how to adjust things so your avatars appear in the right places. Make sure everyone is appearing in the correct location, then stop the server and scanners.

“ Make sure everything starts on boot, then runs in the background ”

11 Automate all the things

The final step is to make sure everything starts on boot, then runs in the background so you don't need to have a Terminal open all the time. There are many methods, but the easiest way to do this is to edit the `rc.local` file and add our requirements.

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

Before the last line that reads `exit 0`, insert a new line and add the following lines.

For the server:

```
/usr/bin/python3 /home/pi/beamap/servermap/
server.py &
```

For each scanner:

```
/usr/bin/python3 /home/pi/beamap/scanner/
scanner.py &
```

To test, reboot each device. Everything should now run in the background.

12 More with beacons

Now let your imagination get involved. What else can you do with these beacons? Try modifying the server to alert your smartphone when someone arrives in a certain place. Or how about a digital Easter egg hunt? Give everyone a battery-powered Pi Zero W and hide beacons. Your score could be generated automatically as you find them. Could you make a box that only opens when someone carrying the right beacon approaches? Over to you. [M](#)

scanner.py

> Language: Python 3

DOWNLOAD
THE FULL CODE:

magpi.cc/Hjhtwi

```
001. import time
002. import requests
003. from beacontools import BeaconScanner
004.
005. serverUrl = "http://127.0.0.1:5000/readings"
006. room = "Kitchen"
007. beacons = {}
008.
009.
010. # This function is called whenever a packet is detected
011. def callback(bt_addr, rssi, packet, additional_info):
012.
013.     # Parse out the type of beacon
014.     typeOfBeacon = type(packet).__name__.split(".").pop()
015.
016.     # Get the ID of the beacon
017.     if typeOfBeacon == "EddystoneURLFrame":
018.         beaconId = packet.url
019.     elif typeOfBeacon == "IBeaconAdvertisement":
020.         beaconId = packet.uuid
021.
022.     # Track how many times we've seen this beacon
023.     if beaconId not in beacons:
024.         beacons[beaconId] = 1
025.     else:
026.         beacons[beaconId] += 1
027.
028. # Scan for all advertisements from beacons
029. print('Starting beacon scanner')
030. scanner = BeaconScanner(callback)
031. scanner.start()
032.
033. while True:
034.
035.     # Allow a 10-second sample to come through
036.     print('Waiting 10 seconds')
037.     time.sleep(10)
038.
039.     # Now send the current scores to the server
040.     print('Sending to server')
041.     try:
042.         response = requests.put(serverUrl, json={"room": room,
043.                                                  "beacons": beacons})
044.
045.         if response.status_code == 200:
046.             print('Success')
047.         else:
048.             print('Got response code: ' + str(response.status_code))
049.     except:
050.         print("Communication error")
051.
052.     # Clean the scores
053.     beacons = {}
```


Buttons and labels with C and GTK



Simon Long

Simon Long is a software engineer working for Raspberry Pi, responsible for the Raspberry Pi Desktop on both Raspbian and Debian.

rpf.io

Make your empty window more interesting and interactive by adding a button and text label

The simple program we looked at in last issue's instalment (magpi.cc/81) draws a window on the screen, but it looks rather bare. There's also that problem that closing the window with the X button leaves the program still running. We're going to fix both of those by adding a button to close the window.

A button is another standard GTK widget, and it supports all the features you'd expect – you can have a label on it; you can see its appearance change when you click it; you can activate it with a keyboard shortcut; you can even disable it temporarily and grey it out.

The first thing to do is to add the button to the window. When we've done that, we can look at making it do something useful. Modify the code for the previous example as follows:

```
#include <gtk/gtk.h>

void main (int argc, char *argv[])
{
    gtk_init (&argc, &argv);
    GtkWidget *win = gtk_window_new
```

An Introduction to C & GUI Programming

For further tutorials on how to start coding in C and creating GUIs with GTK, take a look at our new book, *An Introduction to C & GUI Programming*. Its 156 pages are packed with all the information you need to get started – no previous experience of C or GTK is required!
magpi.cc/GUIbook

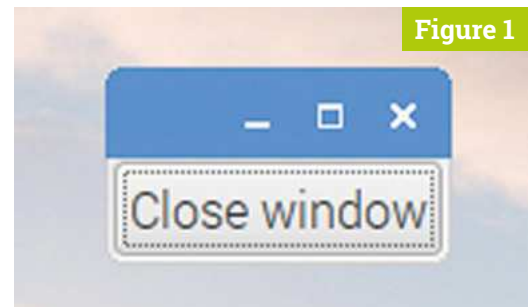
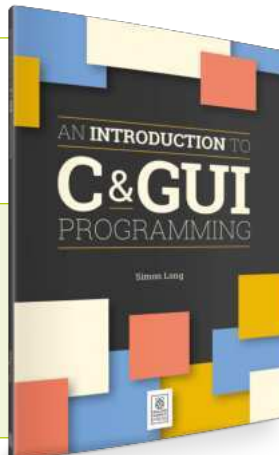


Figure 1

▲ Figure 1 The window with a button on it

```
(GTK_WINDOW_TOPLEVEL);
    GtkWidget *btn = gtk_button_new_with_label
("Close window");
    gtk_container_add (GTK_CONTAINER (win), btn);
    gtk_widget_show_all (win);
    gtk_main ();
}
```

Let's look at the changes:

```
GtkWidget *btn = gtk_button_new_with_label
("Close window");
```

This creates a GtkWidget widget. There is a standard `gtk_button_new` function, but using this version, `gtk_button_new_with_label`, allows us to add a text label to the button at the same time as we create it.

```
gtk_container_add (GTK_CONTAINER (win), btn);
```

The `gtk_container_add` function places the button inside the window.

```
gtk_widget_show_all (win);
```

Finally, the call to `gtk_widget_show` is now replaced with a call to `gtk_widget_show_all` – this tells

GTK to show both the named widget and any widgets it contains. If this line wasn't changed, the window would be shown, but the button would remain hidden.

If you now build and run the code, you'll find that you get a (smaller) window with a button labelled 'Close window' displayed on it (**Figure 1**).

Containers

In the example, we use the line:

```
gtk_container_add (GTK_CONTAINER (win), btn);
```

...to put the button into the window.

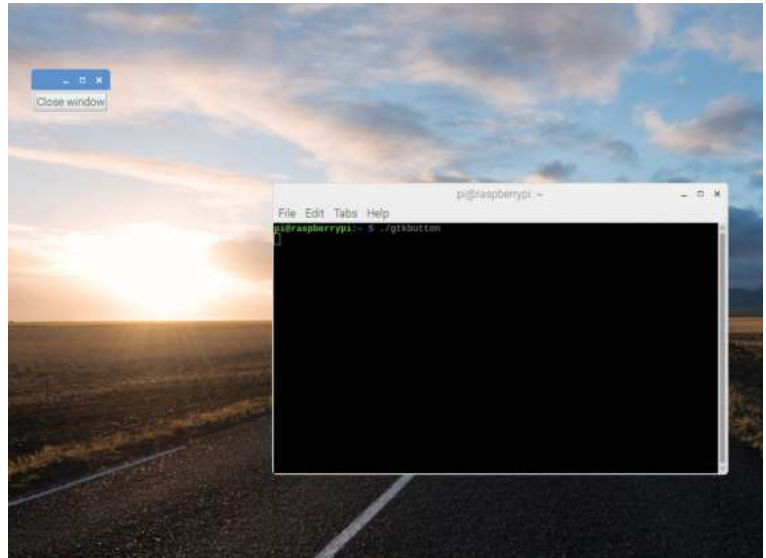
As you've probably noticed, each function call in GTK starts with the prefix `gtk_`, which is then followed by the name of the widget type upon which it operates – so `gtk_window_new` creates a widget of type `GtkWindow`. This function call, therefore, operates on widgets of type `GtkContainer`, but we are using it on `win`, which is a `GtkWindow`. How does that work?

There is a hierarchy of widget types within GTK, and a widget type inherits the properties of its parents within the hierarchy; that widget type can be described as a child of the widget types above it in the hierarchy.

“ Putting one widget inside another like this is something we need to do quite often ”

Because `GtkContainer` is one of the parents of `GtkWindow`, you can use `gtk_container_` functions on `GtkWindow` widgets. Many widgets have `GtkContainer` as a parent, because putting one widget inside another like this is something we need to do quite often.

You will notice that when `win` is supplied as an argument to the `gtk_container_add` function, it is wrapped inside the `GTK_CONTAINER()` function – this casts the argument to a variable of type `GTK_CONTAINER`; in other words, it tells the function call to treat this reference to a `GtkWindow` as a `GtkContainer`. The code will still work if this function isn't included, but the compiler will complain! (GTK includes a `GTK_` function for every



type of widget, as it is quite common to need to cast a widget to a different type.)

So what this line does is to tell GTK to treat the `GtkWindow` widget `win` as a `GtkContainer`, and to put the button widget `btn` into it.

▲ Running the program from a Terminal to make the window appear

Signals

If you run the application, you can click the button with the mouse, and it will highlight to show that it has been clicked; you can also hit the **ENTER** key on the keyboard to do the same thing – this button behaves exactly like the buttons we are used to seeing in applications, but it still doesn't actually do anything useful. The next step is to fix that.

The way to do that is to connect a *handler* function to the signal generated when the button is clicked.

Here's the code with a handler added:

```
#include <gtk/gtk.h>

void end_program (GtkWidget *wid, gpointer ptr)
{
    gtk_main_quit ();
}

void main (int argc, char *argv[])
{
    gtk_init (&argc, &argv);
    GtkWidget *win = gtk_window_new
(GTK_WINDOW_TOPLEVEL);
    GtkWidget *btn = gtk_button_new_with_label
("Close window");
```

► **Figure 2** This time, when you click on the button, the window will close



```
g_signal_connect (btn, "clicked",
G_CALLBACK (end_program), NULL);
gtk_container_add (GTK_CONTAINER (win), btn);
gtk_widget_show_all (win);
gtk_main ();
}
```

The handler function, otherwise known as a *callback*, is called `end_program`, and all it does is to call the GTK function `gtk_main_quit` – as you’d expect from the name, this function exits the loop within `gtk_main` and allows the program to end.

```
g_signal_connect (btn, "clicked",
G_CALLBACK (end_program), NULL);
```

“ All widgets can generate signals under various circumstances – mostly when interacted with by the user ”

This line connects the handler (`end_program`) to the signal called `clicked`, which is emitted by a button widget when the mouse is clicked on it. We use the `GTK_CALLBACK` function to make sure the compiler knows that the `end_program` function is a valid callback.

Signals are used a lot in GTK; all widgets can generate signals under various circumstances – mostly when interacted with by the user, but also when certain system events occur. Each type of widget has a set of signals it can generate; the `g_signal_connect` function is used to ‘connect’ a handler function to them, meaning it will be called when the signal is generated.

Handling signals is one of the most important jobs of the main loop which runs inside `gtk_main` –

if a signal is generated, code in the main loop checks to see if there is a handler associated with that signal, and calls the handler if so.

Try building and running the new code. We now have a button on the window as before, but when we click the button, the window closes and, importantly, the program exits as well.

This still isn’t perfect, though – if we click the X at the top right, the window closes (**Figure 2**) but the program doesn’t exit. Let’s fix that.

The way we do this is to connect another handler to the signal generated when that X is clicked. The name of this signal is `delete_event`, and it is generated from the window widget. So we need to connect the same handler to that event as well:

```
g_signal_connect (win, "delete_event",
G_CALLBACK (end_program), NULL);
```

Note that we connect to `win` rather than to `btn` this time – the `delete_event` signal is created by the window widget rather than the button. Now if the window is closed by clicking the X, the `delete_event` signal causes the `end_program` handler to be called and the main loop then exits.

We’ve now got a GTK application that opens and closes cleanly, but it still doesn’t do anything else. So in the next section, we’ll make it do something more useful...

More advanced layout

We’re going to add another button to the window, and display a count of the number of times the button is pressed. (All we said was that it would be more useful than the previous example; we didn’t say it would be useful for anything people might actually want to do!)

Labels

For displaying text on a window, we use a widget called a `GtkLabel` – a label is any piece of text which can’t be edited by the user. (The button we added in the previous example contains a label widget, which displays the name we gave the button in the `gtk_button_new_with_label` call.)

You can create a label with text already in it, or you can create a blank label and add text to it later on; you can also change the existing text in a label whenever you like.

