BUY IN PRINT WORLDWIDE! magpi.cc/store

# The MagPi

The official Raspberry Pi magazine

Issue 73 September 2018

raspberrypi.org/magpi

## VIDEO GAMES

DESIGN, CREATE & DISTRIBUTE GAMES WITH RASPBERRY PI

Learn video game design

Discover the latest tools

Join game jams & get involved

## LEARN TO CODE BBC TV IS BACK

Rediscover coding with the classic 1980s computer shows

## AIY EDGE REVEALED!

Google's new AI USB dongle for Raspberry Pi

## RACE AI CARS WITH FORMULA PI

Enter the racing competition with a self-driving Raspberry Pi car

## BUILD A TEENAGE DINNER KLAXON

Tired of yelling DINNER at your kids? Technology comes to the rescue

## HOME SECURITY

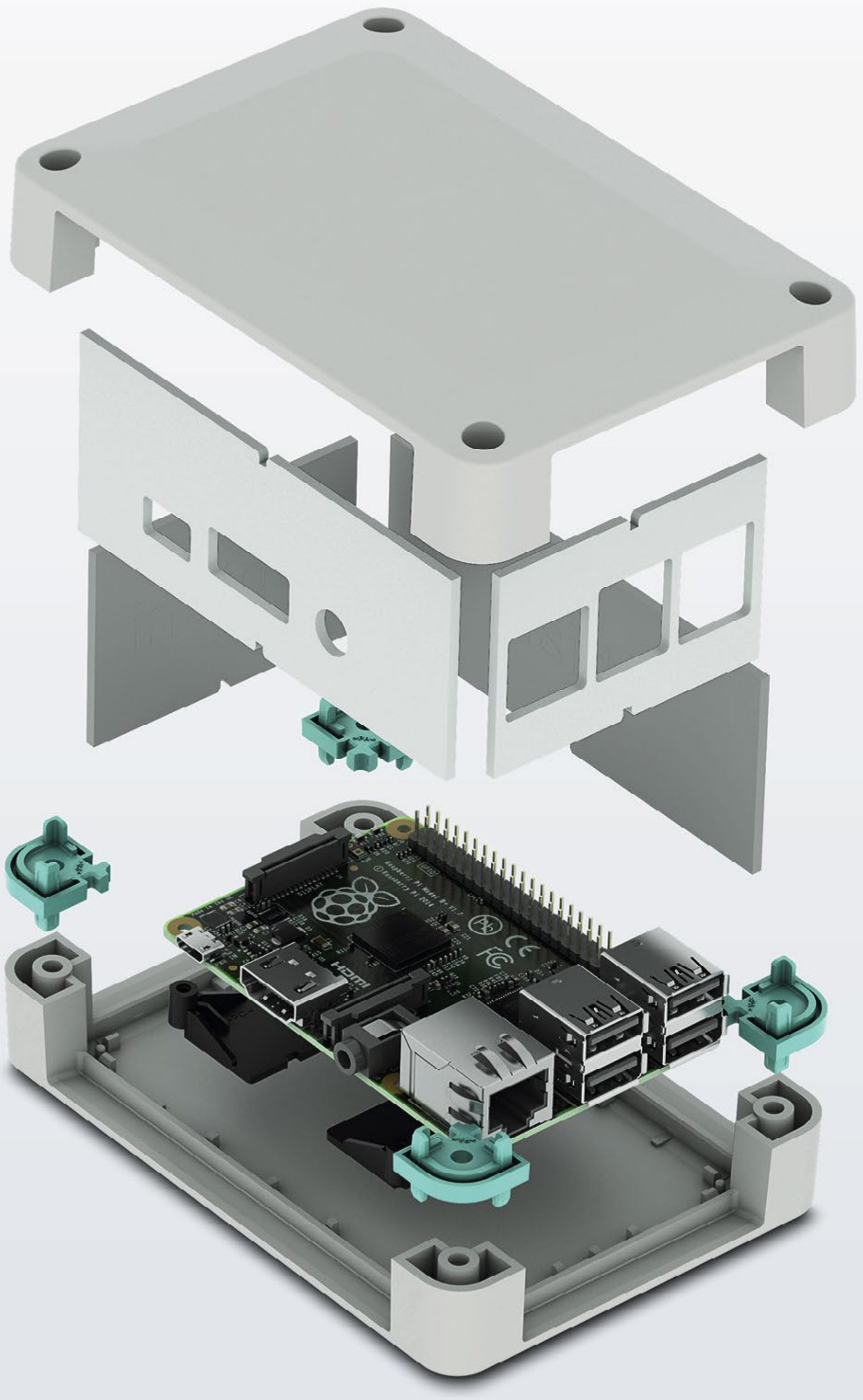
Smart ways to keep the crooks out

## TOP 10 ROBOTS

Discover the best robotics kits

GLOBAL DELIVERY magpi.cc/store

Raspberry Pi PRESS



## Raspberry Pi 3 B+ finds a new home

**Versatile enclosure for Raspberry Pi 3 B+**

The new UCS-RPI Universal Case System is compatible with the recently launched Raspberry Pi model 3 B+. It has pre-milled side walls for easy access to the I/O and power inputs and is available in black or grey and two sizes. Complete with glue dot location posts to secure the single board computer to the case.

For additional information call 0845 881 2222 or visit

**phoenixcontact.co.uk/UCS-RPI**

PHOENIX CONTACT

*INSPIRING INNOVATIONS*

# WELCOME TO THE OFFICIAL MAGAZINE

Making video games is a rite of passage for most programmers. Children of all ages learn through play, and we can think of few better ways to spend the day than making our own video games.

The Raspberry Pi is ideally suited to the creative side of video gaming. Its small, hackable nature makes the Pi a hands-on affair.

Our favourite computer contains enough modern technology to run games like Minecraft while encouraging you to muck around with your own code.

Ever since the dawn of computing, there have been tools and techniques to help budding video game developers. Early coders had to make it up as they went along (you still can), but there's no need to feel so alone these days: there are a lot of tips for designing and making playable games.

Above all else, the Raspberry Pi has a community. So you can share your games with other people and get feedback on them.

We think gaming is an integral part of the computing experience and is especially important for younger and newer coders. So this month's big feature is all about gaming: how to design and create your own games, and share them with other gamers in the Raspberry Pi community.

Let us know what you make!

**Lucy Hattersley**
**Editor**

SUBSCRIBE FROM £5
SEE PAGE 34 FOR DETAILS

## THIS MONTH:

**FIND US ONLINE** raspberrypi.org/magpi

**GET IN TOUCH** magpi@raspberrypi.org

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

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

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

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

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

**CONTRIBUTORS**
**Wes Archer, Brian Beuken, Brian Corteil, Mike Cook, David Crookes, PJ Evans, Rosemary Hattersley, Nicola King, Simon Long, KG Orphanides, Jonathan Pallant, Gary Pickup, Mark Vanstone**

# Contents

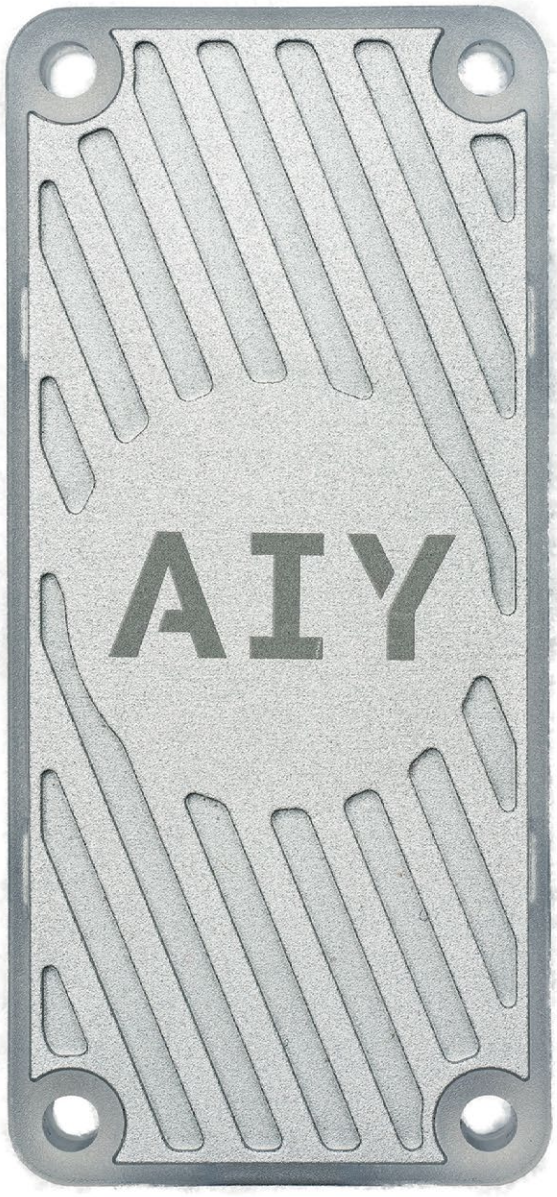
Issue 73 September 2018 raspberrypi.org/magpi

# AIY EDGE TPU ACCELERATOR

Google announces TensorFlow Lite booster for Raspberry Pi

Google has revealed the AIY Edge TPU Accelerator with support for Raspberry Pi. The small USB dongle attaches to a Raspberry Pi to increase the speed of running machine learning (ML) models.

**Below** The stylish Edge TPU Accelerator measures just 65 × 30 mm

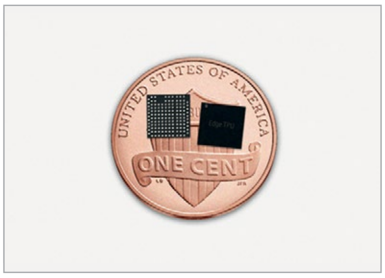


These models use a previously trained neural network to create predictions, such as the next word in a sentence, or what object is in an image. The Edge TPU Accelerator makes it a whole lot faster to perform this prediction on a Raspberry Pi.

The AIY Edge TPU Accelerator is designed to act as a neural network coprocessor for your Raspberry Pi 3 board, massively speeding up the process.

These devices "represent our first steps towards expanding AIY into a platform for experimentation with on-device machine learning," says Billy Rutledge, Google's Director of AIY Projects.

The AIY Edge TPU Accelerator is designed with mounting holes to attach to a Raspberry Pi board and uses a standard USB-C socket to connect. Google says that the device will be available "this fall in the US with other countries to follow." The price is yet to be announced (**magpi.cc/vROTIl**).



**Above** The tiny Google Edge TPU chip is designed to accelerate machine learning inferencing on devices like the Raspberry Pi

The chip itself measures 25 mm square and runs machine learning models locally. The chip provides a vastly improved speed performance over using just a CPU, and at a low power cost.

This first generation of the Edge TPU is designed to run predictions on pre-built models – it does not support training. The models can be trained on desktop computers or much larger machines such as with Google Cloud.

The presence of the TPU will enable a Raspberry Pi to execute previously trained neural network models much faster.

> "We know that performance-per-watt and performance-per-dollar are critical"

## Edge TPU

The Google Edge TPU (Tensor Processing Unit) is a new purpose-built application-specific integrated circuit (ASIC) chip designed to accelerate TensorFlow Lite ML inferencing.

## TensorFlow Lite

The Edge TPU is designed to work with TensorFlow Lite (**magpi.cc/tflite**), a new lightweight ML solution for mobile and embedded devices like the

## AIY EDGE TPU ACCELERATOR SPECIFICATIONS

**ML accelerator:**
Google Edge TPU coprocessor

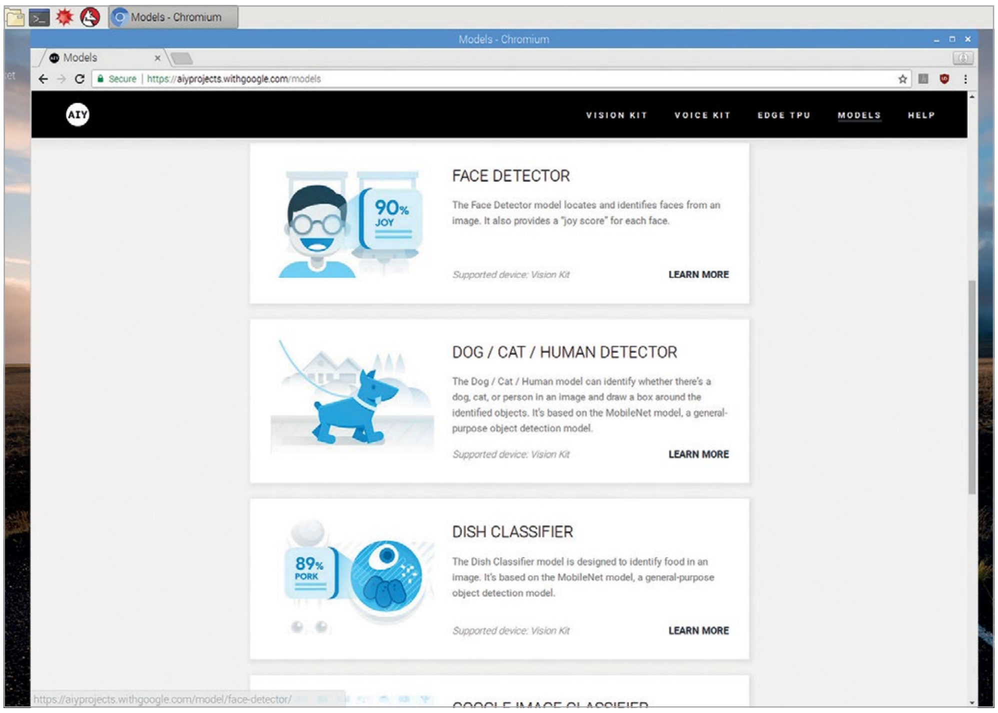
**Connector:**
USB Type-C* (data/power)

*Compatible with Raspberry Pi boards at USB 2.0 speeds only

**Dimensions:**
65 mm × 30 mm

**Supported operating systems (on host CPU):**
Debian Linux, Android Things

**Supported frameworks:**
TensorFlow Lite


Models - Chromium
Models
Secure | https://aiyprojects.withgoogle.com/models
AIY
VISION KIT
VOICE KIT
EDGE TPU
MODELS
HELP
90% JOY
89% PORK
https://aiyprojects.withgoogle.com/model/face-detector/
GOOGLE IMAGE CLASSIFIER

FACE DETECTOR

The Face Detector model locates and identifies faces from an image. It also provides a "joy score" for each face.

Supported device: Vision Kit LEARN MORE

DOG / CAT / HUMAN DETECTOR

The Dog / Cat / Human model can identify whether there's a dog, cat, or person in an image and draw a box around the identified objects. It's based on the MobileNet model, a general-purpose object detection model.

Supported device: Vision Kit LEARN MORE

DISH CLASSIFIER

The Dish Classifier model is designed to identify food in an image. It's based on the MobileNet model, a general-purpose object detection model.

Supported device: Vision Kit LEARN MORE

**Above** Machine learning models for AIY kits are available on the AIY Projects website

Raspberry Pi. It features a set of core operators that have been fine-tuned for mobile devices.

TensorFlow Lite supports a number of models, such as Inception V3 (used for detecting the dominant objects present in an image) and On Device Smart Reply (which is used to make smart replies to text messages – it first appeared on smartwatches). See here for a complete list of supported models: **magpi.cc/tfmodels**. Developers can also create their own models for TensorFlow Lite.

## High performance

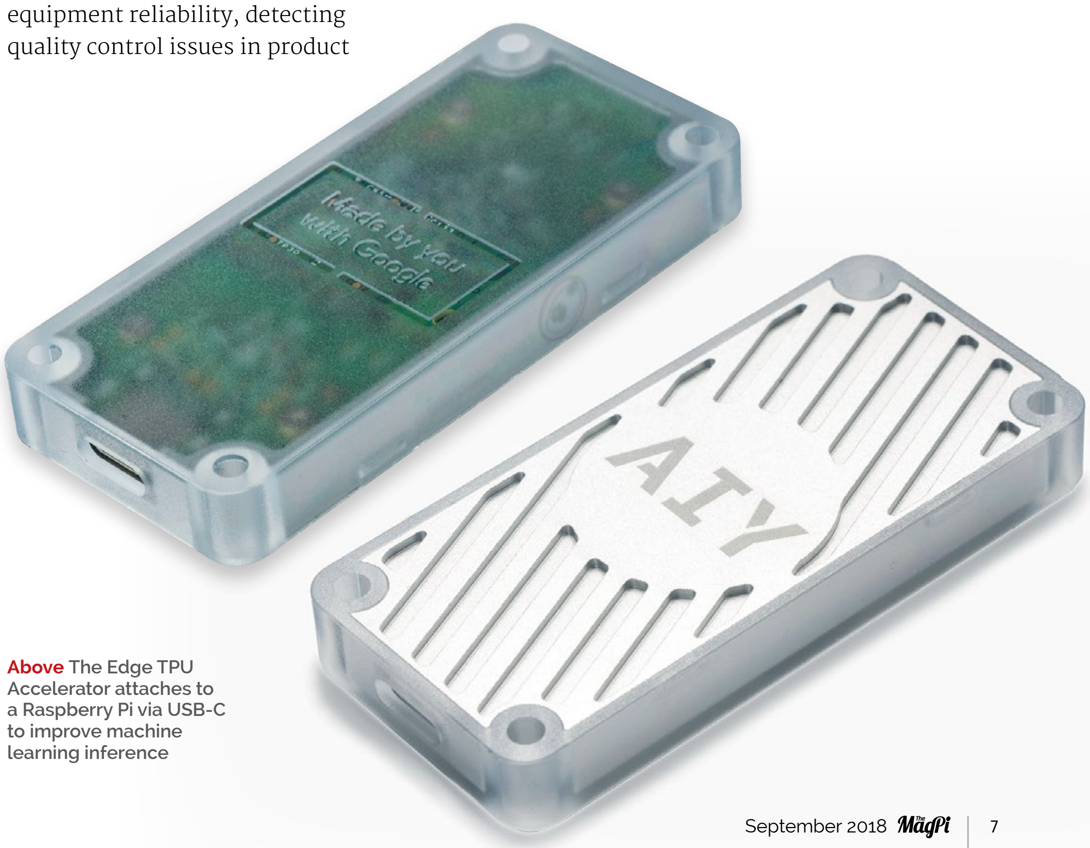
"We know that performance-per-watt and performance-per-dollar are critical benchmarks when processing neural networks within a small footprint," says Billy.

He explains: "The Edge TPU delivers both in a package that's smaller than Lincoln's head on a US penny. It can accelerate ML inferencing on a device or can pair with Google Cloud to create a full cloud-to-edge ML stack. In either configuration, by processing data directly on-device, a local ML accelerator preserves privacy, removes the need for persistent connections, reduces latency, and allows for high performance using less power."

It's early days for machine learning on devices like Raspberry Pi, and it will be exciting to see how small boards with advanced machine learning techniques can be used to solve real-world problems.

Typical examples suggested by Billy include "maintaining equipment reliability, detecting quality control issues in product lines, measuring retail foot-traffic, [and] building adaptive automotive sensing systems." In addition, we're sure you'll all find plenty of unique uses for it.

We're excited by the AIY Edge TPU Accelerator and can't wait to take a look at it. More information can be found on the AIY Projects website: **magpi.cc/edgetpu**.



**Above** The Edge TPU Accelerator attaches to a Raspberry Pi via USB-C to improve machine learning inference

# FORMULA PI ROARS INTO TOP GEAR

Live-streamed races and self-righting cars make for AI thrills for **Rosie Hattersley**

## ENTER FORMULA PI

You can purchase entry to Formula Pi (Winter 2018) races via the PiBorg website (**magpi.cc/oLKiFR**). Entry is £37 and you don't need a robot or any other equipment (all race robots are provided by PiBorg, so everybody uses the same race design – all you need to supply is code).

Along with your entry into the race series, you will receive in the post a Yeti-Lid to decorate. This is returned and attached to your car (so you can identify it in races).

Take a look at the standard race code online at **magpi.cc/hEJNsP**. Entrants edit and update this code to win the race. Code is sent to Formula Pi via FTP and uploaded to a robot. Racers watch the race online with all the other competitors.

"We have some amazing things lined up for the coming seasons," says Tim Freeburn. "Please support the project and help make autonomous racing accessible to everyone!"

Formula Pi has been showing off its latest Raspberry Pi-powered self-driving robot cars ahead of a Kickstarter campaign that aims to take autonomous racing mainstream (see **magpi.cc/jjcVzD**).

Unlike almost every other form of motorsport, this form of track racing isn't prohibitively expensive for ordinary folk to get involved with. Formula Pi race director Tim Freeburn says competitors "do not require a large budget or their own robot to compete, with the cost of entry being about £37 (just $50)." Formula Pi even provides basic example code to get you started, which you can then adapt to improve racing performance.

### Competitive racing

Formula Pi was showcased at the Raspberry Fields festival at the end of June, where cars with the new Raptor chassis were shown cruising round a broadly triangular track. While a great proof of concept for autonomous driving, the smooth running demonstrated at Raspberry Fields only touched on the cars' true abilities. To be really impressed you need to follow the Formula Pi competitive race season. Here, AI cars primed with team-customised code go head-to-head, avoiding collisions, taking the racing line, self-righting, and rebooting in a bid to complete the most laps in ten-minute heats.

*The MagPi* got trackside access to a Formula Pi race at owner PiBorg's Cambridge HQ, where we were able to watch three heats involving five second-generation YetiBorg AI cars.

The races are recorded automatically using multiple cameras around the race track. Contestants around the world log in to watch the live-streamed event at **formulapi.com**, and post-race analysis can be viewed on YouTube at **magpi.cc/KuYzfT**.

Live telemetry on each car is broadcast, along with its track

Competitors design custom Yeti-Lids for their AI cars

**Above** Formula Pi racers must avoid collisions to rack up laps

position. And live commentary is provided by PiBorg developer Claire Pollard (**@thetuftii**). Coder-owners join in with a live chat as the races take place. The overall result is a highly automated global race where code wins the day.

## Distinguishing marks

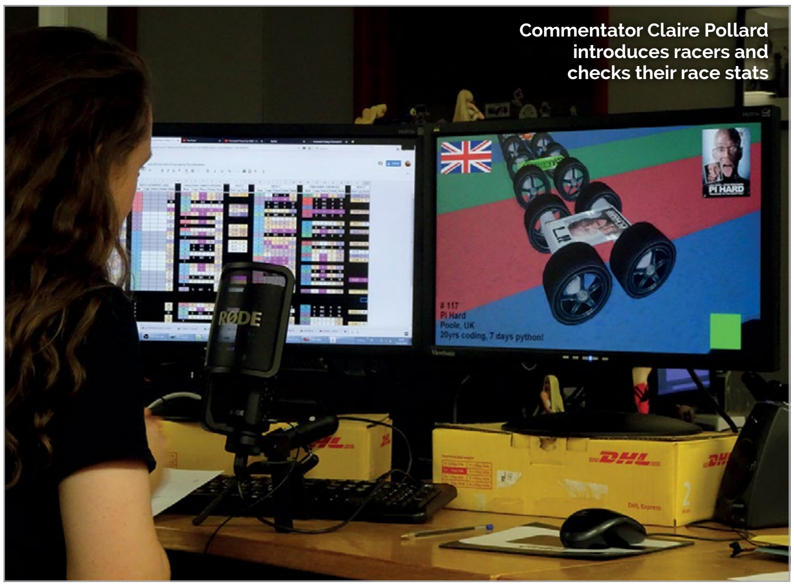
Cars are distinguished by their different coloured lights and by the owner-designed 'lids' – metal tags that screw on to the otherwise identically specced vehicles. Custom code is sent to Formula Pi via FTP or, occasionally, an SD card in the post.

PiBorg and Formula Pi director Tim Freeburn tells us that in the first competitive season of YetiBorg AI racing, entrants simply made their cars "as fast as possible, with results akin to those in Crash Bandicoot." For the 2018 season, competitors had to add crash avoidance to their Raspberry Pi code. The effectiveness of this new rule was in full effect at the Formula Pi track in Earith, just outside Cambridge, where we witnessed groups of five cars dashing round the track at top speed and, mostly, avoiding each other.

As with Formula One, the rules change from season to season. Presently, vehicles that crash or are entangled become obstacles for the other racers. For future races, it's likely that some of PiBorg's club cars will be employed as static obstacles, along with any competitors that develop faults. At present, competitors can initiate a reboot code for 'stalled' cars.

The middle of the race track already has nods to TV show *Robot Wars*, with mini versions of Matilda, Sir Killalot, and the turntable obstacle from Pi Wars (**magpi.cc/ZWvFVH**). In time, these props will become interactive – the windmill's sails already turn – and may become additional obstacles.

> "A highly automated global race where code wins the day"


Commentator Claire Pollard introduces racers and checks their race stats

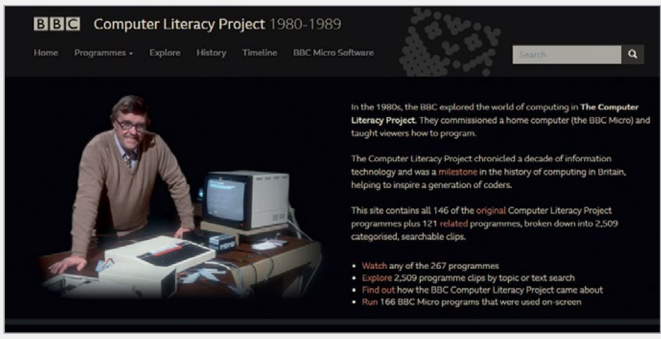
## Faster cars

Autonomous vehicles competing in the 2018 race season are twice the speed of the first generation of Formula Pi cars and can complete a lap in around 15 seconds. The yet-to-be-launched Raptor models are faster still, lapping in around seven seconds. Speed aside, what's likely to really appeal are the customisable moulded chassis, designed by engineers at Rolls-Royce, which make the racers seem like 'real' cars. So if you want to get into competitive racing in which it's your code that gives you the edge, Formula Pi looks like the high 'oct-AI-ne' way to go.

# BBC COMPUTER LITERACY ARCHIVE

The 1980s was a golden era for home computing. Let's explore it all over again, says **Rosie Hattersley**

**Below** The Computer Literacy Project Archive brings together all of the original 146 CLP shows, plus 121 related programmes



In the 1980s, the BBC explored the world of computing in **The Computer Literacy Project**. They commissioned a home computer (the BBC Micro) and taught viewers how to program.

The Computer Literacy Project chronicled a decade of information technology and was a milestone in the history of computing in Britain, helping to inspire a generation of coders.

This site contains all 146 of the original Computer Literacy Project programmes plus 121 related programmes, broken down into 2,509 categorised, searchable clips.

- Watch any of the 267 programmes
- Explore 2,509 programme clips by topic or text search
- Find out how the BBC Computer Literacy Project came about
- Run 166 BBC Micro programs that were used on-screen

If you've ever wanted to know more about the history of British home computing, a whole treasure trove of interviews, TV shows, programming code, and photos has been assembled for your delectation and inspiration. The BBC has launched an archive of its Computer Literacy Project (CLP).

This delightful archive of home computing and information technology (**magpi.cc/nqHDem**) includes all 146 TV programmes from a ten-year endeavour that began 36 years ago.

"The BBC Micro was a major influence on the creation of the Raspberry Pi"

Interviews with heroes of technology and personal computing – including Microsoft co-founder Bill Gates, Apple's Steve Jobs and Steve Wozniak, and Apricot's Roger Foster – are all now accessible from the BBC's Computer Literacy Project Archive.

Launched back in 1982, the BBC's Computer Literacy Project was intended to inspire and encourage a whole generation of coders and home computing enthusiasts. The impetus for its launch was a critical *Horizon* documentary in 1978 that suggested a 'lack of awareness and competitiveness' in the UK that meant Britain was likely to miss out on the social and economic benefits of microelectronics.

Such was the success of BBC Education's CLP and the BBC Micro home computer (among others) that, says Hermann Hauser, co-founder of Acorn Computers, "Britain [was] the most computer literate nation on Earth at the time and with the BBC computer created a generation of UK programmers who have become leaders in their field."

The BBC Micro was a major influence on the creation of the Raspberry Pi, which was designed to capture the coding heyday of the 1980s and increase the number (and quality) of students applying to study computing at Cambridge University.

Eben Upton, Raspberry Pi co-founder, told The Centre for Computing History (**magpi.cc/EOMrxz**): "The first computer I owned was a BBC Micro." The initial idea for the Raspberry Pi arose from talking about redoing the BBC Micro (as a response to MIT planning an Apple II clone).

## Computing history

The original BBC TV shows are bound to fascinate anyone investigating the UK personal computing evolution for the first time, while evoking nostalgia in those who remember the 1980s BBC Computer Literacy Project.

## INSTALL BBC BASIC

Back in 1982, the Acorn-built BBC Micro computer launched as a means of allowing home users to explore programming and coding. Homes and schools quickly bought into the personal computing revolution, with many of today's most ardent computer fans citing the BBC Micro and its BASIC language as their first programming experience.

You can easily install BBC BASIC on your Raspberry Pi. Open a Terminal and enter:

```
sudo apt-get update && sudo apt-get upgrade
sudo apt-get install brandy
```

Then enter **brandy** at the command line to open the Brandy Basic V Interpreter. Note that you will need to use **CAPS LOCK** to enter BBC BASIC code, as you need to type every command in capitals.

The BBC Micro project was instrumental in teaching computing in British schools. (Credit: Marcin Wichary, Wikimedia Commons)

The archive came about due to the difficulty for academics and technology historians to easily access treasures from the CLP. While the TV footage and interviews have been recovered, radio shows broadcast under the CLP banner appear to have been lost.

Many of the TV shows make for interesting viewing from a present-day perspective. In May 1980, *The Silicon Factor* (**magpi.cc/crbMPk**) getting a paint-spraying robot in a car factory to memorise the actions involved in writing a name on a sheet of paper was ground-breaking. "The memory is recording every single unsteady movement of the pen," explained presenter Bernard Falk before showing the robot accurately reproducing his writing.

Meanwhile 1988's *Electric Avenue* issued rather prescient warnings about data sharing and theft: **magpi.cc/JziPxO**.

Mainly, however, the archive is about showing the possibilities and ideas of digital pioneers. As with the original CLP resources, BBC chief technology and product officer Matthew Postgate says the Computer Literacy Project Archive is "a unique resource for teaching and learning that will hopefully encourage a new generation of computer users."

## RETRO COMPUTING

Discover how to turn a Raspberry Pi into a functioning BBC Micro (and lots of other old computer systems) in issue 67 of *The MagPi*: **magpi.cc/67**.

DINrPlate™

The simple way to mount your Pi!

- Industrial DIN rail mount
- Open frame for better airflow
- Integrated USB strain relief

www.DINrPlate.com

# ASTRO PI RETURNS TO THE STARS

Take your code to the International Space Station and beyond, urges **Rosie Hattersley**

The European Astro Pi Challenge, an ESA Education project run in collaboration with the Raspberry Pi Foundation, is gearing up for its next mission. From 12 September until 26 October, Astro Pi will be accepting applications from space fans keen to impress ISS astronauts with their ideas and coding prowess. Part of the Astro Pi Challenge, Mission Space Lab is a four-phase experiment aimed squarely at secondary schools and older primary school pupils.

The European Astro Pi Challenge is an annual science and programming challenge where student-written programs are run on the International Space Station. Last year it had 6800 participants for its two missions: 1500 for Mission Space Lab and 5300 for Mission Zero, the majority from secondary school. This year, the European Astro Pi Challenge aims to attract at least 7500 participants.

## Get your code into space

The aim of the European Astro Pi Challenge is to encourage schoolchildren to conduct scientific investigations in space by writing computer programs that will be run on Raspberry Pi computers aboard the ISS.

For Mission Space Lab, teams first submit their experiment ideas – no code is required yet. The teams with the best ideas are then sent an Astro Pi kit including a Raspberry Pi with both an infrared camera and a Sense HAT, plus a microSD card and instructions on (optionally) 3D-printing your own case.

“Astro Pi will be accepting applications from space fans keen to impress ISS astronauts”

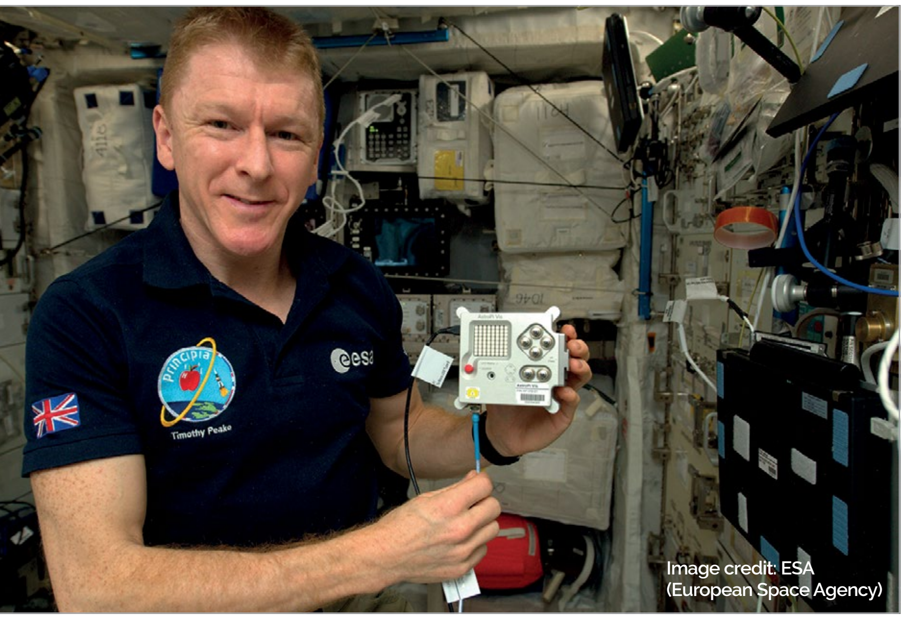


Image credit: ESA (European Space Agency)

Once your team’s code is approved, it is sent up to the International Space Station where your experiment is run on the ISS Astro Pi computers for three hours. Participating teams receive a download of the data for their experiment and a photo of where the ISS was when their experiment was taking place. Finally, students write up a scientific report about the experiment they conducted.

The teams submitting the ten best entries will receive a special certificate.

The Astro Pi Challenge is open to entrants from ESA member and associate member states. For more details, visit **astro-pi.org**.

**Left** Astronaut Tim Peake using a Raspberry Pi aboard the International Space Station

**CHRIS JOHNSON**

Chris is an electrical engineer who normally designs high-speed data acquisition systems for industrial and scientific research applications.

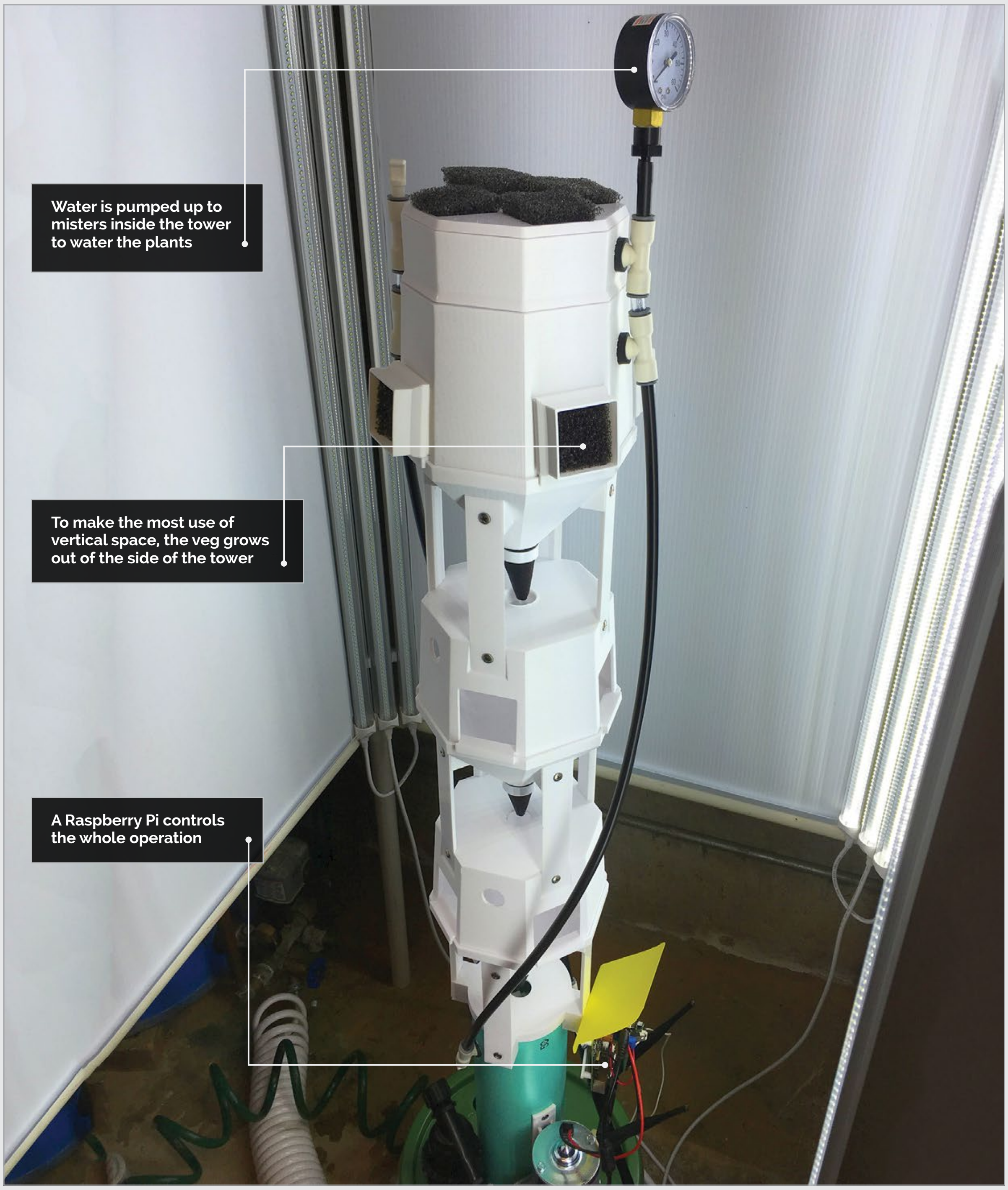
**magpi.cc/TLhRxH**

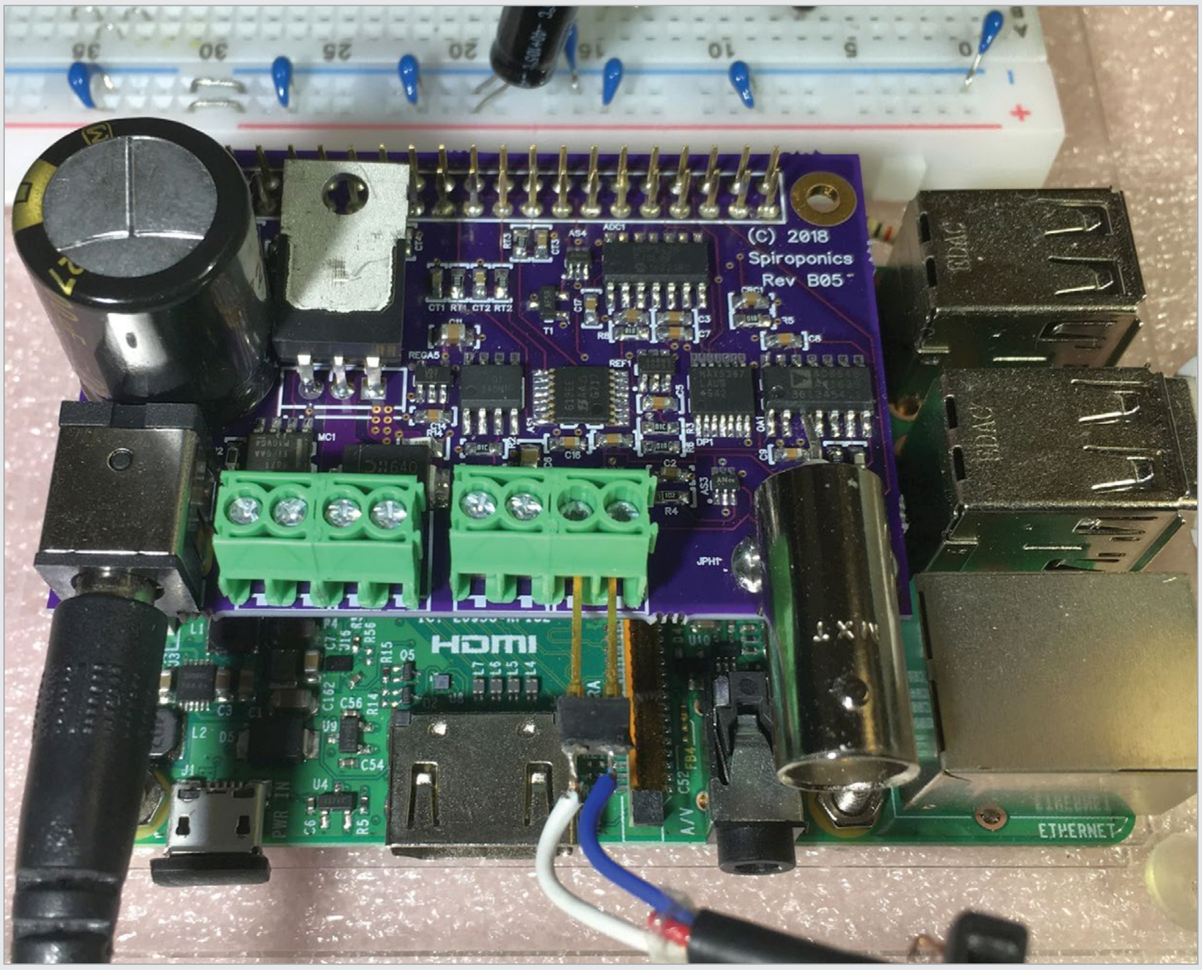
# LITTLE GREEN TOWER

An aeroponics tower that uses a Raspberry Pi to grow some tasty veg

**Quick Facts**

- The tower has been five years in the making
- The project uses a custom HAT board
- It's what's known as a 'vertical farm'
- Water is delivered to the plants as mist
- The system repurposes a motor for door locks, a water pump from an RV, and more


Water is pumped up to misters inside the tower to water the plants
To make the most use of vertical space, the veg grows out of the side of the tower
A Raspberry Pi controls the whole operation



**Left** The SpiroBoard, created by Chris for the project, monitors pH, electroconductivity, air temperature, and water temperature, as well as controlling a motor for the pump

**Below** It's a long way from soil and a watering can, but you should just taste the results

As healthy eating becomes more and more popular, a greater number of people want to grow their own vegetables. The problem is, not everyone has a nice garden or allotment to grow their own stuff – enter hydroponics and, specifically in Chris Johnson's case, aeroponics: the process of growing plants in an air or mist environment.