Here's the code to add a label to our window:

```
#include <gtk/gtk.h>

void end_program (GtkWidget *wid, gpointer ptr)
{
    gtk_main_quit ();
}

void main (int argc, char *argv[])
{
    gtk_init (&argc, &argv);
    GtkWidget *win = gtk_window_new
(GTK_WINDOW_TOPLEVEL);
    GtkWidget *btn = gtk_button_new_with_label
("Close window");
    g_signal_connect (btn, "clicked",
G_CALLBACK (end_program), NULL);
    g_signal_connect (win, "delete_event",
G_CALLBACK (end_program), NULL);
    gtk_container_add (GTK_CONTAINER (win), btn);
    GtkWidget *lbl = gtk_label_new ("My label");
    gtk_container_add (GTK_CONTAINER (win), lbl);
    gtk_widget_show_all (win);
    gtk_main ();
}
```

Let's look at the new code:

```
GtkWidget *lbl = gtk_label_new ("My label");
```

This creates a `GtkLabel` called `lbl`, with the text 'My label' in it. We then call another `gtk_container_add` to add the label to the window. Build and run the code, and see what happens.

Ah. A window with a button, but no label. What's gone wrong? Well, if you look at the Terminal window, you can get a clue – you should see an error message telling you that a `GtkWindow` can only contain one widget at a time, and that this one already contains a `GtkButton`.

So you can only put one widget into a window, but we want to have two – a button and a label; that's not going to work. Unless...

Boxes

A user interface toolkit that only allowed you to put one thing at a time on a window would be a bit limited! So while a window can only contain one widget, GTK offers several widgets which can be used to hold sets of other widgets. The two most useful ones are boxes and tables – we'll

stick to boxes for the time being, as they are easier to use.

There are two types of box widget, which are called `GtkHBox` and `GtkVBox`; these are short for 'horizontal box' and 'vertical box', depending on how the widgets you put into the box are to be arranged. A box widget can hold an unlimited number of other widgets, but appears to its parent as a single widget, so can be used to put multiple widgets into a window.

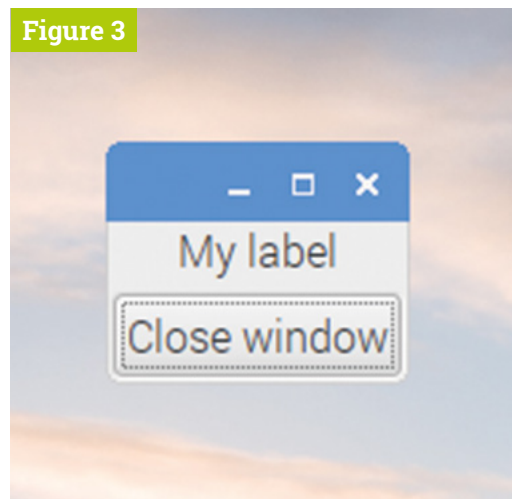
We're going to use a `GtkVBox` for this example, so modify the code as follows:

```
#include <gtk/gtk.h>

void end_program (GtkWidget *wid, gpointer ptr)
{
    gtk_main_quit ();
}

void main (int argc, char *argv[])
{
    gtk_init (&argc, &argv);
    GtkWidget *win = gtk_window_new
(GTK_WINDOW_TOPLEVEL);
    GtkWidget *btn = gtk_button_new_with_label
("Close window");
    g_signal_connect (btn, "clicked",
G_CALLBACK (end_program), NULL);
    g_signal_connect (win, "delete_event",
G_CALLBACK (end_program), NULL);
    GtkWidget *lbl = gtk_label_new ("My label");
    GtkWidget *box = gtk_vbox_new (FALSE, 5);
    gtk_box_pack_start (GTK_BOX (box), lbl,
TRUE, TRUE, 0);
```

Figure 3



Top Tip

Compiling and running

For details of how to compile and run your C program using GTK, refer to part 1 of this guide, in issue 81 (magpi.cc/81).

◀ Figure 3 A `GtkLabel` and a `GtkButton`

```

gtk_box_pack_start (GTK_BOX (box), btn,
TRUE, TRUE, 0);
gtk_container_add (GTK_CONTAINER (win), box);
gtk_widget_show_all (win);
gtk_main ();
}

```

Let's look at the changes line-by-line:

```
GtkWidget *box = gtk_vbox_new (FALSE, 5);
```

This creates a new `GtkVBox` widget. The two parameters to the call are whether or not the box is *homogeneous* – which means it assigns the same amount of space to every widget it holds – and the spacing in pixels between the widgets it holds. So in this case, the box is not homogeneous – each widget will only be allocated as much space as it needs – and there will be a gap of 5 pixels between the two widgets we are putting into it.

```

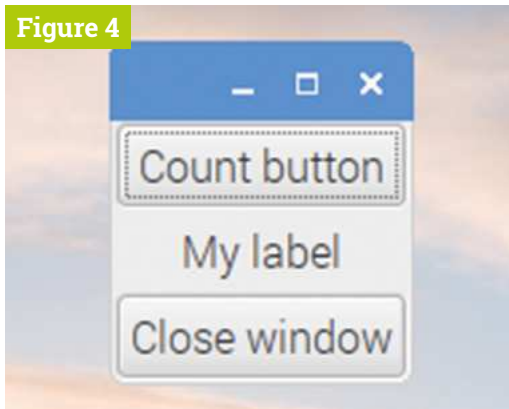
gtk_box_pack_start (GTK_BOX (box), lbl,
TRUE, TRUE, 0);
gtk_box_pack_start (GTK_BOX (box), btn,
TRUE, TRUE, 0);

```

The `gtk_box_pack_start` function puts a widget into a box. Widgets are added in the order that calls to this function are made, and the `_start` at the end indicates that widgets are placed in the box from the 'start' end, which is the top of a `VBox` or the left-hand side of an `HBox`. (There is also a `gtk_box_pack_end` function, which adds widgets from the bottom or right-hand sides.)

The function takes five arguments; the first is the name of the box we are packing the widgets into, and the second is the widget we want to put into the box. The remaining three arguments control how

Figure 4



► **Figure 4** The button counting application before a button is pressed

“ The window only holds one widget, the box, so won't complain about holding too many widgets ”

the widget is positioned in the box – we'll look at this later on.

```
gtk_container_add (GTK_CONTAINER (win), box);
```

Finally, we add the box to the window itself – this means the window only holds one widget, the box, so won't complain about holding too many widgets.

If you now build and run this, you should see a window like **Figure 3** (previous page).

Now we have a window containing the controls we need, let's get back to creating our button push counter. Here's the code:

```

#include <gtk/gtk.h>

int count = 0;

void end_program (GtkWidget *wid, gpointer ptr)
{
    gtk_main_quit ();
}

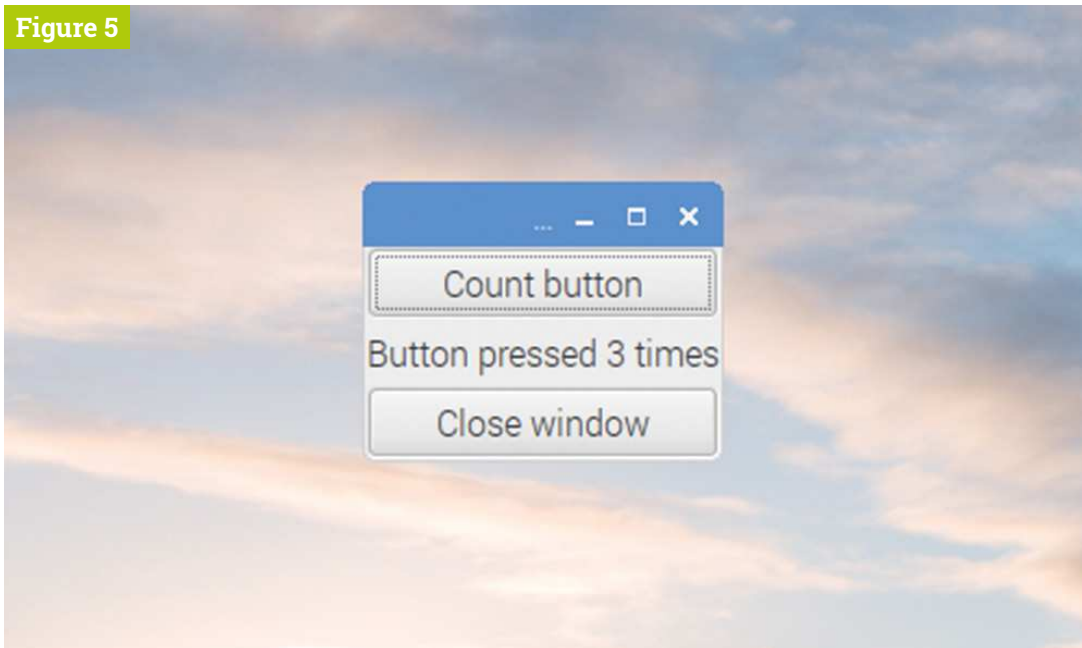
void count_button (GtkWidget *wid, gpointer ptr)
{
    char buffer[30];
    count++;
    sprintf (buffer, "Button pressed %d times", count);
    gtk_label_set_text (GTK_LABEL (ptr), buffer);
}

void main (int argc, char *argv[])
{
    gtk_init (&argc, &argv);

    GtkWidget *win = gtk_window_new (GTK_WINDOW_TOPLEVEL);
    GtkWidget *btn = gtk_button_new_with_label ("Close window");
    g_signal_connect (btn, "clicked", G_CALLBACK (end_program), NULL);
}

```

Figure 5



◀ **Figure 5** After the 'Count button' has been pressed three times

```
g_signal_connect (win, "delete_event",
G_CALLBACK (end_program), NULL);

GtkWidget *lbl = gtk_label_new ("My
label");

GtkWidget *btn2 = gtk_button_new_with_
label ("Count button");
g_signal_connect (btn2, "clicked",
G_CALLBACK (count_button), lbl);

GtkWidget *box = gtk_vbox_new (FALSE, 5);
gtk_box_pack_start (GTK_BOX (box), btn2,
TRUE, TRUE, 0);
gtk_box_pack_start (GTK_BOX (box), lbl,
TRUE, TRUE, 0);
gtk_box_pack_start (GTK_BOX (box), btn,
TRUE, TRUE, 0);
gtk_container_add (GTK_CONTAINER (win),
box);
gtk_widget_show_all (win);
gtk_main ();
}
```

We've added a new button, `btn2`, along with a new handler callback function, `count_button`, which has been connected to the `clicked` signal on the new button.

```
g_signal_connect (btn2, "clicked",
G_CALLBACK (count_button), lbl);
```

Note that whenever we have used `g_signal_connect` before, the last argument has been `NULL`, but in this case it is `lbl`. The final argument of this

function is a pointer – it can be a pointer to anything which the handler might need to use. If you look at the arguments to the handler function, they are:

```
void count_button (GtkWidget *wid, gpointer ptr)
```

The first argument of a handler for any signal is a pointer to the widget that generated the signal. Different signals have different numbers of arguments in their handler functions, but they all have a general-purpose pointer as the last argument, and this receives the pointer that is supplied as the last argument to the `g_signal_connect` call.

In this case, we need to update the text in the `GtkLabel`, so the handler needs access to a pointer to the label, and the general-purpose pointer is a convenient way of doing this. So we pass the `lbl` pointer into `g_signal_connect`, and in the handler we cast this pointer back to a `GtkLabel` so we can use it in the `gtk_label_set_text` function, which updates the text in the label:

```
gtk_label_set_text (GTK_LABEL (ptr), buffer);
```

Sometimes we need to pass more than one reference to something to a handler function; there are various ways around this like the use of structures or global variables, but for simple cases like this, the general-purpose pointer works well.

If you now build and run the code, you should see a window which looks like **Figure 4**.