"For the last five years I have been designing and refining an aeroponics tower system that is particularly well suited to growing leafy green vegetables," Chris tells us. "The goal is to design a computer-controlled, modular, compact, and low-cost aeroponic system that is easy to replicate."

The idea came to him while browsing through some 'particularly bad lettuce' while shopping. "I had always been interested hydroponics," Chris explains. "The consistently poor quality of the lettuce finally provided the push to begin the project."

> "The consistently poor quality of the lettuce finally provided the push to begin the project"

All the years of work have paid off and, as you can see overleaf, Chris has been growing plenty of leafy green veg.

"A neighbour told me that the kale was the best that they had ever had," Chris says. "I personally am not a kale fan,

**Below** Who would have thought such an industrial setup could create something so tasty?

so I'll have to take her word for it. The green leaf lettuce is far superior to what I was getting in the grocery store. The best part is that you pick leaves as you need them and the rest of the plant just keeps on growing."

Chris says that the main issue still to overcome is algae growth in the water. "I am working to mitigate that with different plastic and also different fertiliser.

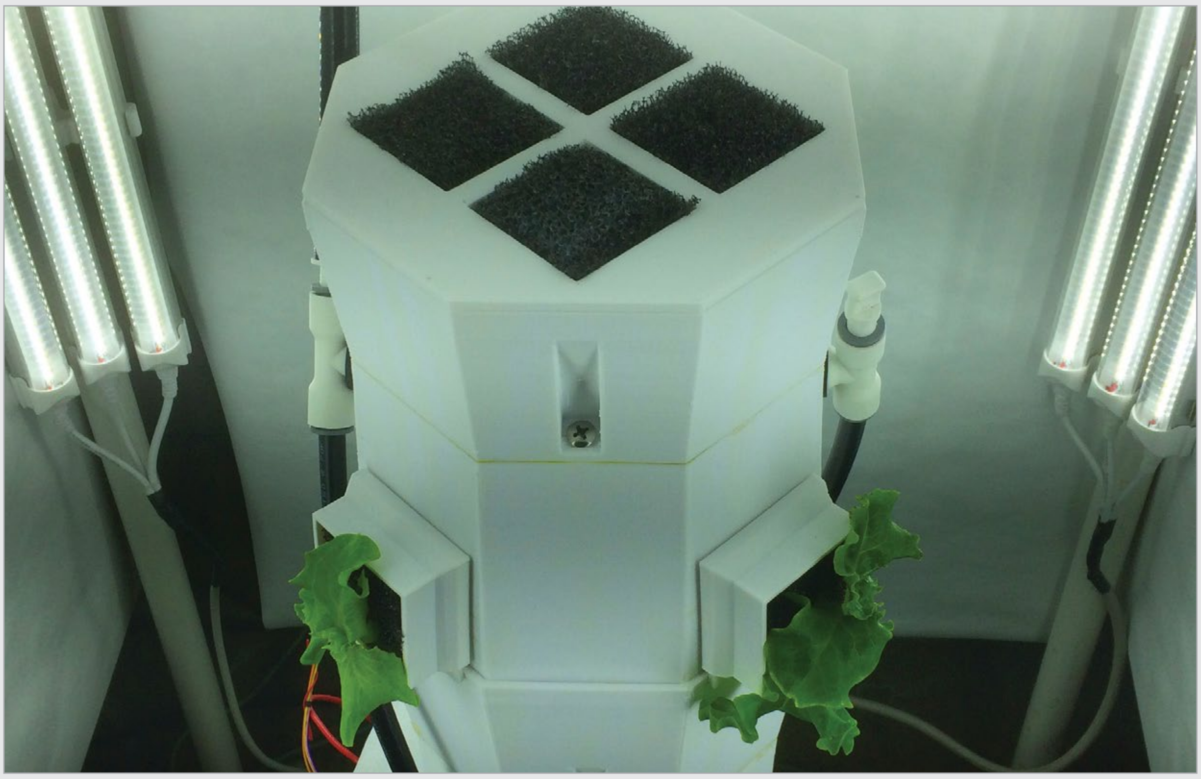
> To reduce algae, the plastic should be opaque so that the algae inside the system gets no light to grow

Green-leaf vegetables need copper and copper kills algae, so I will soon be testing a fertiliser that contains the required trace amounts of copper. To reduce algae, the plastic should be opaque so that the algae inside the system gets no light to grow. Standard white filament, needed for its light reflectivity, is not opaque enough. I am looking to get a filament extruder so that I can make my own more opaque filament."

This will also reduce the cost to reproduce the system: "[It's] printed with PETG, the same thing used to make two-litre drink bottles, since it is food-safe; same as all the hoses and fittings. The water pump is actually a standard RV/caravan water pump."

A Raspberry Pi 3 powers the system, with regular upgrades made to the Pi part of it. "I use the webiopi web interface program to provide the backbone for the system and the control software is written in Python," reveals Chris. "I use the pigpio library for the I²C I/O and to control the waveform timing for a custom controller board that I designed to work with the Pi."

You can find out more details on the Hackaday project page, which Chris is in the process of updating at the time of writing: **magpi.cc/RarKVA**.



**Above** After four weeks, lettuce can be seen growing well!

**Left** It's more spaceship survival pod than English country garden, aesthetically

**ALEX SCHWARZ**

Alex Schwarz is an amateur radio enthusiast with a particular interest in space weather.

**magpi.cc/uldOgt**

# SPACE WEATHER STATION

Alex Schwarz uses the Raspberry Pi as part of an effort to better observe and predict the effects of space weather. **David Crookes** grabs an umbrella

**Quick Facts**

- The Pi helps with space weather predictions
- RF-Seismograph monitors up to six different bands
- It checks the NOAA space weather alert system
- 24-hour monitoring is made possible
- Measurement graphics are uploaded to a web server

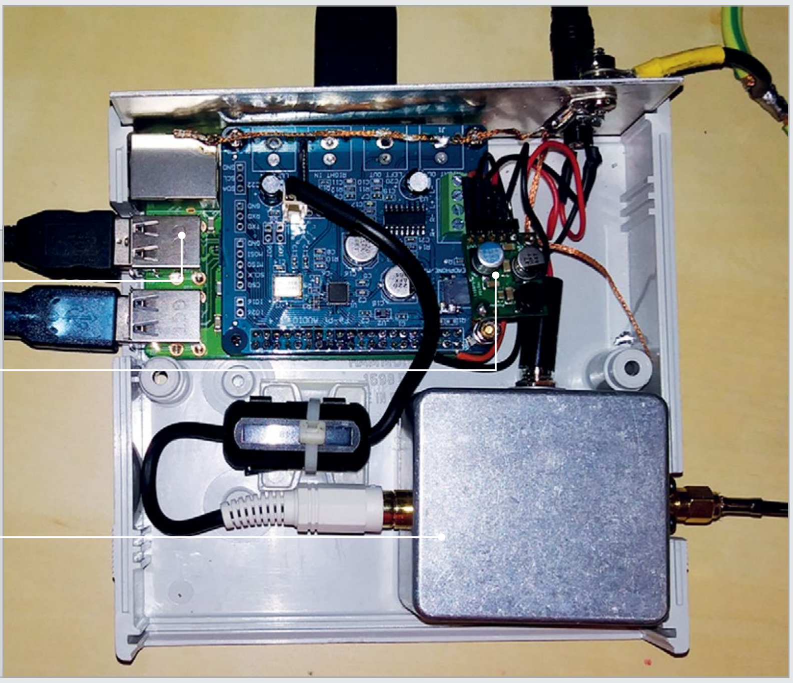
If you want to know what the weather is like, then some would suggest you look out of the window. But that only works if the weather you're after is immediately outside. Should you want to know what's happening elsewhere – in space, perhaps – then you'll have to get a little more technical.

Alex Schwarz has done just that, developing a real-time space weather station which he now runs on a Raspberry Pi 3. It makes use of two well-established amateur radio software packages called the MDSR and RF-Seismograph, as well as some serious hardware including a huge antenna that can be many metres in height. These allow for real-time spectrum analysis and long-term propagation monitoring by detecting particular noise changes that point to specific happenings.

## Listen up

To explain, propagation monitoring detects the behaviour of radio waves as they travel from one point to another in various parts of the atmosphere. As such, the idea is that the weather station is able to 'listen in' for changes in natural space phenomena such as the solar wind and the Earth's magnetic field.

This is then interpreted by the Pi and its installed software, allowing a graphical representation of the weather to be noted. By constantly monitoring the input, it's possible to view patterns and match them to phenomena. So far, the space weather station has picked up on a build-up of energy in the troposphere; at least two hours before the storm hit, it was possible to view the extent of the lightning strikes. It has also detected propagation from meteorite trails. "Now we know how a meteorite



The Raspberry Pi 3 is fitted with a Fe-Pi Audio V1.4 24/16-bit sound card

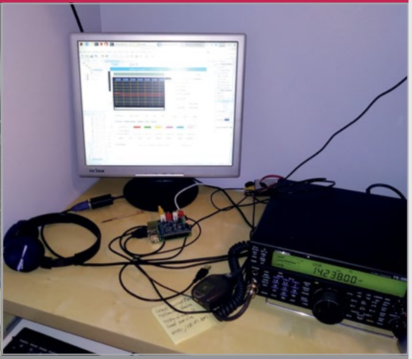
To simplify the setup of the unit, a Buck power regulator is implemented to reduce the 13.8 V to 5 V for the Pi

The LIF-2016 converter allows an existing transceiver to be used, ensuring the signal can be processed using the sound card

## RUNNING THE WEATHER STATION

**>STEP-01**
**Connect to an antenna**
The space weather centre seeks to record local band conditions at all times. In order to do so, it needs to be hooked up to an antenna. Large ones are less prone to interference.

**>STEP-02**
**Monitor background noise**
The waves are picked up by the transceiver, filtered, and sent to the Pi where the RF-Seismograph records the background noise of the connected antenna via the sound card.



**>STEP-03**
**Processing the sound**
The sound is processed by the software on the Raspberry Pi, ready for presenting on screen and it's then possible to pick out the characteristics of various different types of space weather.

looks on the RF-Seismograph and it will enable us to spot more, even during the day," Alex says.

It really is an impressive feat, especially considering Alex and his fellow amateur radio enthusiasts wrote the RF-Seismograph software themselves. "We believe it has great potential to increase the understanding of the ionosphere [the ionised part of Earth's upper atmosphere] and how it protects life on this planet," Alex tells us. "It was originally designed to measure the solar eclipse last year, but in the process we discovered the software is also perfect to measure and show how short-wave radio propagation changes over time."

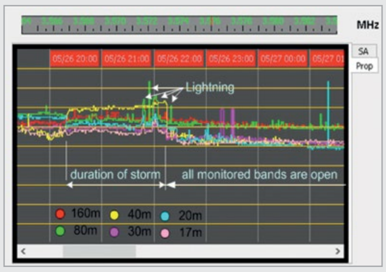
> "It was originally designed to measure the solar eclipse"

### Sounding off

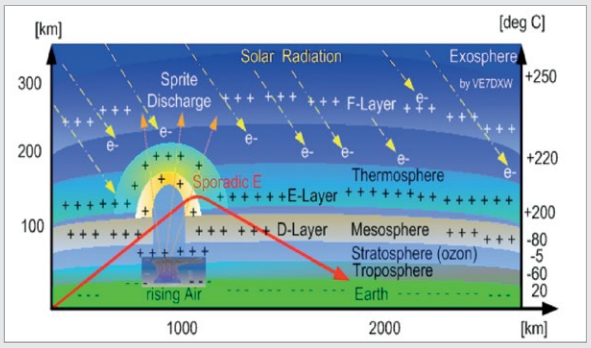
The Raspberry Pi has certainly come to the rescue. Initially, the software had been running on Windows and Linux, but Alex and his friends wanted a way of lowering costs. "The Raspberry Pi 3 is fast enough and so we decided to move all our software over to Raspbian," Alex says of the switch. "The RF-Seismograph runs a little slowly on the Pi so there's room for improvement, but it's a lot simpler and cost-effective in comparison to PCs. The OS is also stable, it's easy to install, and the add-on hardware looks classy."

The hardware for this project included a sound card with stereo line input for picking up the sound waves that are created in the receiver. By scanning the receiver across six different short-wave bands, changes can be monitored and identified. In order to get the raw signal from the receiver, the LIF-2016 converts radio waves to sound waves that the audio card can process. The higher the quality of the sound card input, they figured, the better the results for the received and demodulated signal.

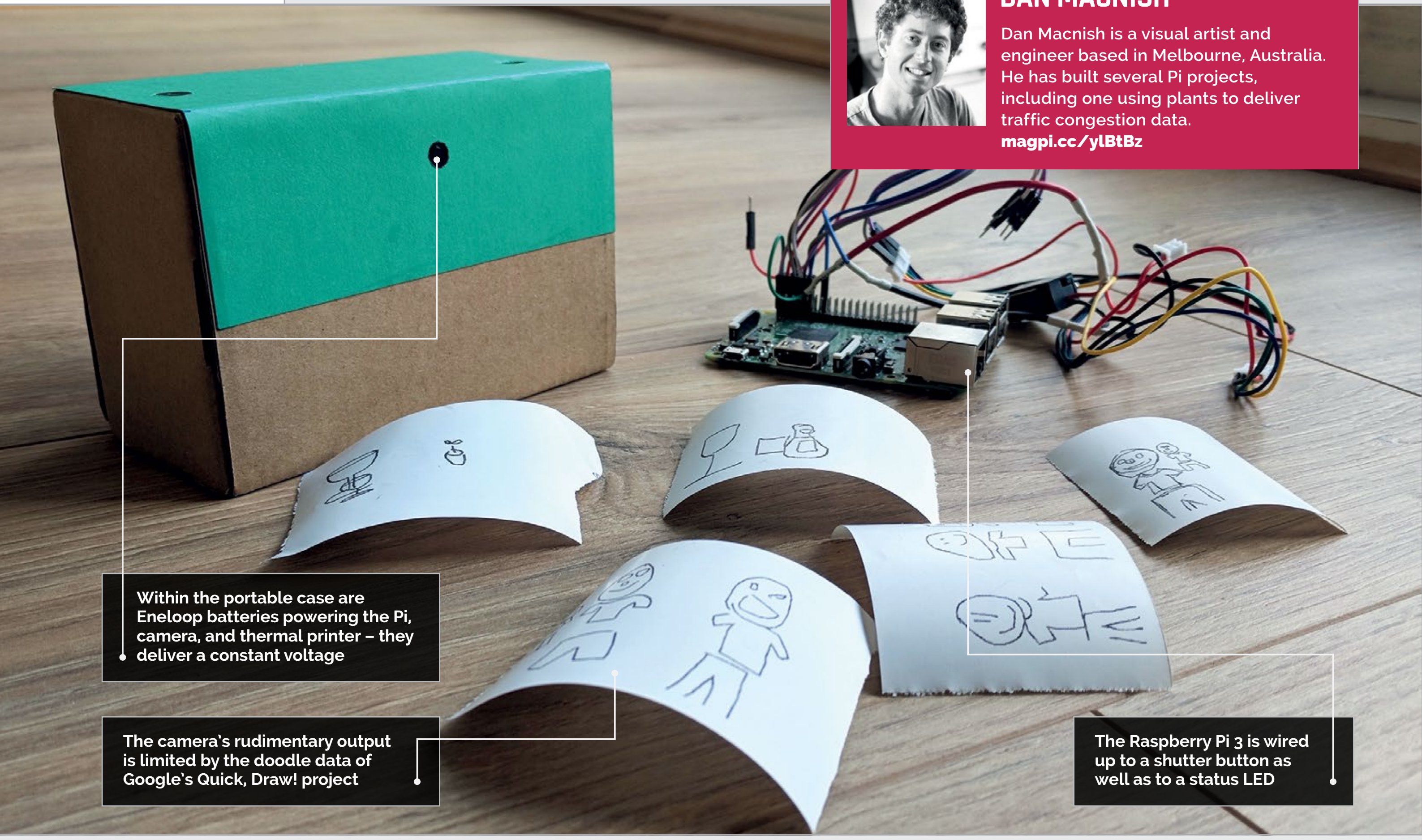
For convenience, a seven-inch touchscreen was also added and the whole setup was connected to the transceiver – the device that connects to a large shortwave antenna and picks up the signal for the Pi to process. "The software is capable of sharing this information with the world by including an FTP server that uploads every ten minutes to the web," Alex adds. "So that is how we keep our website automatically current. It's certainly been worthwhile."



**Above** This image shows the results of a thunderstorm as seen through the space weather centre, with the lightning spikes clearly visible



**Above** A diagram showing the different layers of Earth's atmosphere detectable by the space weather station



Within the portable case are Eneloop batteries powering the Pi, camera, and thermal printer – they deliver a constant voltage

The camera's rudimentary output is limited by the doodle data of Google's Quick, Draw! project

The Raspberry Pi 3 is wired up to a shutter button as well as to a status LED

## DAN MACNISH

Dan Macnish is a visual artist and engineer based in Melbourne, Australia. He has built several Pi projects, including one using plants to deliver traffic congestion data.
**magpi.cc/ylBtBz**

# DRAW THIS

Dan Macnish wants to help people overcome fears about deep learning and artificial intelligence by using a camera that turns photos into cartoons. **David Crookes** strikes a pose

### Quick Facts

- Draw This uses a neural network to recognise objects
- Images are matched with doodles from Quick, Draw!
- The doodles are printed like on a Polaroid camera
- The Quick, Draw! dataset costs nothing to download
- The entire dataset takes up just 5GB of space

We all know how cameras work. You line up your shot, fiddle with the focus, set the shutter speed, and – flash! – you get a perfect image of whatever is in front of you. With Dan Macnish's camera, however, the process and results are a little different. Instead of outputting a faithful photograph, it attempts to turn what it sees into a cartoon doodle, instantly spitting it out on to thermal paper like a quirky Polaroid picture.

To do this, it makes use of two key components: a neural network and the dataset produced by Google's online game Quick, Draw! (**quickdraw.withgoogle.com**). The game is similar to Pictionary in that it suggests an object to sketch, with the twist being that it then uses AI to predict what you're attempting to scribble. The result has been more than 50 million doodles, and Dan's device seeks to match one of them with whatever the neural network reckons has just been snapped.

It's all rather eye-catching. "I got the idea while experimenting with some of the amazing open source research into neural networks," Dan tells us. "There are some great projects coming out of this research, but many of them are focused on neural networks themselves, rather than simple applications for these networks. So my project originated as a super-fun, simple application for this research, to be enjoyed by all sorts of different people – not just engineers and developers."

## Cartoon camera

As such, Dan was keen to keep things simple, fun, and whimsical, and he soon got down to work. He began with some basic sketches and ideas focusing mainly on what the user experience would be. "Through these initial sketches, I set on the idea of a Polaroid camera that draws cartoons, without ever showing the original image," he says. "I particularly focused on the emotions and feelings that people might have: their surprise when seeing a cartoon and the excitement of not knowing what it would look like."

At the same time, he began prototyping the software using Python and Jupyter Notebook on his laptop. He ran a pre-trained network over a few photographs before manually browsing the Quick, Draw! dataset and hand-selecting images which he then copied and pasted together in Paint. "These quick prototypes led me to flesh out the software and I moved on to an integrated development environment (IDE). Once I was getting good results on the laptop, I transitioned the code to the Raspberry Pi."

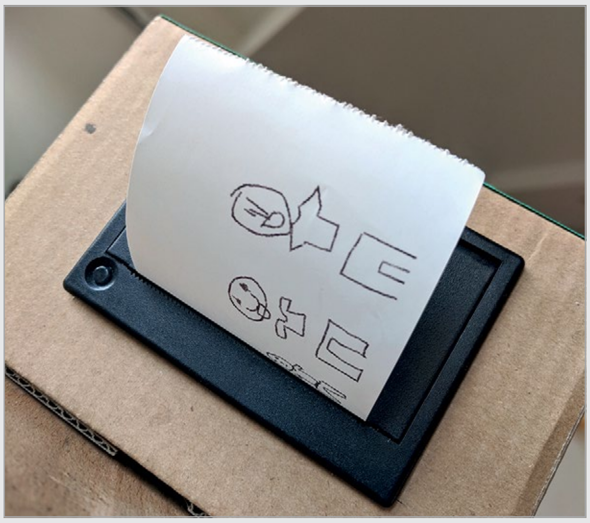
"My project originated as a super-fun, simple application"

## Pi-cture this

At the heart of the device is an internet-connected Raspberry Pi 3 running Raspbian Stretch on a 16GB card, with a v2 Camera Module, outputting to a thermal printer, all housed in a simple cardboard casing. When an attached shutter button is pressed, the software interprets what it sees as one of 345 doodles and outputs them.

"The neural network I used can only recognise a small fraction of the categories in the Quick, Draw! dataset," Dan explains. "So there are still many possibilities for using this dataset that have not yet been imagined."

Even so, the response has been hugely positive. "People love the concept and it's great seeing their reactions to some of the funny cartoons that appear," he says. "One of the most surprising things is that even when the camera prints something completely incorrect, people often interpret the cartoon in a way that makes it seem correct after all. It's quite entertaining."



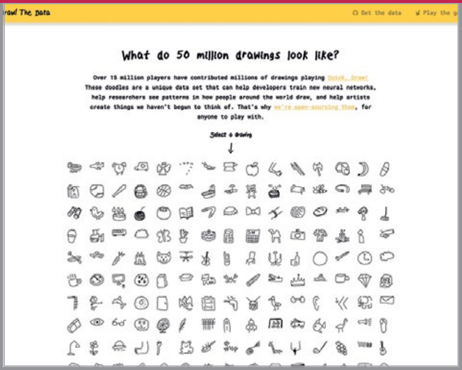
**Above** Some results are stranger than others. Dan's food in a Japanese cafe was turned into an American hotdog, while some selfies have replaced humans with dogs

**Below** Dan used some efficient Python code released by Google to interface with the individual drawings

## CARTOONIFY THE WORLD

### >STEP-01
**Press the button**

Although there is a lens at the front of the casing (a v2 Camera Module), there is no viewfinder around the back. You simply power up, aim, and press the shutter button on top.



raw! The Data

Get the data   Play the g

What do 50 million drawings look like?

Over 15 million players have contributed millions of drawings playing Quick, Draw! These doodles are a unique data set that can help developers train new neural networks, help researchers see patterns in how people around the world draw, and help artists create things we haven't begun to think of. That's why we're open-sourcing them, for anyone to play with.

Select a drawing

↓

### >STEP-02
**Time for processing**

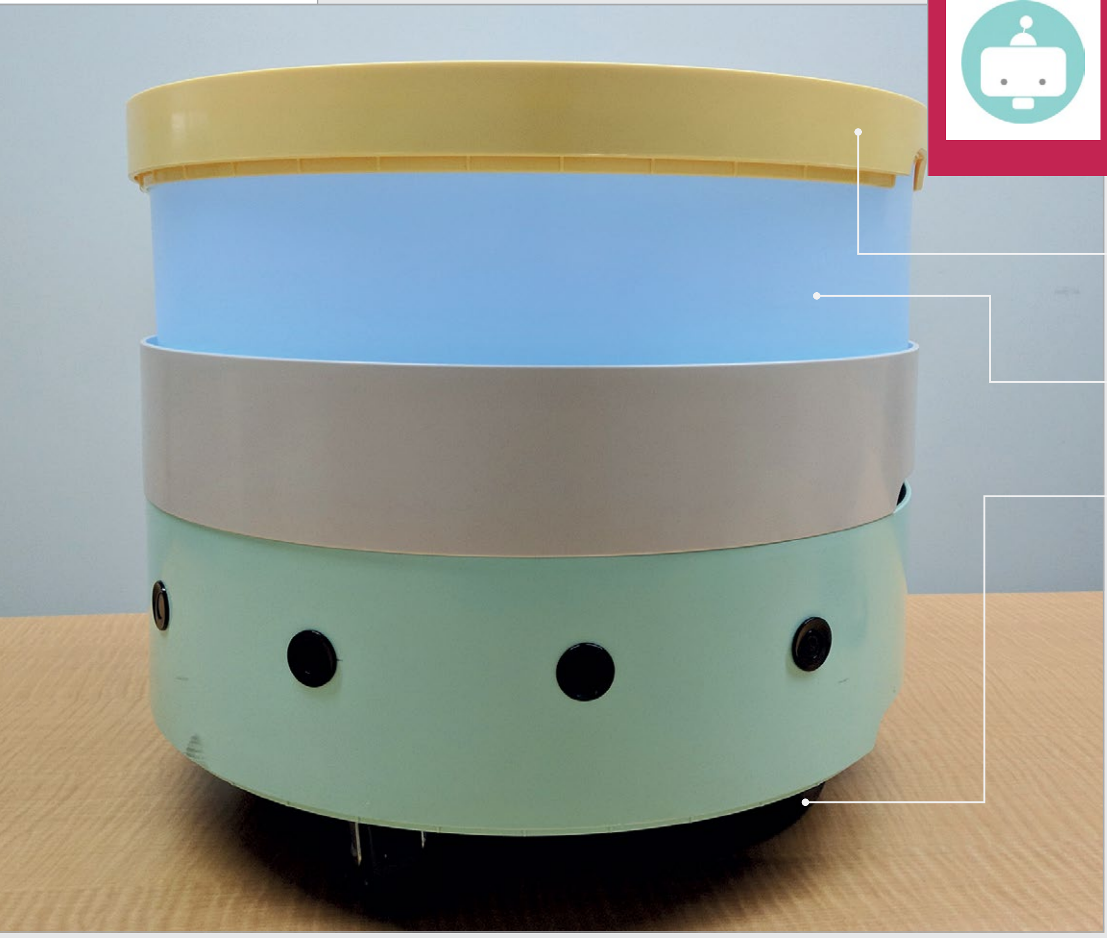
When the shutter button is pressed, the status LED lights for two to three seconds as the Raspberry Pi processes the image. The Quick, Draw! dataset covers hundreds of objects, from forks to faces, beards to bears.

### >STEP-03
**Printing a picture**

Dan says the lack of a screen and the inclusion of a thermal printer enhances the experience. "There is a fantastic feeling of anticipation as the image slowly appears from the printer," he says.

**ISABELLE SIMOVA**

Isabelle enjoys tinkering with electronics, but this is the first project she has open-sourced.
**magpi.cc/CDEbOq**


An IKEA tabletop; other table sections were upcycled for the body
An LED strip provides animated lighting effects
Hoverboard-sourced wheels are supplemented by castors

# HOVERBOT

A robot made from a hoverboard and an old table? **Nicola King** investigates the fascinating world of upcycled bots...

**Quick Facts**

- Hoverbot cost only around $250 to build
- It can run for 4+ hours on a battery charge
- A 60-LED strip is used for illumination
- Hoverbot features a webcam and speaker
- Code and build details are at **magpi.cc/PMnvOi**

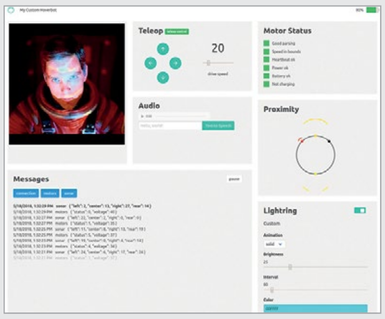
While countless Raspberry Pi robots have been created, most of them are on the small side. Hoverbot, however, bucks this trend and comprises upcycled parts including a hoverboard and an IKEA table.

Maker Isabelle Simova explains that she did not set out to build a robot of a particular size, but first asked herself a number of questions about what makes a useful robot. “I considered a couple of relevant scenarios like moving goods around a store, picking up items at home, or watering plants. For a robot to accomplish those tasks, at the very least it needs to run for a reasonable amount of time (at least a few hours) and be strong enough to carry a good amount of weight (10 kg or more).”

So, Hoverbot was born, a large robot that is equipped with sonar sensors for obstacle detection and is powerful and sturdy enough to transport heavy items. “Hoverbot is strong enough to carry me around,” says Isabelle, “though I wouldn’t recommend it to other people for safety reasons!”

## Recycle, repurpose, regenerate

Taking around three months to build, Hoverbot is essentially an upcycled machine, a concept that was important to Isabelle. “When you repurpose consumer goods you are not only leveraging the economies of scale that make otherwise expensive items affordable, you are also leveraging and extending work that has already been done by someone else,” she



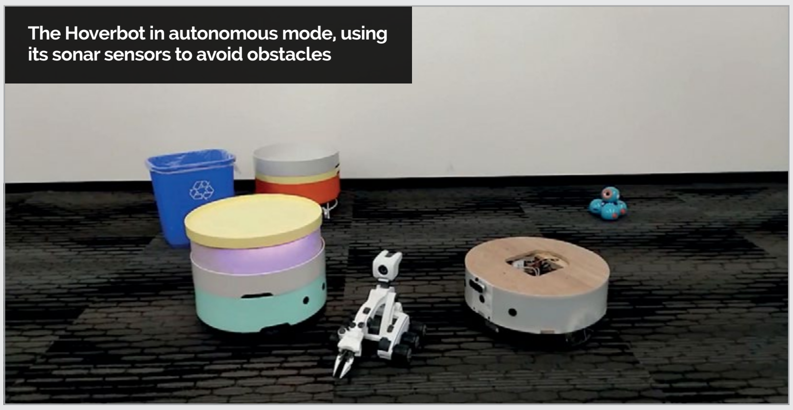
**Above** A web dashboard enables remote control of Hoverbot, including text to speech to make it talk says, adding that "upcycling can be a great forcing function for productivity as well. It can give you a concrete starting point in what is sometimes an intimidatingly large search space .... When you upcycle something you already have, you are putting something to use that might otherwise end up in a landfill." Clearly a point that our consumer-driven societies need to take on board, and something that a lot of makers and hackers generally are already aware of.

## Trial and error

A disassembled hoverboard met the requirements of the project, and at a fairly low cost. Isabelle met various challenges along the way, including reprogramming the motor controller, which required her to write firmware without any manufacturer documentation. "The process involved some trial and error and I accidentally let the magic smoke out of two hoverboard motor controllers."

Having used the Raspberry Pi in other projects, she was confident that it was up to the job: "With a Raspberry Pi you can ... both easily interface with devices over GPIO and run software written in high-level programming languages like JavaScript and Python. With Node.js on the Raspberry Pi, I was able to write all the Hoverbot application logic in JavaScript and leverage existing Node.js modules for serial and SPI communication."

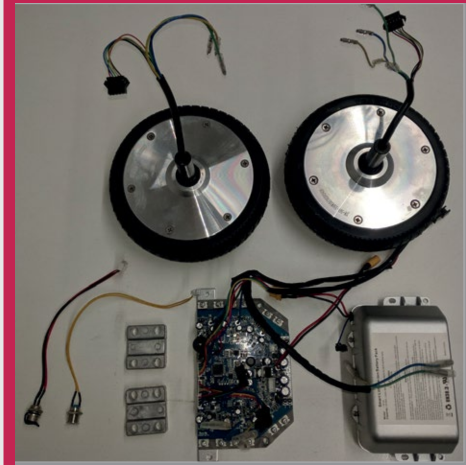
Isabelle tells us she will continue to tweak the Hoverbot, improving the motor controller firmware, and extending the robot's capabilities, as she explains: "Its current incarnation doesn't do much beyond driving around and avoiding obstacles, so it does need a bit more intelligence to accomplish useful tasks."


The Hoverbot in autonomous mode, using its sonar sensors to avoid obstacles

> "Hoverbot is strong enough to carry me around, though I wouldn't recommend it to other people for safety reasons"

It's truly inspiring to see what can be built with a dose of ingenuity and just a few items, some of which may otherwise have been destined for the bin!

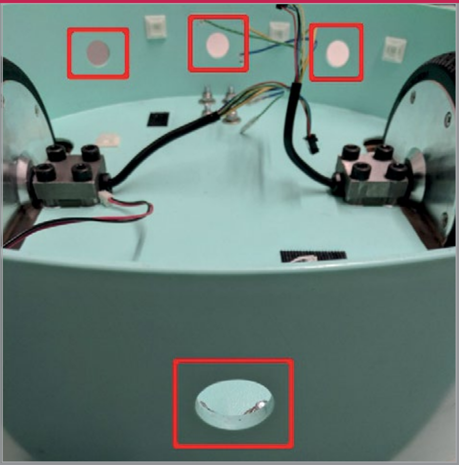
## INNER WORKINGS OF HOVERBOT



**>STEP-01**

**Disassemble the hoverboard**

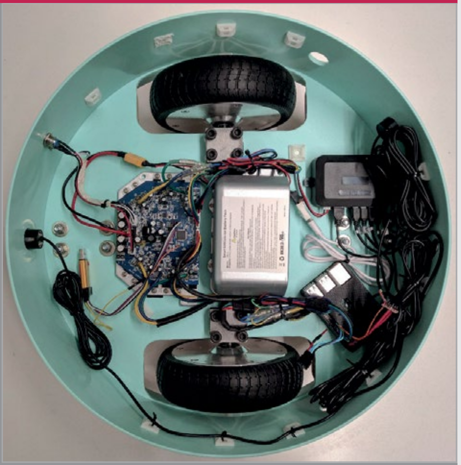
A hoverboard was dismantled to remove parts for the robot, including wheels, motors, motor controller, power switch, charge port, and battery.



**>STEP-02**

**Prepare the table**

An IKEA table was upcycled for the robot's body. Holes were cut in one table section for the wheels, and for four sonar sensors taken from a car reversing kit.



**>STEP-03**

**Electronics placement**

The electronics, including the Raspberry Pi, are mounted in the base using double-sided tape. An LED strip, webcam, and speaker are placed in an upper section.

# MAKE YOUR OWN VIDEO GAMES

DESIGN, CREATE, AND DISTRIBUTE YOUR OWN GAMES WITH OUR GUIDE TO EASY DEVELOPMENT TOOLS FOR THE RASPBERRY PI

**K.G. ORPHANIDES**

K.G. is a developer, software preservationist and avid player of obscure indie games. They're currently developing audio systems for Deck of Bards' Codename Caerus.

twitter.com/kgorphanides

It's easier than ever to make and publish your own games, and the Raspberry Pi is a perfect development platform, thanks to a wide range of tools and a hardware spec that encourages efficient, widely compatible design.

If you're new to programming, there are plenty of development environments that are easy to learn as you go, thanks to clear interfaces and copious tutorials. Whether you want to make an adventure game, a shoot-'em-up or a platformer, we'll help you find the tools you need.

Some of the most innovative games and interactive experiences combine or defy established genres to create something entirely new.

## THINK SMALL

A successful game project is the one you finish. If you're a solo developer working on your first or second game, it's probably a mistake to set out to make a vast open-world RPG.

There's a huge audience for tiny games, and the process of developing a small but perfectly formed gem can teach you a lot about tight design, efficiency, and good development practices.

Best of all, when you've finished your game, you can share it with the world before going to work on your next fantastic project.

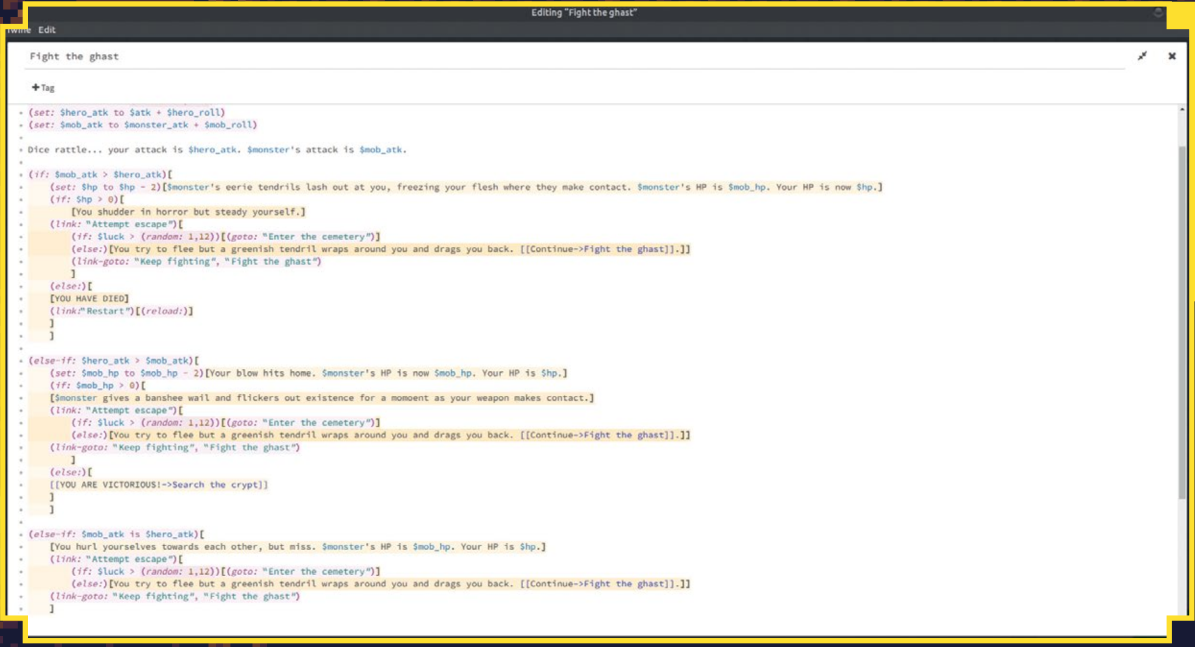
## PLAN YOUR DESIGN

The first stages of game development don't require any tech at all: your brain and some way of taking notes are basically all you're going to need.

What do you want your game to do? Do you want to tell a specific story, create a world for the player to explore, or present a puzzle or physical challenge to be overcome? Which development tools and game styles are most appropriate for your project?

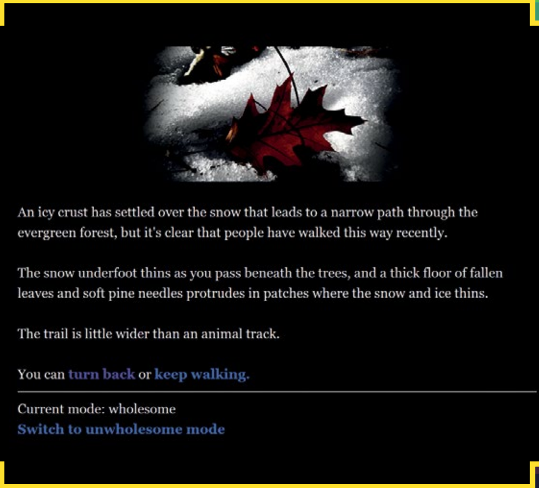
What skills do you have and what resources are available? Are there any particular areas of design and development that you want to try out?