When you click on 'Count button', the text in the label should update on each click (**Figure 5**). [🔗](#)



THE Official
RASPBERRY PI
PROJECTS BOOK

VOLUME 4

**200 PAGES OF
 IDEAS & INSPIRATION**

THE OFFICIAL RASPBERRY PI PROJECTS BOOK VOLUME 4



DIY Games Console



Build a Robot



Set up a Spy Cam



Make a Magic Mirror

55 PROJECTS & GUIDES

200 pages of ideas & inspiration

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

£12.99

200 pages of
Raspberry Pi

THE *Official*

RASPBERRY PI PROJECTS BOOK

VOLUME 4

Amazing hacking and making projects
from the makers of *MagPi* magazine

Inside:

- How to get involved with the Pi community
- The most inspirational community projects
 - Essential tutorials, guides, and ideas
 - Expert reviews and buying advice

Available
now

magpi.cc/store

plus all good newsagents and:

WHSmith BARNES & NOBLE



Available on the
App Store



GET IT ON
Google Play



MODEL RAILWAY PROJECTS

Want to power up your hobby trains?
Let's go Pi-powered

We see so many hobbies making use of the Raspberry Pi in amazing and novel ways. In many cases it makes a hobby more accessible (like robotics), or previously difficult tasks a little easier (such as advanced wearables).

One of the oldest electronic hobbies around, model railways are intricate and beautiful miniaturised train services, long operated in garages and basements. Like anything, the technology in them has improved over the years, and many people have found ways to incorporate a Raspberry Pi into their track layout. So, let's jump aboard and see what projects we can find!

STATIONS

Choose your destination

68

CONTROL
SOFTWARE

69

DECODER
HARDWARE

70

LIONEL TRAIN
SWITCH CONTROL

71

INTERNET
OF LEGO

72

MODEL RAILWAY
AUTOMATION

75

WIMBORNE
MODEL RAILWAY

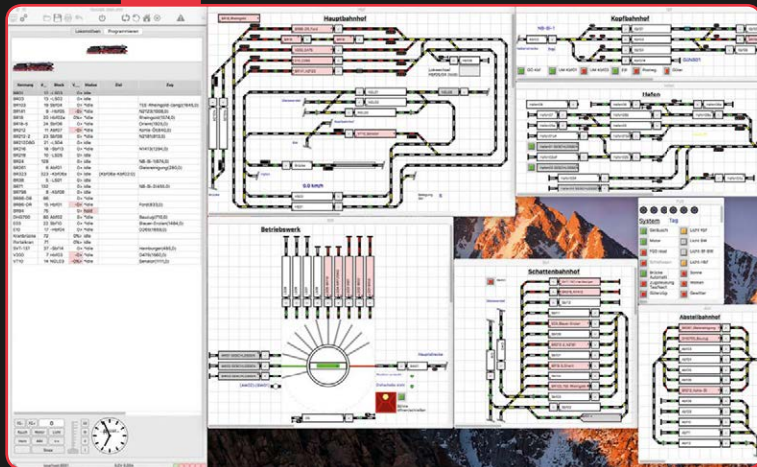
CONTROL SOFTWARE

Some great ways to get started with Pi-powered model railways

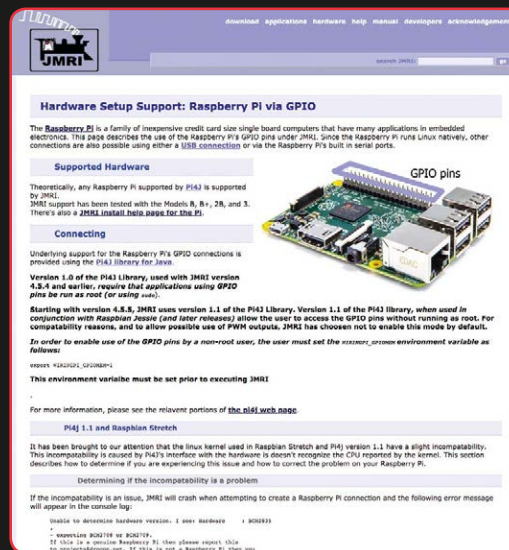
One of the major benefits of adding a Raspberry Pi to a model railway is being able to do some custom coding so that it runs exactly the way you want it to. There's some software to help you with that...

Rocrail | magpi.cc/WaTTaF

Rocrail is free, multiplatform software for controlling model trains. It allows the user to completely, or partially, control an entire railroad system via automation and manual control, depending on their preferences, and includes a more visual, block-based interface for setting up the automation. It even has a web client, making the manual control easier to access from anywhere. This control includes cars, accessories, and lighting – it's pretty powerful!



▲ With Rocrail, multiple user interfaces can be accessed from any web-connected device



▲ JMRI (Java Model Railway Interface) can be run on a Raspberry Pi and used to control your railway

JMR-Pi | magpi.cc/puVuFE

This is a Raspberry Pi port of JMRI, Java-based control software suitable for model railways. It's described simply as a way of listening to and sending messages on the 'command bus'. In conjunction with the software DecoderPro (magpi.cc/wRpmeK), it can be used to control a model railway.

JMR-Pi on a Raspberry Pi will work as a control box / headless server from which to connect to remotely – and thanks to its size, it hides a lot better in a railway than a spare computer.

COMMAND STATIONS

Hardware for controlling railways



SPROG

sprog-dcc.co.uk

A HAT for the Raspberry Pi that turns it into a DCC interface.

Digitrax decoder

digitrax.com

Digitrax makes a number of decoders that work on its LocoNet.



Hornby Elite

magpi.cc/mLFCKg

A bit expensive, but it's compatible with a lot of this software.

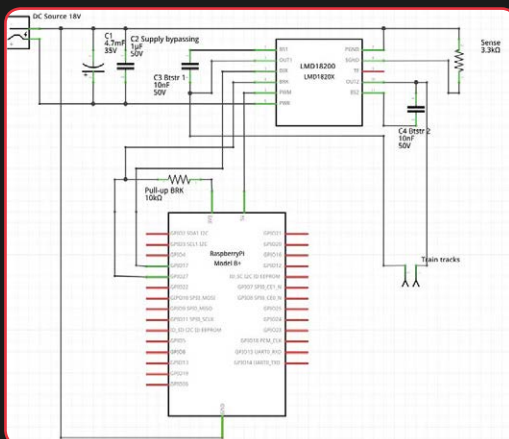


Check the supported hardware on any software you decide to use!

▣ Rocrail even has a web client, making the manual control easier to access from anywhere ▣

dccpi | magpi.cc/ZFMZBJ

A Python library specifically optimised for Raspberry Pi, it lets you work with the DCC protocol, which is used in the control of model railways. It's a fairly simple library, able to control direction and speed, as well as some lights, over GPIO. It also requires a little power boost to make sure it provides a signal to the tracks, but it's a fun, very DIY solution to controlling a train. It makes use of the WiringPi library, which could technically also be used on its own to control the railway, but dccpi is a lot more advanced.



▲ The Raspberry Pi needs an additional booster circuit, like this one, to actually provide the signal to the tracks

AMAZING MODEL RAILWAY PROJECTS

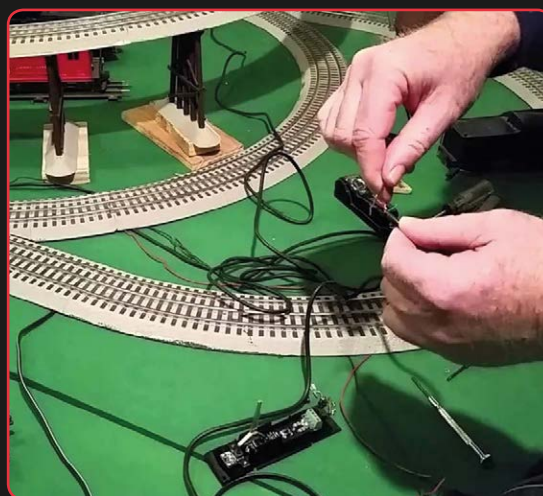
Here are some incredible things people have done with a model railway and a Raspberry Pi!

Lionel Train Switch Control

magpi.cc/yiLZhX

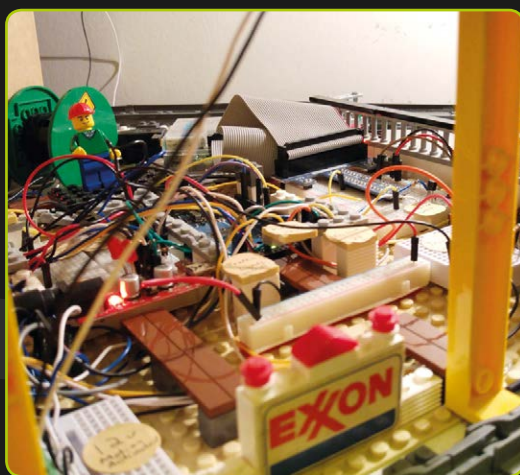
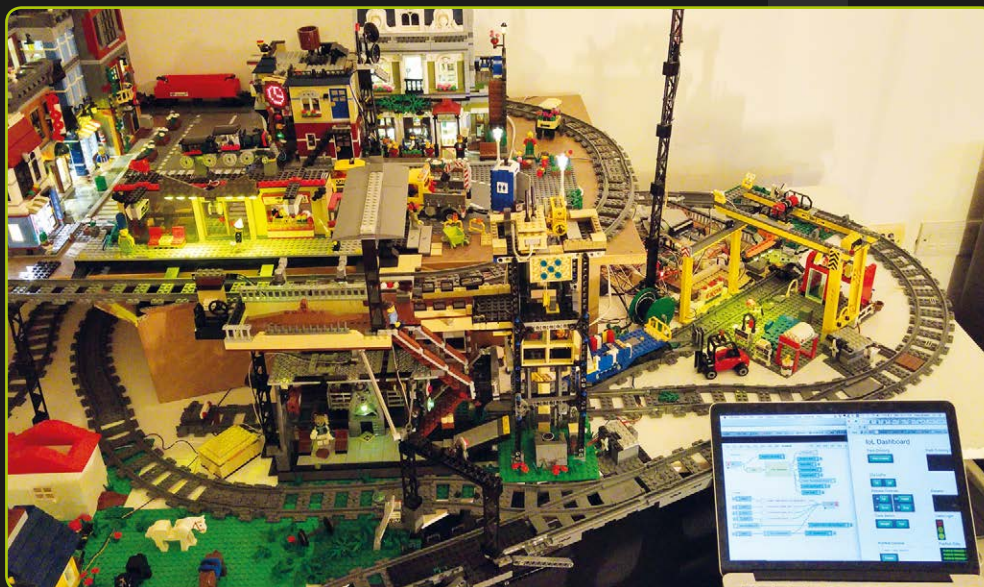
Our friends at Dexter Industries put this guide together a while ago, showing you how to control switches in a model railway using a mixture of Arduino and Raspberry Pi. This uses a more direct method than DCC, as you're controlling the power that activates the switches and stuff manually with power.

While this is quite different to other methods, it's an interesting way to program the setup, and allows you to use sensors hooked up to the Raspberry Pi a little more easily for performing automation or adding your own manual buttons or controls. Perhaps you could 3D-print your own DCC-esque controller with this.



▲ You'll need your relay to handle 20 V AC, so make sure to get the correct relay, and turn the power off before connecting it all

■ The Internet of Lego is a living project where I set out to learn everything about the Internet of Things ■



▲ Multiple Pi devices and Arduinos work in conjunction to control the city

Internet of Lego

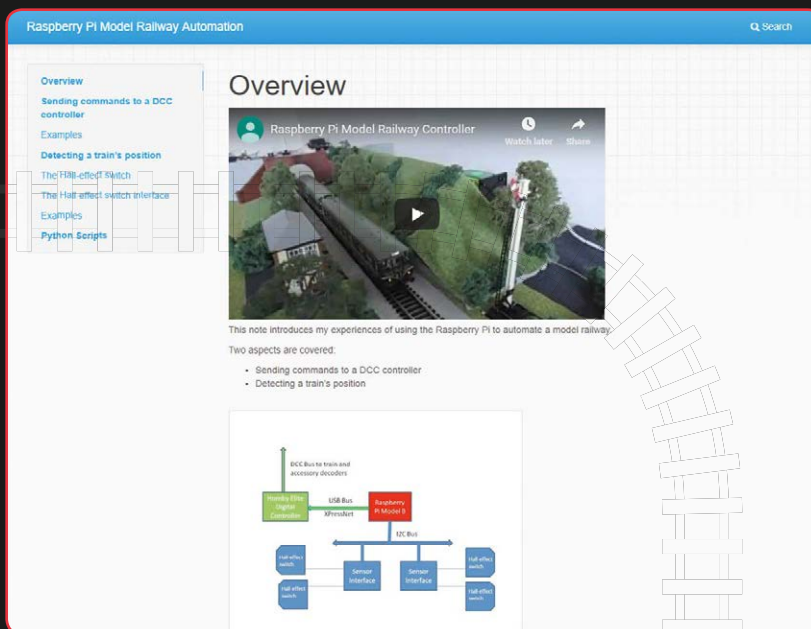
internetoflego.com

More than just a model railway, this is an entire Lego city, including railway, that is controlled by a mixture of Raspberry Pi and Arduino. We spoke to its creator, Cory Guynn, in issue 48 of *The MagPi* (magpi.cc/48).