If you're particularly comfortable with your ability as a 2D artist, then a visual novel could be perfect. Writers and storytellers can try their hand at text-based or simply illustrated narrative games, while keen programmers could



```
(set: $hero_atk to $atk + $hero_roll)
(set: $mob_atk to $monster_atk + $mob_roll)

Dice rattle... your attack is $hero_atk. $monster's attack is $mob_atk.

(if: $mob_atk > $hero_atk)[
    (set: $hp to $hp - 2)[$monster's eerie tendrils lash out at you, freezing your flesh where they make contact. $monster's HP is $mob_hp. Your HP is now $hp.]
    (if: $hp > 0)[
        [You shudder in horror but steady yourself.]
    (link: "Attempt escape")[
        (if: $luck > (random: 1,12))[(goto: "Enter the cemetery")]
        (else:)[You try to flee but a greenish tendril wraps around you and drags you back. [[Continue->Fight the ghast]].]]
        (link-goto: "Keep fighting", "Fight the ghast")
        ]
    (else:)[
    [YOU HAVE DIED]
    (link:"Restart")[(reload:)]
    ]
    ]

(else-if: $hero_atk > $mob_atk)[
    (set: $mob_hp to $mob_hp - 2)[Your blow hits home. $monster's HP is now $mob_hp. Your HP is $hp.]
    (if: $mob_hp > 0)[
    [$monster gives a banshee wail and flickers out existence for a momoent as your weapon makes contact.]
    (link: "Attempt escape")[
        (if: $luck > (random: 1,12))[(goto: "Enter the cemetery")]
        (else:)[You try to flee but a greenish tendril wraps around you and drags you back. [[Continue->Fight the ghast]].]]
    (link-goto: "Keep fighting", "Fight the ghast")
        ]
    (else:)[
    [[YOU ARE VICTORIOUS!->Search the crypt]]
    ]
    ]

(else-if: $mob_atk is $hero_atk)[
    [You hurl yourselves towards each other, but miss. $monster's HP is $mob_hp. Your HP is $hp.]
    (link: "Attempt escape")[
        (if: $luck > (random: 1,12))[(goto: "Enter the cemetery")]
        (else:)[You try to flee but a greenish tendril wraps around you and drags you back. [[Continue->Fight the ghast]].]]
    (link-goto: "Keep fighting", "Fight the ghast")
    ]
```

**Left:** Learning how to work with simple game development tools can help you learn the fundamentals of programming, with a concrete goal at the end of it all


An icy crust has settled over the snow that leads to a narrow path through the evergreen forest, but it's clear that people have walked this way recently.

The snow underfoot thins as you pass beneath the trees, and a thick floor of fallen leaves and soft pine needles protrudes in patches where the snow and ice thins.

The trail is little wider than an animal track.

You can turn back or keep walking.

Current mode: wholesome
Switch to unwholesome mode

**Far left:** You can use game creation tools to write deeply personal stories for a specific audience…

**Left:** …Or a fun, throwaway tribute to the classics, just to learn how they were made

focus on logic or action games illustrated with Creative Commons image assets.

For most games, you'll need a plot or objective (even if it's as simple as 'rescue the kidnapped king', 'collect the blue spheres to replenish the planet's soul', or 'defend the bug-eyed monsters from a human invasion'), probably some graphics and sound effects, and some programming to hold everything together and turn your idea into a functional game.

If you want to experiment with programming and development but don't know where to start, then a 'fantasy console' such as PICO-8 gives you a development environment with a complete but helpfully restricted set of features, so you can focus on making your game without being overwhelmed.

## PUBLISHING YOUR GAMES

Once you've finished your game, it's easy to publish it, whether for free, as pay-what-you-want donorware, or for a fixed price. Two of the best platforms for independent games publishing are Itch.io and GameJolt.com. Both of these popular digital stores let you publish games free of charge and, if users pay for your game, take a small percentage of that fee to cover their costs.

> If you need a bit of inspiration or motivation, try a game jam

## GAME JAMS

If you need a bit of inspiration or motivation, try a game jam. These events invite participants to produce a game on a specified theme, sometimes using a particular game engine, within a set period of time.

Game jams take place both on- and offline. Some are competitive, with plaudits or even prizes for the highest rated titles, but most are just an opportunity to test your skills, make something cool, and talk to other people who share your development interests.

Offline game jams in particular encourage people to work together, giving artists, musicians, writers, and programmers a chance to combine their talents to create something more sophisticated than they could alone. Many online jams encourage the same kind of teamwork, using communication platforms such as Discord to get people talking.

Fans of different game engines or genres hold regular dedicated jams or contests. Notable annual events include IFComp for text games, make-a-game-in-a-weekend jam Ludum Dare, and the Global Game Jam at physical locations around the world. You can also find lists of game jams on Itch and Game Jolt.

Every year, there are hundreds of game jams: free community challenges where people come together to make games within a set theme and time limit.

## GAME JAMS

| | |
|---|---|
| **INDIE GAME JAMS LIST** indiegamejams.com | **ITCH.IO JAMS** itch.io/jams |
| **GLOBAL GAME JAM** globalgamejam.org | **GAME JOLT JAMS** jams.gamejolt.com |
| **LUDUM DARE** ldjam.com | **IFCOMP** ifcomp.org |

# SPOTLIGHT ON PICO-8

**YOU WILL NEED**

- PICO-8 ($14.99) magpi.cc/NIBFUe

PICO-8 is a fantasy console – a tiny virtual machine that pretends to be an 8-bit games console or home computer. It uses an implementation of the Lua programming language with special libraries and features to make it easier to make games.

PICO-8 isn't open source and you'll have to pay for it, but $14.99 gets you perpetual, DRM-free updates and you can sell or give away the games you make without restriction.

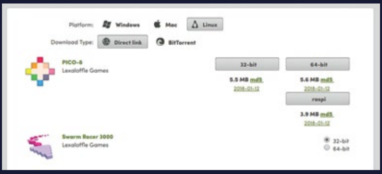
Some of the most important decisions in game design are about what your game won't do. Narrowing your choices and options through mechanical means is an excellent way of doing this. PICO-8's fantasy console conceit means that you only have to deal with a 16-colour display, six control inputs, four audio channels, and a maximum 32kB compiled game size.

Once you've created a game, you can save and distribute it as a virtual cartridge that can be packaged as a standalone game or played via a web browser. You can also import other people's cartridges to take a look at their code.

## STEP 01 DOWNLOAD AND RUN PICO-8

Lexaloffle fulfils PICO-8 orders via the Humble store. After buying it, you'll be emailed a link to download the software and associate it with your **lexaloffle.com** account. It's worth registering on the site to get support, distribute your games, and engage with the community.

For the Pi, you should download the Raspberry Pi version of PICO-8 from Humble or Lexaloffle. Expand the zip file wherever you like, go to that directory, and run the app from the File Manager or the command line. It'll automatically create some hidden directories to store its game files the first time you run it.



Platform: Windows Mac [✓ Linux]
Download Type: [✓ Direct link] BitTorrent

PICO-8 Lexaloffle Games — 32-bit 5.5 MB md5 2018-01-12 | 64-bit 5.6 MB md5 2018-01-12 | raspi 3.9 MB md5 2018-01-12

Swarm Racer 3000 Lexaloffle Games — [✓ 32-bit] 64-bit

## STEP 02 USING PICO-8

The fantasy console opens at the main PICO-8 screen. Like the main interface of early home computers such as the C64 and Spectrum, this is where you can load programs and run simple command strings in Lua.

PICO-8 comes with a demo pack, which we'll install and take a look at…

```
INSTALL_DEMOS
CD DEMOS
LS
```

To play any of the listed games and demos, use the LOAD and RUN commands. For instance, to play The Adventures of Jelpi:

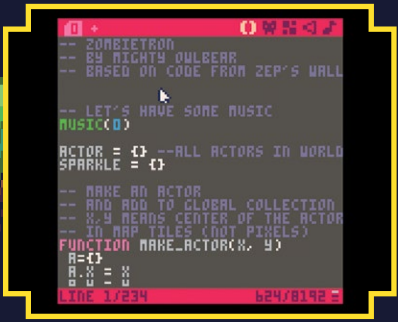
```
LOAD JELPI
RUN
```

…and then press **ESC** when you're ready to quit.

Many other game cartridges are available to download online, both as paid and free games, but make sure you put them in Pico-8's working directory. On the Pi, that's: **~/.lexaloffle/pico-8/carts**.

## STEP 03 THE CODING INTERFACE

To get from the main screen to PICO-8's game creation tools, all you have to do is press **ESC**. If you've loaded a game into memory, you'll be able to browse its code. Otherwise, you'll be presented with a blank input screen.



```
-- ZOMBIETRON
-- BY MIGHTY OWLBEAR
-- BASED ON CODE FROM ZEP'S WALL


-- LET'S HAVE SOME MUSIC
MUSIC(0)

ACTOR = {} --ALL ACTORS IN WORLD
SPARKLE = {}

-- MAKE AN ACTOR
-- AND ADD TO GLOBAL COLLECTION
-- X,Y MEANS CENTER OF THE ACTOR
-- IN MAP TILES (NOT PIXELS)
FUNCTION MAKE_ACTOR(X, Y)
 A={}
 A.X = X
LINE 1/234                624/8192
```

Although it looks basic, PICO-8's editor has far more features than the 8-bit computers that inspired it. Most important of these are syntax highlighting and cut, paste, and selection features using the mouse or keyboard, as you would in any modern GUI editor. You can also search your code by hitting **CTRL+F**.

At the top right, a number shows which tab of the text editor you're on, while a plus sign next to it allows you to create new tabs within the same project, for example to keep different code elements manageably grouped together. When you run your program, it'll treat all tabs as a single program, running the code in them in order.

Icons at the top right let you switch from the editor to PICO-8's other integrated game-building tools. The text editor is signified by a pair of brackets.

While PICO-8's built-in editor is surprisingly sophisticated, you may find a standard text editor more capable and easier on the eyes. Fortunately, you can open and edit .p8 files in anything from Leafpad to Vim.

## STEP 04 SPRITE EDITOR

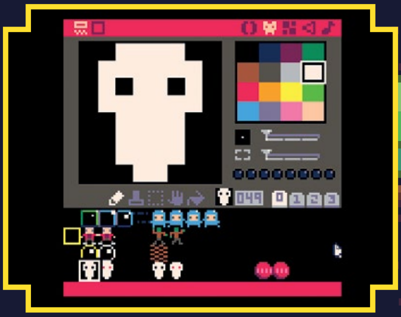
At the top right, a bank of icons lets you shift between edits. To the right of the text editor – with an icon that's either a pointy-toothed mammal or a butterfly – is the sprite editor.

Each sprite is 8×8 pixels in size and can take full advantage of the console's 16-colour palette. You can create up to 128 sprites, with a further 128 that can optionally be stored in memory otherwise used for the game's world map. Sprites are shown in numbered banks towards the bottom of the screen, which you can switch between using the mouse.

If you want to create an animation cycle for an 'actor' (a character or moving object in the game), just place all the sprites in that sequence one after the other and PICO-8's code routines will easily be able to show them all in order – all you have to do is specify the initial sprite's numbered position in the buffer and the number of frames in its animation cycle.

Editing tools allow you to draw, select, copy, paste, move (pan), and flood-fill parts of your sprite. You can flip, vertically flip, and rotate sprites using the **F**, **V**, and **R** keys respectively.

There are eight sprite flags in the form of coloured circles just above the sprite number marker.



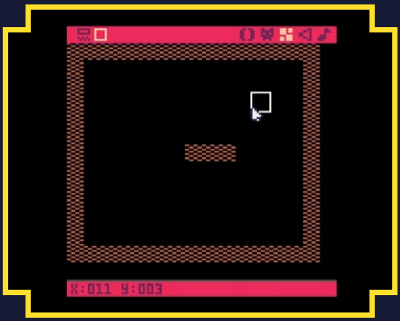
Each of these can be set to on or off, and can be assigned to different characteristics using the code editor, such as whether a given sprite will be solid or not.

Above these are brush size and zoom adjustment bars, which are handy if you're working on sets of sprites that will make up tiles of a larger piece.

## STEP 05 MAP EDITOR

The map editor works similarly to the sprite editor, but with a much larger area. PICO-8's default level map area measures 128×32, although you can expand that to 128×64 if you use the shared space that could otherwise house extra sprites.

Rather than drawing individual pixels, here you'll use the draw tool to place sprites as scenery, background objects, and decorations in the spaces the player will be exploring. Buttons



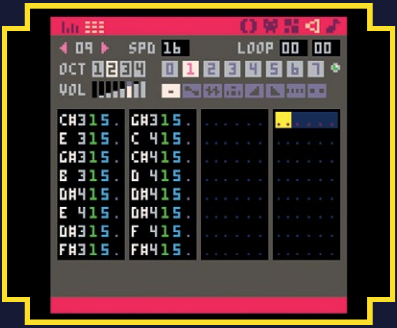
at the top right allow you to switch between having the level occupy the full screen or partly obscuring it by showing the sprite library.

Depending on your game's style and scope, you may need to pan around the level quite a bit – the **SPACE** key activates the click-and-drag to pan tool, and you can zoom in and out using your mouse's scroll wheel or the **<** and **>** keys.

The location of sprites in the level is defined using x and y co-ordinates displayed in the bottom toolbar. You'll be able to place interactive characters and objects in the level by specifying the co-ordinates at which they should appear in your code.

## STEP 06 SFX EDITOR

The built-in SFX (sound effects) editor makes it easy to create a rich soundtrack of 8-bit music and effects for your game. Any SFX track you make can be called in your game's code when an event occurs, or pulled into the music editor to create backing themes.



For sound effects, you'll mostly want an array of blips, beeps, and crashes. These are easily created using pitch editor mode. This lets you draw the shape of your music or sound, using a range of instrument voices inspired by classic consoles.

Press **TAB** or select the appropriate icon at the top left of the screen to switch to tracker mode, which gives you individual control over each note, making it better for more complex music composition and editing the effects you created in pitch mode.

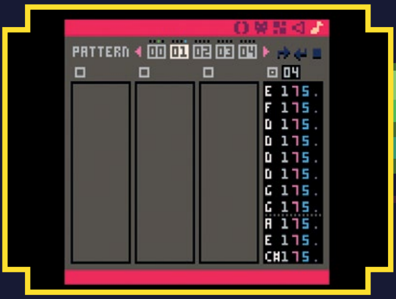
In tracker mode, you can also use your computer keyboard to play in notes one at a time. Remember that inserting the same note repeatedly will result in a single long note at that pitch – to create a silent rest, just reduce the volume of a given note to 0.

Each SFX track can have a maximum of 32 notes, and there's room to store up to 64 individual SFX.

## STEP 07 MUSIC EDITOR

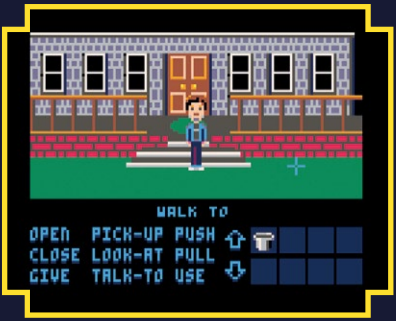
The music editor allows you to assemble your SFX tracks to play simultaneously, consecutively, or in a specified loop to build up your soundtrack. You can have up to four SFX playing simultaneously. By default, each pattern will move on to the next unless it hits a loop end marker – a loop begin marker allows you to determine where the loop will return to.

You can have up to 64 of these patterns, and use the `music()` function to start and stop them in your game's code.



## STEP 08 LEARNING LUA

PICO-8's minimal Lua-based API makes for a clear and consistent programming experience. The official manual's plain text formatting is a little unwelcoming, but it provides plenty of examples of how to use its more complex features.



However, to get to grips with the basics of your new fantasy console, the best thing to do is look through the code of the supplied demo games, copy it, and then modify it to create your own first games.

If you want to make a specific kind of game for the PICO-8, such as a shooter, platform game, or even a point-and-click adventure, there are dedicated game engines and demos for the system that provide all the code you'll need to make different types of game, which you can then work with and customise for your own purposes. Paul 'Liquidream' Nicholas's SCUMM-8 graphic adventure game engine is particularly impressive.

### RESOURCES FOR PICO-8

**PICO-8 MANUAL**
magpi.cc/uXMVws

**SCUMM-8**
magpi.cc/LaGTWu

**SECTORDUB'S PICO-8 FANZINE**
sectordub.itch.io

**PLATFORM GAME DEMO IN 100 LINES**
magpi.cc/nWFCrT

**DOWNLOAD PICO-8 GAMES**
itch.io/games/tag-pico-8
gamejolt.com/tag/pico8
magpi.cc/qkBOds

**FEATURES OF PICO-8 LUA**
pico-8.wikia.com/wiki/Lua

# TELLING STORIES

Narrative games range from simple interactive storybooks to puzzle-filled adventure games and combat-packed RPGs. We're going to look at two of our favourite tools for making story-based gaming worlds in very different styles.

## WEAVING A NARRATIVE WITH TWINE

Twine is a favourite development tool among game writers keen to show off their narrative chops. It lends itself to interactive stories and choose-your-own-adventure style games, but has a scripting language powerful enough to support fully fledged RPGs and strategy games. Both Twine and the games it makes run in a browser.

Your game is built up of passages of text, which are shown on an overview screen as little interlinked cards. Because it supports branching dialogue and plot paths, Twine is a great tool for mapping out the path of narrative games destined for other platforms.

### YOU WILL NEED

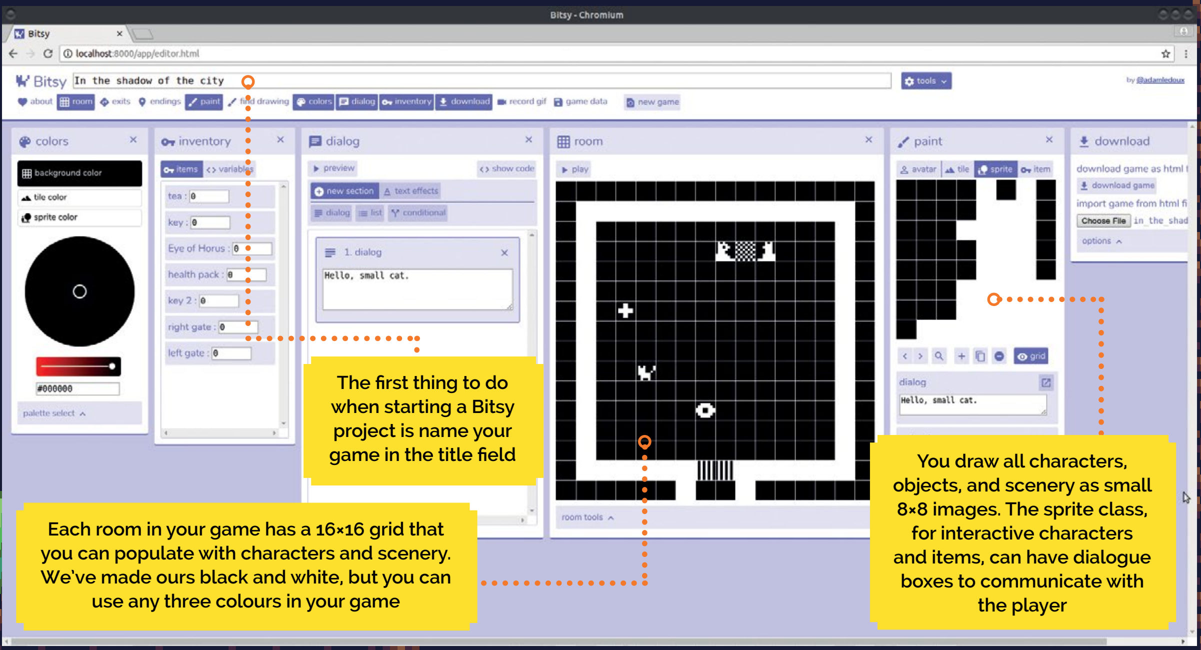
**TWINEJS**
magpi.cc/VAekwV

**TWINE HARLOWE MANUAL**
twine2.neocities.org

**BITSY-NWJS**
magpi.cc/WHCsGD

**BITSY TUTORIAL**
magpi.cc/QxmzMO

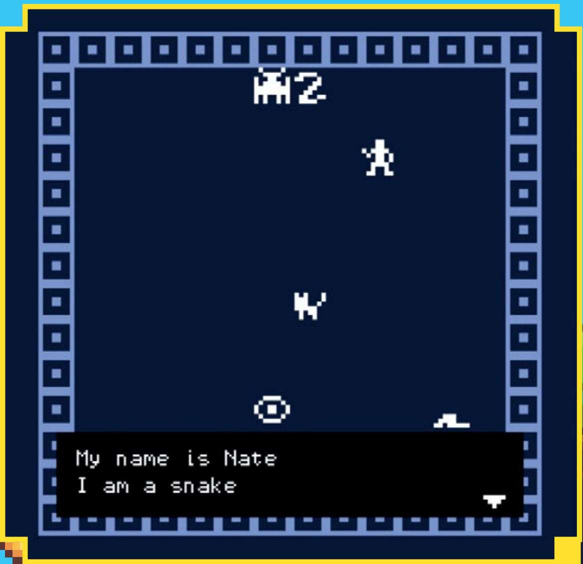
Its most basic function is the `[[link]]` command. If you put any string of text between two sets of



The first thing to do when starting a Bitsy project is name your game in the title field

Each room in your game has a 16×16 grid that you can populate with characters and scenery. We've made ours black and white, but you can use any three colours in your game

You draw all characters, objects, and scenery as small 8×8 images. The sprite class, for interactive characters and items, can have dialogue boxes to communicate with the player



Bitsy games are published as HTML files, which you can upload to a website or email to your friends

## RUN TWINE ON THE PI

There's no standalone Twine desktop client for the Raspberry Pi, but a local version of the engine called TwineJS is maintained by original creator Chris Klimas.

Open a Terminal window and type the following commands:

```
curl -sL deb.nodesource.com/setup_8.x | sudo -E bash -
sudo apt install -y nodejs
git clone github.com/klembot/twinejs.git
cd twinejs
npm install  (to install Twine's dependencies)
npm start  (to start Twine on a local server)
```

Access Twine from Chromium by going to **http://localhost:8000**. Run `npm start` from your **twinejs** directory every time you want to work on your game.

square brackets `[[like this]]`, Twine will automatically create a new passage with that name and display a link to it from your current passage.

## TINY GAME DESIGN IN BITSY

Bitsy is an easy-to-use tool for creating three-colour pixel art adventures in your browser, either on- or offline. First released in early 2017, it already has a huge user community, with regular game jams to inspire to you try out your skills.

Bitsy is self-consciously simple, which makes it ideal for anyone who'd rather tell stories and build worlds than immediately dive into coding. It's particularly good for linear and exploration-based stories or object collection games. You can implement simple puzzles using its conditional dialogue options and even expand Bitsy's core functions using your own or other people's extensions and hacks.

You get an 8×8 pixel grid with two frames of animation to draw four key object types:

| | |
|---|---|
| **Avatar:** | Your game's playable character |
| **Tiles:** | Scenery that can be either solid walls or decorative items that can be walked over |
| **Sprites:** | Interactive scenery and non-player characters |
| **Items:** | Objects that can be picked up |

You can use these objects to design and populate rooms on a 16×16 grid, with small snippets of dialogue associated with sprites and items. A tools button next to your game's title at the top of the screen reveals more options including the exits window, where you can join your rooms together to create a world for the player to explore.

## RUN BITSY ON THE PI

While Bitsy normally runs in a browser directly from the creator's website, there's an offline version that's perfect for Pi development projects. It still works via a browser, but is fully functional without an internet connection. Install and build it with these commands:

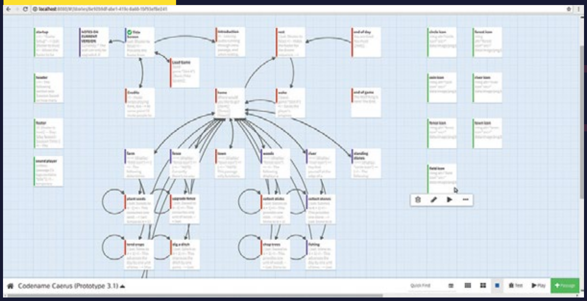
```
curl -sL deb.nodesource.com/setup_8.x | sudo -E bash -
sudo apt install -y nodejs
git clone github.com/ztiromoritz/bitsy-nwjs.git
cd bitsy-nwjs
npm install
npm run build
```

The build can take quite a while, even on a Pi 3 B+, making this an ideal time to go and get a cup of tea. With that done, launch your offline Bitsy server from its directory whenever you want to use it, by typing:

```
npm run serve
```

You can access it in your browser by going to **http://localhost:8000**

Ideal for decision-based narrative games, Twine's overview screen can show you how each passage of text relates to all the others



# A GAME MAKER'S TOOLKIT

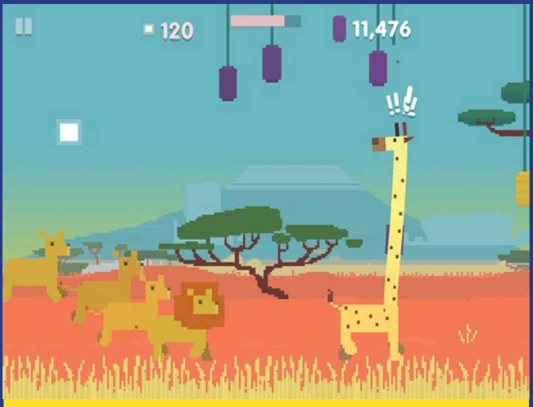
## EIGHT MORE FASCINATING GAME-MAKING TOOLS FOR THE PI

A wide range of game engines, integrated development environments, and markup languages designed for game development work brilliantly on the Pi. We've highlighted eight of the most interesting, with pointers on getting them up and running.

### LÖVE

**love2d.org**

If you've got to grips with PICO-8 and want something a little more flexible, you might be looking for LÖVE. This more advanced Lua-based game engine has been used to make some quite sophisticated titles and is regarded as being particularly good for mobile development. LÖVE 2D is available in Raspbian's repositories (`sudo apt install love`). You'll need experimental OpenGL drivers enabled in raspi-config and a text editor, ideally with a Lua code highlighting module, to write your game in.

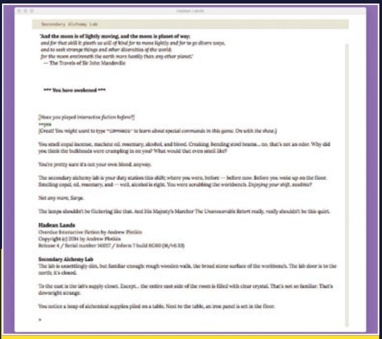


Oh My Giraffe by Nico Prins

### INFORM 7

**inform7.com**

An engine for creating parser text adventures (get lamp, light lamp, eat lamp) that lets you program games by writing sentences in natural English such as 'The Cave is a room. A dragon is here.'



Secondary Alchemy Lab

'And the moon is of lightly moving, and the moon is planet of way;
and for that skill it giveth us will of kind for to move lightly and for to go divers ways,
and to seek strange things and other diversities of the world;
for the moon environeth the earth more hastily than any other planet.'
— The Travels of Sir John Mandeville

*** You have awakened ***

[Have you played interactive fiction before?]
>>yes
[Great! You might want to type "COMMANDS" to learn about special commands in this game. On with the show.]

You smell copal incense, machine oil, rosemary, alcohol, and blood. Creaking, bending steel beams... no, that's not an odor. Why did you think the bulkheads were crumpling in on you? What would that even smell like?

You're pretty sure it's not your own blood, anyway.

The secondary alchemy lab is your duty station this shift; where you were, before — before now. Before you woke up on the floor. Smelling copal, oil, rosemary, and — well, alcohol is right. You were scrubbing the workbench. Enjoying your shift, noobie?

Not any more, Sarge.

The lamps shouldn't be flickering like that. And His Majesty's Marcher The Unremovable Retort really, really shouldn't be this quiet.

**Hadean Lands**
Overdue Interactive Fiction by Andrew Plotkin
Copyright (c) 2014 by Andrew Plotkin
Release 4 / Serial number 141007 / Inform 7 build 6G60 (I6/v6.33)

**Secondary Alchemy Lab**
The lab is unsettlingly dim, but familiar enough: rough wooden walls, the broad stone surface of the workbench. The lab door is to the north; it's closed.

To the east is the lab's supply closet. Except... the entire east side of the room is filled with clear crystal. That's not so familiar. That's downright strange.

You notice a heap of alchemical supplies piled on a table. Next to the table, an iron panel is set in the floor.

>

Hadean Lands by Zarfhome Software

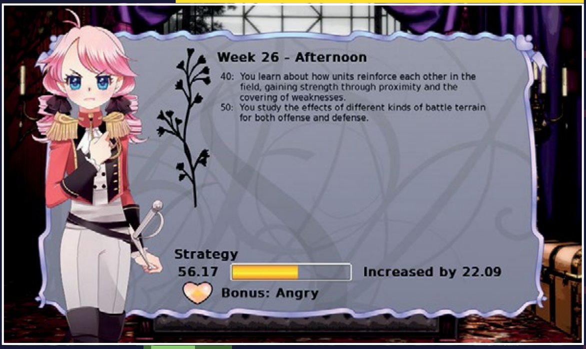
It's easy to get started by writing out a few sentences to describe a room and the items it contains, but the well-documented programming language has tremendous power that you can use to create anything from complex conversations to environmental physics. Only the command-line compilation tool is available for 32-bit systems like the Pi, but you can follow the documentation to write your game in a standard text editor.

### REN'PY

**renpy.org**

Ren'Py is a Python-based game engine for making visual novels – a popular form of illustrated adventure game that originated in Japan. These can focus on dialogue or include puzzles, and typically have rich still graphics and at least mildly choice-based plots. Ren'Py has been used to make commercial hits like Long Live the Queen and can produce very polished results. Bear in mind, though, that Raspberry Pi support is still experimental and requires Fake KMS OpenGL drivers and extra graphics memory to be set in raspi-config.

Long Live the Queen by Hanako Games



Week 26 - Afternoon
40: You learn about how units reinforce each other in the field, gaining strength through proximity and the covering of weaknesses.
50: You study the effects of different kinds of battle terrain for both offense and defense.

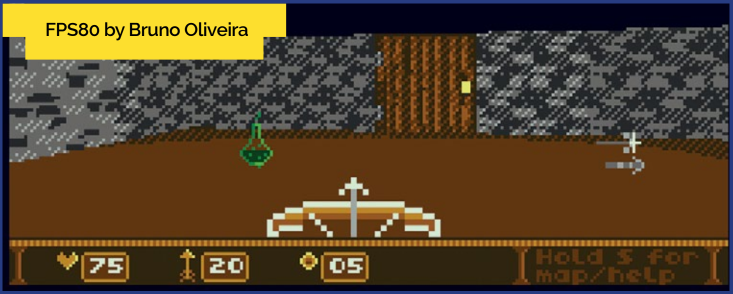
Strategy
56.17 Increased by 22.09
Bonus: Angry

## TIC-80

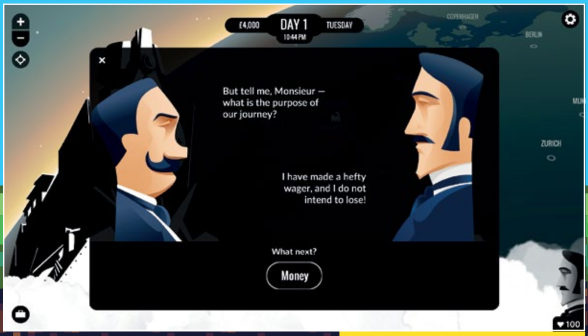
**tic.computer**

If you like the look of the fantasy computer trend but would prefer to go for a free and open-source option, the TIC-80 tiny computer may tickle your fancy. It's virtually equipped with a 240×136 display, 16-colour palette, 256 8×8 sprites, and four-channel sound, and has a full set of integrated tools to edit code, create graphics, build levels, and write music.

It also supports a wider range of programming languages than PICO-8, with JavaScript and Moonscript as well as Lua. Compiling from source works best on the RetroPie operating system and there's also a pre-built JavaScript version available to download and run in your browser.



FPS80 by Bruno Oliveira



But tell me, Monsieur – what is the purpose of our journey?

I have made a hefty wager, and I do not intend to lose!

80 Days by Inkle Ltd

## INK

**magpi.cc/pBmaCl**

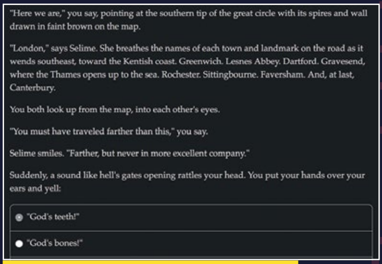
A choice-based markup language, famously used to develop the award-winning 80 Days and Sorcery games, Ink can be used as middleware alongside other game engines or to create standalone narrative games. The open-source Inky editor allows you to test games as you go, and export them to a web-playable version for release. You'll need to install mono-complete from Mono's own repositories to build Inky – full dependencies are listed in its readme file.

## CHOICESCRIPT

**magpi.cc/XSfqoo**

ChoiceScript is the markup language behind the very popular Choice Of series of interactive fiction games for mobile and web platforms. They're simple choose-your-own-adventure style games, ranging from horror to romance, with an emphasis on quality writing. Choice Of Games even has a paid publication path with a 25% royalty rate for new writers and advances available to experienced authors.

All you need is a text editor and the supplied files to make your own ChoiceScript games. The instructions on the download page are a little out of date – you'll now find the extracted test game script at **/web/mygame/scenes/startup.txt**.


"Here we are," you say, pointing at the southern tip of the great circle with its spires and wall drawn in faint brown on the map.

"London," says Selime. She breathes the names of each town and landmark on the road as it wends southeast, toward the Kentish coast. Greenwich. Lesnes Abbey. Dartford. Gravesend, where the Thames opens up to the sea. Rochester. Sittingbourne. Faversham. And, at last, Canterbury.

You both look up from the map, into each other's eyes.

"You must have traveled farther than this," you say.

Selime smiles. "Farther, but never in more excellent company."

Suddenly, a sound like hell's gates opening rattles your head. You put your hands over your ears and yell:

● "God's teeth!"

● "God's bones!"

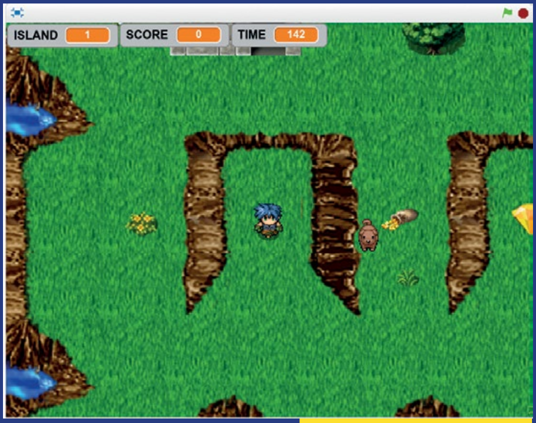
The Road to Canterbury by Choice of Games

## SCRATCH

**scratch.mit.edu**

Scratch comes with every Raspbian install. It's well documented and is explicitly designed to be easy for beginner programmers, with a drag-and-drop interface that allows you to build up sequences of action and logic, including graphical and audio elements.

The visual programming interface uses instruction blocks that bolt together and can be used to define text, control systems, variables, and more. A huge range of games, slideshows, and stories are available on the official Scratch website, and you can look at their code to learn by example.



MyLands by PsiBorg

## PYGAME AND PI3D

**pygame.org • magpi.cc/ZEIPjL**

Pygame and Pi3D are two sets of games programming libraries for Python that are ideal for Pi users who are learning the fundamentals of game development. Pygame comes installed on Raspbian by default and is packed with useful functions for everything from physics to animation, with loads of examples and documentation available online.

You'll have to install Pi3D separately, but it's a great Python toolkit for getting the Pi to do something no one really expects of it: 3D graphics rendering. Don't forget to allocate extra memory to the GPU before running Pi3D's demos.

Pi3D forest walk demo

## SUBSCRIBE TODAY FROM JUST £5

SAVE UP TO 35%

£5 FREE! MODMYPI VOUCHER FOR ALL SUBSCRIBERS

## Pricing

### Rolling Subscription

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

* Leave any time applies to Rolling Subscription only

### Subscribe for a year:

£55 (UK)

£80 (EU)

£90 (USA)

£95 (Rest of World)

### Subscription benefits:

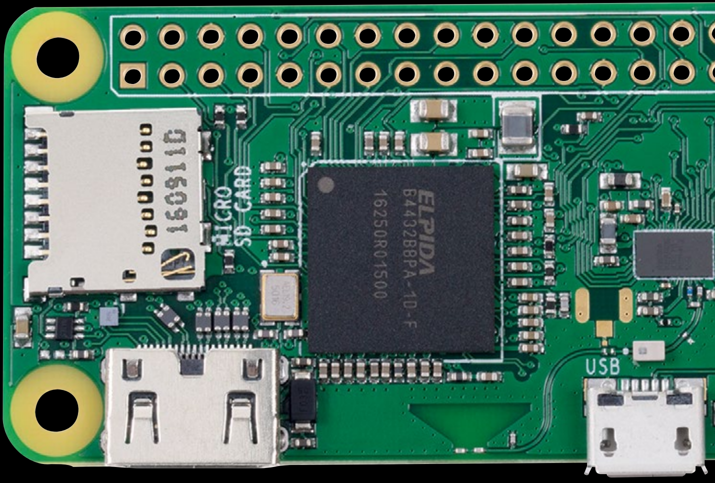
- FREE! Delivery to your door
- EXCLUSIVE! Raspberry Pi offers and discounts
- NO OBLIGATION! Leave any time*

magpi.cc/subscribe

JOIN FOR 12 MONTHS AND GET A

# FREE PI ZERO W STARTER KIT

WITH YOUR SUBSCRIPTION



WORTH £20

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

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

## SUBSCRIBE ON APP STORES

Available on the App Store

GET IT ON Google Play

FROM £2.29

# TAKE SCREENSHOTS ON A RASPBERRY PI

Use screenshots to show others how you set up the software for your projects

**You'll Need**

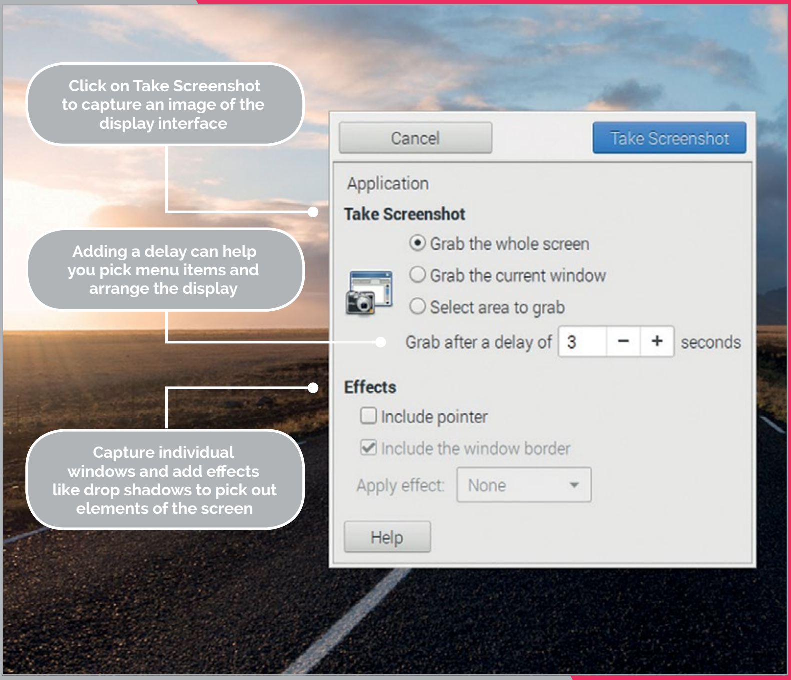
- Raspbian
- GNOME Screenshot
- Internet connection (for installation)

When you've made an amazing project, it's natural to want to share it with other folks (in the hope that they will follow in your footsteps).

Capturing screenshots (images of the desktop display) is a vital component of putting together an effective tutorial. An image is worth a thousand words, and images of your software installation can certainly make it easier for others to understand what you're doing.

Some beginners opt for taking physical photographs of their display, but due to the screen refresh this rarely looks good. It's better to use software to grab the digital screen on the computer and save it as an image file. It's pretty easy too.

You capture images on your Raspberry Pi using a built-in command-line program called Scrot. Alternatively, install a more advanced program called GNOME Screenshot (this is what we typically use). Once this software is installed, you can quickly capture your screen display.


Click on Take Screenshot to capture an image of the display interface

Adding a delay can help you pick menu items and arrange the display

Capture individual windows and add effects like drop shadows to pick out elements of the screen

Grab the whole screen [✓]
Grab the current window
Select area to grab
Grab after a delay of 3 seconds

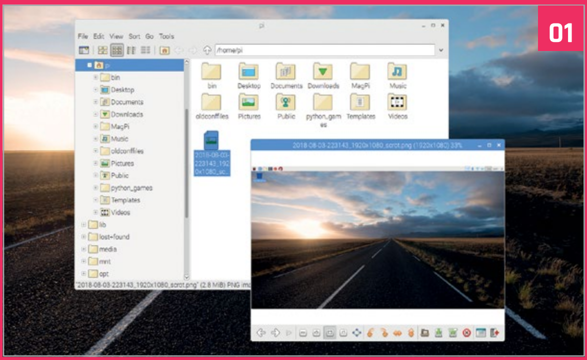
Include pointer
Include the window border [✓]
Apply effect: None

## >STEP-01

### Using Scrot

The easiest way to take a screenshot is using Scrot. This software comes with Raspbian so you don't need to install anything.

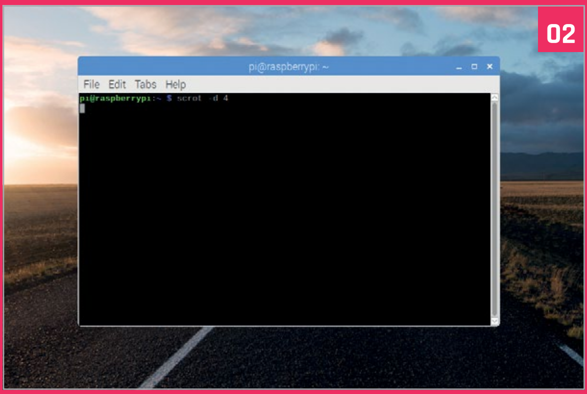
If your keyboard has a **PrtSc** (Print Screen) key, then simply tap it. Scrot will capture an image of the desktop and save it to your home folder. Click on File Manager in the taskbar and double-click the .png file that starts with the date.



## >STEP-02

### Scrot delay

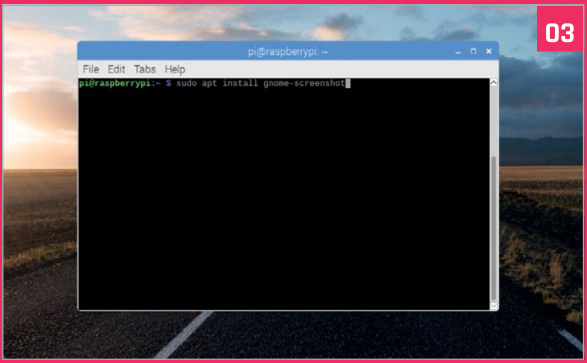
You can also use Scrot from the command line. Enter **`scrot`** there to capture the whole screen (including the Terminal window). If you want to capture the screen without the Terminal window, you can set a delay (then close the Terminal window).



```
pi@raspberrypi:~ $ scrot -d 4
```

```
scrot -d 4
```

With the above command, Scrot will delay for four seconds before capturing the screen. Typically this is enough time to close down the Terminal window. Add a longer time if you want to arrange the display to include other items (such as menus).



```
pi@raspberrypi:~ $ sudo apt install gnome-screenshot
```

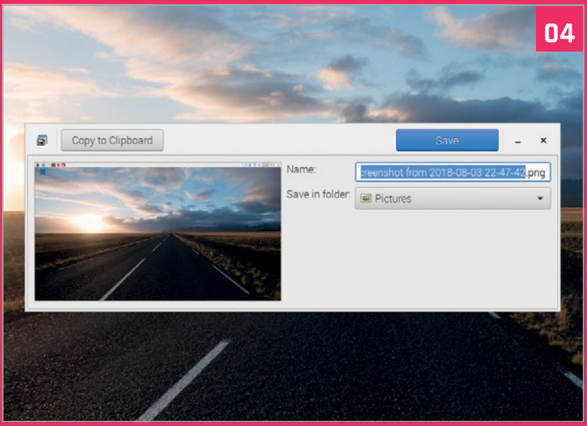
## >STEP-03

### Install GNOME Screenshot

Make sure your Raspberry Pi is connected to the internet. Click on Terminal in the taskbar to open a new Terminal window, and enter the following to update the software and install GNOME Screenshot:

```
sudo apt update && sudo apt upgrade
sudo apt install gnome-screenshot
```

Enter **Y** when prompted, to install the new software and upgrades.

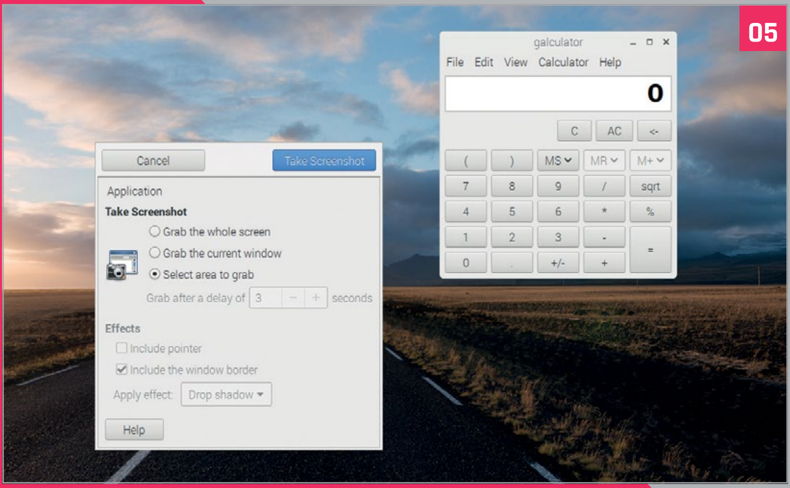


Name: reenshot from 2018-08-03 22-47-42.png
Save in folder: Pictures

## >STEP-04

### Screen capture

GNOME Screenshot provides you with a desktop interface for taking screenshots (along with a bunch of extra features). You'll find it in Menu > Accessories > Screenshot. Click on Take Screenshot and the interface will vanish and a screen will be captured. Unlike Scrot, you get an interface allowing you to name the screenshot and decide where it will be saved. Click on Save to finish taking the screenshot.



Take Screenshot
- Grab the whole screen
- Grab the current window
- [✓ Select area to grab]

Grab after a delay of 3 seconds

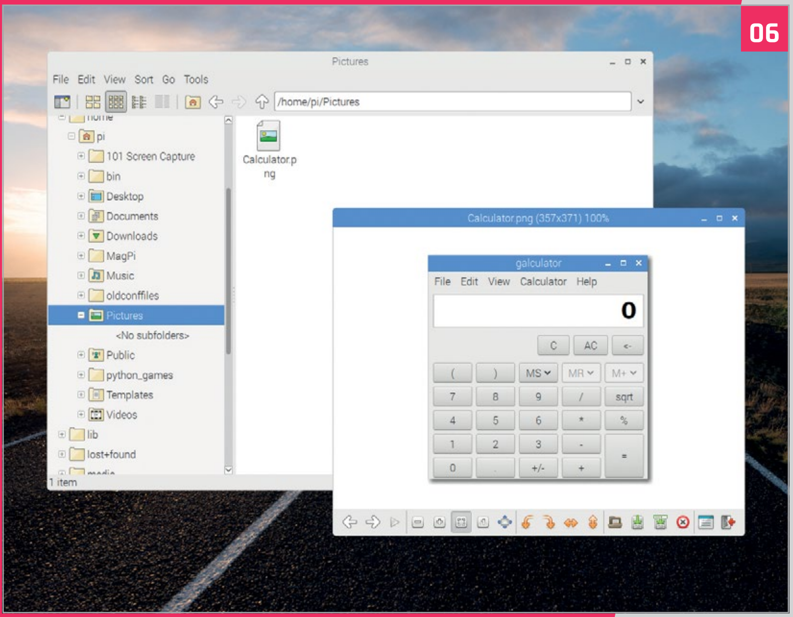
Effects
- Include pointer
- [✓ Include the window border]

Apply effect: Drop shadow

## >STEP-05

### Grab areas

You can focus on a particular part of the screen by capturing an area, rather than the whole screen. Click the radio button marked 'Select area to grab' and click Take Screenshot. The GNOME Screenshot interface will disappear. Now click and drag a marquee around the area you want to capture (in our case the Calculator app) and GNOME Screenshot will capture just that part of the screen.



## >STEP-06

### Grab window

A more effective method is to grab individual windows. These can have effects added, such as drop shadows and borders to make them stand out. Choose the 'Grab the current window' radio button and choose 'Drop shadow' from the 'Apply effect' window. Make sure the window you want to capture is highlighted and click Take Screenshot. An image of the window, with the drop shadow effect, is saved.

## MIKE'S PI BAKERY

**MIKE COOK**

Veteran magazine author from the old days and writer of the Body Build series. Co-author of *Raspberry Pi for Dummies*, *Raspberry Pi Projects*, and *Raspberry Pi Projects for Dummies*.
**magpi.cc/259aT3X**

PART 01

# THE MATRIX

The Matrix is an undedicated array of switches and lights that can be put to any number of uses – animated light displays, games, and controlling synthesizers, to name but three. Best of all, it won't cost you a fortune

### You'll Need

- 16 × 3-LED push lights **magpi.cc/iEkiUY**
- Red, green, and blue transparent paint **magpi.cc/vBcYrB**
- 16 × 510 Ω 125 mW resistors **magpi.cc/mDWDHU**
- 1280×18×6 mm Strip pine **magpi.cc/iXECkc**
- Gorilla Glue **magpi.cc/eluwAl**
- 320×320 mm MDF 6 mm thick **magpi.cc/SPBOlx**

In a thrift shop, we spied white LED stick-on battery powered lights at only £1 for two. That set us off thinking what we could make with them. We came up with an illuminated switch matrix that could be used for many things. At the end of the project (in two or three issues' time), after showing you how to interact with it, we thought we could set you a challenge to see if you could write your own software for a specific application. This is a project that will keep on growing.

We are going to use a MAX7219 chip to drive the LEDs; this removes any real time burden off the Pi. Then, by a crafty modification, we are going to use the same chip to also read the state of the push switches. As these are latching switches, we will then use software to find out which switches have changed since we last looked at them. As three white LEDs are not too interesting, we will use transparent paint, normally used for painting on glass, to make our three LEDs show different colours.

01

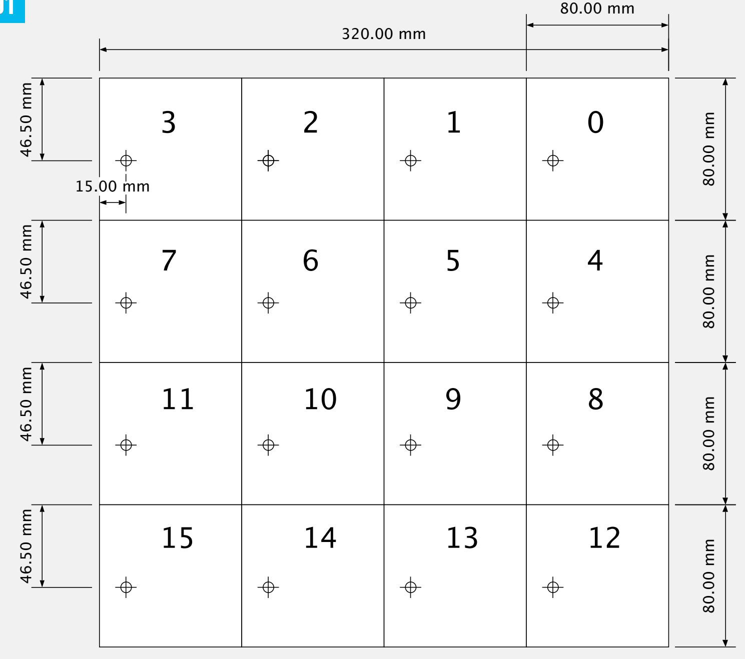


**Figure 1** Marking the underside of the board

### >STEP-01
**Mark out the box lid**

We didn't do this step ourselves first, but on reflection we thought it would be easier to do before the sides were put on the box. Cut a 320×320 mm piece of 6 mm MDF and mark out the squares for each illuminated switch, then number them. This is going to be the back of the box, hence the squares are numbered back to front. Each light will be in its own 80 mm space and you could change this if you want a bigger or smaller board. Then mark the position for the light's fixing holes 15 mm from the left-hand side and 46.5 mm from the top. The dimensions are shown in **Figure 1**.

### >STEP-02
**Mark out the front**

Flip the board over and draw a light pencil line horizontal line across the whole board 46.5 mm down from the top. Repeat this 46.5 mm up from the bottom. Finally, draw two more lines 80 mm down from the top line and 80 mm up from the bottom line. This will serve as marker for when you later fix the lights. Now flip the board over and drill the 3 mm holes marked in **Figure 1**. Like all holes, drill onto a scrap piece of MDF to ensure a clean edge to the hole, and clean up any bits of MDF round the hole with a scalpel.

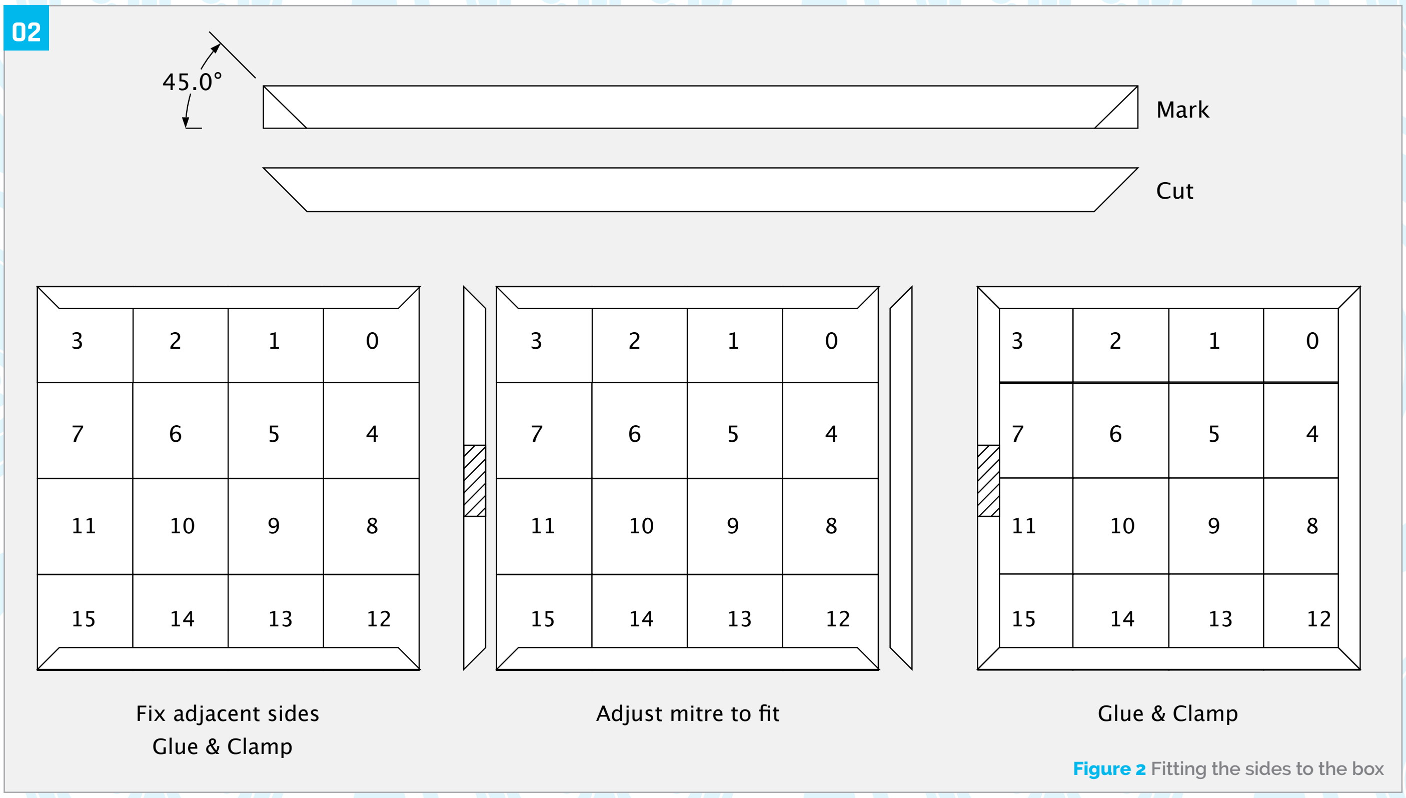
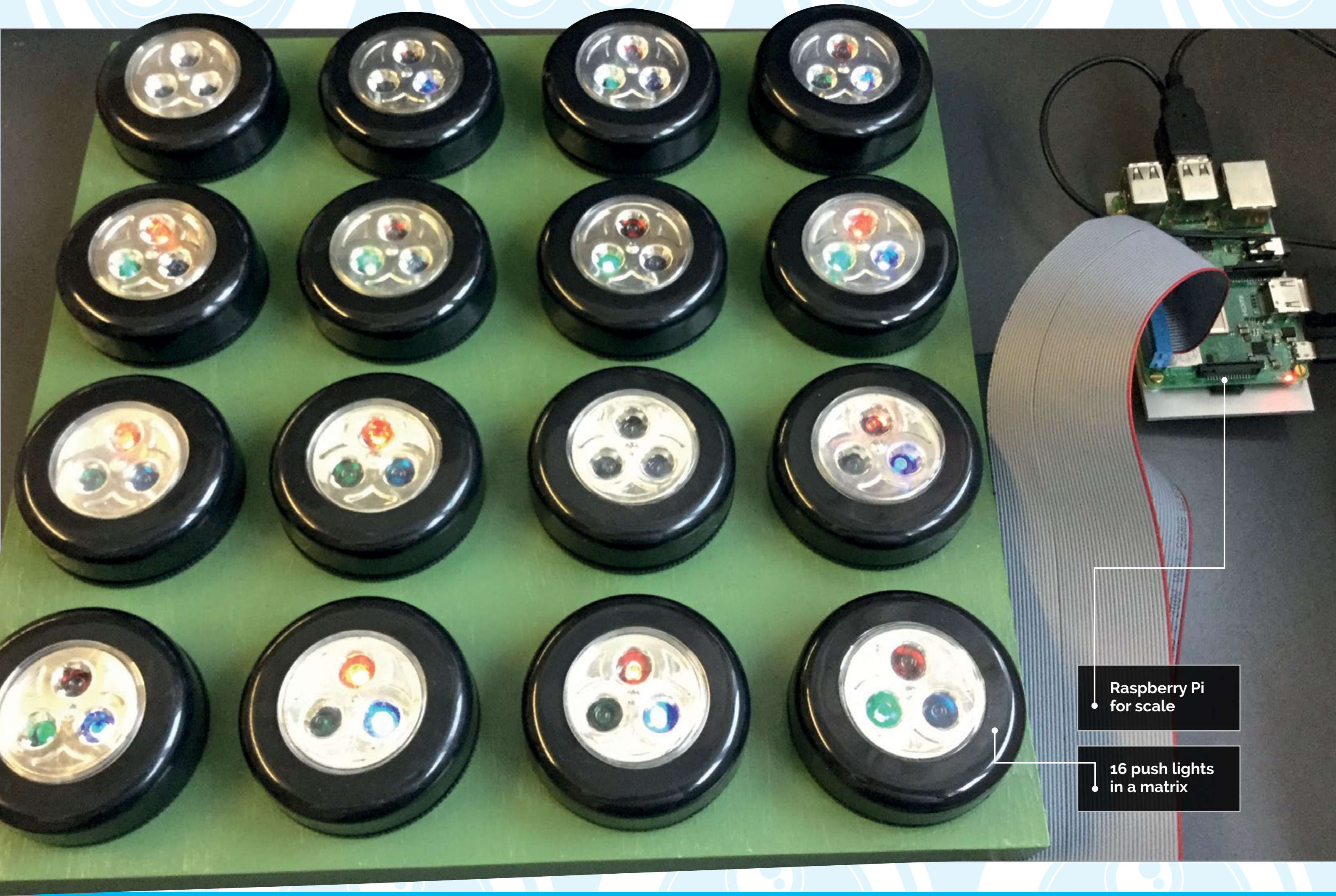


**Figure 2** Fitting the sides to the box



03

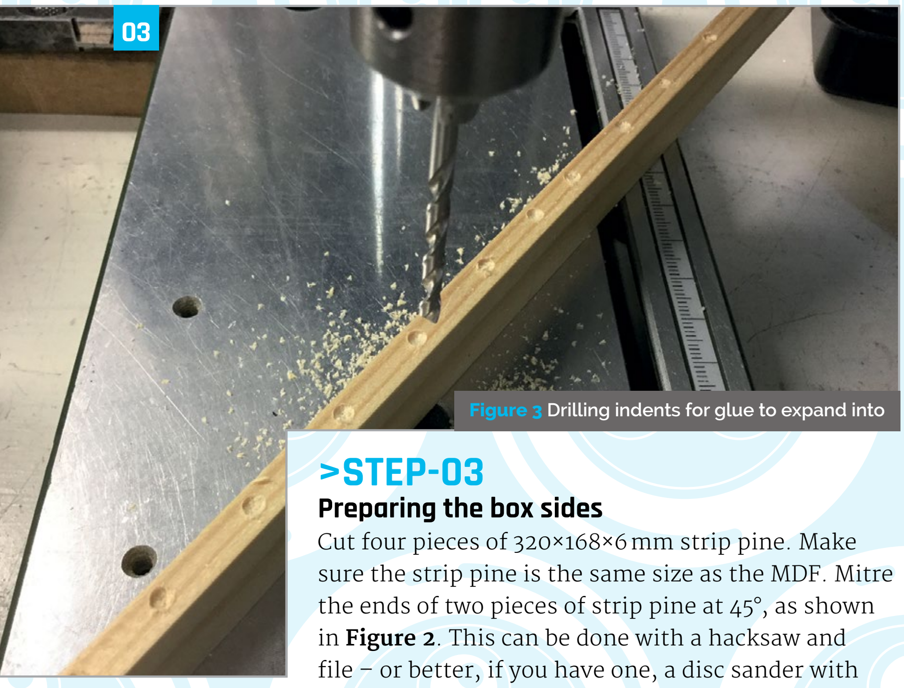


Figure 3 Drilling indents for glue to expand into

## >STEP-03

### Preparing the box sides

Cut four pieces of 320×168×6 mm strip pine. Make sure the strip pine is the same size as the MDF. Mitre the ends of two pieces of strip pine at 45°, as shown in **Figure 2**. This can be done with a hacksaw and file – or better, if you have one, a disc sander with the guide set to 45°. Draw a line, using the strip pine as a rule down each side. Use a 3 mm drill and put a sequence of very shallow indents at approximately 30 mm spacing along the length of the strip pine, as shown in **Figure 3**. Then do the same for the MDF. This is to allow the glue to expand into the indents and have more grip.

05

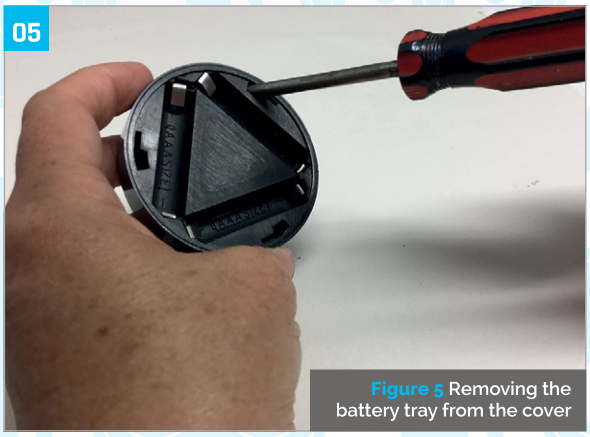


Figure 5 Removing the battery tray from the cover

04

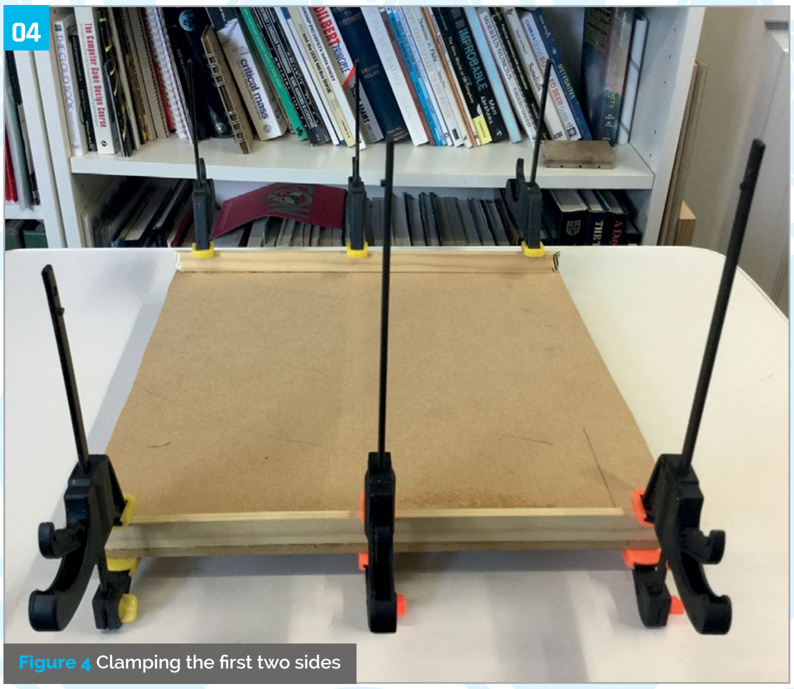


Figure 4 Clamping the first two sides

## >STEP-04

### Fitting the box top and bottom

Gorilla glue is a foaming expanding glue that works by the absorption of moisture. When you are ready, spray a small amount of water from a plant mister spray onto (or wipe with a damp rag) the surfaces you are going to join together. These will be the top and bottom sides. Apply a thin layer of glue to each side and clamp the two sides together. The strength of the glue joint depends on the two sides being clamped together. Allow the glue to set for at least two hours – see **Figure 4**. When set, trim off any glue that has seeped out of the joint on the outside with a scalpel.

## >STEP-05

### Fixing the box sides

Referring back to **Figure 2**, take the other two lengths of strip pine and mitre them. This time, make sure there is a good fit between the two strips already attached. This might involve taking off a little more

06

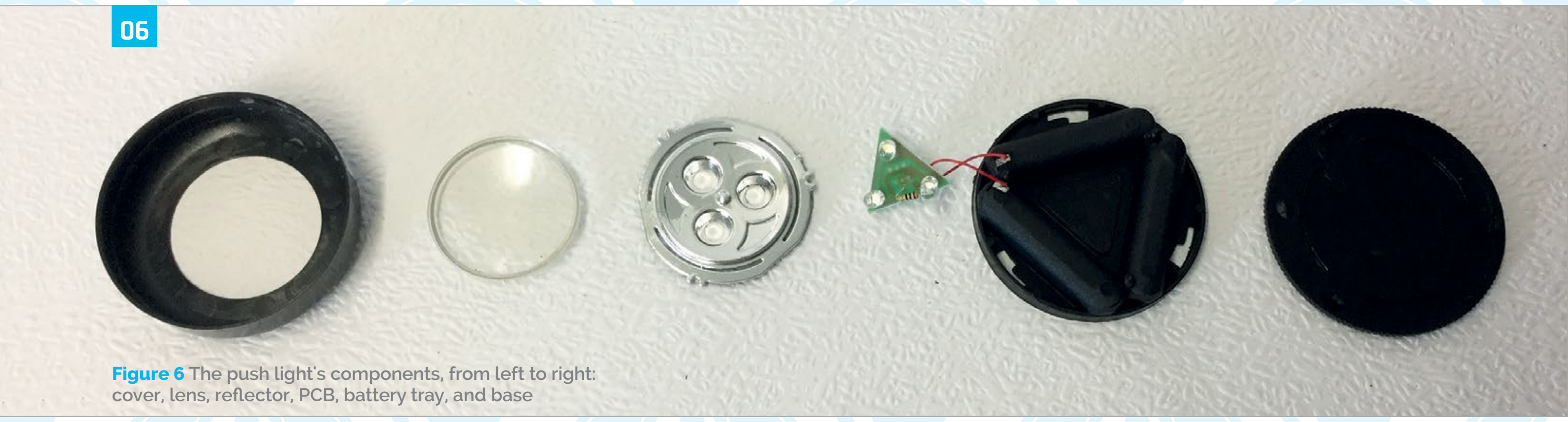


Figure 6 The push light's components, from left to right: cover, lens, reflector, PCB, battery tray, and base

07

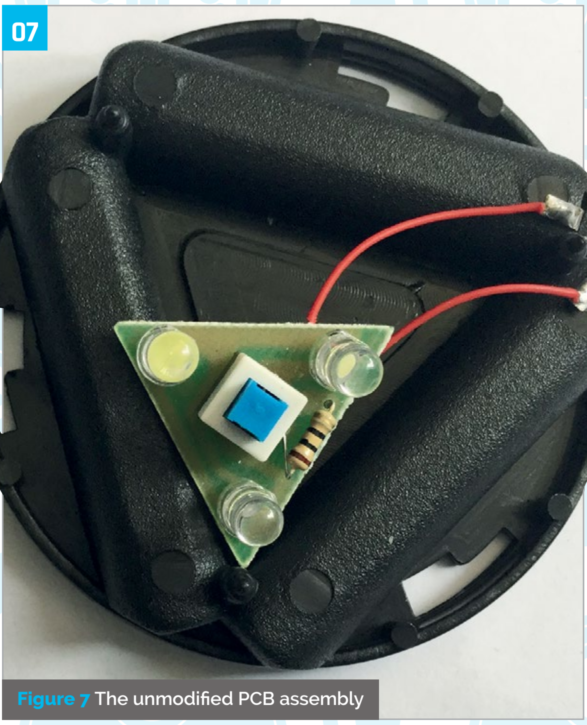



Figure 7 The unmodified PCB assembly

08

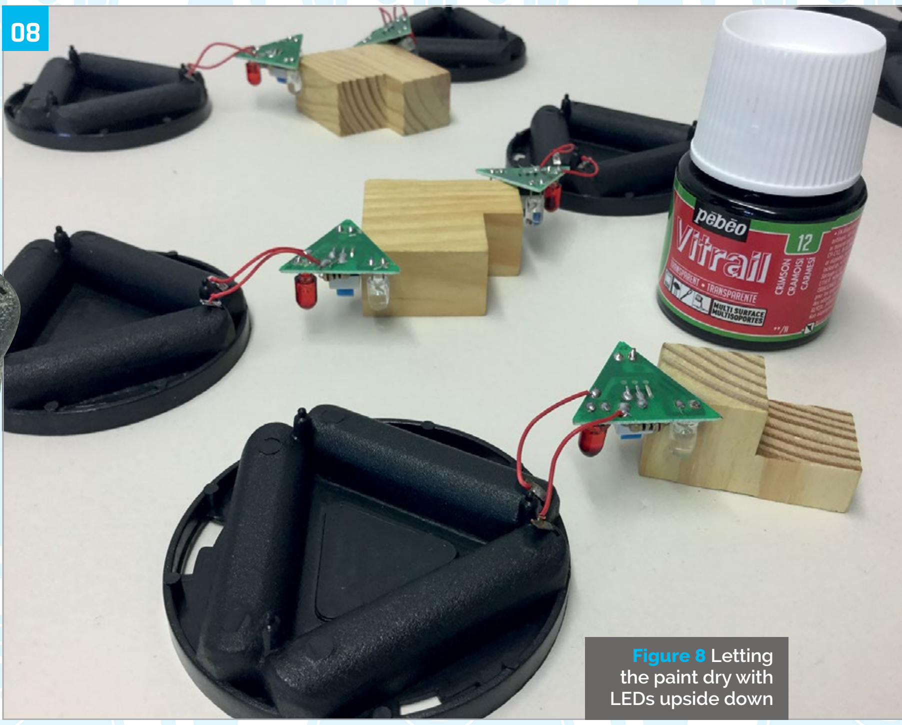



Figure 8 Letting the paint dry with LEDs upside down

until the side strips fit snugly. Now take the left-hand strip and file a shallow indent 51 mm wide and 2 mm deep in the centre of the strip. This will allow a ribbon cable to the Raspberry Pi to slip out of the box when a base is fitted. This is represented by the hatched area in **Figure 2**. Again, drill shallow indents in the strips, spray with water, then glue and clamp to form the tray.

## >STEP-06

### Assessing the lights

Take one of the LED push lights and take it apart; if you have any trouble detaching the battery tray from the base, see 'Detaching the battery tray' box. Then lay out all the parts as in **Figure 6.** To make sure we refer to them consistently, they are, from left to right: cover, lens, reflector, PCB, battery tray, and base. At this stage, the PCB and the battery tray are joined with two wires; let's keep it like this for the time being.

## >STEP-07

### Preparing the LEDs

**Figure 7** shows the PCB with the LEDs on it attached to the battery tray with two wires. This might be a good time to test the unit by inserting the three AAA batteries and making sure the central switch turns the LEDs on and off. In order to make the light displays a bit more interesting, we are going to paint each LED in the light's cluster with a transparent paint – this is normally used for making easy stained glass projects. You can get paint like this from any craft store. We chose the colours red, green, and blue, although you could use any you like.

## >STEP-08

### Painting the first LEDs

Paint the first LED – that is, the one between the wires attaching it to the battery tray – red. Make sure the paint is spread all over the LED and turn it upside down to dry. This is to ensure that any surplus paint collects over the tip of the LED, where the light is stronger. We used the base and odd bits of wood to make sure the wet paint did not touch the table – see **Figure 8**. If too much paint gathers, just touch the forming drop with a brush to drain the paint from it.

09

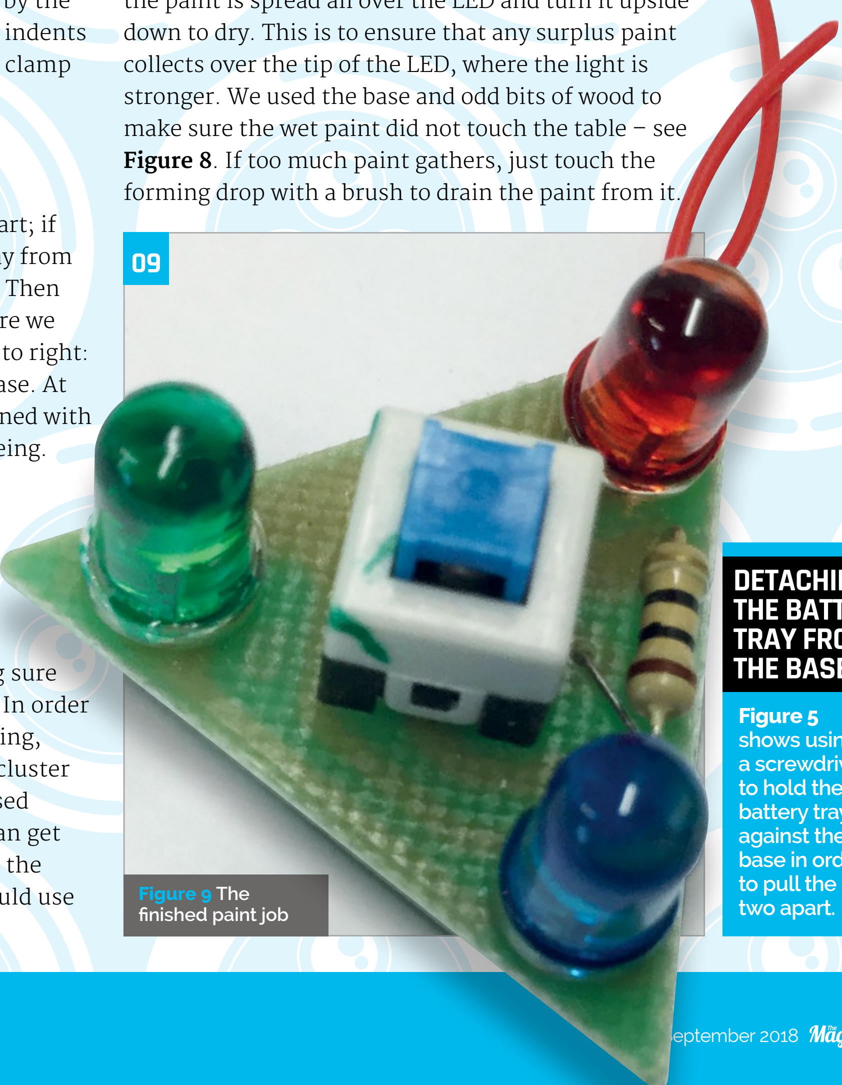



Figure 9 The finished paint job

### DETACHING THE BATTERY TRAY FROM THE BASE

**Figure 5** shows using a screwdriver to hold the battery tray against the base in order to pull the two apart.

10

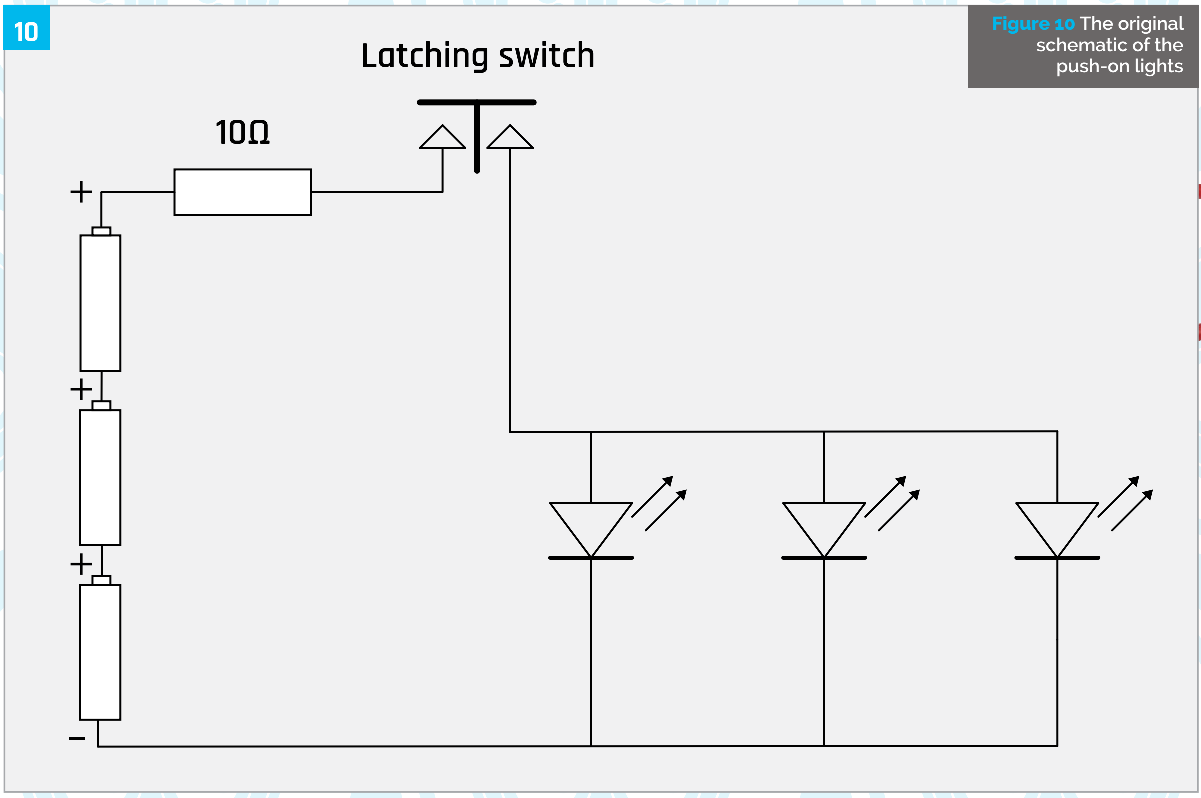


Figure 10 The original schematic of the push-on lights

## >STEP-09

**Finish off the painting**

Allow the paint to become touch-dry, which can take two hours, and then paint the LED to the left of the red one green and again allow to become touch-dry. Finally, paint the LED to the left of the red one blue and this time leave it for 24 hours to become hardened. **Figure 9** shows the finished painted LEDs on the PCB board. Again, check with batteries that the LEDs work and that there are no unpainted clear bits. At this stage, if you think the colours are washed out, you can always add a second coat of paint, although on the final project the LED brightness will be adjustable.