“The Internet of Lego is a living project where I set out to learn everything about the Internet of Things,” Cory told us back then. The train system is his favourite part, as he loves to see stuff in motion. There’s a train-scheduling system built upon the Transport for London API, showing the destination on an OLED screen and changing the tracks to match it.

It makes use of WiFi, infrared, wired sensors, and more to control not only the train and switches, but also the screens and other parts of the giant IoT Lego city.

It's a great way to learn about all the low-level electronics and code involved in a pre-existing system



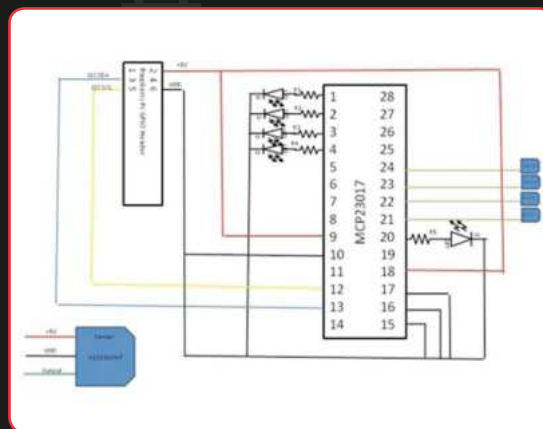
Learn all about Hall effect switches from Petter Wallen's setup

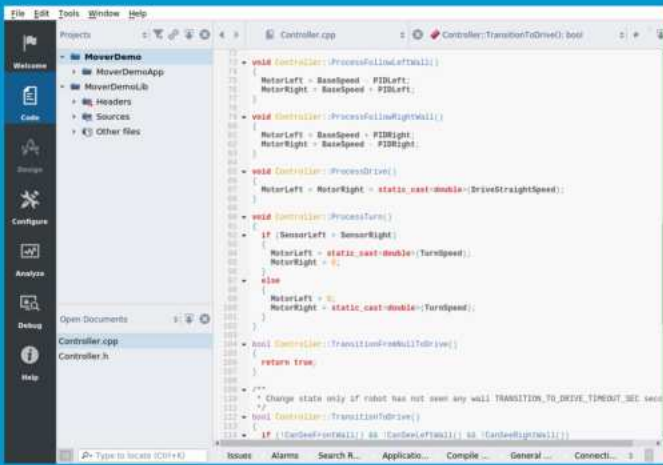
Raspberry Pi Model Railway Automation

magpi.cc/FXtwjt

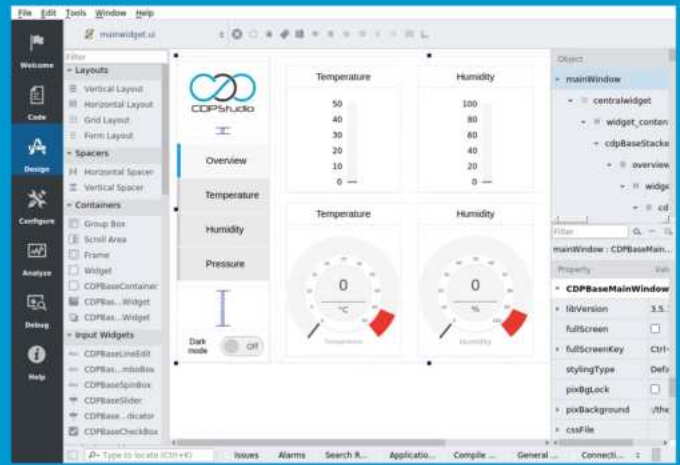
If you want to get into the real nitty-gritty of controlling your model railway, we recommend taking a look at Peter Wallen's project. It makes use of the XPressNet protocol, which is one of the many ways you can control DCC devices. While the code is written in Python, it sends commands to the command line to activate specific addresses of things connected to the Pi, so it's very low-level.

It's a great way to learn about all the low-level electronics and code involved in a pre-existing system, though, and there are some example scripts available to help you get started integrating it into your own railway.

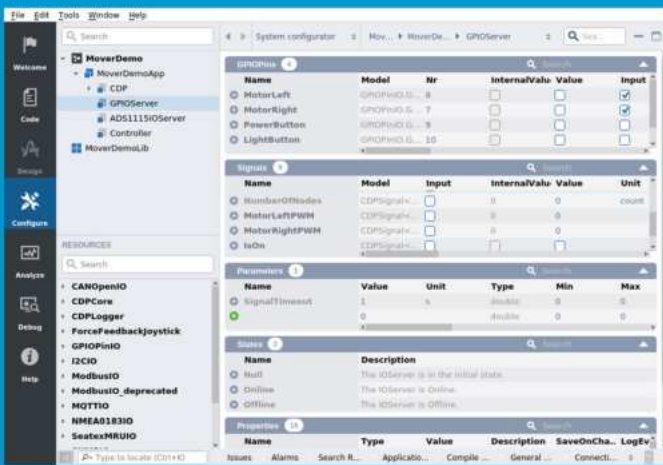




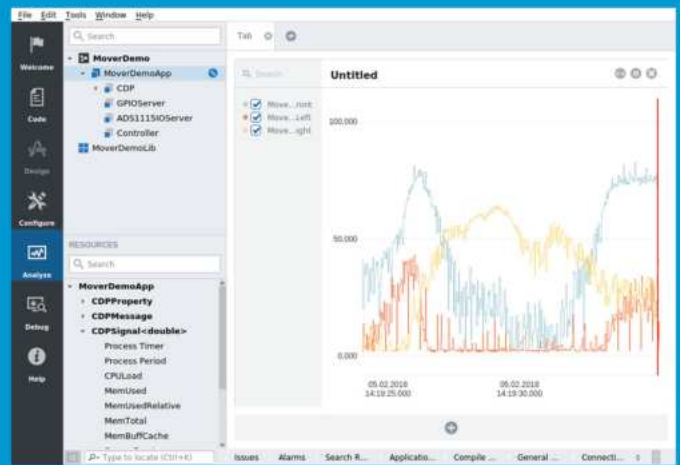
Code



Design



Configure



Analyze

Now free for home projects

A professional control system development tool

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

Free download on www.cdptech.com

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

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



(Hello World)

THE MAGAZINE FOR COMPUTING AND DIGITAL MAKING EDUCATORS

SUBSCRIBE
FREE
IN PRINT AND DIGITAL

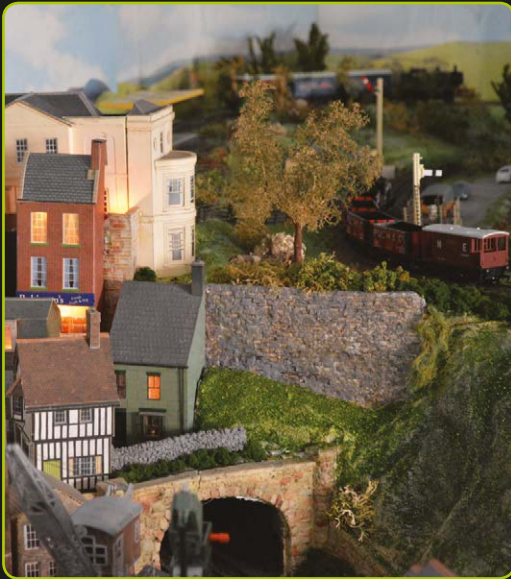
FIND US ONLINE:

helloworld.cc

 @HelloWorld_Edu

 fb.com/HelloWorldEduMag





Wimborne Model Railway

magpi.cc/wScJqw

Model railways are more than just the locomotive making its way around – there's also the miniature land set up around it! Wimborne Model Town in Dorset is home to a large model railway in a very intricate miniature town. All the houses have lights, and there several lamps and street lights in the model as well.

A few years ago, a volunteer named Terry Coles at the model town upgraded the light system so it could be controlled via a Raspberry Pi. There are two sets of lights: the ones in the miniatures, and a big strip of LEDs that acts as a sun for the model railway. A day-night cycle exists in the model railway, with the LEDs lighting up the entire thing as if it was the middle of the day, before slowly turning off to represent a sunset. As this happens, lights in the model start turning on in sections, resulting in a dark night highlighted by the little lights of the model.

It's programmed in C and makes use of WiringPi. One of the cleverer parts of the project is having the OS load into memory so that when power is pulled from the Pi, it won't have a chance to corrupt the SD card. [M](#)

▲ The lights turn on and off in sections depending on the time of the day cycle



PiBug 2WD Robot

SPECS

MOTORS:

2 x DC TT 'Yellow' motors, 1:48 ratio, nominally 200 rpm at 6V, with JST connectors

POWER:

6 x AA batteries (not supplied)

OPTIONAL SENSORS:

Line-follower board (2 x IR sensors), ultrasonic HC-SR04P

► 4tronix ► magpi.cc/oUtoHY ► £24 / \$31

An easy-to-build, two-wheeled robot that uses the Raspberry Pi as part of its chassis

The annual Pi Wars event always inspires more people to get involved in robotics, but where's the best place to start? Probably an inexpensive, entry-level kit like the PiBug. What's really clever about this two-wheeled robot is that your Raspberry Pi is used as a structural component of the robot's body, so you don't need to buy or create a separate chassis. This makes the robot much easier and quicker to get up and running, particularly as no soldering is required.

Buggy building

Online step-by-step assembly instructions (magpi.cc/xDQQTW) show how to assemble all the parts to build your PiBug, which should take around 30 minutes.

Screw-together brass hex pillars are used as mounts for the two DC motors, creating a sturdy rectangular framework between them. One thing to note is that the 30 mm vertical pillars have slightly different ends, so you need to check that the rounded ones are on the bottom.

The other pillar ends are connected to two of the Raspberry Pi's mounting holes using short (9 mm) pillars. The semicircular castor board is then fitted under the front of the Pi, using more pillars. Note that if you want your robot to do line-following, the standard castor board can be replaced with one featuring two line sensors (available separately for £6).

Next, the PiBug motor controller board is slotted onto the Pi's GPIO header, then secured with more



► All the parts you need to build your PiBug robot – you just need to supply the Raspberry Pi



short pillars – onto which the battery holder board is mounted. The robot and Pi are both powered by six AA batteries; rechargeables are recommended.

The final stage involves some simple connections. The motor wires have JST connectors that fit into slots on the motor controller board – much easier than screw terminals. If using line sensors, their female connectors slot onto pins at the rear of the motor controller board; this leaves six spare pins, connected to GPIO pins and power/ground, to use as you wish – along with a push-button on the board. An optional HC-SR04P ultrasonic distance sensor (available for £2.22) is connected via four small holes in the battery board.

In the driving seat

With the PiBug robot assembled and wired, you can start playing with it! A flick of a switch on the side turns on the power to the robot and Raspberry Pi. Once booted up, you can start coding using the PiBug Python library, which comes with a trio of examples to get you started.

As well as code to test the IR line-follower and ultrasonic distance sensors, if fitted, there's a

“ Your Raspberry Pi is used as a structural component of the robot's body ”

program that enables you to drive the PiBug around manually via key presses on a remote computer via an SSH connection. The robot is pretty nippy and agile, zipping along at pace and spinning on its two wheels to quickly change direction. Ours didn't run in a perfectly straight line by default, but you could always alter the library code to calibrate the motors. The PiBug also struggled to set off from a stationary position at the lower speed settings, although it worked fine at a lower speed after slowing down from a faster pace. The low position of the castor board also prevents it navigating major bumps in the terrain.

Still, for an entry-level robot, it works well enough and is a lot of fun. Naturally, you can use the Python library to create your own code to make the PiBug move autonomously, using the optional sensors to avoid obstacles and/or follow lines. **M**

▲ Along with numerous hex pillars, the Raspberry Pi forms part of the robot's chassis

Verdict

While it's a little more expensive than the CamJam EduKit #3, the use of the Raspberry Pi and hex pillars to form a chassis is ingenious and makes for a cool-looking buggy.