## >STEP-10

**Assessing the circuit**

**Figure 10** shows the schematic of the LED's PCB (printed circuit board). Trace through the circuit and make sure that your lights are wired the same. The 10 Ω

11

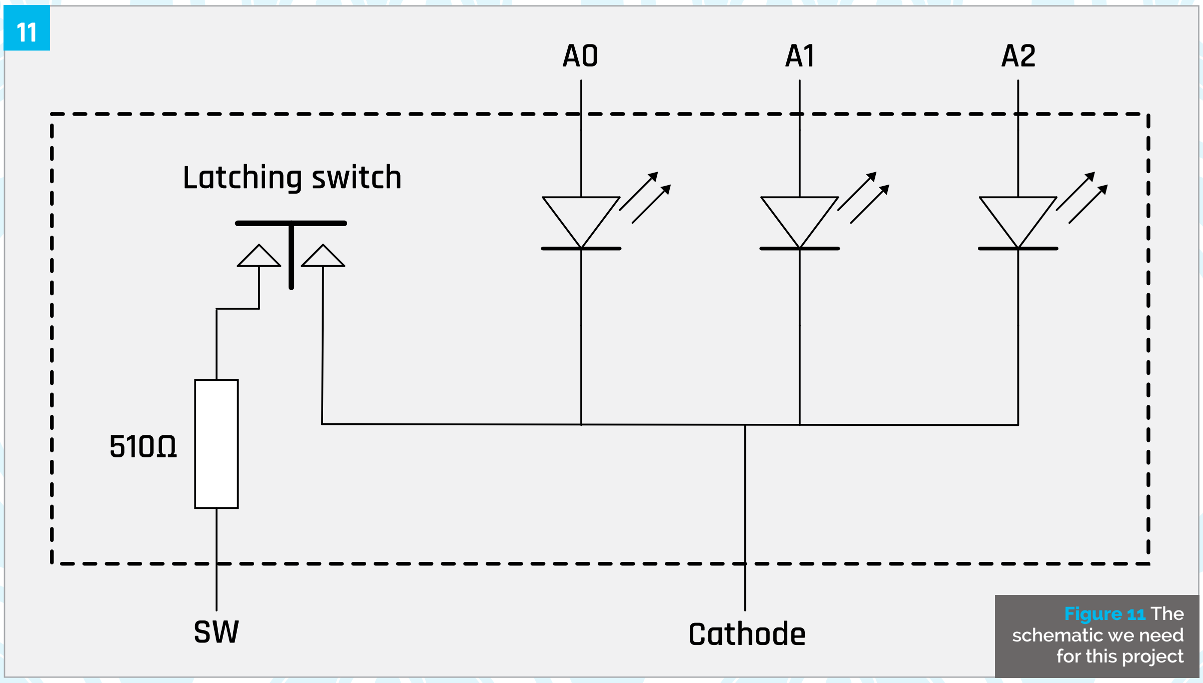


Figure 11 The schematic we need for this project

13

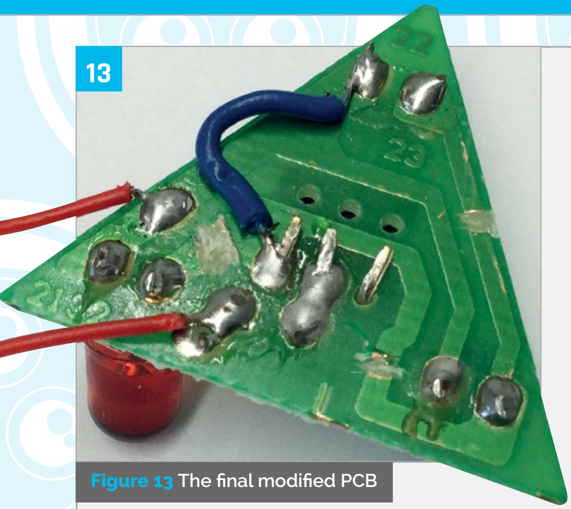


Figure 13 The final modified PCB

12

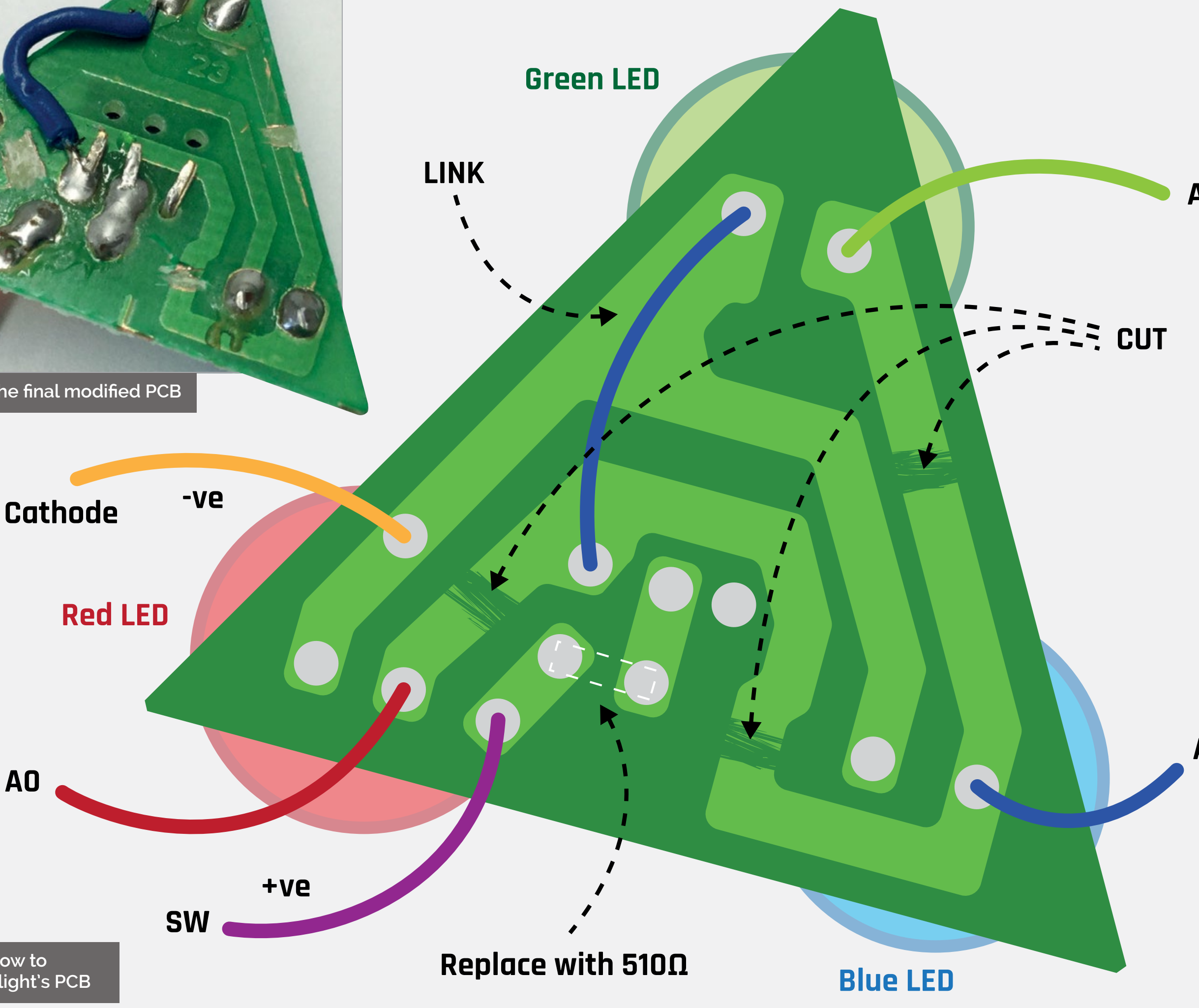


Figure 12 How to modify the light's PCB

resistor and the batteries' internal impedance act as a crude limit to the current through the LEDs. The LEDs themselves are wired in parallel, which is not good practice, and the latching switch connects the positive anode of the LEDs to the positive of the battery supply. Other makes of light might be wired differently; be cautious if your PCB is not like the one shown here.

## >STEP-11

### The circuit we want

We want to control the LEDs separately with a MAX7219 driver chip, which requires common cathode LEDs, so we must modify the PCB in order that the circuit is compatible with the MAX7219. **Figure 11** shows the schematic of what we want the PCB's circuit to be. Each LED's anode needs to be capable of a separate connection to the MAX7219, the cathodes need to be common, and the switch should connect to these cathodes through a 510 Ω resistor. This can be a replacement to the existing resistor, but if we make it a 125 mW one it will fit properly between the holes in the existing PCB.

## >STEP-12

### Modifying the PCB

**Figure 12** shows a drawing of the PCB with the modifications we need to make. We need to change the resistor, make three cuts to the PCB tracks, and wire a link between two points. At this stage we need not wire anything to the three LEDs nor the cathode or SW pads; we will do that later. Start off by removing all the resistors and replacing them with 510 Ω ones. Then you need to make three cuts in the PCB. This can be done with a scalpel, making two cuts very close together and peeling away the PCB between. But a better way is to use a Dremel cut-off wheel and just make the gentlest of touches to remove the copper strip. Finally, use a small piece of wire to link the switch to the cathode lines.

That concludes this month's article. Next month we will see how to make the control board for the matrix.

**FINISHED PCB**

**Figure 13** shows the finished modified PCB. These cuts were made with a scalpel. The PCB can now be detached from the battery tray.

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

PYGAME ZERO PART 03

# THE SCRAMBLED CAT GAME

A challenging game with minimal code using Pygame Zero Actors. In this third tutorial of the series, we introduce several new game programming techniques

**You'll Need**

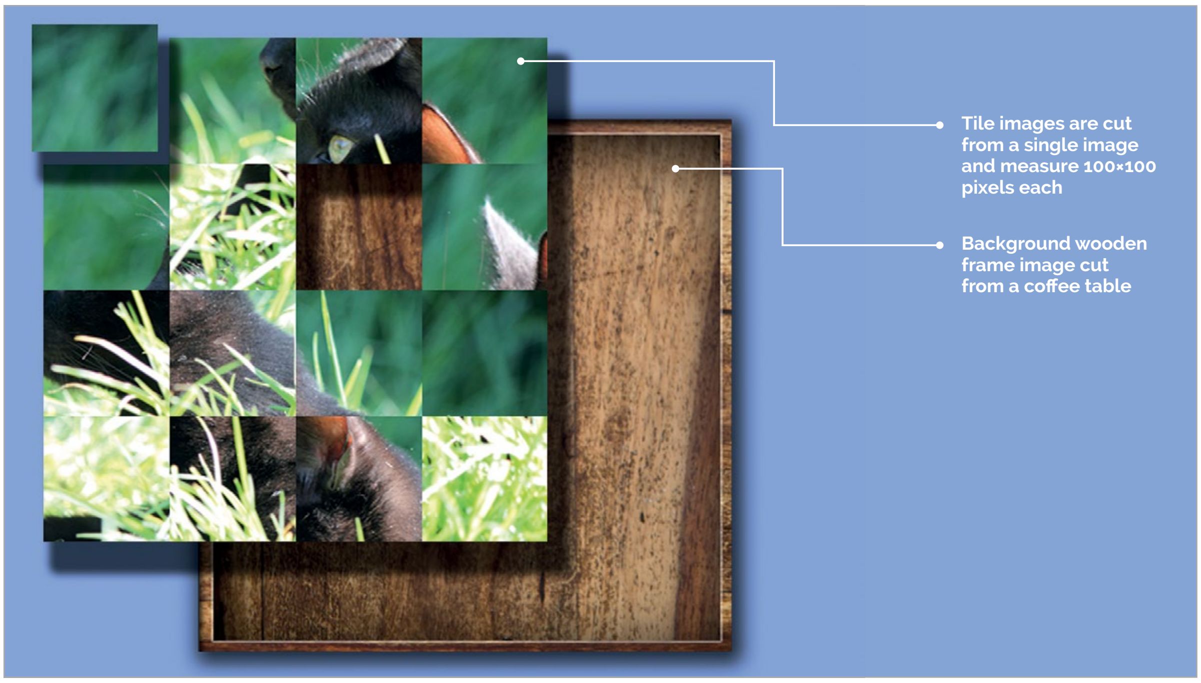
- Raspbian Jessie or newer
- An image manipulation program such as GIMP, or images from **magpi.cc/xdgVLJ**
- The latest version (1.2) of Pygame Zero
- A lot of patience to play the game

When your author first came across this type of game, it was in the form of a plastic frame with numbered tiles and was handed to him by his grandmother to keep him quiet for a while. It's an infuriating game much like a 2D forerunner to the Rubik's Cube, where the player must move jumbled up tiles around the frame to put them back into the correct order, except that there is only one spare space to move tiles into and you seem to end up with an ever more jumbled collection of tiles. In this version, the author's cat, Widdy, has very kindly donated himself to be scrambled.

## >STEP-01

**Decisions, decisions**

When first approaching a sliding tile game, our first thought would be to have a matrix variable (or two-dimensional list) representing the frame with its tiles. You would then have a process function to handle the tiles moving around, changing the values in each of the matrix positions. However, Pygame Zero has a few tricks up its sleeve which work very well in this circumstance. Pygame Zero Actors are your friends and can save you a huge wedge of coding time. Let's get stuck in and set


Tile images are cut from a single image and measure 100×100 pixels each
Background wooden frame image cut from a coffee table

You can find suitable images for games around the house. For this game we used a cat and a coffee table

### WRITING YOUR PROGRAM STRUCTURE

When creating your own programs, you can start by writing a structure of empty functions before writing the detailed code. This can help to visualise the way the program will work.

your program up with a few standard Pygame Zero basics, as in **Figure 1**.

## >STEP-02
### Getting the groundwork done

Before we get into the visuals, you may notice that there are several functions in **Figure 1** with just the keyword **`pass`** in them. In Python we are not allowed to have an empty function, so if we write **`pass`** in it, it means that the function should do nothing. This may sound a bit strange but it means that we can build the basic structure of our program before we start thinking about the finer details. Now that we have our structure, we can add some background graphics. If you have downloaded the graphics from our GitHub link (**magpi.cc/xdgVlJ**), we can add a background colour with **`screen.fill(red, green, blue)`** and then 'blit' a frame using **`screen.blit('board', (150, 50))`** in the **`draw()`** function.

## >STEP-03
### A night on the tiles

You may have noticed the rather stylish wooden frame that is now displayed if you run the program. It was carved out of the author's coffee table for this tutorial and will serve as the holder for the tiles we are about to create. All we need to do to create the tiles is to define a list to hold the tile Actors and then a function to create and arrange the new Actors in the frame. We can do this with a double loop to set the x and y co-ordinates of each tile. So first, define the tile list near the top of the code using **`tileList = []`** and then a function to make the tiles. Have a look at **Figure 2** (overleaf) to see how we can do that. We can call our **`makeTiles()`** function at the end of the code, just before **`pgzrun.go()`**.

## >STEP-04
### Stating the obvious

One technique that we have used in both previous tutorials in this series is to use a variable to keep track of the overall state of the game, and this game is no different. We need to know if the player is allowed to interact with the tiles or if they have completed the puzzle and a few other things too. For this we can define a global variable **`gameStatus = 0`** near the top

### Figure1.py

Language >PYTHON

DOWNLOAD: magpi.cc/xSnagK

```python
import pgzrun
WIDTH = 800
HEIGHT = 600
gameStatus = 0

def draw(): # Pygame Zero draw function
    pass

def update(): # Pygame Zero update function
    pass

def on_mouse_down(pos):
    pass

pgzrun.go()
```

**Figure 1** The basic setup for our Pygame Zero program

## COLLISION TESTING

There are several ways to test collisions. For our border tests, we could have used border Actors and tested collision with them instead of using a mathematical check.

of our program. In this game we will use the value 0 to allow the player to interact with the tiles, then 1 to show that a tile is being moved, 2 will mean that we are preparing the game (more on this later), and 3 will show that the player has completed the puzzle.

## >STEP-05

### Point and click

In this game we will be using two different types of user input to give the player a choice of how to control the tiles. Players will be able to click on a tile to get it to move into the spare space, but we will provide the option to use the arrow keys too. These two input methods work quite differently, so let's look at mouse input first. Pygame Zero provides a function **on_mouse_down(pos)** so that we can capture a mouse click. We then compare the co-ordinates in the variable **pos** to the tiles to see if they collide. If they do, then the player has clicked on a tile.

## >STEP-06

### Which tile?

To check which tile was clicked, we can use a **for** loop. We know how many tiles we have (15), so we can say **for t in range(15):**, where the variable **t** will be our counter. We can then check each tile to see if it has been clicked, with **if tileList[t].collidepoint(pos):**. While we are dealing with moving the tile, it's a good idea to stop the player from interacting with the game, otherwise we may get several tiles trying to move at once. We can change the **gameStatus** variable to 1 to lock the input. See **Figure 3** to see how this works in code.

## >STEP-07

### Let's get things moving

You may notice in **Figure 3**, in the middle of the **for** loop there is a call to **moveTile(tileList[t])**. This is a function that will determine which way a tile can move. We are going to use the **colliderect()** method of our tile Actors to test all directions. If no collision is detected in a certain direction then we know that we can move the tile that way. We will also include a condition to test that the tiles cannot move outside the frame, so we will be using two types of collision detection: one with the built-in Pygame Zero Actor objects and one with a boundary check.

## >STEP-08

### Repel borders!

Sorry for that terrible pun, but first we need to stop the tiles being allowed through the frame border. We can do this with a simple **if** condition for each direction. So for the right border we would say **if(tile.x < borderRight):** and then we would do our Actor collision test. We will need a border test for each direction. What we are actually doing in the **moveTile()** function is saying: "Can we move the tile right, or left, or up, or down?" If the answer to any of those is yes (and it can only be one of them or none), then tell us which way to move.

## >STEP-09

### Tiles to the left, tiles to the right

There is a cunning plan you can use to test if a tile can move in a certain direction without seeing it move. If you add 1 to the x co-ordinate of the tile and then test for a collision – we are going to use a function called **checkCollide(tile)** – then we will know if a tile is to the right of our test tile. We then set the tile back to its original position by subtracting 1 from the x co-ordinate, and all this happens before the Pygame

## Figure2.py

```
def makeTiles():
    global tileList
    xoffset = 251
    yoffset = 151
    x = y = c = 0
    while y < 4:
        while x < 4:
            if(c < 15):
                tileList.append(Actor("img"+str(c),
pos = (xoffset+(x*100),yoffset+(y*100))))
            c += 1
            x += 1
        x = 0
        y += 1
```

**Figure 2** The makeTiles() function. We loop from 0–3 using y and inside the loop from 0–3 using x

## Figure3.py

**Figure 3** Using a locking system while player inputs are being processed

```
def on_mouse_down(pos):
    if (gameStatus == 0):
        doLock()
        for t in range(15):
            if tileList[t].collidepoint(pos):
                m = moveTile(tileList[t])
                if(m != False):
                    animate(tileList[t],on_
finished=releaseLock, pos=(tileList[t].x+m[1],
tileList[t].y+m[2]))
                    return True
        releaseLock()

def releaseLock():
    global gameStatus
    gameStatus = 0

def doLock():
    global gameStatus
    gameStatus = 1
```

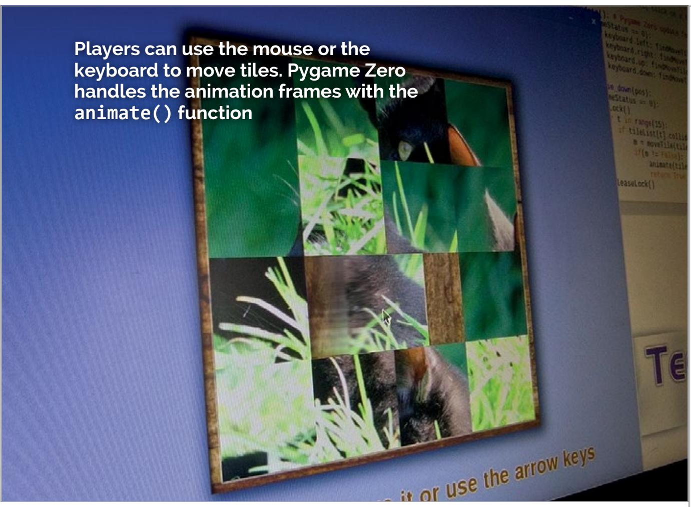

Players can use the mouse or the keyboard to move tiles. Pygame Zero handles the animation frames with the `animate()` function

**Figure4.py**

**Figure 4** An example of testing border collision and then using the Actor collision detection

```python
def moveTile(tile):
    borderRight = 551
    rValue = False
    if(tile.x < borderRight): # can we go right?
        tile.x += 1
        if(not checkCollide(tile)): rValue = "right", 100, 0
        tile.x -= 1

    return rValue

def checkCollide(tile):
    for t in range(15):
        if tile.colliderect(tileList[t]) and tile != tileList[t]: return True
    return False
```

Zero **`draw()`** function is called again so you never see anything move. We can also do the same test for moving right, up, and down.

## >STEP-10

### Collision course

We need to write a function that will test to see if a collision has happened when we moved our tile 1 unit (pixel). The **`checkCollide()`** function loops through the tile list and checks to see if any of the tiles are colliding with the tile that the player has clicked on. We just loop through the tile list and use the **`colliderect()`** method to test for collision (and also make sure we are not testing the tile that has been clicked) and if there is, return **`True`**. If no collision was detected then the function will exit with a **`False`** return value. Have a look at **Figure 4** to see how we test a border and check for tile collision.

## >STEP-11

### Very animated

We will need to add tests for left, up, and down to our **`moveTile()`** function; when that is done, it will return what is known as a tuple. This is a return value with several parts: the direction as a string, the x offset to move the tile, and the y offset to move the tile. This tuple gets sent back to our **`on_mouse_down()`** function and if it's not **`False`** then we know we can move the tile based on the return values. We now call the **`animate()`** function. This is a Pygame Zero function to move an Actor from one place to another.

## >STEP-12

### Locking player interaction

While we are animating the tile moving, we don't want the player to be able to click on any other tiles. We have used our **`doLock()`** function to change the **`gameStatus`** to 1 so no mouse clicks are reacted to. When the animation has finished, we want the player to make their next move, so in our call to **`animate()`** we include **`on_finished=releaseLock`**. This will call the function **`releaseLock()`** which will set the **`gameStatus`** back to 0. This will mean that the player can click on another tile. You will notice from **Figure 3** that if the mouse click is not on a tile then the lock is released at the end of the function.

## >STEP-13

### Using the arrows

The second way that we can capture the player's input is to use the arrow keys. This form of input relies on the fact that if an arrow key is pressed (for example the up arrow), there is only one tile that is able to move in that direction – or alternatively, no tiles are able to move in that direction. So all we need to do is work out which tile can move in the direction of the arrow pressed. We are going to do this with a function called **`findMoveTile(moveDirection)`** and we pass it the direction of movement that we want it to look for.

## >STEP-14

### Scanning the keyboard

First, we must check if the player is pressing one of the arrow keys. We do this in our **`update()`** function. We want to check if the **`gameStatus`** is 0 before processing anything; if it is, we can check the left arrow key by writing **`if keyboard.left: findMoveTile("left")`**. We then write a similar line of code for each of the other arrow keys, passing a different string to our **`findMoveTile()`** function. In the latter function we use the same **`moveTile()`** function as we did with the mouse click input, but this time we check for the direction that has been passed by the keyboard press check.

## >STEP-15

### Which tile is it?

Now all we need to do is write the **`findMoveTile()`** function that scans through the tile Actors and

**MULTI-PURPOSE FUNCTIONS**

Try to write your functions so that they can be used in different ways, like our `moveTile()` function. If there is less code to write, there is less code to debug.

**WATCH OUT FOR MULTIPLE INPUT**

When using the Pygame Zero keyboard object to test for key presses, it doesn't return just a single event. It will continuously read as True while the player holds the key, which may not be what you are expecting.

## Figure5.py

**Figure 5** Moving tiles from the detection of a key press on the arrow keys

```python
def update(): # Pygame Zero update function
    if (gameStatus == 0):
        if keyboard.left: findMoveTile("left")
        if keyboard.right: findMoveTile("right")
        if keyboard.up: findMoveTile("up")
        if keyboard.down: findMoveTile("down")

def findMoveTile(moveDirection):
    doLock()
    for t in range(15):
        m = moveTile(tileList[t])
        if(m != False):
            if(m[0] == moveDirection):
                animate(tileList[t],on_finished=releaseLock, pos=(tileList[t].x+m[1], tileList[t].y+m[2]))
                return True
    releaseLock()
    return False
```

## Figure6.py

**Figure 6** The `scrambleCat()` function. Pass a list of simulated keystrokes to `findMoveTile()`, just like if we were pressing the keys

```python
scrambleCountdown = 30
scrambleList = [2, 0, 2, 0, 3, 1, 1, 1, 3, 0, 0, 2, 1, 2, 1, 3, 3, 0, 3, 0, 2, 0, 2, 2, 1, 3, 1, 3, 3, 1, 2]


def scrambleCat():
    global gameStatus, scrambleCountdown, scrambleList
    tileDirs = ["left", "right", "up", "down"]
    if(scrambleCountdown > 0):
        mt = False
        while(mt == False):
            mt = findMoveTile(tileDirs[scrambleList[scrambleCountdown]])
            scrambleCountdown -= 1
        gameStatus = 2
    else:
        gameStatus = 0
```

finds the one that can move in the direction that the player has pressed. We loop though our tile list and try to move each tile. If we find it can move then we check the direction of movement. If it matches the direction that we are looking for then we have a match and we can initiate the animation to move the tile. **Figure 5** shows this whole process, from key press to animation. Note that we are locking and unlocking the **`gameStatus`** while this is happening, to avoid multiple moves at once.

## >STEP-16

### Time to scramble the cat

The game is not much fun if we start with a completed puzzle, so we need to add a system for mixing up the tiles. There are various ways we could do this, but we thought it might be quite nice if we start the game with the tiles being scrambled move by move. We can do this by simulating key presses using the same functions as in the previous steps. We can use a new **`gameStatus`** of 2 to indicate that the player can't interact, then cycle through a predefined or random set of simulated key presses until everything is jumbled up. We have used a predefined list in this case.

## >STEP-17

### How scrambled is scrambled?

For our scrambling, we can have a list of movements. If we alter the number of movements we are using then the game becomes more or less difficult. We can start at 30 and see how that works. Our scrambling function, **`scrambleCat()`**, calls the **`findMoveTile()`** function and uses the **`scrambleCountdown`** variable to check if we have made enough moves. After each animation, we are using a bit of a cunning trick: we always call the **`releaseLock()`** function after an animation so we can slip in a test to see if gameStatus is 2 and if so, call **`scrambleCat()`** again. See **Figure 6** for the **`scrambleCat()`** function.

## >STEP-18

### Have we won yet?

At some point in the game, there is a slim chance that the player will actually rearrange the tiles back into the correct order. We will need to write a check to see if this has happened each time a tile has been moved. We know that the positions of the tiles were correct before we scrambled them, so what we can do is make a list of the x and y co-ordinates of each tile when we first make the tile Actors. Then all we need to do is to compare that list with the current x and y values of our tile Actors and if they all match, we have a winner!

## >STEP-19

### Finishing touches

You can see from the full program listing the **`checkSuccess()`** function and how that works, and we can also add some text prompts into our **`draw()`** function based on the value of **`gameStatus`**. That's about it! We have a working Scrambled Cat game. You may want to add some features such as a score based on how many moves the player makes, or you may want to change the images. You could have scrambled egg or a scrambler bike or make the tile matrix larger for an even more difficult challenge.

## scrambledcat.py

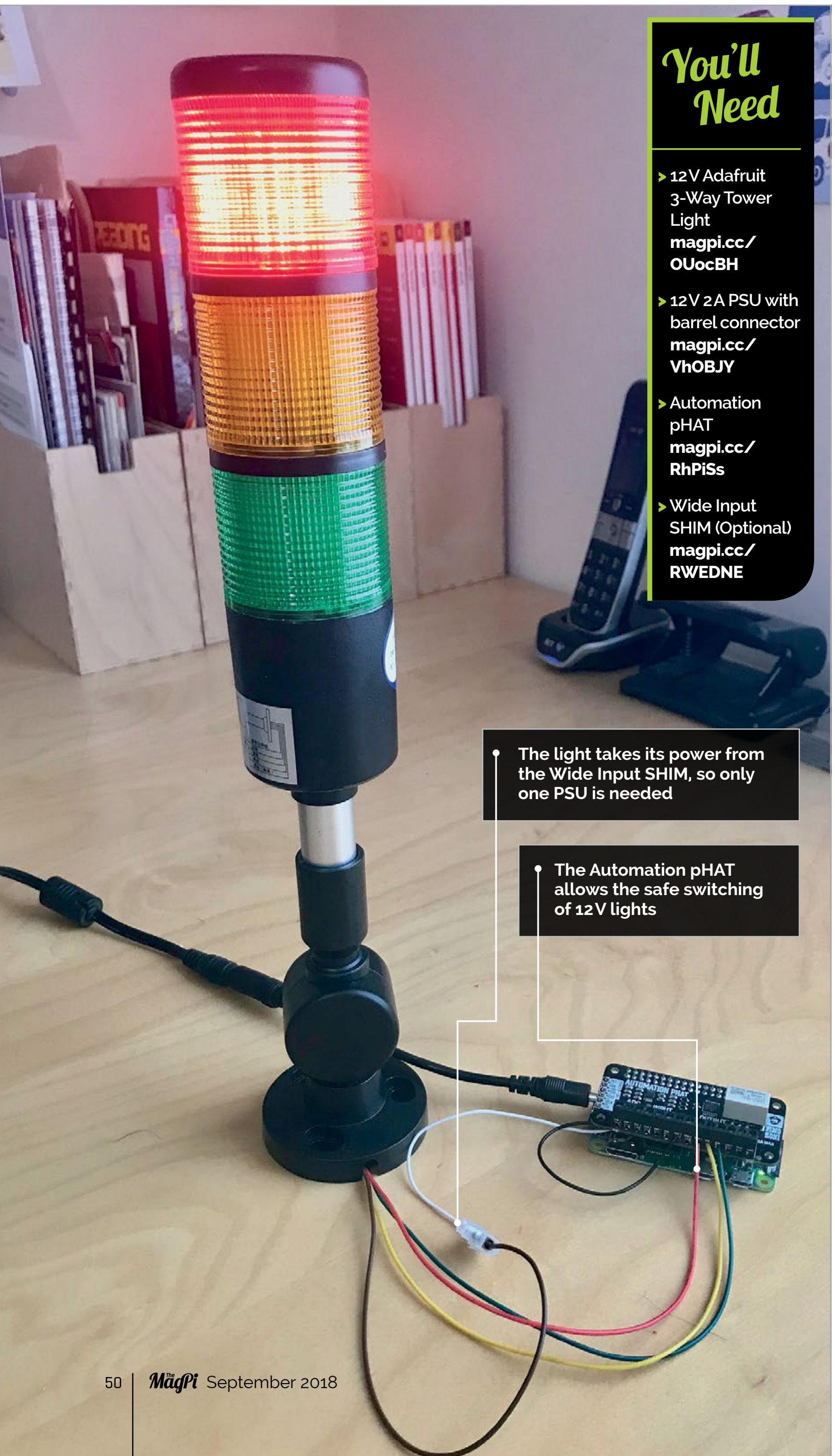
```python
import pgzrun
WIDTH = 800
HEIGHT = 600
gameStatus = 0
tileList = []
correctList = []
scrambleCountdown = 30
scrambleList = [2, 0, 2, 0, 3, 1, 1, 1, 3, 0, 0, 2, 1, 2, 1, 3, 3, 0, 3, 0, 2, 0, 2, 2, 1, 3, 1, 3, 3, 1, 2]

def draw(): # Pygame Zero draw function
    global gameStatus
    screen.fill((141, 172, 242))
    screen.blit('board', (150, 50))
    for t in range(15):
        tileList[t].draw()
    if (gameStatus == 3): screen.draw.text("Success!", (315, 20), owidth=0.5, ocolor=(255,255,255), color=(128,64,0) , fontsize=60)
    if (gameStatus == 2): screen.draw.text("Please wait while we scramble the cat", (135, 540), owidth=0.5, ocolor=(255,255,255), color=(128,64,0) , fontsize=40)
    if (gameStatus <= 1): screen.draw.text("Click on a tile to move it or use the arrow keys", (95, 540), owidth=0.5, ocolor=(255,255,255), color=(128,64,0) , fontsize=40)

def update(): # Pygame Zero update function
    if (gameStatus == 0):
        if keyboard.left: findMoveTile("left")
        if keyboard.right: findMoveTile("right")
        if keyboard.up: findMoveTile("up")
        if keyboard.down: findMoveTile("down")

def on_mouse_down(pos):
    if (gameStatus == 0):
        doLock()
        for t in range(15):
            if tileList[t].collidepoint(pos):
                m = moveTile(tileList[t])
                if(m != False):
                    animate(tileList[t],on_finished=releaseLock, pos=(tileList[t].x+m[1], tileList[t].y+m[2]))
                    return True
        releaseLock()

def findMoveTile(moveDirection):
    doLock()
    for t in range(15):
        m = moveTile(tileList[t])
        if(m != False):
            if(m[0] == moveDirection):
                animate(tileList[t],on_finished=releaseLock, pos=(tileList[t].x+m[1], tileList[t].y+m[2]))
                return True
    releaseLock()
    return False

def releaseLock():
    global gameStatus
    if(gameStatus == 2): scrambleCat()
    else: gameStatus = checkSuccess()

def doLock():
    global gameStatus
    gameStatus = 1

def checkSuccess():
    for t in range(15):
        if(tileList[t].x != correctList[t][0] or tileList[t].y != correctList[t][1]):
            return 0
    return 3    # we have success!

def makeTiles():
    global tileList, correctList
    xoffset = 251
    yoffset = 151
    x = y = c = 0
    while y < 4:
        while x < 4:
            if(c < 15):
                tileList.append(Actor("img"+str(c), pos = (xoffset+(x*100),yoffset+(y*100))))
                correctList.append((xoffset+(x*100),yoffset+(y*100)))
            c += 1
            x += 1
        x = 0
        y += 1
    scrambleCat()

def scrambleCat():
    global gameStatus, scrambleCountdown, scrambleList
    tileDirs = ["left", "right", "up", "down"]
    if(scrambleCountdown > 0):
        mt = False
        while(mt == False):
            mt = findMoveTile(tileDirs[scrambleList[scrambleCountdown]])
            scrambleCountdown -= 1
        gameStatus = 2
    else:
        gameStatus = 0

def moveTile(tile):
    borderRight = 551
    borderLeft = 251
    borderTop = 151
    borderBottom = 451
    rValue = False
    if(tile.x < borderRight): # can we go right?
        tile.x += 1
        if(not checkCollide(tile)): rValue = "right", 100, 0
        tile.x -= 1
    if(tile.x > borderLeft): # can we go left?
        tile.x -= 1
        if(not checkCollide(tile)): rValue = "left", -100, 0
        tile.x += 1
    if(tile.y < borderBottom): # can we go down?
        tile.y += 1
        if(not checkCollide(tile)): rValue = "down", 0, 100
        tile.y -= 1
    if(tile.y > borderTop): # can we go up?
        tile.y -= 1
        if(not checkCollide(tile)): rValue = "up", 0, -100
        tile.y += 1
    return rValue

def checkCollide(tile):
    for t in range(15):
        if tile.colliderect(tileList[t]) and tile != tileList[t]: return True
    return False

makeTiles()
pgzrun.go()
```

**PJ EVANS**

PJ is a software engineer who runs the Milton Keynes Raspberry Jam. His home automation projects have got completely out of hand.

**mrpjevans.com**
**twitter.com/mrpjevans**

# TEENAGE DINNER KLAXON



The light takes its power from the Wide Input SHIM, so only one PSU is needed

The Automation pHAT allows the safe switching of 12 V lights

**You'll Need**

- 12 V Adafruit 3-Way Tower Light **magpi.cc/OUocBH**
- 12 V 2 A PSU with barrel connector **magpi.cc/VhOBJY**
- Automation pHAT **magpi.cc/RhPiSs**
- Wide Input SHIM (Optional) **magpi.cc/RWEDNE**

Always yelling 'Dinner!' to a teenager or shed-dweller? Save your vocal cords by installing a tower light you can control from your phone

If you're blessed with teenagers or others who like to lock themselves away in their room/shed/craft-room, you may well be fed up with shouting up the stairs or out into the garden when you need their attention, especially if headphones are involved. Why not signal your loved ones visually from your phone using a cheap industrial tower light, a Raspberry Pi, and some Python? You'll learn about interfacing the Pi with normally incompatible devices and how little code is required to create a web server.

This project was inspired by James West's klaxon: **jameswest.site/the-tea-time-klaxon**.

## >STEP-01

### Get your Pi ready

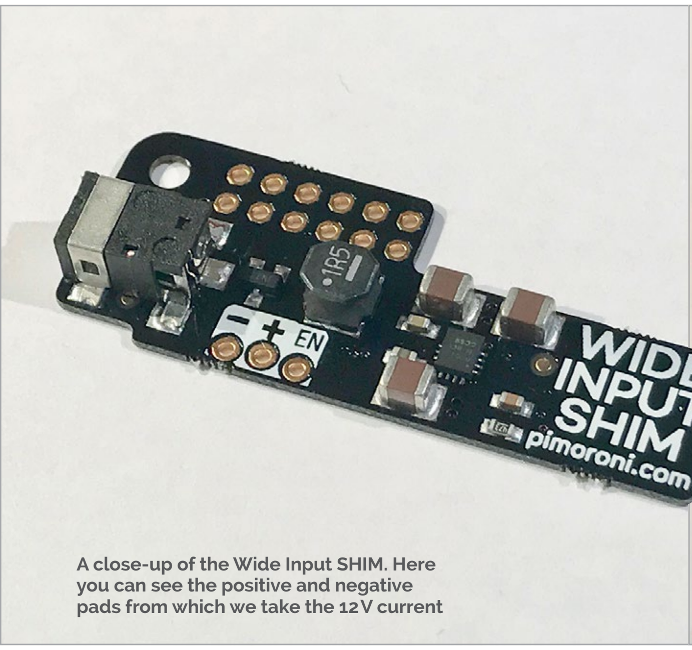
This project can be built using any 40-pin version of the Raspberry Pi capable of WiFi and it's perfect for the Zero W. Once complete, the tower can run 'headless', so no monitor or keyboard will be required. Raspbian Lite is ideal for this as there is no need for a fancy user interface.

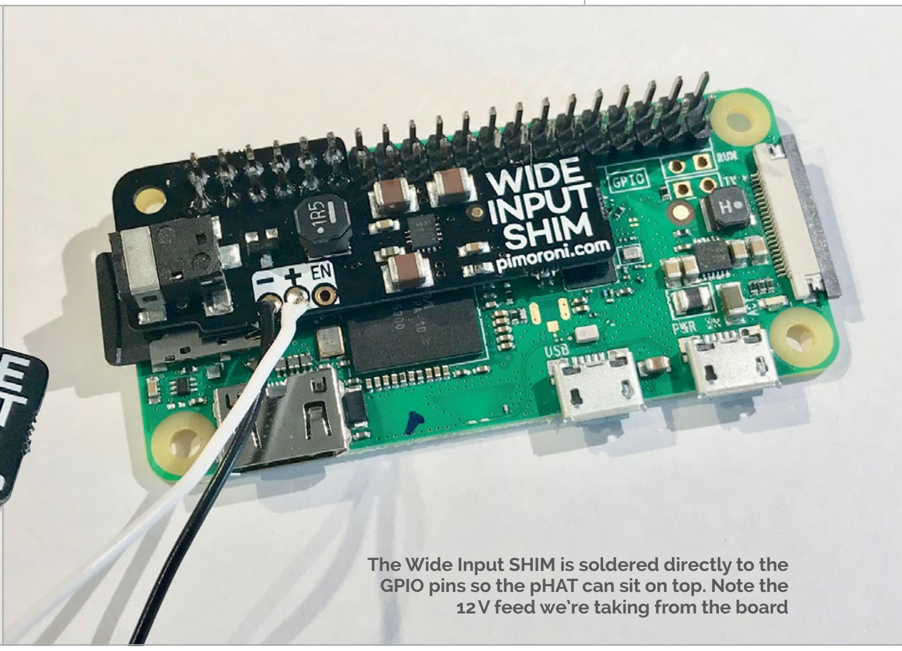
Start by installing Raspbian Lite on your microSD card (see **magpi.cc/etcher**), then booting up and running **`sudo raspi-config`**. Get WiFi set up and under 'Interfacing', make sure both I²C and SSH are enabled. Finally, set the Pi's host name under 'Networking' as we'll need it later on.

## >STEP-02

### Prepare the Wide Input SHIM

So we can safely power the lights and the Raspberry Pi from a single 12 V PSU, we're using the Pimoroni Wide Input SHIM, a tiny power regulator for the Pi. To power the lights, we're going 'tap' the SHIM to get a 12 V feed by using the on-board power inputs as a supply. Carefully solder two wires to the positive and negative pads on the SHIM. We don't want them shorting out on the Pi when the SHIM is in place,


A close-up of the Wide Input SHIM. Here you can see the positive and negative pads from which we take the 12 V current


The Wide Input SHIM is soldered directly to the GPIO pins so the pHAT can sit on top. Note the 12 V feed we're taking from the board

so make sure the solder points are trimmed back as much as possible and covered with insulation tape.

## >STEP-03
### Install the Wide Input SHIM

The Wide Input SHIM mounts on the first twelve pins of the GPIO. It is intended to sit flush at the base of the pins, so HATs can still be used on top. Therefore, the board needs to be precisely soldered directly on to the GPIO pins.

Place the SHIM onto the GPIO pins, aligning with physical pin 1 (see photo). Secure with masking tape and solder into place starting with pins 1 and 12, then you can remove the tape. Take your time. You should now be able to power up your Pi using the barrel connector and 12 V supply.

## >STEP-04
### Add the Automation pHAT

Although we can now power both the Pi and the lights from a single 12 V supply, the Pi cannot switch the lights directly without releasing the magic smoke. Pimoroni's Automation pHAT is a general-purpose collection of inputs and outputs for projects such as this one. For our lights, it features three switchable outputs that can handle 12 V just fine.

You'll need to solder the supplied GPIO connector and the screw terminals to the pHAT. Check and check again that the orientation is correct. Once ready, attach the pHAT to the Pi with the Wide Input SHIM sitting in-between.

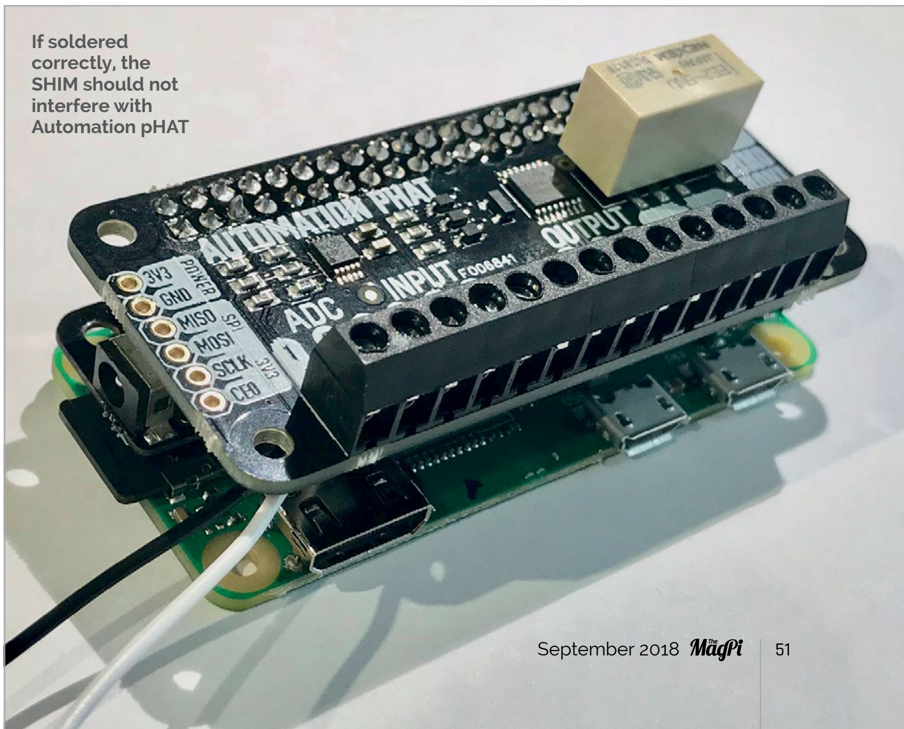
## >STEP-05
### Wire up the tower light

Now our control device is complete, we can attach the tower light itself. There are five wires: one for each colour, a buzzer (that we're not using), and 12 V in. Following the diagram, connect the wires to the Automation pHAT. The brown wire is the power in, so we connect that to the 12 V feed we took from the SHIM earlier. The three colour wires (red, yellow, and green) are connected to the three screw terminals labelled 'Output Sinking 24 V' on the pHAT. Finally, the earth wire from the SHIM needs to connect to the earth on the pHAT.

## >STEP-06
### Install Automation pHAT software

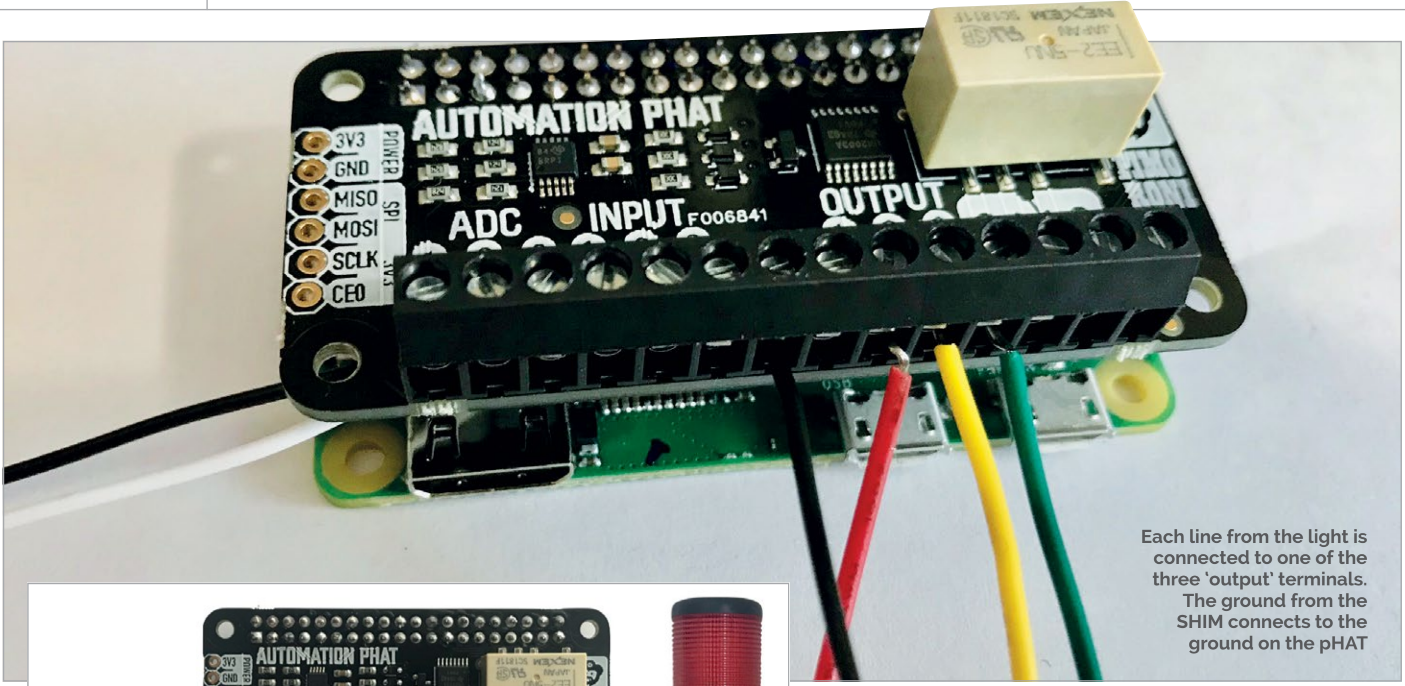
Your hardware is ready, so it's time to get some code running. Power up and, from the command line, make sure you're up to date with **`sudo apt update && sudo apt upgrade`**. Pimoroni takes away all the pain of installing drivers for the Automation pHAT by providing a simple one-line install:

```
  curl https://get.pimoroni.com/
automationhat | bash
```

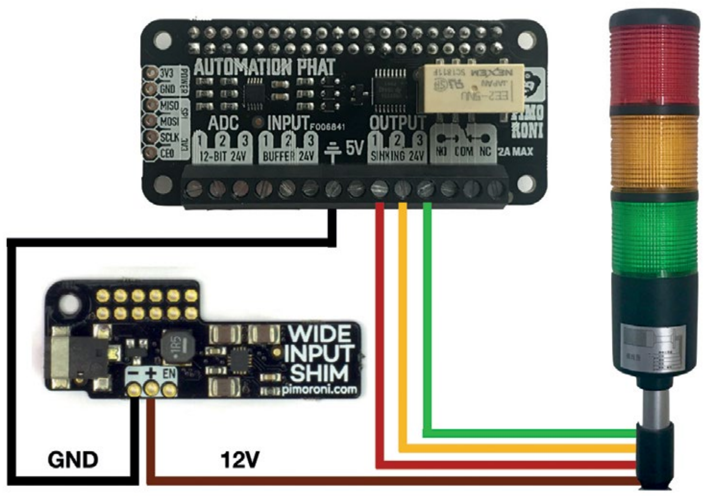

If soldered correctly, the SHIM should not interfere with Automation pHAT

**SOLDERING THE SHIM**

When attaching the SHIM, use a thin tip on your iron and be very careful not to touch components on the Pi.



Each line from the light is connected to one of the three 'output' terminals. The ground from the SHIM connects to the ground on the pHAT



Here's how all our devices connect. The SHIM provides the power, then the Automation pHAT safely switches the 12 V current

The script takes a few minutes to run. Once completed, allow it to reboot the Pi (if required) and then you're ready to try things out.

```
cd ~/Pimoroni/automationhat/examples
python output.py
```

Do you see two blinking lights on the tower? If so, it's all working. Press **CTRL+C** to stop.

## >STEP-07

### Install Flask

There are many options for remotely controlling the lights, such as Twitter or even email. One of the easiest is to run up a simple web server on the Raspberry Pi itself. To keep the code as simple as possible, we'll use the Flask web application framework for Python, which does all the heavy lifting for us. Installation is as simple as:

```
sudo -H pip install flask
```

**CAREFUL WITH THAT POWER!**

When using the SHIM, do not attempt to power the Pi using a regular USB supply at the same time.

## >STEP-08

### Code!

From the **/home/pi** directory (if you're not sure, just type **cd** followed by **RETURN**), issue the following command:

```
mkdir ~/towerlight && cd ~/towerlight
```

Now create a new file:

```
nano towerlight.py
```

Enter the code shown here (or you can download it from GitHub). Save the file (**CTRL+X**, then **Y**). When run, this script will fire up a basic web server.

The code uses the Flask and Automation pHAT Python libraries to allow simple switching of the output terminals, allowing the individual lights on the tower to illuminate. Every time a web request comes in, all the outputs are switched off, then the selected one is illuminated.

## >STEP-09

### Testing time!

To start the web server, just run:

```
python ~/towerlight/towerlight.py
```

You'll see a few announcements on screen and then it will wait for requests. Using the host name you set earlier, or the IP address of the Raspberry Pi (you can run **ifconfig** to discover this), go to **http://hostname.local:5000/** or **http://ip-address:5000/** from any web browser. If the '.local' address does not work and you're on Windows, you may need to install Apple's Bonjour Services (**magpi.cc/JSUBrT**).

The simple website you should now see allows you to switch each light on and off by clicking the links.

## >STEP-10

### Start on boot

To start the web server whenever we boot the Pi, we need to create a service file:

```
sudo nano /lib/systemd/system/towerlight.service
```

And populate it as follows:

```
[Unit]
Description=TowerLight
After=multi-user.target

[Service]
Type=idle
ExecStart=/usr/bin/python /home/pi/towerlight/towerlight.py

[Install]
WantedBy=multi-user.target
```

Now, install the service:

```
sudo chmod 644 /lib/systemd/system/towerlight.service
sudo systemctl daemon-reload
sudo systemctl enable towerlight.service
```

The server will now start automatically on reboot.

## >STEP-11

### Assemble!

Once you're happy with testing, you can shorten the leads as you see fit. It's straightforward to mount the tower on top of a small project box and feed the wires through. Mount the Pi inside with hot glue or sticky pads, with the large barrel connector lead supplied with the SHIM accessible from the outside.

Congratulations, you now have the perfect teenager/shed-dweller alerting device in a single unit that can be controlled from your phone or any web browser.

## >STEP-12

### Make it your own

What we've made here is fairly basic, so there's lots to build on. Why not try improving the website and learning about Flask templates? How about a disco mode? The tower comes with a (very loud) buzzer; could you add it in? What are the other things that could trigger the lights? Twitter? A remote button? Also, don't forget there are plenty of inputs to play with on the Automation pHAT, so the lights could also react to sensors. Check out **magpi.cc/TSEFxL** for an alternative version of the klaxon software with extra features.

## towerlight.py

Language >PYTHON

DOWNLOAD: magpi.cc/Uayjbk

```python
import automationhat
import time
from flask import Flask
app = Flask(__name__)

# Generate a simple HTML page. You could also use Flash's built-in Jinja Templating.
def makePage(body):
    return '''
    <!DOCTYPE html>
    <html>
        <head>
            <meta name="viewport" content="width=device-width, initial-scale=1.0">
        </head>
        <body>''' + body + '''
        <hr/>
        <ul>
            <li><a href="red">Red</a></li>
            <li><a href="yellow">Yellow</a></li>
            <li><a href="green">Green</a></li>
            <li><a href="off">All Off</a></li>
        </body>
    </html>
    '''

# This function runs every time a request is received before routing.
# We switch off all the lights here.
@app.before_request
def allOff():
    automationhat.output.one.off()
    automationhat.output.two.off()
    automationhat.output.three.off()

# Our default home page
@app.route('/')
def index():
    return makePage('Index')

# The next three functions switch on the three lights
@app.route('/red')
def red():
    automationhat.output.one.on()
    return makePage('Red')

@app.route('/yellow')
def yellow():
    automationhat.output.two.on()
    return makePage('Yellow')

@app.route('/green')
def green():
    automationhat.output.three.on()
    return makePage('Green')

# Switch everything off
@app.route('/off')
def off():
    return makePage('Off')

# Let's announce ourselves by making the lights blink
automationhat.output.one.on()
time.sleep(0.2)
automationhat.output.one.off()
automationhat.output.two.on()
time.sleep(0.2)
automationhat.output.two.off()
automationhat.output.three.on()
time.sleep(0.2)
automationhat.output.three.off()

# Start the web server on port 5000
if __name__ == '__main__':
    app.run(host='0.0.0.0')
```

GARY PICKUP

An experienced IT Consultant who enjoys all things geek and sharing knowledge, who once appeared in a Marvel comic book.

**geektechstuff.com**
**twitter.com/geektechstuff**

# AN INTRODUCTION TO CRON

**You'll Need**

- Raspbian
- Some tasks to automate

Ever find yourself loading the same webpages around the same time daily for a catch-up on the latest Raspberry Pi news? Get your Pi to do it automatically with cron

Cron is a piece of software that is built into the Raspbian operating system and is used to schedule (automate) tasks to run at different intervals (e.g. every five minutes), times, or dates. Cron was originally designed to automate system administration within UNIX systems; however, due to its diversity, we can use it within Raspbian to automate tasks such as opening applications, loading webpages, and running Python scripts.

## >STEP-01

### Launch Terminal

To configure cron, we will be using a text editor within the Terminal – the latter can sometimes seem daunting, but it opens up lots of possibilities and access to utilities like cron. To quote Douglas Adams, 'Don't panic!' By default, the Terminal has a shortcut at the very top of the Raspbian desktop; click it to receive a prompt that says **pi@raspberrypi:** – and it's time to start.

## >STEP-02

### Launching cron

Cron is configured via files called crontabs. Each user account on the Raspberry Pi can have its own crontab; however, this tutorial will be dealing with creating a crontab for the default Pi user. At the **pi@raspberrypi:** prompt, type **`crontab -e`**. the '-e' is very important as this tells Raspbian that we want to edit our crontab. The Terminal will output '**no crontab for pi – using an empty one**'; this is because we are creating our first crontab.

## >STEP-03

### Welcome to your first crontab

The Terminal will open a crontab in the default text editor, which for Raspbian is nano. Nano uses keyboard combinations to save and exit files, unlike graphical text editors where clickable menus are used. The crontab already contains a lot of text that has been commented out using hashes (#); this text is an introduction to cron and worth reading. Once read, use the down arrow on the keyboard to get to bottom of the file, just below the **`m h dom mon dow command`** line.

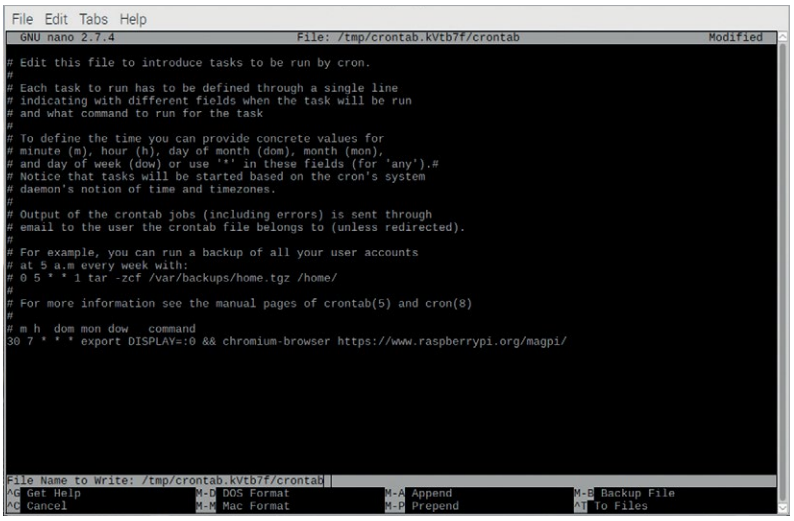
**MORE CRONTAB OPTIONS**

`crontab -e` allows for users to edit a crontab

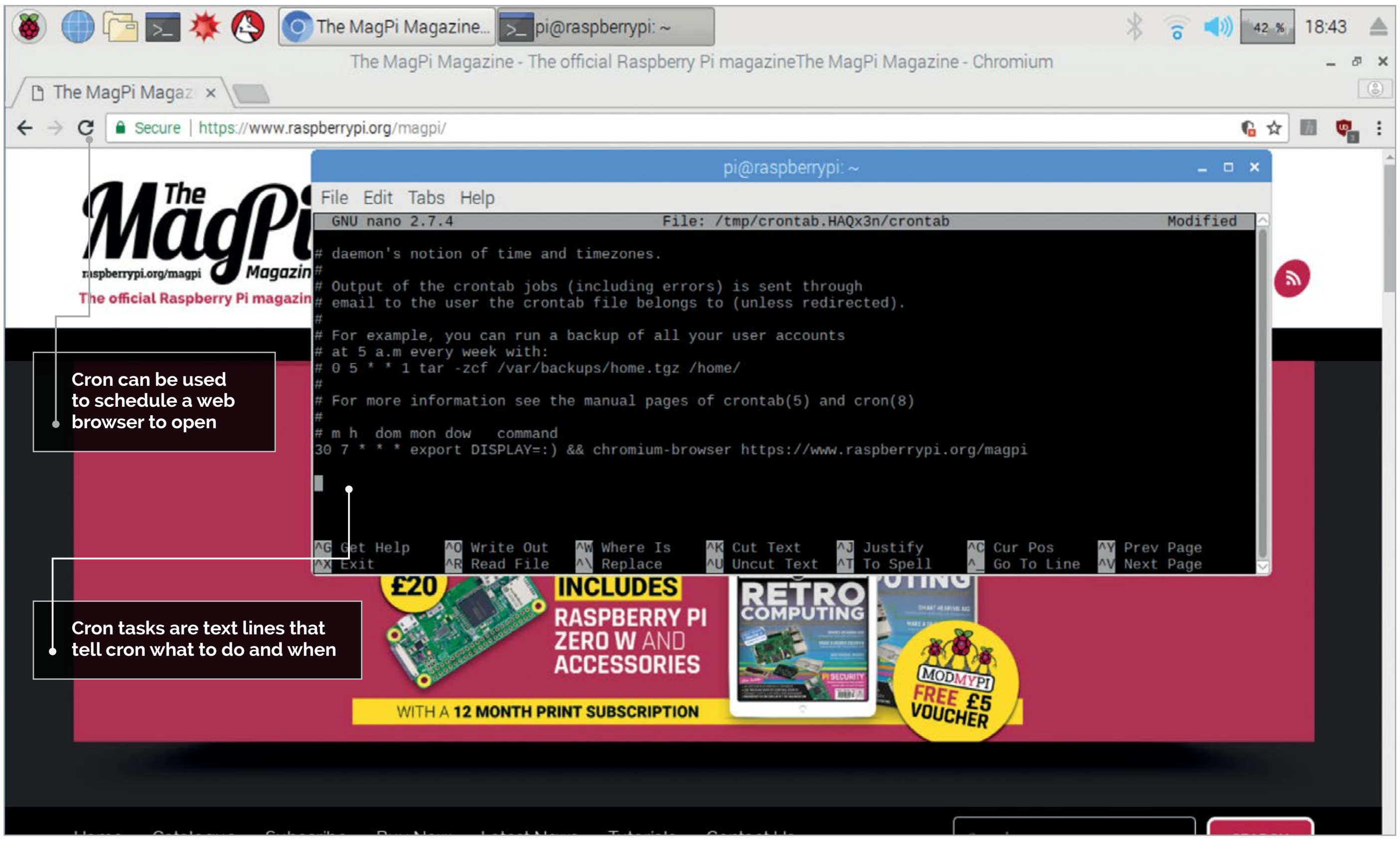
`crontab -l` lists all cron tasks set to run for a user

`crontab -r` removes a user's crontab

**Below** Using nano to edit text within the Terminal relies on using the keyboard without the mouse



```
  GNU nano 2.7.4                File: /tmp/crontab.kVtb7f/crontab                 Modified

# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
30 7 * * * export DISPLAY=:0 && chromium-browser https://www.raspberrypi.org/magpi/

File Name to Write: /tmp/crontab.kVtb7f/crontab
^G Get Help              M-D DOS Format            M-A Append                M-B Backup File
^C Cancel                M-M Mac Format            M-P Prepend               ^T To Files
```


The MagPi Magazine...
pi@raspberrypi: ~
The MagPi Magazine - The official Raspberry Pi magazineThe MagPi Magazine - Chromium
Secure | https://www.raspberrypi.org/magpi/
The MagPi Magazine
raspberrypi.org/magpi
The official Raspberry Pi magazine
£20
INCLUDES RASPBERRY PI ZERO W AND ACCESSORIES
WITH A 12 MONTH PRINT SUBSCRIPTION
RETRO COMPUTING
MODMYPI FREE £5 VOUCHER

```
pi@raspberrypi: ~
File  Edit  Tabs  Help
  GNU nano 2.7.4               File: /tmp/crontab.HAQx3n/crontab               Modified

# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
30 7 * * * export DISPLAY=:) && chromium-browser https://www.raspberrypi.org/magpi



^G Get Help   ^O Write Out   ^W Where Is   ^K Cut Text   ^J Justify   ^C Cur Pos     ^Y Prev Page
^X Exit       ^R Read File   ^\ Replace    ^U Uncut Text ^T To Spell  ^_ Go To Line  ^V Next Page
```

Cron can be used to schedule a web browser to open

Cron tasks are text lines that tell cron what to do and when

## >STEP-04

### Scheduling a task

The **m** (minute) **h** (hour) **dom** (date of month) **mon** (month) **dow** (day of week) **command** line gives a fleeting look at how cron can schedule a task. We will be using it to open our favourite website at 7.30am daily, but this could very easily be 7.30am every Thursday, or on the first day of every month. When writing a value for these options, we must provide either a number or an asterisk (*****). The ***** means any.

## >STEP-05

### Let's launch a web browser

To launch a web browser at 7.30am every day, we need to enter the following line of text:

```
  30 7 * * * export DISPLAY=:0 && chromium-
browser https://www.raspberrypi.org/magpi/
```

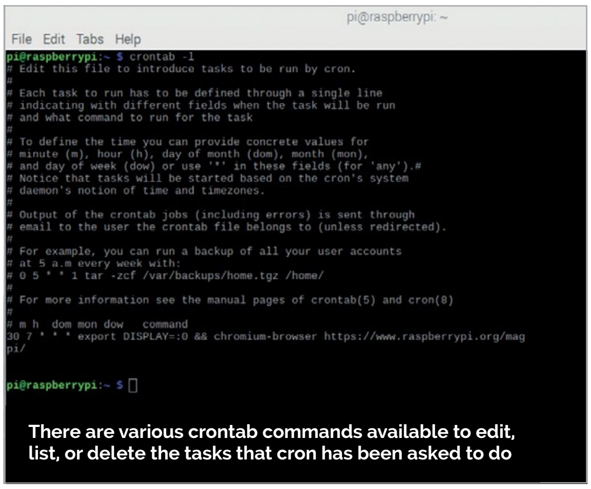
This line says at 30 minutes past the seventh hour on every day of every week in every month, run the command **export DISPLAY=0 && chromium-browser**, which opens the web browser Chromium and tells it to open the website **raspberrypi.org/magpi**. The **export DISPLAY=:0 &&** part is required to make sure cron displays the web browser on screen.

## >STEP-06

### Saving the crontab

Once the line of text has been entered, we need to save our crontab. Unlike a graphical word processor, nano requires that keyboard combinations are used to save the crontab and exit. Press **CTRL+O** to open the save file option, and then press **ENTER** to save the crontab. Don't worry about the name, as Raspbian will name the crontab appropriately. With the crontab saved, press **CTRL+X** to exit nano. You can now sit back and watch as cron runs your automated task at the scheduled time.

" Don't worry about the name, as Raspbian will name the crontab appropriately "



```
pi@raspberrypi:~ $ crontab -l
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
30 7 * * * export DISPLAY=:0 && chromium-browser https://www.raspberrypi.org/mag
pi/


pi@raspberrypi:~ $ 
```

There are various crontab commands available to edit, list, or delete the tasks that cron has been asked to do

**CRON SHORTCUTS**

Cron also supports wording rather than the `* * * * *` format of `m h dom mon dow`. Remove the asterisks and use `@reboot`, `@hourly`, `@daily`, `@weekly`, or `@yearly` for a relevant result.

**BRIAN BEUKEN**

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

# CODING GAMES ON THE RASPBERRY PI IN C/C++ PART 09

It's time to go a bit retro and make our game look better

**You'll Need**

- Code::Blocks `sudo apt-get codeblocks`
- FreeType2
- stb_image.h (see last issue)

We've done most of the 'hard' work so far: we have enemies, we have control, we can interact with our map, and we are almost ready to interact with our baddies. But as a game, it's dull: one big window with the whole game visible. That's not how we remember our old retro games.

It's time to consider the appeal of our game and make changes to the display and the map, to create satisfying gameplay.

There's a very good reason in this game why we kept our window to a size that would allow us to see the whole game: it allowed us to see the whole game! That might seem very obvious, but if we had reduced the view area to a window, it would mean we'd have to actually play the game so we could get to a point where we could see if our characters were behaving as we expected. At the moment, while they're a bit small and hard to distinguish, we can comfortably see

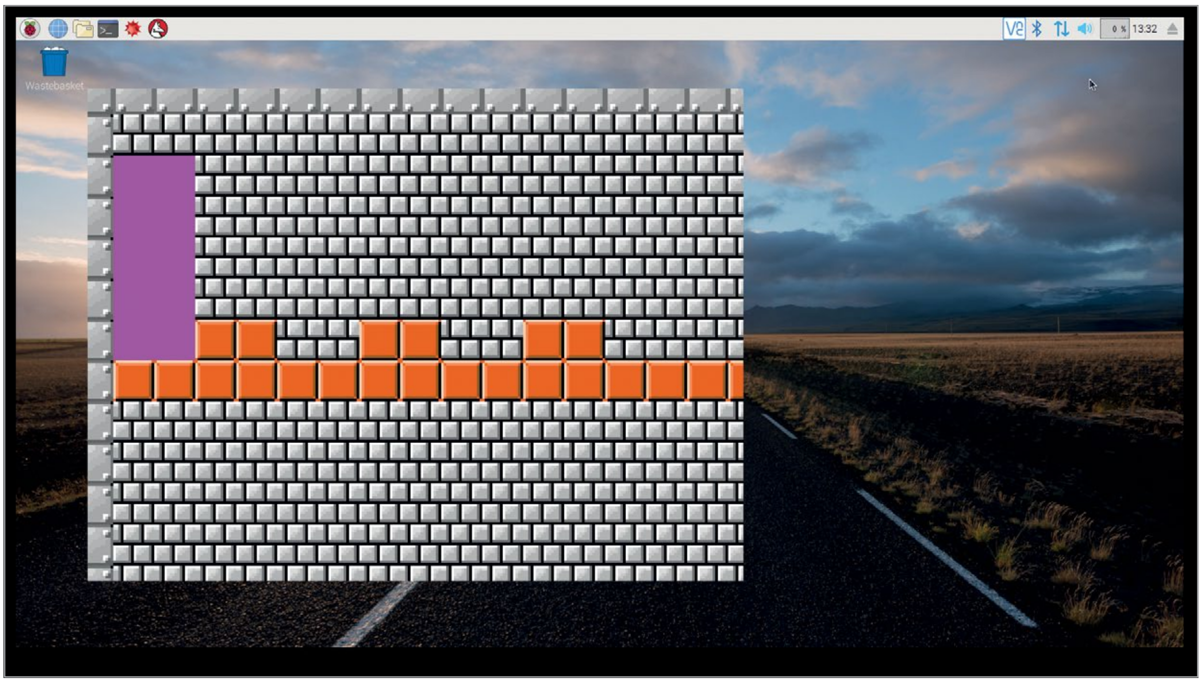


**Figure 1** Scaling up the graphics

**USE SWITCHABLE VIEWS**

Even though we now have a full-screen display, being able to switch back to a window makes placement and observation nicely easy.

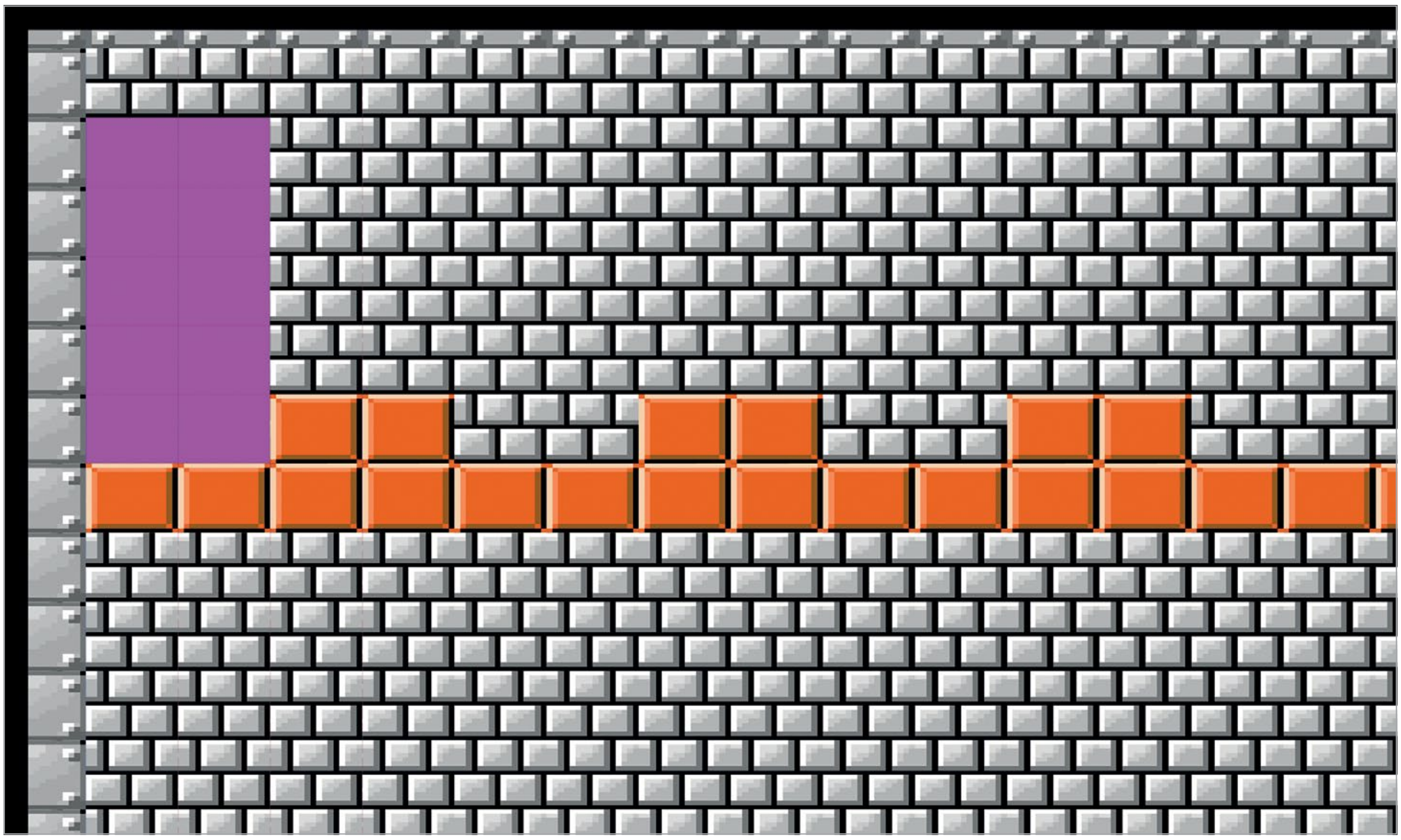


**Figure 2** Full-screen glory, but where's Bob?

that all the enemies are where we expect them, and do what we expect of them.

## Step through it

A second point is much harder to get excited about, as for the most part you've had code to download which you knew was working, so there would be little reason to actually inspect and step through your code. But it is an important thing to be able to do. This is one reason why most coders like to use two systems to work on: one to edit, compile and debug, and one to run on. This is possible using two Raspberry Pi systems or a PC and Pi, but it does need a bit of setting up – we will cover this another time.

Back to viewing the whole game. What we are in fact currently doing is visualising the whole of the play area, or map. We've allowed our map co-ordinates to directly reflect our screen co-ordinates, but we have the capacity to scale things up and view a window into our map that we can drag along with our player.

But before we can do that, we need to acknowledge that some things we did previously to make life easy for us are now going to bite us.

## Magic numbers are bad

We've mentioned before, mostly in the code, that hard-coding our tile sizes to actual numbers like 16 isn't a good idea. Yes, it's the actual number of pixels the image has, but now we have a predicament: if we're going to scale up the visuals to allow for 64×64 pixel tile size, we have to go through all our code and take note of every use of the number 16 as it relates to the size of a tile, and change it manually (**Figure 1**).

We can't just do a global find and replace, because there are some cases where the number 16 is actually fully legit. We are still interacting with our map using 16×16 tiles; it's only the display that will change. These so called 'magic', or hard, numbers are very prone to error if you miss one when trying to change, and it's rather tedious to go through an entire project checking and changing every instance, once it gets beyond a certain size.

This is where we need to use one of the standard concepts of C/C++ to define our value in a way that's easy to change.

```
#define A_CHANGEABLE_VALUE 16
```

Now in your code when you are referring to the magic value, use the label **A_CHANGEABLE_VALUE**, which gives us a nice helpful way to alter things because you only have to change the **#define** once!

The **Game.h** file has set the size of the tiles as a **#define**; since they are square, we use the same value for x and y sizes. Try altering the sizes and you will see that one change allows you to have any size of screen. Don't make it too big, though: since our original 16 size was only 16×16 pixels, you can see our scaling is stretching things quite a bit.

We can also scale up our sprites using a similar system, as a proportion of our actual tile sizes vs our image size.

**LOAD ONLY WHAT YOU NEED**

We will cover this in more detail next time, but take a small note of how much you are loading and how often. Be aware of things like this.

**AVOID MAGIC NUMBERS**

If you are likely to change a value, don't hard-code it. Magic numbers can cause a lot of bugs when you find you have to change them.

Now when we draw our tiles and sprites, we need to remember that our sizes and positions have to reflect these new scale values and they will therefore only be visible in the part of the screen we are focusing on, which sadly to begin with is going to be the top-left corner (0,0).

## OK screen, follow me

Now we can display the screen any size we want, while the map is busy doing its updates independently. We want to get the screen to focus on our area of interest, namely where our player is. He has to be the controlling system that allows the screen to scroll around and lets us see what is directly around him.

That's not so hard to do. If we keep note of where in the map our player is, we can consider that the corner of the screen that we want to see is relative to his position; that is easy to calculate.

Imagine Bob has a pole that's a fixed length, and he ties it to the screen; as he moves, he pulls the screen 'window' around with him, so the window itself has co-ordinates that start at a set distance behind Bob. We know that the corner of the screen will be a certain x distance away, and a certain y. Voilà, we have the basics of a scroll system that centres on our main character (**Figure 3**). We can make minor adjustments to deal with the windows co-ordinates are at the edges of the map, and even make small allowances to allow Bob to move freely within a centre window area without scrolling. Check out the **Bob** class to see how that is worked out and passed back to the tile render system to reposition things.

One more step to full arcade mode – and this is an easy one to do on the Raspberry Pi – is to set our screen to full. The Raspberry Pi range has a couple of nice features that let us ask the display code how big the screen is, which in turn lets us set the size. Set **#define FULLSCREEN** to true in **Game.h,** and see the result (**Figure 2**).

There has been one niggle all through our development of our game. When we press the keys in our game, it often does something to folders or open projects on our desktop; we really should stop that. Because our OS is running, it is naturally going to read keys, but we saw early on that using the OS to read keys in our game wasn't ideal. But having both our own key reader and the OS reader causes niggles.

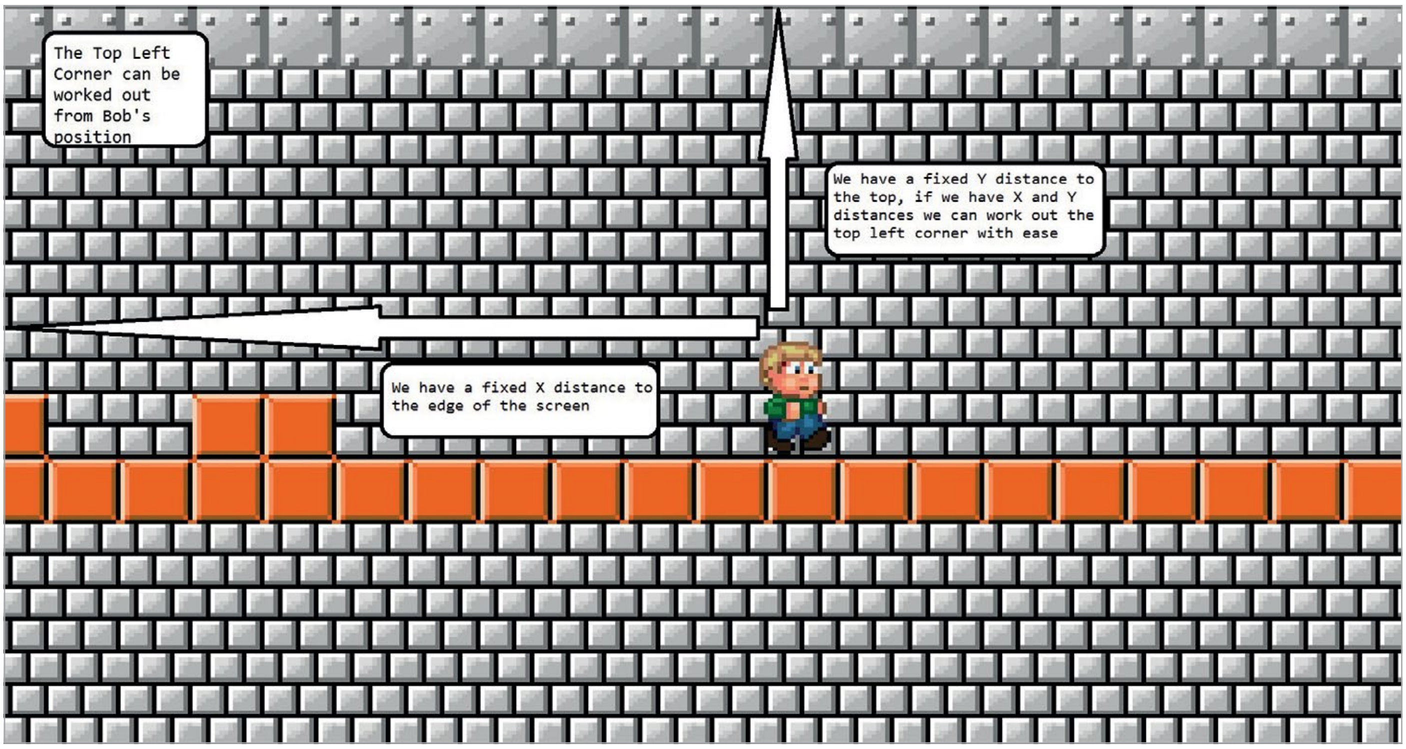
The redirection code can be found in the game startup code, and importantly the restore code as the game exits. We don't really have to do that, but it's good practice to leave our system as close to the way we entered it as we can.

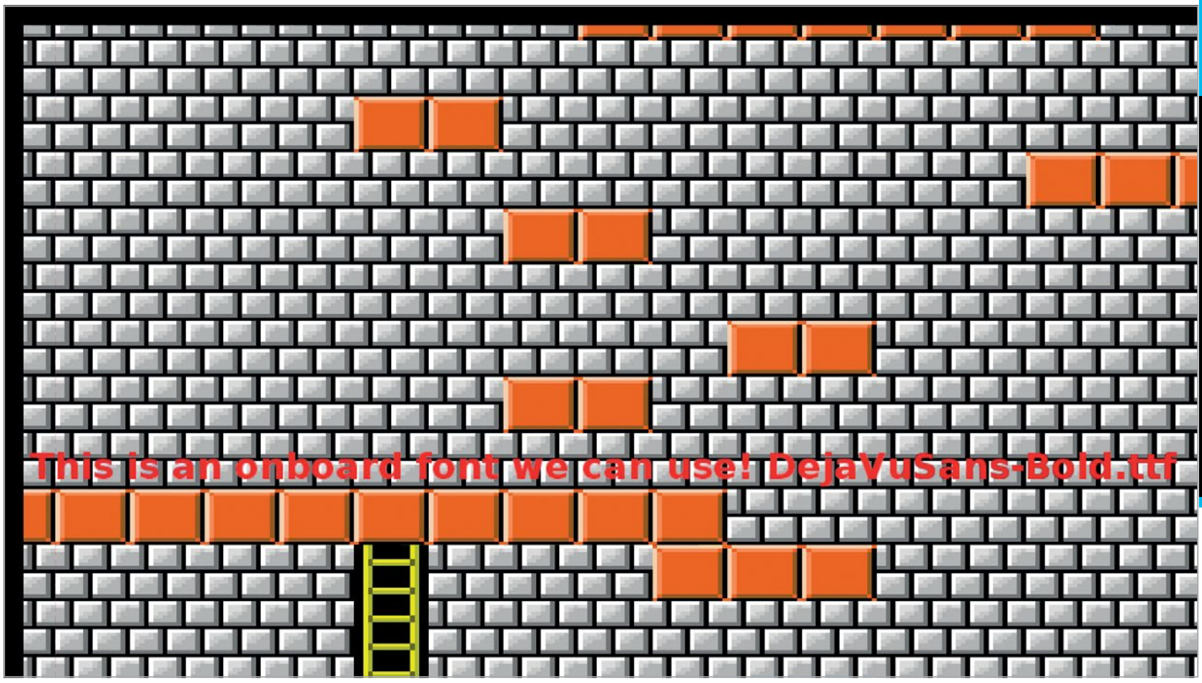
## Finally some text

Printing text is easy, isn't it? We just use a **printf** function or a **cout** or some other variation. Hmm, well no, sending text to buffers and console display is easy, but actually putting the letters A–Z and numbers 0–9 on screen in our game when we want it, at the right point, in the right colour, in the right size... no, that's not actually easy at all.