8/10

LibreELEC 9.0 Leia

SPECS

KODI VERSION:
18 'Leia'

COMPATIBILITY:
All Raspberry Pi
versions

INSTALLATION:
Graphical
wizard or
burnable image

**EMULATOR
API:**
Libretro

IMAGE SIZE:
123MB

► LibreELEC ► libreelec.tv ► Free

The latest version of this 'Kodi OS' brings Kodi 18's star new feature: retro game emulators. **Rob Zwetsloot** sees if they're worth the fuss

We love testing out a new Kodi version with new features here at *The MagPi* – the Raspberry Pi powers media centres the world over, and we like to see how much they improve. Kodi's development is changing slightly, so this may be the last time we make sure to do a review for a new numbered version, or in this case LibreELEC.

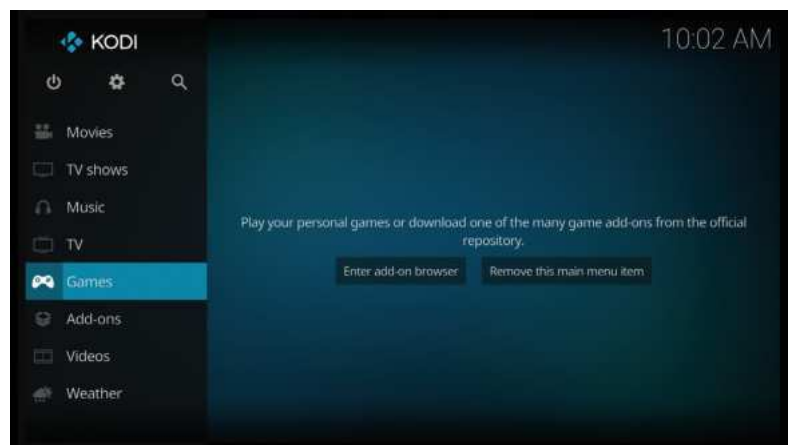
LibreELEC 9.0 is the latest version of our favourite HTPC OS for the Pi, built around the newest Kodi 18. As well as the usual minor tweaks and bug fixes behind the scenes (including a nice little update to the settings layout), the real selling-point of the new Kodi is native support for retro game emulators.

Join the party

For years now, you have been able to play games on Kodi. However, it was not something properly supported by Kodi itself – you just had to install the right add-ons and such.

That's all changed here, with a wide selection of popular console emulators and controller maps made available using Libretro, which is part of the core Kodi system.

▼ The familiar interface is still there



▲ Not all the emulators run as well as we'd like – the homebrew *Blade Buster* was slugging on some, with washed out colours

The tech is all there, and it's all pretty old, so it should be a simple addition to Kodi and LibreELEC. Or at least you would think, especially going by the handy new Games item in the default menu. This menu lets you install any of these emulators, but that's about it. Games can be launched by going into the file manager in settings and finding your ROM, which is quite awkward.

This is only temporary, according to the dev team, with a game browser coming in the future. For now, though, all the rest of its media abilities work just as well as before – it just might be best to temper any expectations about the game-playing aspects. **M**

Verdict

The game emulation stuff generally works, but it's not properly baked in yet. Still, if you just want to watch your videos or play music, there's nothing better.

7 / 10



BUILD, PLAY, WIN!



RETRO GAMING SALE



Vilros Raspberry Pi 3 Model B
Retro Pi Arcade Gaming Kit with
Classic USB Gamepad
~~\$79.99~~ **\$75.99**






Vilros Raspberry Pi 3 Model B+
Retro Arcade Gaming Kit With
Retro Gaming Controller Set
~~\$114.99~~ **\$109.99**



Vilros Raspberry Pi 3 Model B
Plus Retro Gaming Kit with 2
Gamepads & Fan-Cooled Retro
Gaming Case
~~\$89.99~~ **\$85.49**



Vilros Retro Gaming USB Classic
Controller Set of 5
~~\$39.99~~ **\$37.99**

USE THE CODE **RETRO** FOR 5% OFF] Check us out at www.vilros.com    /vilrosusa

Crack Coupon Code for ADDITIONAL 20% Savings!

Restrictions Apply" *Offer Expires May 31, 2019"

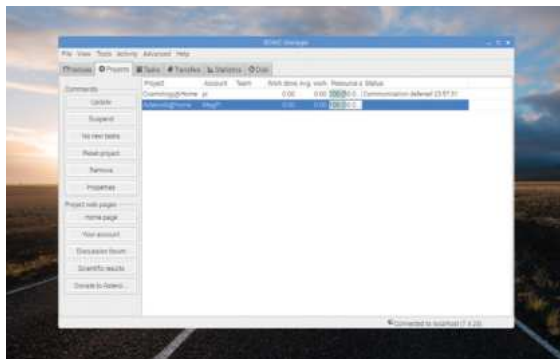


10 Best:

Health projects

Get a clean bill of health with these Raspberry Pi projects

One of the things we love to see is amazing medical and health applications of the Raspberry Pi. In our experience, they're usually some of the most impressive displays that we see at Maker Faires, Coolest Projects, and other events where folks are showing off amazing projects. Here are ten of the best to inspire you. [M](#)



BOINC

Distributed computing for science

Lending spare CPU cycles to help cure diseases? Not as crazy as you might think: BOINC has been around for ages, and uses your idle CPU to help fold proteins or solve equations that can one day result in cures and treatments.

► boinc.berkeley.edu



Patient monitor

Vital signs HAT

This project uses the HealthyPi, a HAT for the Raspberry Pi that is specifically for medical applications. With it, you can create a full ECG which can measure heart rate, SpO2, respiration, temperature, and blood pressure.

► magpi.cc/ikhuAK

Heartfelt

Heart monitoring through feet

Preventative care can save the NHS a lot of money, and the Heartfelt monitor is able to detect the symptoms of an oncoming cardiovascular event from the feet of at-risk patients. When an issue is detected, a carer is notified, and a quick doctor's visit and medicine will mean the patient avoids a trip to the hospital.

► hftech.org



LiV Pi

Air pollution monitor

Air quality is a big deal in Hong Kong, and this Pi-compatible device lets you know just how clean your air is. It's aimed towards businesses, but it's also very useful at home if you're living in the big city and have concerns over air pollution.

► livpi.com

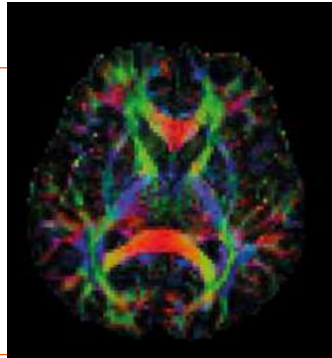


MRI analysis

Medical image processing

A very early but excellent Pi project, where a research scientist turned his Pi into an MRI analysis computer. The key trick to getting this to work was hardware-accelerated calculations, as it wouldn't be as well optimised in software.

► magpi.cc/SWxdSi

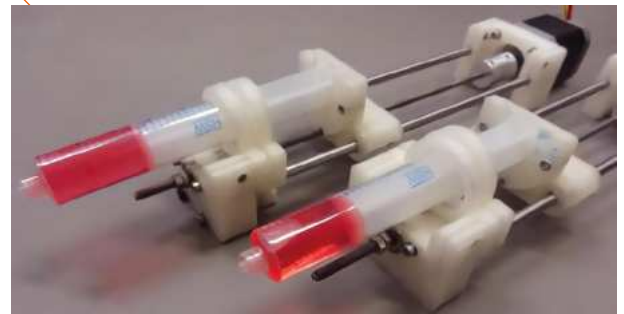


Open-source syringe pump

Perfectly timed doses

A usually expensive piece of equipment that allows for timed, precise doses of drugs or chemicals from a syringe – whether for a patient with an IV, or a scientist running a long-term experiment. This version, powered by a Pi, is much cheaper.

► magpi.cc/wPGyPy



Artificial Pi Pancreas

AI insulin pump

A DIY solution to a normal continuous glucose monitor (CGM) that helped Dana Lewis perfectly control her insulin injections thanks to a bit of machine learning. The Pi itself controls the insulin pump using the data from the CGM.

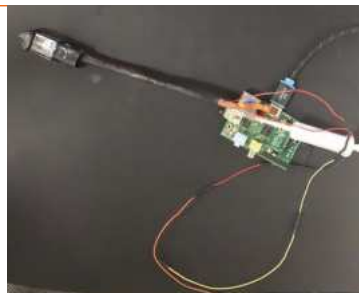
► diypr.org

Gastric cancer screening device

Low-cost medical tech

Not every part of medicine is glamorous. However, like this screening device, it can still save lives. The idea is that it's very cheap and can be used in poor countries where the incidence of gastric adenocarcinoma is high.

► magpi.cc/przGLX

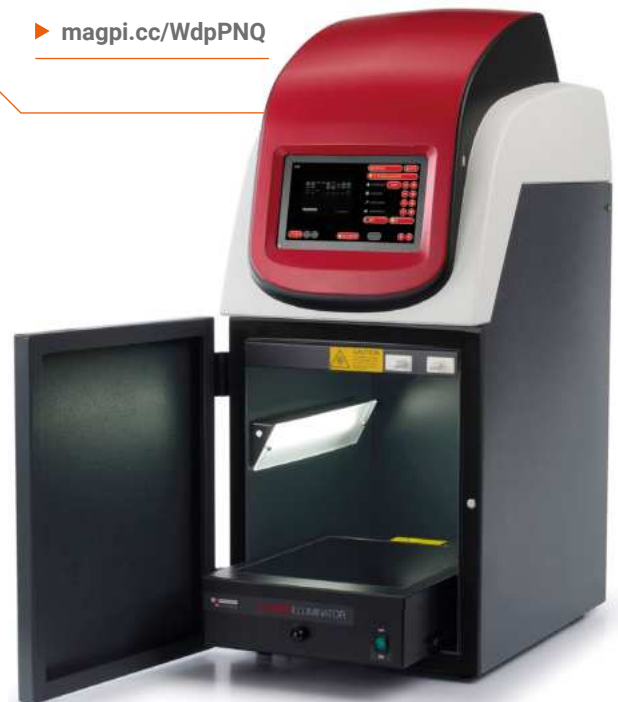


NuGenius

DNA gel imaging

This is an incredibly fancy piece of medical tech – a DNA imager that uses a Raspberry Pi. It's quick, has modern tech conveniences such as a touchscreen, and is an all-in-one device.

► magpi.cc/WdpPNQ



Heartbeat monitor



Exercise stats

This project pairs the Pi with Bluetooth heartbeat sensors that are popular with gym equipment, enabling you to get more useful stats from a workout than your usual treadmill display. Daniel here has made data analysis very flexible with the code.

► magpi.cc/2kBj0aM

Learn artificial intelligence with Raspberry Pi

With these resources, your Pi can think for itself. **PJ Evans** gets (machine) learning

Google AI Education

AUTHOR

Google


Price:
Free

ai.google/education

The Raspberry Pi is a powerful tool when it comes to artificial intelligence (AI) and machine learning (ML). Its processing capabilities, matched with a

small form factor and low power requirements, make it a great choice for smart robotics and embedded projects. Google is a champion of the Pi's place in the AI world, its AIY voice recognition system being given away with this very magazine (issue 57, no longer available in print – magpi.cc/57).

Google's AI Education site is an ideal place to start your machine-learning journey. If you want to really understand how AI/ML works 'under the hood', there are lot of principles to comprehend before you

even get to coding. Google has provided a self-guided suite that starts with a 'Crash Course' in machine learning, then expands to cover the basics of problem framing and data gathering. The main online course comprises 25 lessons over 15 hours (approximately, you can set your own pace) and comes in the form of reading materials, interactive sections, programming exercises, and video tutorials. This is then backed by a substantial collection of follow-on courses. A superb resource. 