Text is funny stuff when displayed on screen. It comes in all kinds of shapes and styles, called fonts, all kinds of sizes, all kinds of emphasis. Even within a font there are all kinds of issues with the width of different letters. Consider W and I; W is one of the

**Figure 3** Drag the screen along using fixed distances


The Top Left Corner can be worked out from Bob's position
We have a fixed Y distance to the top, if we have X and Y distances we can work out the top left corner with ease
We have a fixed X distance to the edge of the screen



**Language**

>C++

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

**DOWNLOAD:**
magpi.cc/MzGDri

**Figure 4** Here's a very simple font display using an on-board font

widest letters in the Roman alphabet, I is one of the thinnest, but we want a word like WIDTH to keep the letters close together. We call that proportional spacing, where we keep the gap between letters to a comfortable minimum. W I DTH just looks wrong.

We could just treat text like a sprite, keeping a suitable drawn tile for each letter and number in a fixed order which matches the standard ASCII character set system (A = 65, B = 66, etc.). But though simple, it's rather an inflexible system and the font is also going to be a fixed size, so we have to make adjustments when drawing to keep the proportional spacing we desire to look good and if we want to use a different font, we may have to store new info on spacing.

Oh, and we mentioned the Roman (also called Latin) alphabet – that's the standard English alphabet – but there are several others: Cyrillic, Kanji, Arabic, etc. There has to be some kind of standard, though, which computers can use to display the same text in any chosen font or emphasis! The most common is the TrueType standard, which keeps all the data needed to proportionally space our letters, and info on any emphasis systems we want to use.

There are many sites where you can download free to use TrueType fonts – a quick web search for free TrueType fonts will find dozens of sites you can browse for just the right font you want, but make sure it's free to use. Once you download it, treat it like any other asset and load it as you need it.

So, to repeat, printing text is *not* easy, not easy at all… it's such a lot of work in fact, it's in our interests to make use of code that's been written by far more experienced people who actually enjoy working with such demanding tasks.

There are many font-handling libraries around, some you have to pay for, but most are free. We're going to use a common free one that's already on our Raspberry Pi, called FreeType. It's fairly simple to use and once we have included the relevant files, we can let that library take the strain.

Review the new `Text` class, which lets us use the FreeType lib in easy-to-understand ways. We also have a fair number of pre-installed fonts we can use; our demo will use what's available on your Raspberry Pi. Just add **/usr/include/freetype2** to your `#include` file list, and the library itself is called **freetype**.

Now that we can print text, we can also use it to report on values in the game. Since our debugger is not so easy to use now, we are full-screen – but we'll come to that soon, as we're going to start holding back a little on providing fully working code so you can write your own.

There are some interesting new concepts in our text shader and setups, not least of which is their use of a mathematical concept called a matrix. Now, these are truly wonderful things which have a tendency to scare beginners, but really all a matrix does is allow us to do all the maths we do in our current hacky shaders much faster. We're going to discuss this properly in the next lesson.

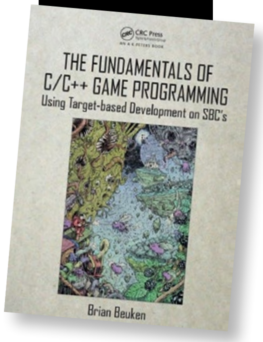
## Next time

Our main core of the game is done, but we don't have a very organised game: there's no intro; no menu; only one, rather bad level; our jumps and timing need tweaking; and we don't as yet interact with other objects. Next time, we'll correct these and add pick-ups and a score system, as well as introducing some concept of losing lives and game over.

We'll also take our first important steps in using matrices, and take a bit more care of our asset management.

## LEARN MORE ABOUT C

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

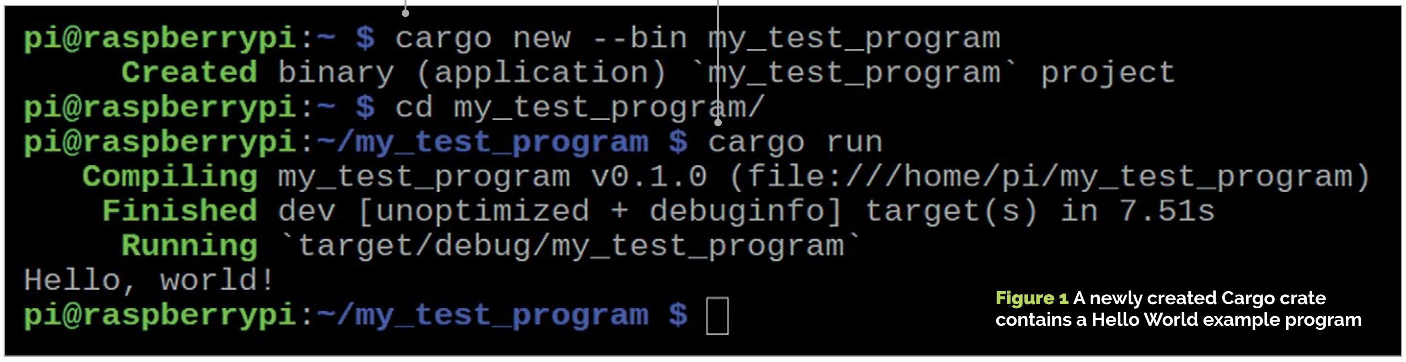

CRC Press
THE FUNDAMENTALS OF C/C++ GAME PROGRAMMING
Using Target-based Development on SBC's
Brian Beuken

**JONATHAN PALLANT**

A professional embedded systems programmer based in Cambridge, Pi Wars veteran, and member of the Rust Embedded Working Group.
**keybase.io/thejpster**

The `cargo new` command creates a new Rust crate. The `--bin` argument makes it a program, not a library

Cargo automatically re-compiles your code when you try to run it



```
pi@raspberrypi:~ $ cargo new --bin my_test_program
     Created binary (application) `my_test_program` project
pi@raspberrypi:~ $ cd my_test_program/
pi@raspberrypi:~/my_test_program $ cargo run
   Compiling my_test_program v0.1.0 (file:///home/pi/my_test_program)
    Finished dev [unoptimized + debuginfo] target(s) in 7.51s
     Running `target/debug/my_test_program`
Hello, world!
pi@raspberrypi:~/my_test_program $ 
```

**Figure 1** A newly created Cargo crate contains a Hello World example program

# READING SENSE HAT DATA WITH RUST

**You'll Need**

- Rust **rust-lang.org**
- Sense HAT **magpi.cc/sense-hat**

Want to learn a new programming language that's ergonomic and safe, yet runs really fast? Dive in to Rust

We've all heard about security bugs in programs written in C, a language that dates back nearly 50 years. Yet it remains the default choice for systems programming. In Rust, a new programming language from Mozilla, it's almost impossible to introduce security flaws, yet the compiler produces fast machine code. To introduce you to Rust, we'll walk through the installation of the compiler and a simple example program for the Sense HAT.

## >STEP-01

### Installing Rust

Rust is fairly straightforward to install and has native support for the ARM processor in the Raspberry Pi. If you visit **rust-lang.org** and select Install Rust, it will prompt you to run the following command: **`curl https://sh.rustup.rs -sSf | sh`**. This will download and run the installation script, detect your Pi, and fetch the correct version of the compiler automatically. Just follow the on-screen instructions.

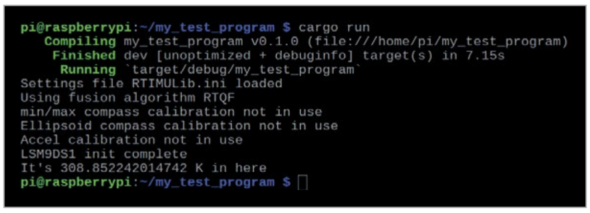
## >STEP-02

### Create and run your first program

Rust programs and libraries are called 'crates' and are built using a tool called 'Cargo'. Let's ask Cargo to create a new 'binary' crate with a bare-bones example program inside, using the **`cargo new`** command (see **Figure 1**). You can give your new program any name you like, but make sure you use underscores instead of spaces. We can then enter the new folder that has been created and ask Cargo to run our program. Cargo correctly works out that our program must be re-compiled before it can be

## listing1.rs

```rust
// Here we access the crate we specified in Cargo.toml.
extern crate sensehat;

// The use statements pull some types into scope, to
// save us some typing.
use sensehat::SenseHat;

// main() gets called when we run our program
fn main() {
    // The variable 'hat' represents our SenseHat. We mark
    // it mutable (otherwise it would be const, or read-only).
    let mut hat = SenseHat::new().unwrap();
    // Get the temperature and unwrap the result.
    let temp = hat.get_temperature_from_humidity().unwrap();
    // Tell everyone how hot it is using the println! macro.
    println!("It's {} in here", temp);
}
```



```
pi@raspberrypi:~/my_test_program $ cargo run
   Compiling my_test_program v0.1.0 (file:///home/pi/my_test_program)
    Finished dev [unoptimized + debuginfo] target(s) in 7.15s
     Running `target/debug/my_test_program`
Settings file RTIMULib.ini loaded
Using fusion algorithm RTQF
min/max compass calibration not in use
Ellipsoid compass calibration not in use
Accel calibration not in use
LSM9DS1 init complete
It's 308.852242014742 K in here
pi@raspberrypi:~/my_test_program $ 
```

**Figure 2** This is what **listing1.rs** should look like when you run it. This library defaults to using kelvin for temperature

run and takes care of that for us. Compiling our code checks it for correctness and then converts it from the source code into a machine code file the computer can run, so it's harder for bugs to hide in Rust, unlike Python where each line isn't checked until it's executed.

## >STEP-03

### Fetch some crates

Rust gives you access to a range of software crates written by other Rust programmers, at **crates.io**. By making changes to our crate's **Cargo.toml** file, we can tell Cargo we'd like to make use of one of these crates in our program. This saves us a bunch of typing, but it's often better to build on the work of others. Edit the **Cargo.toml** file in your project's folder in your favourite text editor (such as Geany) to add this line, under **`[dependencies]`**:

```
sensehat = "1.0"
```

Next time you build your code, this will pull in version 1.x (anything compatible with 1.0) of your author's Sense HAT driver from **magpi.cc/issfxz**. Because we used some other crates when writing this driver, when you next build, Cargo will automatically pull all of those in as well, and all the crates they need, and so on.

## >STEP-04

### Reading from the Sense HAT

Our program lives in a file called **src/main.rs**. Let's load that up, delete the 'hello world' example placed in there by default, and do something with our Sense HAT by copying the code in **listing1.rs**. We can run this again with **`cargo run`** and it should look like **Figure 2**. Don't worry, it will only take a while to compile the first time around.

## >STEP-05

### Let's loop

Rust has all the usual programming language features like **`for`** loops and **`if`** statements. Let's expand our program to wait until someone shakes our Pi, using the accelerometer function, and then print them a message on the LED screen – see **listing2.rs** and **Figure 3**.

## >STEP-06

### Read the docs!

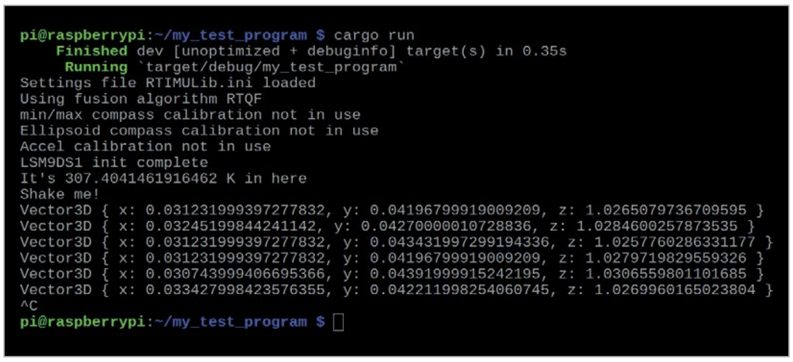
Hopefully we've given you a little taste for the Rust programming language and shown that it's not much more complicated than Python or JavaScript. For further reading you can check out the Rust Book, which is available for free online at **magpi.cc/HAoyYu**, or as a printed copy from your favourite book vendor. If would like to do some experiments with LEDs and buttons, check out Rahul's excellent port of GPIO Zero to Rust at **magpi.cc/eYpDfk**. You can also read the documentation for the **sensehat** crate, to find what other features it has to offer, at **docs.rs/sensehat**.

## listing2.rs

Language
>RUST

DOWNLOAD:
magpi.cc/apkZUw

```
// Here we access the crate we specified
in Cargo.toml.
extern crate sensehat;

// The use statements pull some types into scope, to
// save us some typing.
use sensehat::{Colour, SenseHat, SenseHatError};
use std::{thread::sleep, time::Duration};

// We've declared main as returning a Result, which
// means we can use the ? operator to bail out early
// if our code finds any errors at run-time.
fn main() -> Result<(), SenseHatError> {
    let mut hat = SenseHat::new()?;
    let temp = hat.get_temperature_from_humidity()?;
    println!("It's {} in here", temp);
    // Clear the screen
    hat.clear()?;
    // Instructions for the user
    println!("Shake me!");
    loop {
        // Get the G readings from the sensehat
        let accel = hat.get_accel_raw()?;
        // Show them on screen in debug format.
        println!("{:?}", accel);
        // Check if any acceleration value is too high
        if [accel.x, accel.y, accel.z].iter().any(|g| *g > 1.5) {
            // If so, print a warning message!
            hat.text("Earthquake!", Colour::WHITE, Colour::RED)?;
            // Pause on the final character
            sleep(Duration::from_millis(1000));
            // Turn the screen off
            hat.clear()?;
        }
        // Let our CPU rest for a short while to slow down
        // the output.
        sleep(Duration::from_millis(250));
        // Press Control + C to quit this loop!
    }
}
```



```
pi@raspberrypi:~/my_test_program $ cargo run
    Finished dev [unoptimized + debuginfo] target(s) in 0.35s
     Running `target/debug/my_test_program`
Settings file RTIMULib.ini loaded
Using fusion algorithm RTQF
min/max compass calibration not in use
Ellipsoid compass calibration not in use
Accel calibration not in use
LSM9DS1 init complete
It's 307.4041461916462 K in here
Shake me!
Vector3D { x: 0.031231999397277832, y: 0.04196799919009209, z: 1.026507973670959 5 }
Vector3D { x: 0.03245199844241142, y: 0.04270000010728836, z: 1.02846002578735 35 }
Vector3D { x: 0.031231999397277832, y: 0.043431997299194336, z: 1.0257760286331177 }
Vector3D { x: 0.031231999397277832, y: 0.04196799919009209, z: 1.0279719829559326 }
Vector3D { x: 0.030743999406695366, y: 0.043919999152421 95, z: 1.0306559801101685 }
Vector3D { x: 0.033427998423576355, y: 0.042211998254060745, z: 1.0269960165023804 }
^C
pi@raspberrypi:~/my_test_program $ 
```

Figure 3 This Rust Sense HAT driver uses the same RTIMUlib C++ library as the Python Sense HAT driver

# FREQUENTLY ASKED QUESTIONS

Your technical hardware and software problems solved...

**NEED A PROBLEM SOLVED?**

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

## PROGRAMMING ON PI

### HOW CAN I PROGRAM ON THE RASPBERRY PI?

**Basics with Scratch**
Depending on your skill level, the easiest entry point is Scratch. It uses programming blocks that you drag and drop with your mouse to create code you can see – this teaches you the basic structure and logic of coding, while also letting you see instant results for what you've done.

**Learning code**
The step above Scratch is actually typing out code in something like Python. It's quite easy to read and understand and, like Scratch, lets you see the results of your work pretty quickly. You can find lots of Python tutorials in *The MagPi*'s archives and on the Raspberry Pi projects site: **rpf.io/projects**.

**Programming objects**
Once you're confident enough with making programs in Python, you can then graduate to controlling different components and reading from sensors. This can be as simple as turning on an LED hooked up to a Pi, or more complex, like recording environmental data to control a robot.

### WHAT'S IS THE DIFFERENCE BETWEEN THE PYTHON IDES?

**IDLE**
IDLE is the default IDE (integrated development environment) that comes with Python. It lets you write your code and test it quickly without having to compile it into software. It offers basic debugging of your code, and it's a lot better than just typing something into a text document.

**Thonny**
Thonny comes with Python 3.6, and has been developed to be a lot easier to use for beginners compared to IDLE. It includes the shell and code in one window, and gives you info on variables in extra columns around the interface – almost everything you'll need to know at a glance.

**Mu**
Designed for true beginners in mind, this new option is meant to be the first Python IDE to use. It offers specific modes to make coding your project easier – you can select standard Python, or even a Pygame Zero mode to make some games with.

### WHAT OTHER LANGUAGES CAN I USE?

**Anything you desire!**
All modern programming languages are available to use on the Raspberry Pi, such as C, C++, Ruby, and JavaScript. You may need to install extra software to truly get the most out of them, but a quick Google search will help you there.

**Node-RED**
This programming language uses Node blocks to create code, somewhat similarly to Scratch, albeit designed with creating the code as a workflow. It's very powerful and has been used to animate dinosaurs, hack Minecraft, and much more.

**Wolfram Mathematica**
Wolfram and Mathematica are supplied for free in Raspbian and allow you to do some serious coding with access to a lot of data. We've seen them used for wonderful graph plots that look like pieces of art, as well as making it simple to mod Minecraft Pi and post to Twitter.

# FROM THE RASPBERRY PI FAQ RASPBERRYPI.ORG/HELP

**WHAT IS THE USER NAME AND PASSWORD FOR THE RASPBERRY PI?**
The default user name for Raspbian is **pi** and the default password is **raspberry**. If this does not work, check the information about your specific operating system on the downloads page (**magpi.cc/1MYYTMo**).

**WHY DOES NOTHING HAPPEN WHEN I TYPE IN MY PASSWORD? DID MY RASPBERRY PI FREEZE?**
To protect your information, Linux does not display anything when you are entering passwords in the Bash prompt or the Terminal. As long as you were able to see the user name being typed in, your keyboard should be working correctly.

**HOW DO I CONNECT A MOUSE AND KEYBOARD?**
The Model A/A+ has one USB port, the Model B has two ports, and the Model B+, Pi 2 Model B, and Pi 3 Model B/B+ have four ports. These can be used to connect most USB 2.0 devices. Additional USB devices such as mice, keyboards, network adapters, and external storage can be connected via a USB hub. The Pi Zero and Pi Zero W have a single micro USB port; this requires a USB OTG cable to connect devices such as keyboards or hubs.

**WHERE IS THE ON/OFF SWITCH?**
There is no on/off switch! To switch on, just plug it in. To switch off, if you are in the graphical environment, you can either log out from the main menu, exit to the Bash prompt, or open the Terminal. From the Bash prompt or Terminal, you can shut down the Raspberry Pi by entering `sudo halt -h`. Wait until all the LEDs except the power LED are off, then wait an additional second to make sure the SD card can finish its wear-levelling tasks and write actions. You can now safely unplug the Raspberry Pi. Failure to shut the Raspberry Pi down properly may corrupt your SD card, which would mean you'd have to re-image it.

## THE MAGPI APP

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

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

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

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

GET IT ON Google Play

Available on the App Store

# SECURE YOUR HOME WITH RASPBERRY PI

MAKE YOUR HOUSE SAFER WITH THESE SIMPLE PI PROJECTS

Home security is important. Whether you live in a high-crime area or are currently at a bad time in your life, feeling safe in your own home is essential. The Raspberry Pi can help you here.

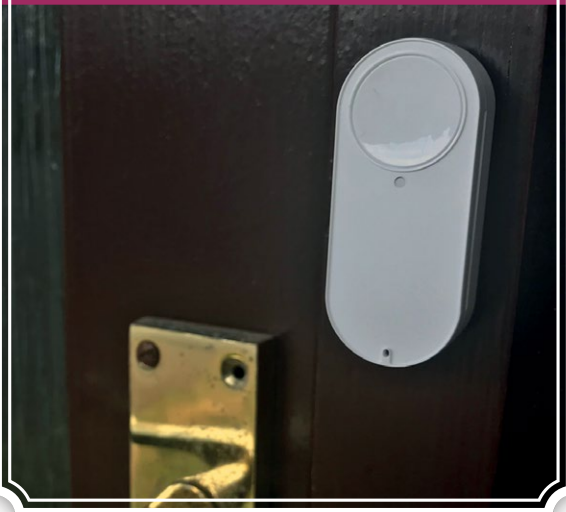
With the rightt tools you can set up a CCTV network. Or have a special doorbell that lets you know whenever someone is at your door, whether you're at home or not. Over the next few pages we'll delve into these kind of projects that you can make much more cheaply than a home security system.

Of course, you could also use all of the above to keep an eye on your cat during the day, or make sure you hear the postman. However you want to use it, make sure to have a spare Raspberry Pi handy, and let's get started.

66 | RASPBERRY PI HOME CCTV NETWORK

68 | DOOR SECURITY MONITOR

70 | AMAZON DASH BUTTON DOORBELL



EE Unlocked 62%

14:59

Wednesday 18 July

PUSHOVER now

**Door opened!**
The door is open!

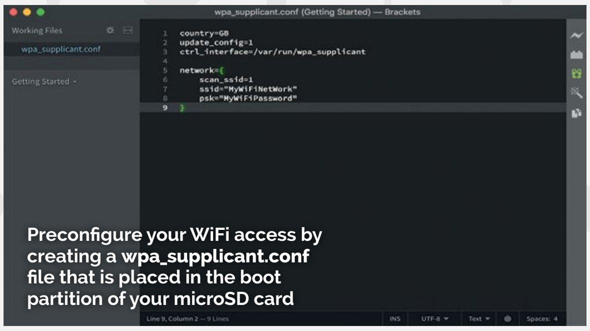
# RASPBERRY PI HOME CCTV NETWORK

Combine multiple Raspberry Pi boards and their Camera Modules to create a home CCTV network using motionEyeOS

**YOU'LL NEED**

- Two or more Pi Camera Modules **magpi.cc/camera**
- motionEyeOS **magpi.cc/2iiCyMs**
- Wide-angle lens **magpi.cc/PyxWgR**

Home CCTV systems and IP cameras are becoming increasingly popular nowadays, but can be expensive to buy and install, especially for multiple cameras. In this guide, we'll show you how to use multiple Raspberry Pi boards and their Camera Modules to create a home CCTV network using motionEyeOS, so that you can keep an eye on your home at all times. You can even be alerted whenever any of your cameras detects movement, so that you never miss anything!

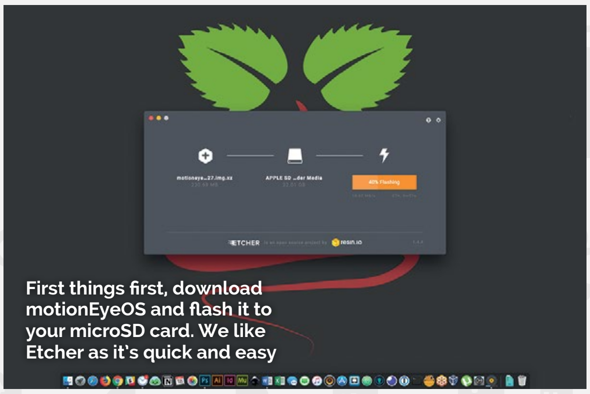


```
country=GB
update_config=1
ctrl_interface=/var/run/wpa_supplicant

network={
    scan_ssid=1
    ssid="MyWiFiNetWork"
    psk="MyWiFiPassword"
}
```

Preconfigure your WiFi access by creating a wpa_supplicant.conf file that is placed in the boot partition of your microSD card

## STEP 01 DOWNLOAD AND FLASH MOTIONEYEOS

Every camera you want to include in your Raspberry Pi home CCTV network must be running motionEyeOS. Download the image based on the Raspberry Pi model you are using and then flash it to the microSD card. We love using Etcher as it is simple to use and works with compressed image files, but you can do this any way you like. Don't boot your Pi just yet, though, as we need to preconfigure the WiFi connection.


First things first, download motionEyeOS and flash it to your microSD card. We like Etcher as it's quick and easy

## STEP 02 PRECONFIGURE YOUR WIFI CONNECTION

If you are running motionEyeOS on a Pi Zero W, you will need to preconfigure the WiFi connection before booting for the first time. Using a text editor, create a file called **wpa_supplicant.conf**, then add your network credentials (more information can be found here: **magpi.cc/BrUXgR**). Save the file and then copy it to the boot partition on your Pi's microSD card – we will reuse this file for the other Pi boards as we add them, to save repeating this every time.

## STEP 03 BOOT YOUR MASTER CCTV PI

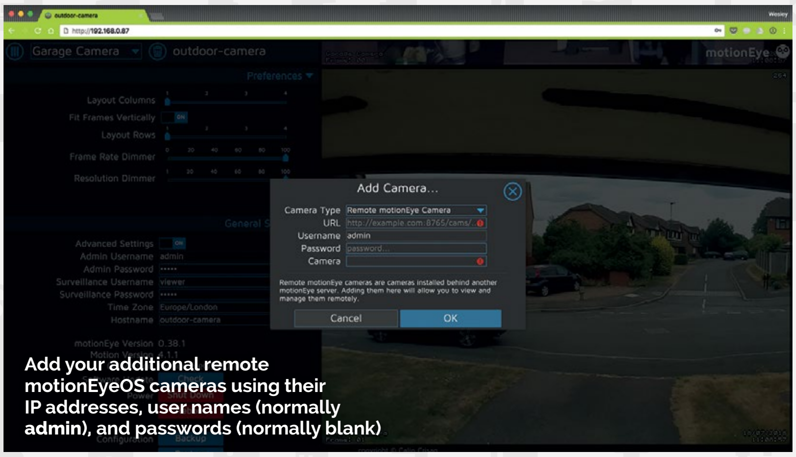
We'll use the first Pi as our 'master'. The first boot can take longer as motionEyeOS will optimise the microSD card. Access the motionEyeOS interface by going to the Pi's IP address, where you should be greeted with a login prompt. Use **admin** as the user name; the default password is blank.

## STEP 04 SET UP ADDITIONAL PI BOARDS

Repeat Steps 1 and 2 on each additional Pi you want to add. They should all be running motionEyeOS, be connected to WiFi, and you should be able to access motionEyeOS on each using its IP address – if not, double-check that you added the **wpa_supplicant.conf** file to the boot partition and try again. Now log back in to the 'master' CCTV Pi and click on the three horizontal blue lines in the top-left corner. Next to that is a drop-down menu – select 'add camera' and in the Camera Type, select 'Remote motionEye Camera'. Enter the remote camera's IP address, user name (**admin**), and password (blank), and you should be able to select the camera. Do this for each additional camera you have set up earlier.

## STEP 05 CONFIGURE CAMERAS IN ONE PLACE

Once you have added your cameras to the 'master' CCTV, you should see them in the live view on motionEyeOS accordingly. Congratulations! Now you can customise and configure each camera from the 'master'. You may want to name each camera, as well as tweak the resolution, or even add email alert notifications when motion has been detected. Your home CCTV system is now complete!



Add Camera...

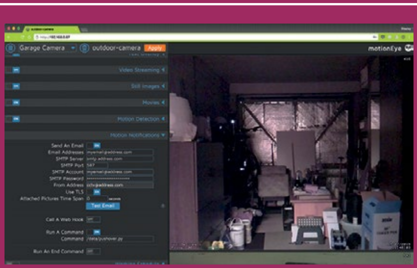
Camera Type: Remote motionEye Camera
URL: http://example.com:8765/cams/...
Username: admin
Password: password...
Camera:

Remote motionEye cameras are cameras installed behind another motionEye server. Adding them here will allow you to view and manage them remotely.

Cancel OK

Add your additional remote motionEyeOS cameras using their IP addresses, user names (normally admin), and passwords (normally blank)

## CONFIGURE MOTION ALERTS WHEN DETECTED

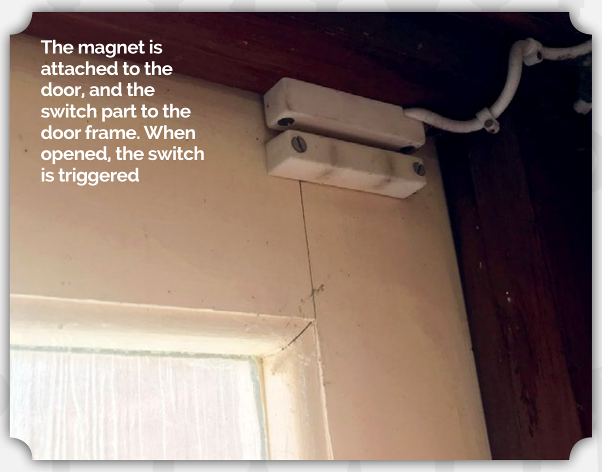


You can configure motionEyeOS to alert you when it detects motion. This can be via email, using push notifications (see our guide on Pushover notifications from issue 43: **magpi.cc/43**) or even via a custom command triggered by a webhook, and is a great way of keeping informed when you are away from home. You may want to tweak the motion detection sensitivity before you do this, especially if the camera is likely to be triggered by trees blowing in the wind, otherwise you can end up with a lot of emails or push notifications!

The official Pi Zero Case is perfectly suited to the Camera Module
A wide-angle lens is inexpensive and enhances the field of view on the camera
The Pi Zero W is a great choice due to its size and wireless capabilities

# DOOR SECURITY MONITOR

Connect magnetic door security switches to your Raspberry Pi and receive notifications when they are opened, via motionEyeOS


The magnet is attached to the door, and the switch part to the door frame. When opened, the switch is triggered

## YOU'LL NEED

Magnetic reed switch

Pushover account **pushover.net**

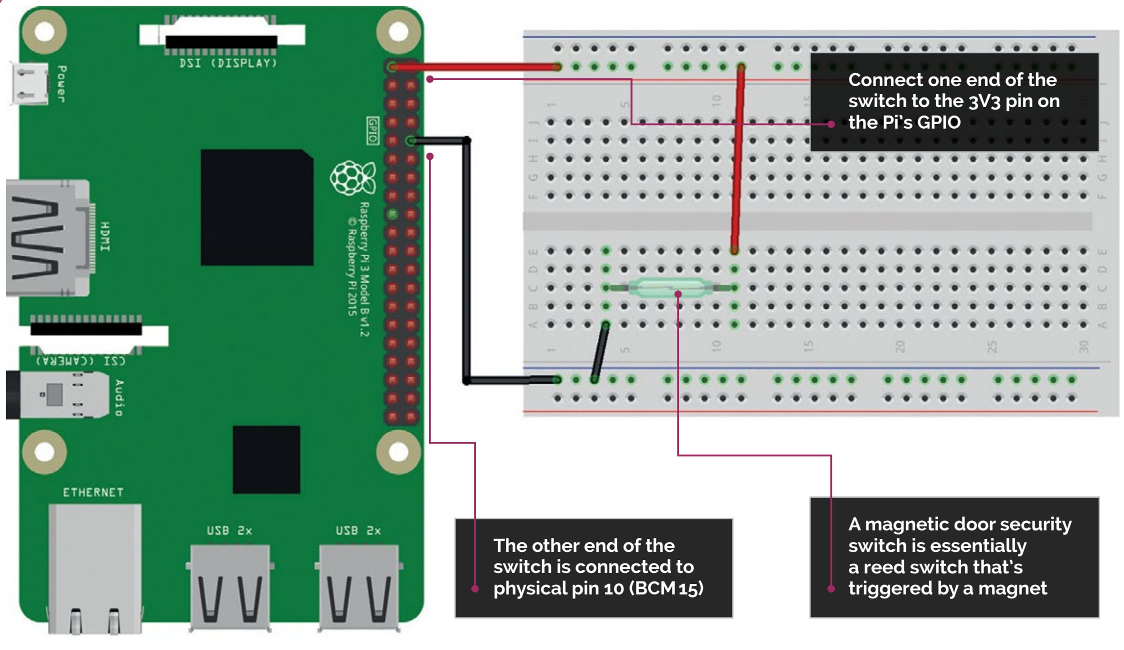
Prototyping wire

Mini breadboard

Hopefully, you saw how simple it is to use multiple Raspberry Pi cameras with motionEyeOS to create a home CCTV system in the last guide. Why not go one step further by using the Raspberry Pi's GPIO to monitor simple magnetic reed switches to see whether a door or window is open or closed and receive notifications via motionEyeOS and Pushover too? In this guide, we will show you how to do just that!

## STEP 01 BUILD THE PHYSICAL CIRCUIT

Before you connect everything to the door or window you want to monitor, it is best to test the circuit first. Use a breadboard and connect a jumper jerky cable to the 3V3 pin on the Pi and the positive rail on the breadboard



Connect one end of the switch to the 3V3 pin on the Pi's GPIO

The other end of the switch is connected to physical pin 10 (BCM 15)

A magnetic door security switch is essentially a reed switch that's triggered by a magnet

– like our diagram. Connect another to GPIO physical pin 10 (BCM 15) and the negative rail. You can then connect the magnetic switch to the positive and negative rails – it doesn't matter which way here.

## STEP 02 CREATE A PUSHOVER APP

You'll need to create a Pushover app for this to work. There is a seven-day trial, and it's $4.99 / £3.83 for a licence. Head over to **pushover.net**, log in, and then create an application. Give it a name (Door Monitor) and upload an icon – there's one in our source code if you want to use that! You will then get an API Token / Key, and you can find your user token on the main screen. Keep these safe!

## STEP 03 CREATE THE CODE ON MOTIONEYEOS

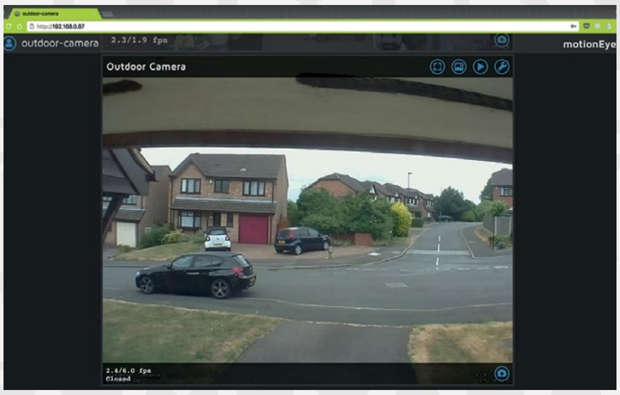
Unlike Raspbian, motionEyeOS does not allow you to clone GitHub repositories easily. The quickest way is to SSH into the Pi (usually as root) and then navigate to the **/data/etc** directory and type `nano monitor_1`. The file does not need an extension and the number has to match the camera ID it will work with. Now add our code, replace the app and user tokens for Pushover, then save and exit. Lastly, run `chmod +x monitor_1` to make the script executable.

## STEP 04 CHECK IF IT WORKS

By now, you should have prototype circuit, a Pushover app, and have added our code to your Pi running motionEyeOS. If you have motionEyeOS open in your browser, reload the page first as this is required. Next, click on the image of the camera feed and you should see the word 'Closed' in the bottom-left corner of the screen. If you break the circuit – i.e. pull the magnet away from the switch – this should change to 'Open' and you should receive a Pushover notification.

## STEP 05 INSTALL THE SWITCH PROPERLY

Now that we know the switch and code work, you can disassemble your prototype circuit and install it all properly. It's worth powering off your Pi first, though. The magnet part of the switch should go on the door/window itself, and the switch part should be on the frame – there are normally installation instructions supplied with the switch as well as screws etc., for mounting correctly. Once complete, boot up the Pi and everything should be working properly.

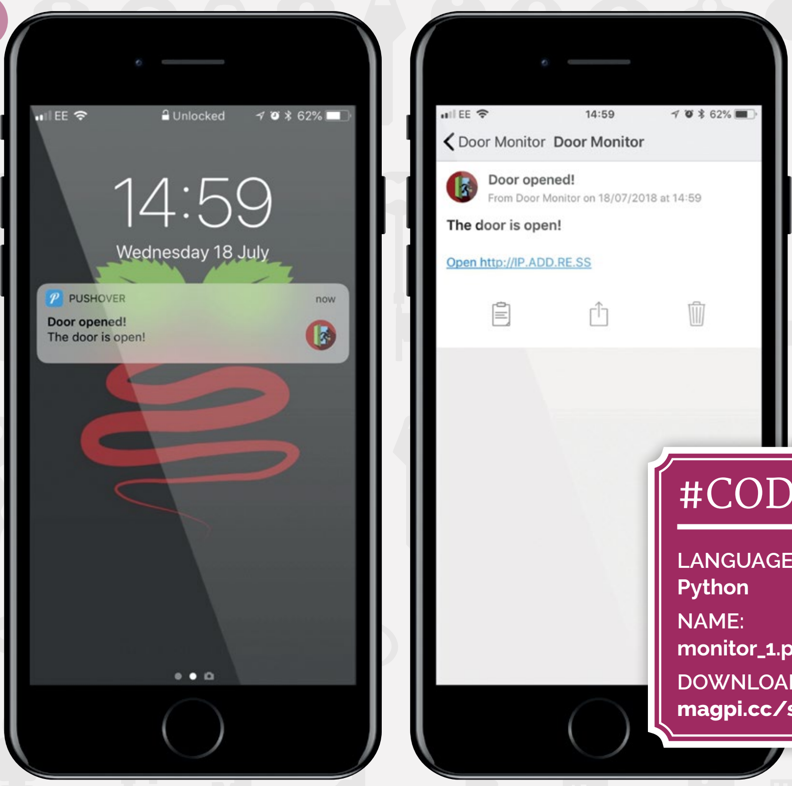


**Left** In motionEyeOS, you can see the status of the door in the bottom-left corner when you click on the image – it is closed here

## STEP 06 CUSTOMISE THE CODE TO SUIT

Our code displays a simple 'Open' or 'Closed' message on the camera feed in motionEyeOS, as well as in the Pushover notification. You can easily change this by modifying the relevant parts in our code. By default, and to prevent you from being spammed with repeated Pushover alerts, we've set our code to run every 300 seconds (5 minutes). You can also tweak this, but we found it to be a good time period.

**Below** You can also receive Pushover alerts when a door has been opened – handy for when you're not home!



#CODE

LANGUAGE: **Python**
NAME: **monitor_1.py**
DOWNLOAD: **magpi.cc/skHZrp**

# AMAZON DASH BUTTON DOORBELL

Want to do more with the Amazon Dash Button apart from ordering shopping? Put it to good use and make your own IoT doorbell

## YOU'LL NEED

Amazon Dash Button **magpi.cc/mHeanP**

Pushover account and app **pushover.net**

The Amazon Dash Button is a cheap, hackable internet-enabled device that was designed to order shopping online with a push. Whilst this is a great idea, why not make the Dash Button do something more constructive by creating your very own IoT doorbell using Pushover notifications and a Raspberry Pi? In this guide, we'll show you how to do exactly that so you can be alerted to visitors even when you are away from home.

Once pushed, the Dash Button triggers a Pushover notification on your smartphone or tablet

The LED status light will pulse white when pushed – the red flash can safely be ignored

The button can be simply stuck to the frame with the supplied adhesive

## STEP 01 FIND YOUR DASH'S MAC ADDRESS

Before you can use the Dash Button, you will need to start the process of setting up the button on Amazon. Follow the steps outlined here (**amzn.to/2mNhAqt**), but stop and exit the app before you choose the product. It's important to do this otherwise you could order products! Next, you'll need to find the MAC address of your Dash Button – this is unique to your button and identifies it online. Read our previous tutorial on how to do this here: **magpi.cc/2rwtTxP**.