Essential websites

Providers of popular AI/ML tools

GOOGLE CORAL

Recently featured in *The MagPi* #79, this exciting new USB accelerator from Google transforms the Raspberry Pi's AI capabilities by adding a dedicated neural network processor. Also, don't miss Google's AIY site. coral.withgoogle.com

OPENCV

The tool of choice for many robot builders, the Open Source Computer Vision Library

not only gives your Pi sight, but the ability to 'comprehend' what it sees. A powerful tool for intelligent object recognition.

opencv.org

TENSORFLOW

If you want implement machine learning on a Pi, chances are you'll be using TensorFlow to do it. The official site not only features full documentation, but also a range of courses.

tensorflow.org




Complete Guide to TensorFlow for Deep Learning with Python

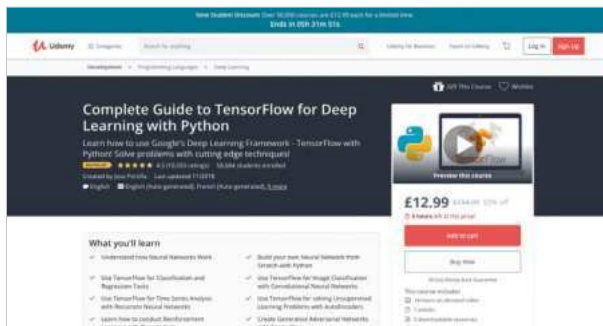
AUTHOR

Jose Portilla

Price:
£195 (often discounted)
magpi.cc/rbAdNE

TensorFlow is, without a doubt, the most popular software library for machine learning on the Raspberry Pi. If you're keen to get started and write code, TensorFlow will

probably be your tool of choice. You can get a great introduction to TensorFlow in *The MagPi* #71 (free download: magpi.cc/71), but if you are after a deep dive, Udemy offers a comprehensive 14-hour video course that covers not only the theory of machine learning, but also the practicalities of setting up the software with real-world examples and programming exercises. If you're after a hands-on learning approach, this may well suit. Don't be put off by the steep price – this course is often promoted and was £13 at time of writing. 



Learn by example

Do you learn by doing? Try these

HOW TO BUILD DIY AI PROJECTS USING GOOGLE TENSORFLOW AND RASPBERRY PI

A collection of AI/ML projects to build or provide inspiration. From introductions to TensorFlow to a wide range of projects including magic mirrors and an impressive cucumber sorting machine!
magpi.cc/gLsMHK

AI ON RASPBERRY PI WITH THE INTEL NEURAL COMPUTE STICK

Like Google, Intel has also released a USB-based neural co-processor. This tutorial is a great 'getting started' guide, talking you through installation and on to your first facial recognition app.
magpi.cc/jSShvE

RASPBERRY PI POKÉDEX

PylmageSearch is an incredible resource for learning OpenCV. This detailed tutorial is ideal for younger minds, using a Pi and the official touchscreen to create a Pokédex that can 'recognise' plush Pokémon.
magpi.cc/Ergqjd



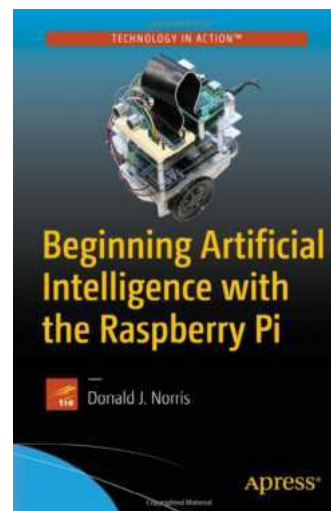
Beginning Artificial Intelligence with the Raspberry Pi


AUTHOR

Donald J Morris

Price:
£20 (digital £16)
magpi.cc/VfrZbO

This book is perfectly tailored to the Raspberry Pi community. Not only does it cover the principles behind concepts such as neural networks, fuzzy logic, and shallow versus deep learning, it also provides practical, fun projects to code and build. Starting with simple examples of learning, you can play your Pi at noughts-and-crosses and Nim. Along the way, the projects are made fun through the use of the Pi's GPIO header, using LEDs and switches to bring code to life. You then progress to robotics, covering obstacle avoidance and light



seeking. A steady learning curve culminates in the building of 'Alfie', your very own artificially intelligent robot vehicle. If you fancy building the winner of the next Pi Wars, this could be the perfect reading material. 



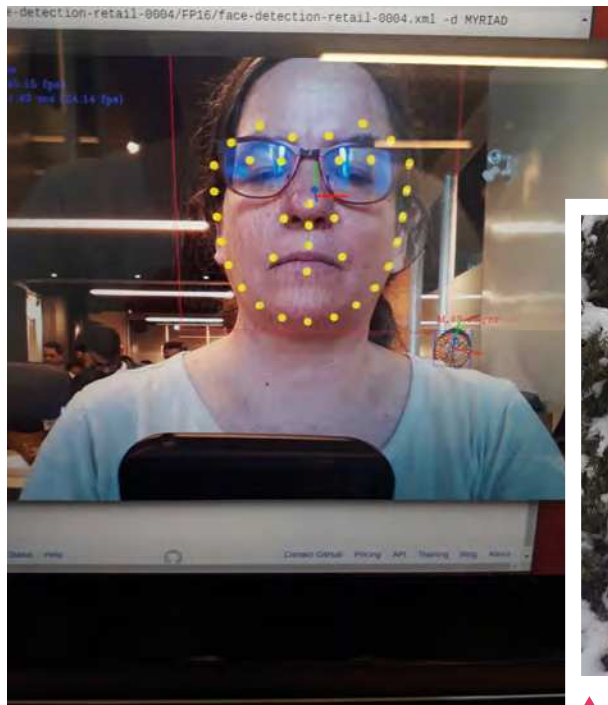
Nicole Parrot

Home-schooling her kids led Nicole to help others learn to code

> Category **Educator** | > Day job **CTO** | > Website nicole.parrot.ca

As part of the wider educational mission of the Raspberry Pi Foundation, there's been an ongoing effort to address the low percentage of girls and women in computing and related spaces. While the overall gender balance is disappointing, there are still some amazing women who have always been part of the community, such as Nicole Parrot.

▼ Computer vision on the Pi is remarkably powerful – here's Nicole showing off its face detection



“Last millennium I was a software developer in the special effects industry,” Nicole, who currently works at Dexter Industries, tells us. “Followed by five years at Microsoft in 3D graphics.”

Nicole quit in the late nineties to home-school her kids, which included teaching them and other children how to code. Her home coding lessons soon moved over to schools, and she's helped teachers bring code to the classroom.

“That gives me approximately 15 years of experience in exposing kids to code,” Nicole says. Her work at Dexter Industries reflects this experience.

What is your role at Dexter Industries?

I am CTO for Dexter Industries. As such, I manage a small team to bring new products to market, and create projects. I also manage curriculum writing. We want to offer teachers the full experience, from a stable robot in the classroom to prepared lessons, to open-ended projects. Our main product is the GoPiGo, now in its third iteration. We've learned a lot about what's needed to make a robot for the classroom, and this particular robot is the end result. It comes packaged with DexterOS, based on Raspbian with tools that make it usable out of the box. [...] Within DexterOS



▲ This fake snowman is powered by Pi, and manages to survive the cold Québécois winters



we have Bloxter, a block-based language, and Python available, again all within the browser. I'm quite proud of this project, to be honest.

What inspires you?

Kids! And this is from a person who used to be entirely career-oriented. I used to avoid kids, and flee from the room if someone would bring their kid with them. Then I did a full 360 degrees when I got my own. Kids are such a gift; their way of thinking is magical. Their reactions are just awesome when they figure out something like a tricky piece of code, or lighting up an LED for the first time. Sharing knowledge (remember that I home-schooled, I did all subjects, all years, and then some more!) is the most inspiring experience I can think of.

I'm also a yarn artist, I love natural yarns, and I try to buy from local breeders when possible. I love to know that this particular yarn came from an

“ Within DexterOS we have Bloxter, a block-based language, and Python available, all within the browser ”

animal with a name, and I get to know that name too. I have a source for alpaca yarn that allows me to visit the animals in question. That way I know they are well-treated.

How did you learn about the Raspberry Pi?

I was an early adopter of the Pi. I got my first one at Christmas 2012. I could not get one before that because they were not for sale in Canada where I reside – the first available ones were in December. I've been around since the beginning. However, I had been out of tech for a long time already by that point, didn't remember any electronics, my UNIX was entirely forgotten except for 'vi'. I could use vi and even exit it if I wasn't thinking about it. Muscle memory is a



▲ Nicole teaching data analysis with Python for beginners

funny thing. But there wasn't much I could do with the Pi at that point.

I spent quite a few years playing catch-up, getting my tech groove back. It was funny how simple things were tricky to get back – I remember having trouble getting two embedded loops to work – but more advanced concepts like recursion were a breeze. **W**

◀ Nicole's activities don't just limit her to Canada – she's also been to Cambridge (UK) for Pi Wars, and been to the Pi Store

Amazing projects

Nicole has built some impressive things with Raspberry Pi – here are just a few of them:

GrovePi Zero bike trips

magpi.cc/MePSpa

Based on a GPS tracker and time-lapse photo creator that Nicole created for long road trips in a camper van, you can use Google Maps to see your trip.

Holiday wreath

magpi.cc/DcmEKu

A Pi-powered wreath to put on your door during Christmas!

Dancing snowman

magpi.cc/RPFDkp

This snowman dances to music created by Sonic Pi.

This Month in Raspberry Pi

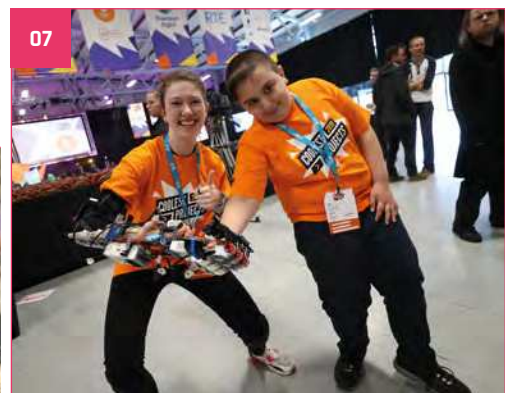
Coollest Projects International 2019

Kids from around the world show off their incredible projects in Dublin!

We love seeing photos from any **Coollest Projects event**. The makers, coders, and STEAM superstars of tomorrow are always at these technology fairs, showing off some amazing builds. Coollest Projects International is the original and largest of the events, and this year was no exception. [M](#)

01. The venue was huge, and full of amazing young makers
02. Representatives from all the countries participating got to show their projects
03. It was even a chance to meet some amazing YouTubers
04. Making in all its forms was present at the event
05. Giant Rock-'em-sock-'em robots kept everyone entertained!
06. Smaller robots also made quite a splash at the event
07. Ever wanted a giant Lego robot hand? We do now
08. Cosplay is definitely making





MagPi Monday

Amazing projects direct from our Twitter

Every Monday we ask the question: have you made something with a Pi over the weekend? Every Monday, our followers send us amazing photos and videos of the things they've made. Here is a small fraction of them. Follow along at #MagPiMonday. [👉](#)

DDO @dandomingue Follow

Replying to @TheMagPi

Singing rabbit with a pizero, speakerPhat and controlled via home assistant



02

Brian Cortell @Cannonfodder Following 01

Replying to @TheMagPi

My teenage son Bill aka @jedifodder made this for his mum's birthday, lit by @pimoroni rainbow phat and Pi Zero.



01

Leo White @LeoWhitesTweets Follow 03

Replying to @TheMagPi

I built a PS3 controller to Atari DB9 joystick adaptor out of a RPi Zero and a couple of logic level converters so I can play games on my Commodore 64.




03

Charly Kühnast @CharlyKuehnast Follow 04

Replying to @TheMagPi

There's a lux sensor above the Pi's display. As light decreases, the Pi gradually fires up the lights (Philips Hue and IKEA Tradfri) across the living room. The display and two tiny OLEDs show the sensor's readings and light status.

#RaspberryPi #lego



04

8BitsandaByte @8BitsandaByte Follow 05

Replying to @TheMagPi

An internet controlled and tracked hamster wheel! 🐹



05

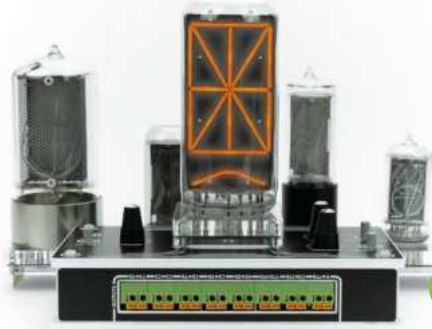
- 01. We love the craftsmanship behind this wonderful gift
- 02. We haven't seen a video of this, but we hope it sounds fine and not creepy
- 03. Now this is a very cool technical project – it's a work in progress, but we love the idea
- 04. The presentation of this project alone is enough to warrant a spot here
- 05. Hamsters and the internet are always a great combination

Crowdfund **this!** Raspberry Pi projects you can crowdfund this month

Universal PRO Nixie Tester and Healer

From the makers of the Nixie HAT, which lets you attach a Nixie tube to the top of your Raspberry Pi, comes the tester and healer. Nixie tubes may experience problems, which are just a natural part of how they work, and this device will test and heal many common issues they have.

► igg.me/at/pro-nixietester



CROWDFUNDING A PI PROJECT?

If you've launched an irresistible Pi-related project, let us know!
magpi@raspberrypi.org

Raspberry Pi Cameras & Temperature Sensors: An IoT Guide

Not all projects are physical items – this one is a book which has wise words on environmental sensors and the Raspberry Pi Camera Module in relation to IoT. The campaign will be finished by the time you read this, but it's already successful so give it a look!

► kck.st/2IPhQ7P

Best of the rest!

Here are some other great things we saw this month

EVERGREEN WARNING

"Accidentally fried my Pi, so I turned it into a reminder not to do it again...". Extremely wise words from Reddit user *ironically_maiden*.

► magpi.cc/uFxUrF



RASPBERRY PI STORE VR

If you don't live in Cambridge, you may never have been to the Raspberry Pi Store. Fear not – if you have Google Cardboard VR Goggles, you can now look around it with this amazing recreation.

► magpi.cc/EKfGcV



From USD\$49.99 Only

Most convenient DAC for Pi No compromise in Audio Quality

- 192kHz Sampling Rate / 24bit Resolution DAC
- 2 x Precise Crystal as master clock for lower jitter
- Ultra low noise voltage regulator
- Up to 1.5" OLED Colour Display
- Power switch with graceful shutdown
- Full integration with Volumio
- Open source software on github
- Optional Case, Amp, CD Upsampling & Extraction Add-ons

<http://nanomesher.com/nanosound>





Raspberry Jam Event Calendar

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

01. Beijing Raspberry Jam

- 📅 Monday 3 June
 - 📍 South Side, No. 9 An Hua Street, Beijing, China
 - ▶ magpi.cc/bTFQeK
- Anyone who wants to program, or just to get familiar with the Raspberry Pi, can join.

02. Bake-n-Make your own Raspberry Pi

- 📅 Tuesday 4 June
 - 📍 Seattle Makers, Seattle, WA, USA
 - ▶ magpi.cc/cmzdwG
- Come gather with other Pi enthusiasts. Any level of experience is welcome.

03. The All London Raspberry Pi Jam

- 📅 Saturday 8 June
 - 📍 The Microsoft Reactor London, London, UK
 - ▶ magpi.cc/NKrvfe
- Aimed at teaching children and parents about technology and programming.

04. Bridport Raspberry Jam

- 📅 Saturday 15 June
 - 📍 Bridport Library, Bridport, UK
 - ▶ magpi.cc/FehSPV
- Bridport Jam is back after a hiatus, with some cool demos and robots to show off.

05. Bognor Regis Raspberry Jam

- 📅 Saturday 15 June
 - 📍 University of Chichester, Bognor Regis, UK
 - ▶ magpi.cc/veSiyh
- An ideal event if you have a Pi and aren't sure what to do with it, or you are seeking inspiration for new ideas.

06. Seattle Raspberry Jam

- 📅 Wednesday 19 June
 - 📍 Bellevue Library, Bellevue, WA, USA
 - ▶ magpi.cc/NvsBzz
- Participate in the monthly project, share your knowledge, or show a project you've created.

07. Cornwall Tech Jam

- 📅 Saturday 8 June
 - 📍 Penwith College, Penzance, UK
 - ▶ cornwalltechjam.uk
- For anyone interested in technology, of all ages and abilities. Ask questions and learn about programming.

08. Stafford Raspberry Jam

- 📅 Tuesday 11 June
 - 📍 Stafford Library, Stafford, UK
 - ▶ magpi.cc/ioiLD0
- Welcoming anyone that wants to show off their projects, or see other people's builds!

FULL CALENDAR

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



FIND OUT ABOUT JAMS

Want a Raspberry Jam in your area?
Want to start one?
Email Ben Nuttall about it:
jam@raspberrypi.org



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

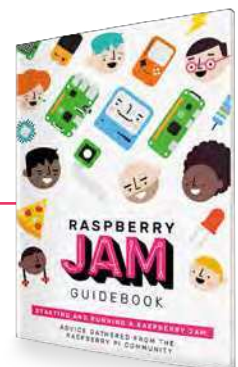


Raspberry Jam advice: Finding a venue

“I got one of the first Raspberry Pis and was looking to find other people like me who had little experience of programming and Linux to help and support each other. I approached the library to see if they would give us the space and they agreed.”

Simon Belshaw – Exeter Raspberry Jam

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



Your Letters



International stamps

I love the Raspberry Pi stamps you have! I wish I could get them in the USA – is there any chance you’ll be releasing them over here?

Jeffrey via Twitter

The stamps are not actually made by us or Raspberry Pi, they’re made by the Royal Mail, so it’s not something we can really sell you in America. The USPS probably wouldn’t accept them. You can buy a presentation pack from the Royal Mail store in the UK (shop.royalmail.com) and have them sent to the USA (delivery is £2.15 via Royal Mail). You can read more about why the stamps exist in our story about them on page 6.

Pi Zero serial interfaces

I am using a Raspberry Pi Zero for my project. Currently I am facing a problem as I need to use both a Raspberry Pi Camera Module and a Raspberry Pi 7-inch touch display. Unfortunately, the Pi Zero only includes one CSI connector port and I don’t want to connect that display with HDMI. Is there any way to map the HDMI to GPIO pins?

Akhil via email

This is a bit of a tricky one – we do know of a few Raspberry

▼ The HyperPixel works via GPIO pins, and may be a good alternative for some Pi Zero projects

Pi displays that work via the GPIO pins. Pimoroni and Adafruit have made a few (see the HyperPixel for an example: magpi.cc/WgSMAA); however we’re unable to find specific instructions on how to do this for the official display.

All we can suggest is to look for a different 7-inch display, or even switch it out for the HyperPixel 4.0, as it is a very nice screen.

Contact us!

- ▶ Twitter [@TheMagPi](https://twitter.com/TheMagPi)
- ▶ Facebook [magpi.cc/facebook](https://facebook.com/magpi.cc/facebook)
- ▶ Email magpi@raspberrypi.org
- ▶ Online raspberrypi.org/forums



Event photos

I like seeing photos from the Coolest Projects and other events – I assume there’s always more than what we see in the magazine, though. Is there any way we can get to see all the photos from these events?

Margaret via Facebook

Keep an eye on the websites and blogs related to these specific events and you may see many more photos than those we fit in the magazine. We have been trying to get a public photo repository going, but it’s not as easy as uploading an album to Facebook.

This issue, we have some photos that came from Coolest Projects International, which you’ll be able to find more of on the CoderDojo website (coderdojo.com).

Article repo


I needed a contents list for the issues of your excellent magazine to keep track of articles and where to find them. So I have created an Excel spreadsheet from the contents pages of each issue, which I have converted to a PDF file. Would this be of interest to the wider readership? Here is a link to the folder on Dropbox which contains the contents list and status PDFs: magpi.cc/SPLiNH.

Delaware via email

We do get people asking us about contents lists like this every now and then, so this should be very useful to a lot of

Mag	Issue	Article Title	Author	Category
Mag 12	52	GETTING THE AIR PROJECTS VOICE OUT	BOON ON DALE	LETTERS
Mag 12	53	COOL COOL	Therrie the next generation of coders	Small World
Mag 12	54	THE COOL PROJECTS	How do you contribute projects to Raspberry Pi?	Advertise
Mag 12	55	COOL PROJECTS YOU MUST HAVE	Not too hard to have projects for you to build	Advertise
Mag 12	56	COOL PROJECTS YOU MUST HAVE	Do you have an idea for an article?	Advertise
Mag 12	57	COOL PROJECTS YOU MUST HAVE	How to get your articles published	Advertise
Mag 12	58	COOL PROJECTS YOU MUST HAVE	How to get your articles published	Advertise
Mag 12	59	COOL PROJECTS YOU MUST HAVE	How to get your articles published	Advertise
Mag 12	60	COOL PROJECTS YOU MUST HAVE	How to get your articles published	Advertise
Mag 12	61	COOL PROJECTS YOU MUST HAVE	How to get your articles published	Advertise
Mag 12	62	COOL PROJECTS YOU MUST HAVE	How to get your articles published	Advertise
Mag 12	63	COOL PROJECTS YOU MUST HAVE	How to get your articles published	Advertise
Mag 12	64	COOL PROJECTS YOU MUST HAVE	How to get your articles published	Advertise
Mag 12	65	COOL PROJECTS YOU MUST HAVE	How to get your articles published	Advertise
Mag 12	66	COOL PROJECTS YOU MUST HAVE	How to get your articles published	Advertise
Mag 12	67	COOL PROJECTS YOU MUST HAVE	How to get your articles published	Advertise
Mag 12	68	COOL PROJECTS YOU MUST HAVE	How to get your articles published	Advertise
Mag 12	69	COOL PROJECTS YOU MUST HAVE	How to get your articles published	Advertise
Mag 12	70	COOL PROJECTS YOU MUST HAVE	How to get your articles published	Advertise

people! We are testing out an open-source list for GitHub; it might make it easier to update by the community.

We will admit, though, we sometimes need to go through old issues and find that the odd article is not always listed in the contents, so don’t always treat them as complete. 

DAC+ ADC

Play back your music.
And record it.



Up to 192kHz/24bit

Connect existing audio equipment and stream music over the network

Create a Karaoke box

Use the Raspberry Pi as an effects processor

Record audio

Create a web radio station

www.hifiberry.com

HackSpace

TECHNOLOGY IN YOUR HANDS

THE **NEW** MAGAZINE
FOR THE **MODERN MAKER**



SUBSCRIBE AND
SAVE UP TO
35%
on the cover price



OUT NOW

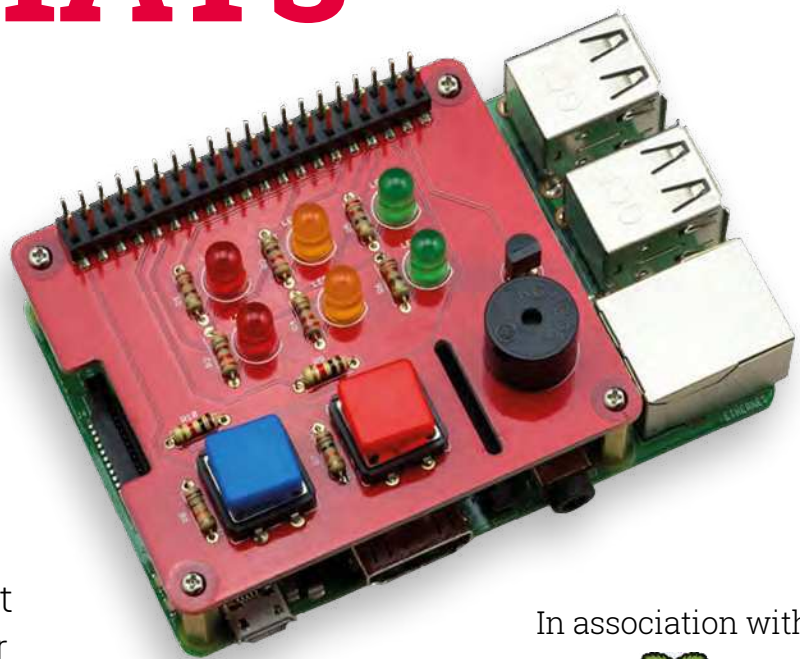
hsmag.cc



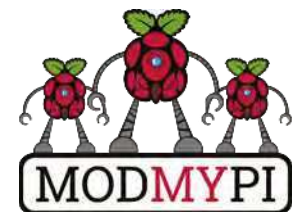
WIN One of ten JAM HATs

We reviewed the JAM HAT in the last issue (magpi.cc/81) and thought it was a really "neat solution to adding buttons, LEDs, and a buzzer to a Raspberry Pi." We've now got a few to give away.

“ The JAM HAT is the perfect add-on board for taking your first steps with the Raspberry Pi GPIO pins. Designed for beginners with GPIO Zero code and examples ”
– ModMyPi



In association with



Head here to enter: magpi.cc/win | Learn more: magpi.cc/jMxZCU

Terms & Conditions

Competition opens on 29 May 2019 and closes on 28 June 2019. Prize is offered to participants worldwide aged 13 or over, except employees of the Raspberry Pi Foundation, the prize supplier, their families, or friends. Winners will be notified by email no more than 30 days after the competition closes. By entering the competition, the winner consents to any publicity generated from the competition, in print and online. Participants agree to receive occasional newsletters from The MagPi magazine. We don't like spam: participants' details will remain strictly confidential and won't be shared with third parties. Prizes are non-negotiable and no cash alternative will be offered. Winners will be contacted by email to arrange delivery. Any winners who have not responded 60 days after the initial email is sent will have their prize revoked. This promotion is in no way sponsored, endorsed or administered by, or associated with, Instagram or Facebook.

NEW

Wireframe

Join us as we lift the lid
on video games



Visit wfmag.cc to learn more

NEXT MONTH | MagPi

EDITORIAL

Editor

Lucy Hattersley
lucy@raspberrypi.org

Features Editor

Rob Zwetsloot
rob.zwetsloot@raspberrypi.org

Sub Editors

Phil King and Nicola King

ADVERTISING

Charlotte Milligan
charlotte.milligan@raspberrypi.org
+44 (0)7725 368887

DESIGN

criticalmedia.co.uk

Head of Design

Lee Allen

Designers

Sam Ribbits, Harriet Knight

Illustrator

Sam Alder

CONTRIBUTORS

Alex Bate, Mike Cook, David Crookes, PJ Evans, Rosie Hattersley, Nicola King, Simon Long, Sean McManus

PUBLISHING

Publishing Director

Russell Barnes
russell@raspberrypi.org

Director of Communications

Liz Upton

CEO

Eben Upton

DISTRIBUTION

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

SUBSCRIPTIONS

Unit 6 The Enterprise Centre
Kelvin Lane, Manor Royal,
Crawley, West Sussex, RH10 9PE
+44 (0)1293 312193
magpi.cc/subscribe
magpi@subscriptionhelpline.co.uk



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

The MagPi magazine is published by Raspberry Pi (Trading) Ltd., Maurice Wilkes Building, St John's Innovation Park, Cowley Road, Cambridge, CB4 0DS. The publisher, editor, and contributors accept no responsibility in respect of any omissions or errors relating to goods, products, or services referred to or advertised in the magazine. Except where otherwise noted, content in this magazine is licensed under



a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported (CC BY-NC-SA 3.0).
ISSN: 2051-9982.

The Best SCIENCE PROJECTS

With
Raspberry Pi

THE MAGPI #83

ON SALE 27 JUNE

Plus!

Make retro games with Pico-8
Rescue old electronics

The best portable Pi projects
Build a mobile cinema robot

Use Google Duo with a doorbell

DON'T MISS OUT!

magpi.cc/subscribe

TWITTER @TheMagPi

FACEBOOK fb.com/MagPiMagazine

EMAIL magpi@raspberrypi.org



It's all about community

Being important is nice, but being nice is important. By **Alex Bate**

As Social Media Manager for Raspberry Pi, I always try to encourage the members of our wonderful community to publish their makes online: write blog posts, create online tutorials, publish videos. We don't mind what you do, so long as we get to share in the excitement of those people using our product, and see what incredible creations you're all building.

But when we ask others to share their creativity publicly, we're also asking them to expose a greater part of themselves to the wider online

world, and, sadly, to the negativity that can often exist across social media platforms, such as Twitter, YouTube, and Reddit.

“ Ensure that the small bubble of the internet in which you exist is a positive one ”

I've always strived to help create an online safe space for our community. While some may feel the need to post negativity and spread anger, I've always aimed to ensure that our small chunk of the internet continues to be a safe, welcoming environment for those who wish to join in with the conversation, and expand their Raspberry Pi knowledge. Whatever

Stay positive

you share with us, we appreciate it. And while I may not dedicate a blog post to your build, I can assure you that I've discussed it aloud, or shared it with the team via Slack or email.

Recently, while interviewing an influencer for my podcast, How Does Social (howdoessocial.com), about her experiences as a YouTuber, I became even more aware of the way in which so many creatives are exposed to negativity online. And while we can continue to tell


ourselves that the motivation of online trolls is often due to jealousy and misinformation, this doesn't lessen the blow when someone posts harsh comments about a project you've dedicated your blood, sweat, and tears to – a project you're infinitely proud of.

And what I want to say is this: please don't be put off. For every troll, every negative comment, there are a hundred people in awe of what you've done. And every time you share your excitement at your

latest creation, we're there in the background, cheering you on.

A little respect

Remember to always protect your mental health when it comes to existing in the online world, and to treat others how you wish to be treated. Take five minutes between writing a tweet and sending it, to ensure you're happy with it. Post positive comments on YouTube, share the good points of your day, save customer service complaints for email and webforms, and ask a friend to read the comment section for you if it gets too much. Ensure that the small bubble of the internet in which you exist is a positive one, and that the message you share online is one of respect and peace. Never feed the trolls. Ignoring negativity is the best way to ensure it doesn't grow – because who continues talking when it's obvious that no one is listening?

Peace and love to you all. Your friendly neighbourhood internet caretaker, Alex. 

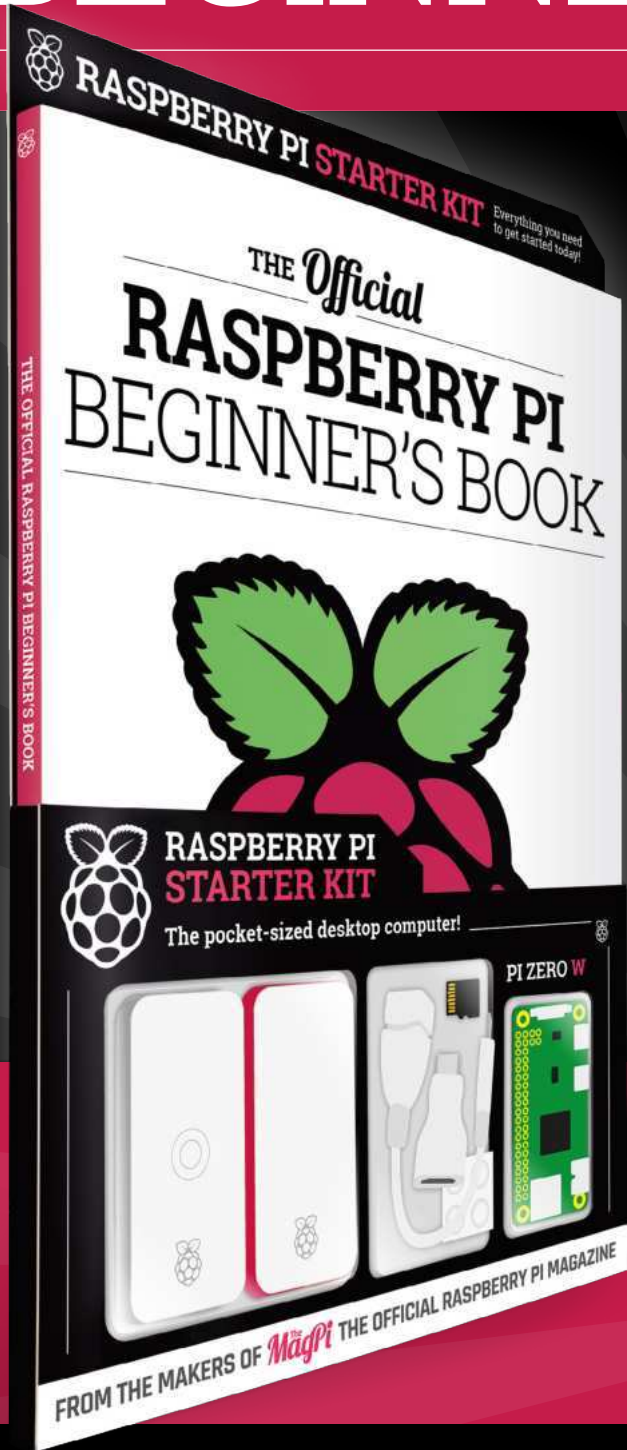
AUTHOR Alex Bate

Social Media Manager for Raspberry Pi. Podcaster and maker.

[@alexjrsocial](https://twitter.com/alexjrsocial)
howdoessocial.com

THE *Official*

RASPBERRY PI BEGINNER'S BOOK

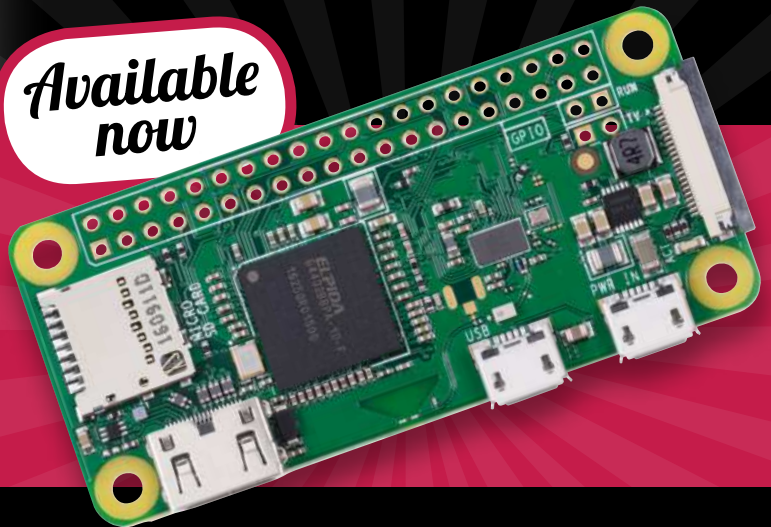


LEARN COMPUTING THE EASY WAY!

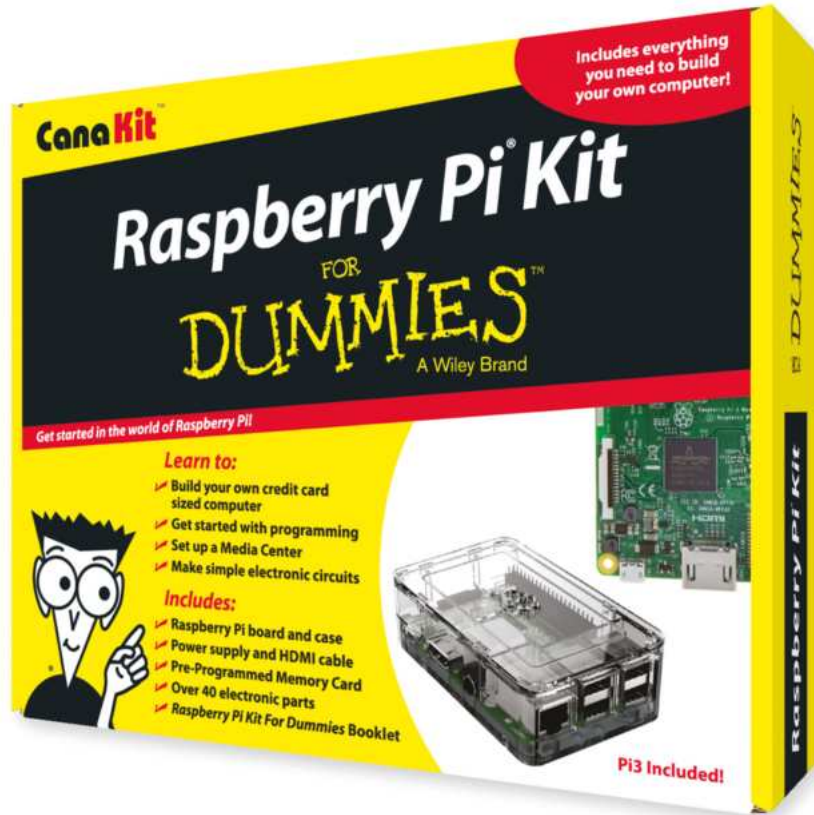
Includes

- Pi Zero W computer
- Official case with three covers
- USB and HDMI adapters
- 8GB microSD card
- 116-page beginner's book

*Available
now*



Buy online: magpi.cc/store



Kit Includes:

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

FOR DUMMIES
A Wiley Brand

Available for worldwide shipping at:

WWW.CANAKIT.COM

Available in Europe through RS Components



\$89^{.99}
US DOLLARS

£69^{.99}
EXCLUDING VAT

Raspberry Pi is a registered trademark of the Raspberry Pi Foundation. For Dummies and the Dummies Man logo are trademarks or registered trademarks of John Wiley & Sons, Inc. Used under license. RS logo is a registered trademark of RS Components Ltd. CanaKit is a registered trademark of Cana Kit Corporation.