## STEP 02 CREATE A PUSHOVER APP

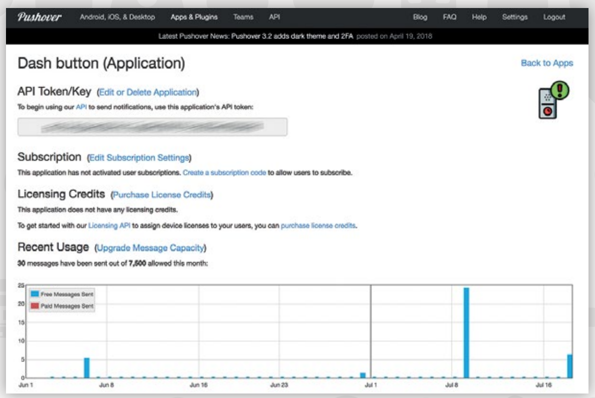
You'll need to create a Pushover app for this to work. There is a seven-day trial, and it's $4.99 / £3.83 for a licence. Head over to **pushover.net**, log in, and then create an application. Give it a name (Dash Doorbell) and upload an icon – there's one in our source code if you want to use that! You will then get an API Token/Key, and you can find your user token on the main screen. Keep these safe!

## STEP 03 GRAB OUR PYTHON CODE

Clone our Python code from GitHub with:

```
git clone https://github.com/
raspberrycoulis/dashdoorbell.git
```

...and then you will need to make a few simple edits to get this up and running.


Pushover · Android, iOS, & Desktop · Apps & Plugins · Teams · API · Blog · FAQ · Help · Settings · Logout

Latest Pushover News: Pushover 3.2 adds dark theme and 2FA posted on April 19, 2018

Dash button (Application) — Back to Apps

API Token/Key (Edit or Delete Application)

To begin using our API to send notifications, use this application's API token:

[redacted]

Subscription (Edit Subscription Settings)

This application has not activated user subscriptions. Create a subscription code to allow users to subscribe.

Licensing Credits (Purchase License Credits)

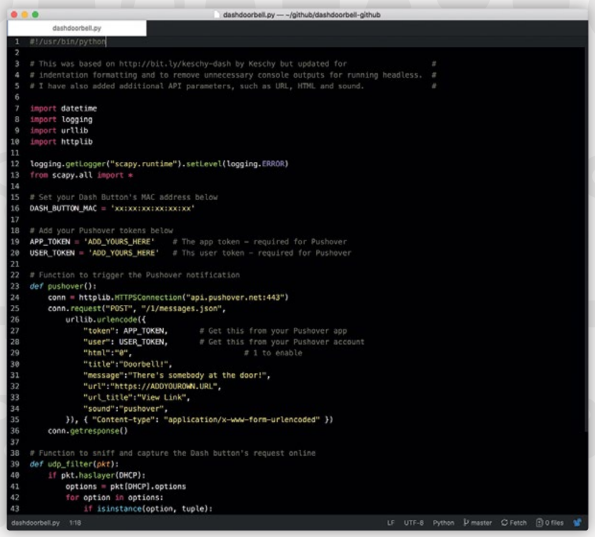
This application does not have any licensing credits.

To get started with our Licensing API to assign device licenses to your users, you can purchase license credits.

Recent Usage (Upgrade Message Capacity)

30 messages have been sent out of 7,500 allowed this month:

**Above** Create your Pushover app and then grab your user and API token keys. Why not use our app icon in the source code?


dashdoorbell.py — ~/github/dashdoorbell-github

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

# This was based on http://bit.ly/keschy-dash by Keschy but updated for              #
# indentation formatting and to remove unnecessary console outputs for running headless.  #
# I have also added additional API parameters, such as URL, HTML and sound.           #

import datetime
import logging
import urllib
import httplib

logging.getLogger("scapy.runtime").setLevel(logging.ERROR)
from scapy.all import *

# Set your Dash Button's MAC address below
DASH_BUTTON_MAC = 'xx:xx:xx:xx:xx:xx'

# Add your Pushover tokens below
APP_TOKEN = 'ADD_YOURS_HERE'    # The app token - required for Pushover
USER_TOKEN = 'ADD_YOURS_HERE'   # Ths user token - required for Pushover

# Function to trigger the Pushover notification
def pushover():
    conn = httplib.HTTPSConnection("api.pushover.net:443")
    conn.request("POST", "/1/messages.json",
        urllib.urlencode({
            "token": APP_TOKEN,       # Get this from your Pushover app
            "user": USER_TOKEN,       # Get this from your Pushover account
            "html":"0",                         # 1 to enable
            "title":"Doorbell!",
            "message":"There's somebody at the door!",
            "url":"https://ADDYOUROWN.URL",
            "url_title":"View Link",
            "sound":"pushover",
        }), { "Content-type": "application/x-www-form-urlencoded" })
    conn.getresponse()

# Function to sniff and capture the Dash button's request online
def udp_filter(pkt):
    if pkt.haslayer(DHCP):
        options = pkt[DHCP].options
        for option in options:
            if isinstance(option, tuple):
```

#CODE

LANGUAGE:
**Python**

NAME:
**dashdoorbell.py**

DOWNLOAD:
**magpi.cc/JYfoFv**

**Left** You'll need to add your Dash Button's MAC address and Pushover user and application tokens to the **dashdoorbell.py** script before it'll work

Alternatively, you can type the code yourself using your favourite code editor, but cloning is a lot quicker! Assuming you are in the **dashdoorbell** directory, run **`nano dashdoorbell.py`** to edit the code, replacing the MAC address, app and user tokens where shown, then exit (**CTRL+X**) and save (**Y**) to keep the changes.

## STEP 04 TEST IT OUT!

Before you affix your Dash Button to your door frame, you might want to just check that everything works as it should. After you have added your MAC address, app, and user tokens in the previous step, you can test the Python code by running **`sudo ./dashdoorbell.py`** in the Pi's Terminal. You won't see anything in the printout, but push the button once or twice and you should get a Pushover notification on your smartphone or tablet!

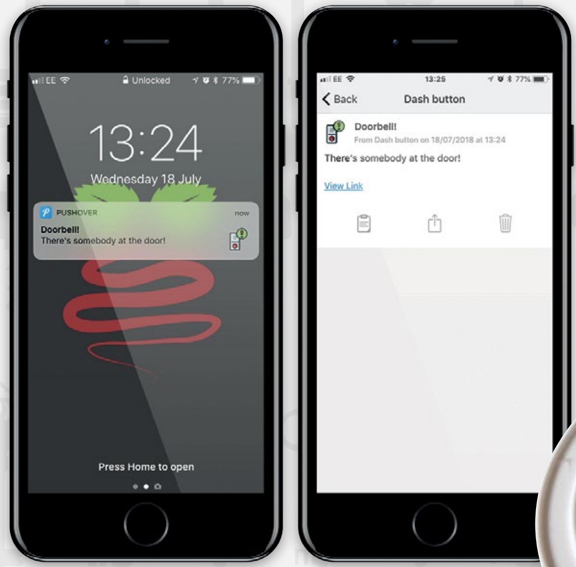
## STEP 05 MOUNT THE DASH BUTTON

Now everything is working, you can install your Dash Button wherever you want it. The button comes with an adhesive sticker on the bottom, so you can just use that to affix the button, but please keep in mind that this is semi-permanent and also remember that Dash Buttons are not waterproof! You could also put it on your bedroom door to stop your nosey parents, sister, or brother coming in unannounced.

## STEP 06 EDIT THE MESSAGE SENT

The default message in our code is basic and just tells you that 'There's somebody at the door!' with the title 'Doorbell!'. However, if you want to you can easily change this to whatever you like by editing the relevant parts in the **dashdoorbell.py** code. Using the Pushover API is very simple, and well documented over at **pushover.net/api**, but feel free to experiment. You can even change the sound by changing the appropriate part in the code!

**Below** Why not create your own label? Just peel the old one off and use the template in the source code to create your own



Doorbell!
There's somebody at the door!

Doorbell!
From Dash button on 18/07/2018 at 13:24
There's somebody at the door!
View Link

**Left** If set up correctly, when you push the Dash Button you should receive a push notification that tells you somebody is at the door!

**Maker Says**

"An extremely compact industrial server based on Raspberry Pi Compute Module"

**Sfera Labs**

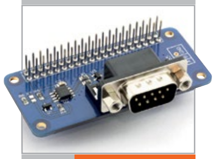
# STRATO PI CM

Need a robust housing/motherboard for your Raspberry Pi Compute Module? **Brian Corteil** gives the Strato a spin

**Related**

**RS485 PI**

This only covers one of the features of the Strato Pi CM, at a fraction of the cost. It's not as well protected as the RS-485 port on the Strato Pi CM.



£13 / $17

magpi.cc/OwvjKJ

The Strato Pi CM is a housing/motherboard designed for all versions of the Raspberry Pi Compute Module (CM) and is from a line of industrial products for the Pi/CM from Sfera Labs. The Strato Pi CM can be supplied with or without a Compute Module.

Designed for the IoT market, it enables a Raspberry Pi to interface with industrial equipment/sensors to the internet. Possible roles for the Strato include collecting sensor/diagnostics data, as a web interface to the collected data, to bridge old equipment via its RS-485 port to the internet, and for a webcam CCTV network video recorder (NVR).

Opening the box, we found a standard two-gang DIN module that can be mounted to a DIN rail type normally used in electrical distribution boards and industrial control panels. The removable connector block is clearly labelled along with the LEDs on the front panel. On close inspection, the module has an RS-485 interface, two USB ports, and an Ethernet port. Other features include 9 to 28 V DC power voltage, a real-time clock, controllable LED, and a hidden push-button.

Obtaining access to the PCB proved very simple: you just need to remove the DIN rail retaining clip, power/RS-485 block and, using a small bladed screwdriver, the PCB is exposed. There are a couple of new interfaces: a micro USB port for programming the CM, and a microSD holder if you are using a CM Lite board and the hidden button.

Your reviewer handed the Strato to friendly electronic engineer and fellow Makespace member Rob Karpinski for his view of the construction and design. Having given it a once-over, he said that it was well made and that the RS-485 inputs were well protected. When told the cost of the unit, he choked a little but did say that it was expected for a device designed for industrial use.

magpi.cc/e0FNbF

From £264 / $290

Sfera Labs sent the Strato Pi CM Solo for review; this version does not come with a CM, but installing one is easy: just line it up and push down until it clicks. The next step is to install Raspbian. Reading the included Quick Reference, we were pointed to the user guide on the Sfera Labs website and then instructed to follow the guide on the Raspberry Pi site – **magpi.cc/UtAcOj**. We followed the guide and used the micro USB port; the user guide did not mention that the Strato had to be powered at the same time to install Raspbian Stretch Lite on the CM eMMC. We remembered to include a file called **ssh** to enable remote access and then connected to the Strato via its Ethernet port using PuTTY.

Following the user guide, we installed the Strato Pi utility and the software supporting the real-time clock. We played around with the utility, turning on/off the LED, and restarting the CM using the watchdog feature – although it turns out you don't need to use the utility to access these features or the button.

## Test script

After installing the GPIO Zero Python library, we wrote a simple script to flash the LED every couple of seconds for ever. This script has now been running for over 14 days; the Strato module has been sitting in your reviewer's garden office / toy room, in temperatures over 35°C with no issues.

Next thing to test was the RS-485 serial port. Luckily, we had an RS-485 to RS-232 and a serial-to-USB converter in the test equipment box. Using PuTTY on the PC, we were able to access the CM's console and use the command line as normal. With the correct cable, this means we could use the console or have sensors/equipment installed up to 1000 metres away.

## Niche market

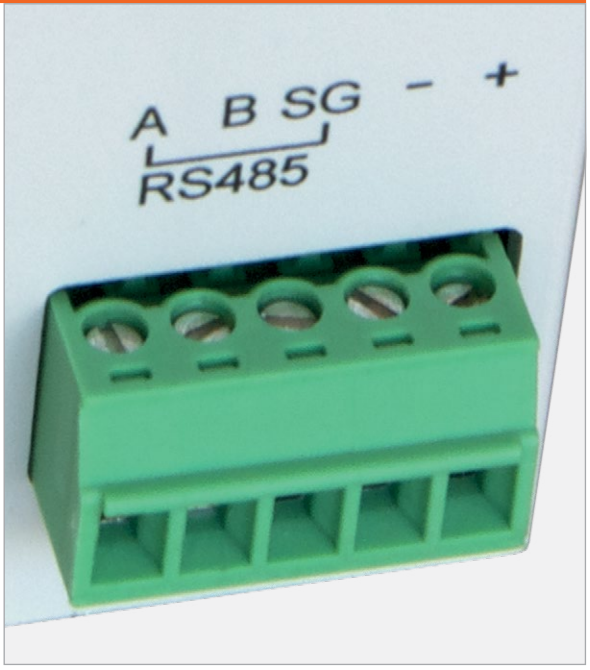
The Strato Pi CM is for a niche market that requires the versatility of the Raspberry Pi in a small compact format that the Compute Module allows, and the professional user requires. The build quality is excellent and with a well-thought-out design. The inclusion of a real-time clock, isolated RS-485 serial interface, LED, and button, plus the wide range of voltages the Strato Pi can take for its power input, are its strengths. Adding all these functions to a standard Pi would require the use of several HATs stacked on top of each other. Your reviewer would personally like to see some GPIO pins broken out from the Compute Module with the same protection as the RS-485 port, to allow the adding of I²C, SPI devices, sensors, and extra LEDs and buttons.

> "The build quality is excellent and with a well-thought-out design"

### Last word

**If you need the power of the Raspberry Pi 3 Compute Module in a compact industrial case with an RS-485 interface, we don't think you would go wrong with the Strato Pi CM.**

★★★★☆



Right Why stack HATs when you can just use this?

magpi.cc/WgSmAA

£40 / $51

Maker Says

"A high-resolution, high-speed 4.0″ TFT display for your Raspberry Pi

Pimoroni

# HYPERPIXEL 4.0

**Rob Zwetsloot** looks at this bigger touchscreen for your Pi that is still palm-sized

It was only last year that we reviewed the 3.5-inch HyperPixel from Pimoroni, itself a fantastic touchscreen offering great pixel density in a diminutive, Pi-sized form factor.

> The screen is super-lovely once working – crisp and clear visuals thanks to the pixel density

This new, bigger HyperPixel model keeps the same high-quality parts but gives you more screen real estate to work on.

The larger size does mean that it doesn't fit within the confines of the (standard) Raspberry Pi form factor, with a good chunk of the screen hanging off the SD card side of the board. It's really not that bad, but it does mean if you have any cases that previously allowed you to squeeze in the 3.5-inch HyperPixel or other displays, it likely won't work with the new HyperPixel. It's an interesting dilemma as, due to the existence of stuff like the A+ and the Pi Zero, the standard Model B form factor is not as ubiquitous as it once was. If the old 3.5-inch screen would outsize a Pi Zero, then why not make it a little bit bigger anyway?

## One-line install

As usual, this bit of Pimoroni kit is really simple to install. All you need to do is attach the screen to a Pi, hook the latter up to the usual monitor and keyboard arrangement before turning it on, and then enter a one-line command in the Terminal to get all the software properly installed. It doesn't take long and you'll then be immediately ready to go!

The screen is super-lovely once working – crisp and clear visuals thanks to the pixel density, with responsive touch controls and wide viewing angles due to the display tech used. It's basically everything you'd want from a screen, and the Raspberry Pi can handle it just fine.

If the touchscreen controls aren't a necessity, you can get a HyperPixel without them and save a few quid (and probably a bit of power draw). Either way, it's another top Pimoroni product.

Related

**OFFICIAL TOUCH DISPLAY**

A larger screen at the same resolution as the HyperPixel 4.0, its size may be more attractive for some projects.

£60 / $76

magpi.cc/2mftBGX

Last word

A beautiful screen that's easy to install with a responsive touchscreen perfectly suited for every Pi with 40-pin GPIO. What more could you ask for?

★★★★★

THE MAGAZINE FOR COMPUTING & DIGITAL MAKING EDUCATORS
(Hello World)
DATABASES AND BIG DATA
Why do many struggle with databases - and what can you do about it?
INSPIRING ENGINEERS!
Coding and INCLUSION
ORGANISING A COMPUTING DAY OUT
PSSST! IS IT OKAY TO COPY?
Why building on existing code is great for teaching

# 10 BEST RASPBERRY PI ROBOT KITS

Make the perfect Raspberry Pi automaton with these excellent robot kits

The Raspberry Pi has helped create a new boom in hobby robotics, largely thanks to its price, easy hackability, size, and a great community willing to share their robot expertise. It's never been easier to join in yourself, thanks to a plethora of robot kits for the Pi. Here are ten of the best, in no particular order.

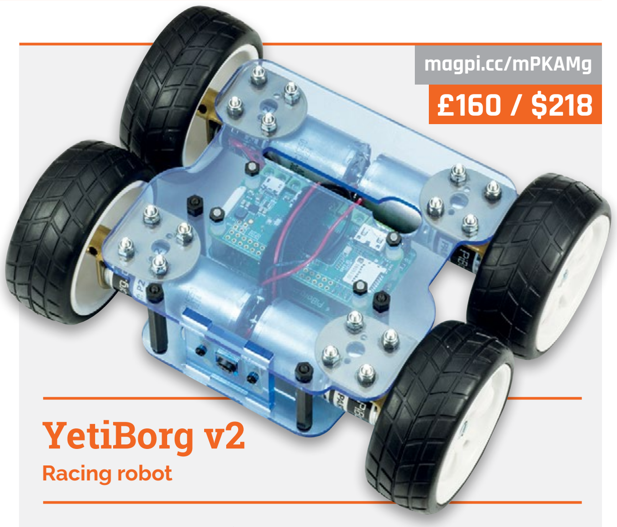
## CamJam EduKit 3

**Minimalist beginners kit**

This fantastic, simple kit has been around for a while to promote Pi Wars, the annual Raspberry Pi robot challenge. It includes all the basic stuff you need to create a robot – all you need to do is supply your own Raspberry Pi! You can even use the box as the robot's chassis.

magpi.cc/NhrRMf

£18 / $24



## YetiBorg v2

**Racing robot**

magpi.cc/mPKAMg

£160 / $218

A more beginner-friendly PiBorg kit compared to the DiddyBorg, the YetiBorg is a smaller design that is perfect for people relatively confident in their coding skills who want to apply them to something physical. It's also used in the Formula Pi racing series, where automated robots compete against each other using code from teams around the world.

## RaPiRo

**Cute biped**

This adorable robot may be on the expensive side, but it's unusual for a Raspberry Pi robot as it walks. And waves at you. Construction is a bit tricky, and coding is trickier still, but once you have it walking around and interacting with things, it's all well worth it.

magpi.cc/iuuzdy

£426 / $539

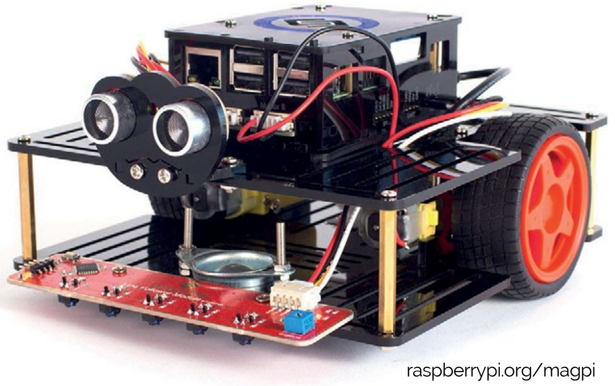
## Smart Robot Car

**Fully automated**

magpi.cc/NzjPRy

£107 / $140

While all the robot kits here can be made autonomous, this kit is one of the few that comes with all the equipment you'll need to make it so from the start. It has a line sensor, an ultrasonic distance sensor, and voice recognition. You can even program it in Scratch.



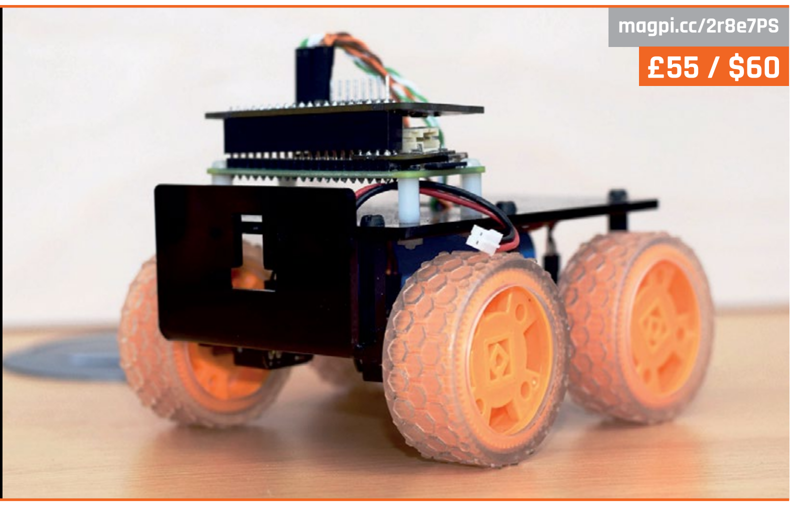
## Tiny 4WD

**Competition ready**

This robot kit from maker Brian Corteil is based on a robot-building feature he wrote way back in *The MagPi* #51, improving upon the design and supplying it one complete box. It's powerful, extremely quick, and very sturdy. Thanks to the Explorer pHAT motor controller, it's also very flexible and can be modified to any of your requirements.

magpi.cc/2r8e7PS

£55 / $60



## Build a robot!

Want to skip the kits and build a robot from scratch? Check out issue 51 of *The MagPi* for tips on what you need to make your own robo-companion: **magpi.cc/51**

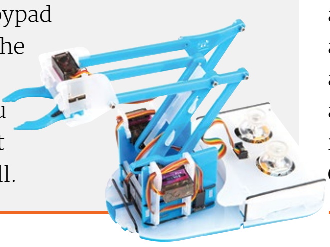
## MeArm Pi

**Robot arm**

magpi.cc/eErXww

£70 / $76

While not a very mobile robot, the MeArm is a robot nonetheless, albeit more like one you'd find in a Peugeot factory signing Picasso cars. One of the coolest things about the project is that it comes with a very neat joypad setup that fits on top of the Pi and controls the arm. With a bit of hacking, you can probably repurpose it for something else as well.



## Smart Video Car

**Remote controlled**

magpi.cc/cLKQMe

£75 / $99

This great kit lets you control a robot out-of-the-box from an Android phone or browser. The camera can even be tilted and panned to look around. The kit is aimed at beginners and uses a webcam rather than the Pi Camera Module.

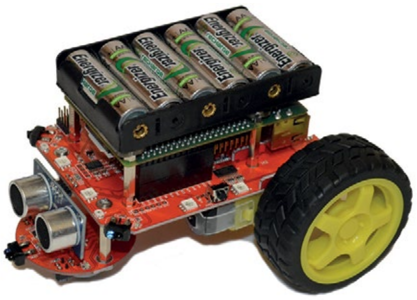
## GoPiGo 3

**Educational automaton**

While a lot of Raspberry Pi robot kits have an educational aspect to them, the GoPiGo goes one step further by not only offering an easy-to-use series of base kits for individuals, but also special kits with enough robots for an entire classroom. They'll even help you get the right educational information.

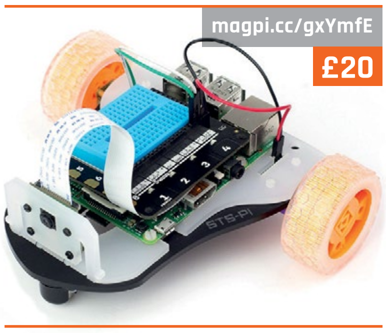
magpi.cc/MURCkG

£96 / $100



magpi.cc/gxYmfE

£20



## STS-Pi

**Spaceship for beginners**

This very simple Pi robot looks like a Space Shuttle (hence the name STS, which was used for Space Shuttle mission numbers), giving it immediate cool factor. It's a simple build with not much ability to be developed further as easily as other kits, but it's great for beginners and especially younger makers.

## DiddyBorg v2

**Serious robotics**

magpi.cc/eUEseN

£210 / $286

This beast of a kit from the robotics whizzes at PiBorg is powerful, sturdy, and endlessly upgradeable. It's everything you'd want from a robot kit that you plan to use as a test platform, or as the base of a bigger, meaner build. There's even a faster version called the Red Edition.

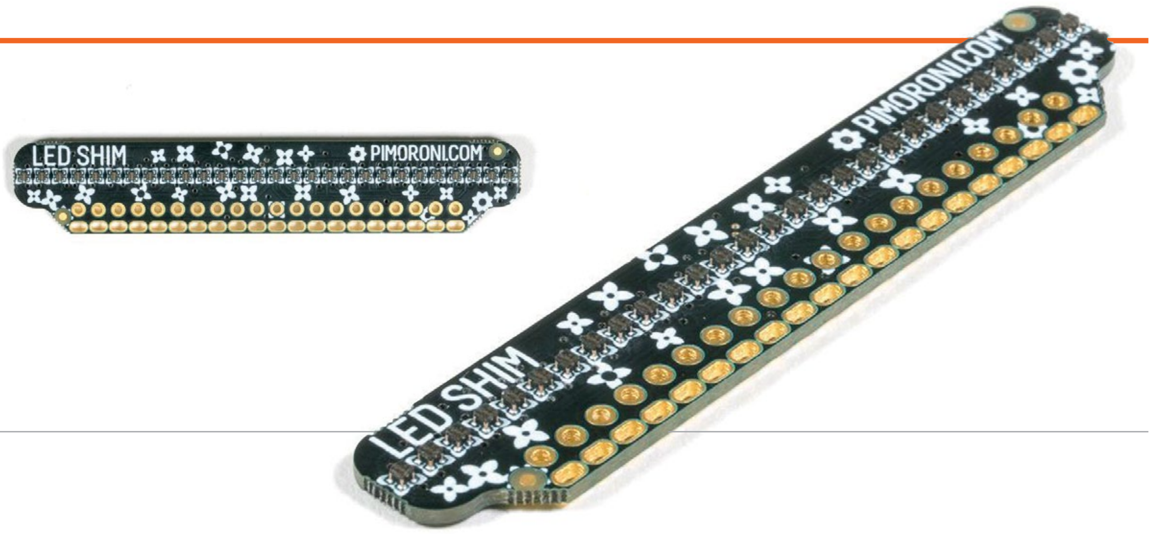
magpi.cc/vAvVlG

£8 / $8

Maker Says

“28 tiny RGB LED pixels in a single row”

Pimoroni



# LED SHIM

This slimline LED strip could give your Pi the (illuminated) edge, as **Phil King** discovers

After upping the pixel count on its Unicorn HAT and Scroll pHAT, Pimoroni has done a similar trick for its Blinkt! LED strip, but this time it comes in super-slim SHIM form. Packing 28 tiny RGB pixels – about half the size of the Pi’s own status LEDs – the LED SHIM simply slips onto the GPIO pins with no soldering required. The ‘friction fit’ keeps it in place, even when turned upside down. While it worked fine on most Pi boards we tried, we did have an issue with a poor pin connection on one Pi that resulted in an I/O error. A bit of jiggling usually fixes the issue but if not, there’s always the option of soldering the SHIM to the GPIO or adding a female header.

At a mere 0.8 mm thick, the SHIM leaves plenty of room on top to add a HAT or pHAT. And since it only uses two I²C pins, there should be no pin conflict issues. The LED strip protrudes from the edge so it’s still perfectly visible with another board on top; the only downside is that you might have trouble fitting the Pi in a case.

> “The SHIM leaves plenty of room on top to add a HAT or pHAT”

## Bright and beautiful

Arranged in a single row, and driven by the same LED matrix chip used on the Scroll pHAT HD, the 28 LED pixels are tiny but bright. A one-line installer command adds the software library and a host of examples (**magpi.cc/AoDrqc**). The latter demonstrate numerous possible use-cases – such as a VU meter, Twitter status, and data display – as well as some impressive animated effects.

There are also some examples of using the SHIM with other boards such as the Enviro pHAT – for which there’s a colour-coded direction meter and a spirit level.

Coding it is similar to on the Blinkt!, using a ‘set’ function to select a pixel’s RGB shade, then a ‘show’ function to light it. So it’s simple to start creating your own lighting effects.

## Related

**BLINKT!**

Equipped with eight larger RGB LEDs, the original Blinkt! has a female header to fit onto the GPIO.

£5 / $5

magpi.cc/tnbkRL

## Last word

**Ideal for a status display, or some cool lighting effects on the edge of your Pi, this super-slim SHIM has numerous possible uses and can work with add-on boards on top.**

★★★★☆

CoderDojo

# Can I start a CoderDojo club in my local area?

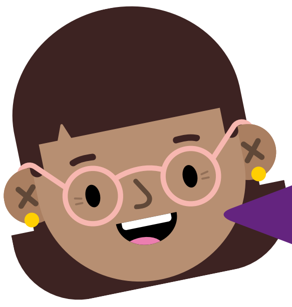
CoderDojo is a global network of free, volunteer-led, project-based programming clubs for children aged 7–17. Dojos are championed by individuals all around the world who are passionate about giving young people the opportunity to learn to code.

## Starting a Dojo is a fun and incredibly rewarding experience

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

We're ready to support you by providing:

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



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

- Maroes, CoderDojo NL

**Start your own club. Join us at CoderDojo.com**

magpi.cc/uvJcQa

£5-£20 / $7-27

Maker Says

Components and useful bits for everyday electronics and tech projects

Pimoroni

# MAKER ESSENTIALS

A selection of neatly packaged parts for makers

One of the coolest things about coding with a Raspberry Pi is its integration with electronic components.

Every maker starts out by discovering how to set up the GPIO pins to a breadboard, and create simple circuits with LED lights, resistors, and switches. Typically the next step is building and controlling a robot from simple motors and wheels.

There's no shortage of starter guides on how to blink an LED with a Raspberry Pi, or use buttons to interact with code. But it's often weirdly confusing for a beginner to get hold of components to go with online tutorials.

Which is where Pimoroni hopes its Maker Essentials kits will step in. This collection of low-cost components offers a neat and simple way to get basic equipment without having to buy packs of parts from random eBay or Amazon sellers.

There are five different packs available: LEDs & Resistors (£5.50), Various Headers (£5.50), Mini Breadboard & Jumpy Jerky (£9.20), Switches & Potentiometers (£7.26), and Micro-motors and Grippy Wheels (£22).

## Quality components

Each pack is sealed and contains a selection of the appropriate parts. Quality is high across the board, although some packs seem to make more sense than others to us.

The collection of LEDs & Resistors is valuable for any beginner. The Mini Breadboard & Jumper Jerky would be a good pack, but one larger breadboard would be infinitely more practical than two mini boards.

The Switches & Potentiometers is a useful pack of parts, but does anybody really need the 48 headers in the Various Headers pack? (We're sure folks will get in touch saying they can never have enough headers.)

Perhaps the most interesting pack is Micro-motors & Grippy Wheels, which offers a neat path to a micro robotics kit. The motors and wheels are dinky, but you'll need a motor controller (which you could get for the same price included in some other kits).

We couldn't find any online instructions for the kits. Given the simple nature of the components, that's not a major issue: there's no shortage of tutorials for any of them. Even so, beginners will be better served with a package that combines parts with instructions.

Related

CAMJAM EDUKIT #2

CamJam EduKit #2 comes with a larger breadboard, a range of LED lights and sensors, and an online set of detailed worksheets.

£8 / $11

magpi.cc/IBortt

Last word

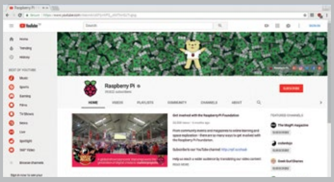
**These are neat packs of high-quality components that serve a simple purpose, but do it well. Total beginners should look to more comprehensive kits with project instructions, though.**

★★★★☆

# LEARN PYTHON WITH RASPBERRY PI

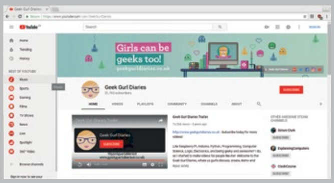
## YOUTUBE VIDEOS

The best Python YouTube sites



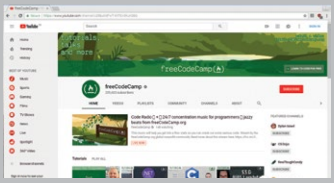
### RASPBERRY PI

The Official Raspberry Pi channel is packed with resources for learning to code, and almost all of them feature Python. You'll also find advice on setting up a Raspberry Pi, plus how to make awesome things with Raspberry Pi, Python, and electronics. Be sure to subscribe!
**youtu.be/QAHJVe2jq_E**



### GEEK GURL DIARIES

This fantastic website offers a crash course on computer science and programming by none other than Code Club's Carrie Anne Philbin. It covers a range of boards and technologies, but be sure to take a look at the Raspberry Pi tutorials.
**youtu.be/bZwubV-m9JM**



### LEARN PYTHON

A more advanced collection of video tutorials put together by Free Code Camp. This comprehensive course isn't Raspberry Pi specific, but covers everything from getting started with strings all the way up to objects and inheritance.
**youtu.be/rfscVSovtbw**

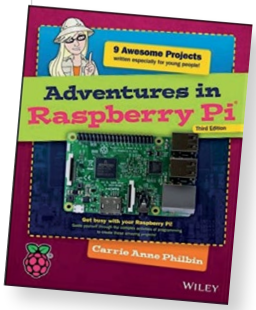
## ADVENTURES IN RASPBERRY PI

New to Raspberry Pi? Want your children to learn coding and have fun connecting up lights, sensors, or even marshmallows? Or even use the Pi to make a dedicated device? You've come the right place. Philbin is a technology educator with plenty of experience of bringing alive the Pi's possibilities for a young audience.

After setting up the Pi, and learning some command-line basics, readers get to try using the Scratch programming language to animate a monkey sprite, then start building a role-playing game, before diving into turtle graphics to make shapes with both Scratch and Python. Sonic Pi continues the coding – making music with Ruby code. Throughout these divergent adventures, additional notes, definitions, and tips and tricks combine with the lessons to more broadly educate new programmers.

The last chapters are on hardware: using the GPIO with those sensors and marshmallows, and building a Pi jukebox. Readers who already have the second edition (see *The MagPi* #32) will want to know what's new – a whole extra chapter on experimenting with cameras, and expanding your Pi's capabilities with add-on HATs – but won't need any persuasion to buy extra copies as Christmas presents for young relatives!

**Author:** Carrie Anne Philbin
**Publisher:** Wiley
**Price:** £20.99
**ISBN:** 978-1119269069
magpi.cc/2zt24rO



## TEACH YOUR KIDS TO CODE

Dr Payne promises "programming so easy a parent can do it!" Starting with turtle graphics, the reader is drawn in, and Python seems natural, easy, yet still a thing of wonder.

Learners are encouraged to experiment, rather than overloaded with details of how and why – but where details are necessary, such as number types and operators, they are introduced.

As concepts are brought into play, we feed them into turtle graphics – so conditionals lead to fractal spirals, and user input selects the shape drawn.

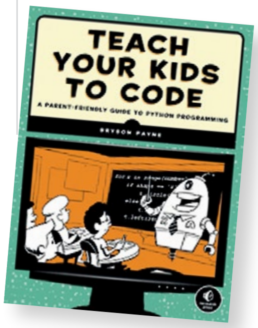
Programming challenges at the end of each chapter – turning the High Card game into War, or adding sound effects to the Pong game – steer further learning.

Aimed at children aged nine and up, there's enough here for everyone – parents can take younger children through some of the projects, and teens shouldn't feel talked down to.

Payne has the balance right between giving enough in plentiful, fun projects to keep interest, and introducing programming concepts to build real understanding almost by osmosis.

Python and Pygame installation are banished to an appendix, where you'll also find instructions on creating your own modules! And once more, delightful illustrations by Miran Lipovača lift an already excellent No Starch book to another level. Strongly recommended for learners of all ages.

**Author:** Bryson Payne
**Publisher:** No Starch
**Price:** £19.99
**ISBN:** 978-1593276140
magpi.cc/TYkZLo



## PROGRAMMING THE RASPBERRY PI

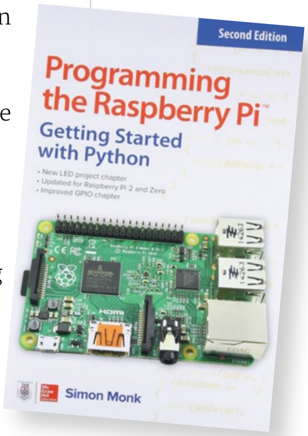
This could be the perfect introductory programming book to give to someone who's just got a Raspberry Pi for Christmas.

With no wasted words, Monk introduces the Pi and its operating system, then teaches both Python and using Python with the Pi, in a direct and easily absorbed text that harks back to the best beginner guides of the 8-bit era.

Skip the first two dozen pages if you are not new to the Pi, and dive into Python with an introduction that – through well-chosen examples, such as a dice rolling simulation – will have you learning conditionals, control flow, and comparison operators before you know it. Next, Hangman introduces functions, as well as strings, lists, and dictionaries. OOP is touched on with a temperature converter, then file handling and GUI programming (with Tkinter) through building on the earlier code examples. The Pygame chapter makes use of many of the techniques learned, then moves on to refactoring.

The same concise style is used to cover the Pi hardware for the rest of the book: GPIO pins, breadboard prototyping, connecting an Arduino, then a range of sample projects culminating in a Raspberry Pi robot. Unreservedly recommended for confident beginners of all ages.

**Author:** Simon Monk
**Publisher:** Tab Electronics
**Price:** £11.64 / $15
**ISBN:** 978-1259587405
**magpi.cc/224fOlh**


Second Edition
Programming the Raspberry Pi
Getting Started with Python
• New LED project chapter
• Updated for Raspberry Pi 2 and Zero
• Improved GPIO chapter
Simon Monk

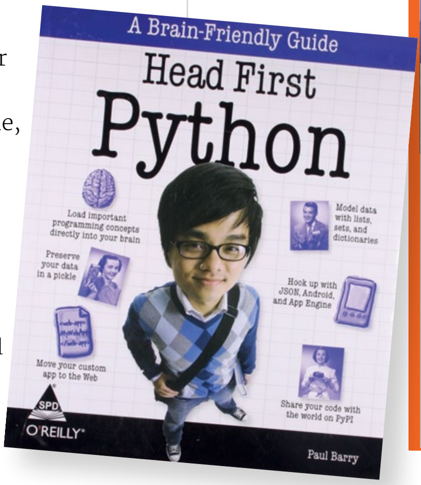
## HEAD FIRST PYTHON

Whether or not you like the quirky, visual style of O'Reilly's Head First books, this is an excellent introduction to Python, well written and well paced. Aimed at those with at least a little coding experience in another language, Paul Barry's text starts iconoclastically by sidestepping Hello World and diving into a more in-depth first program, bringing in the list data structure, importing from the standard library, and introducing the powerful in operator.

The data-first approach, with lists, then dictionaries, tuples, and sets, is a firmer foundation than starting with control flows. Functions follow, and making your own modules, which you then install in the cloud at PythonAnywhere, to support the web app you write with Flask. Difficult subjects like decorators, comprehensions, and generators are painlessly introduced, as our app is constantly refactored, and communicating with an SQL database is carefully taught.

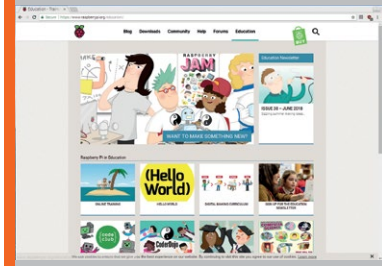
PEP8 gets an early mention, but otherwise testing is left. Nor is there anything on Jupyter or doing data science, for example, but this is a book that teaches a lot, very quickly, and confident learners who find other introductory books too slow-paced will get a head start in Python with this Head First guide.

**Author:** Paul Barry
**Publisher:** O'Reilly
**Price:** £39.99
**ISBN:** 978-1491919538
**magpi.cc/2nfiscb**


A Brain-Friendly Guide
Head First Python
Load important programming concepts directly into your brain
Preserve your data in a pickle
Move your custom app to the Web
Model data with lists, sets, and dictionaries
Hook up with JSON, Android, and App Engine
Share your code with the world on PyPI
O'REILLY
Paul Barry

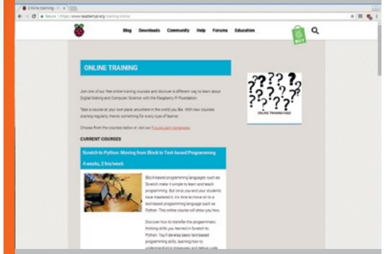
## INTERACTIVE WEBSITES

### Python web resources



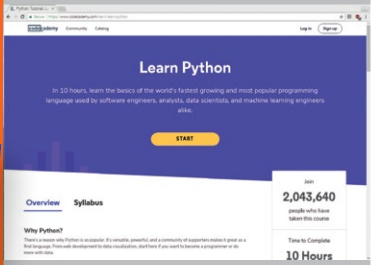
### RASPBERRY PI

The Raspberry Pi Education section on the official website should be your first port of call when learning anything with a Raspberry Pi, and Python is no exception. Here you'll find tutorials, blog posts, and education resources. Sign up to the Education Newsletter while you're there.
**rpf.io/education**



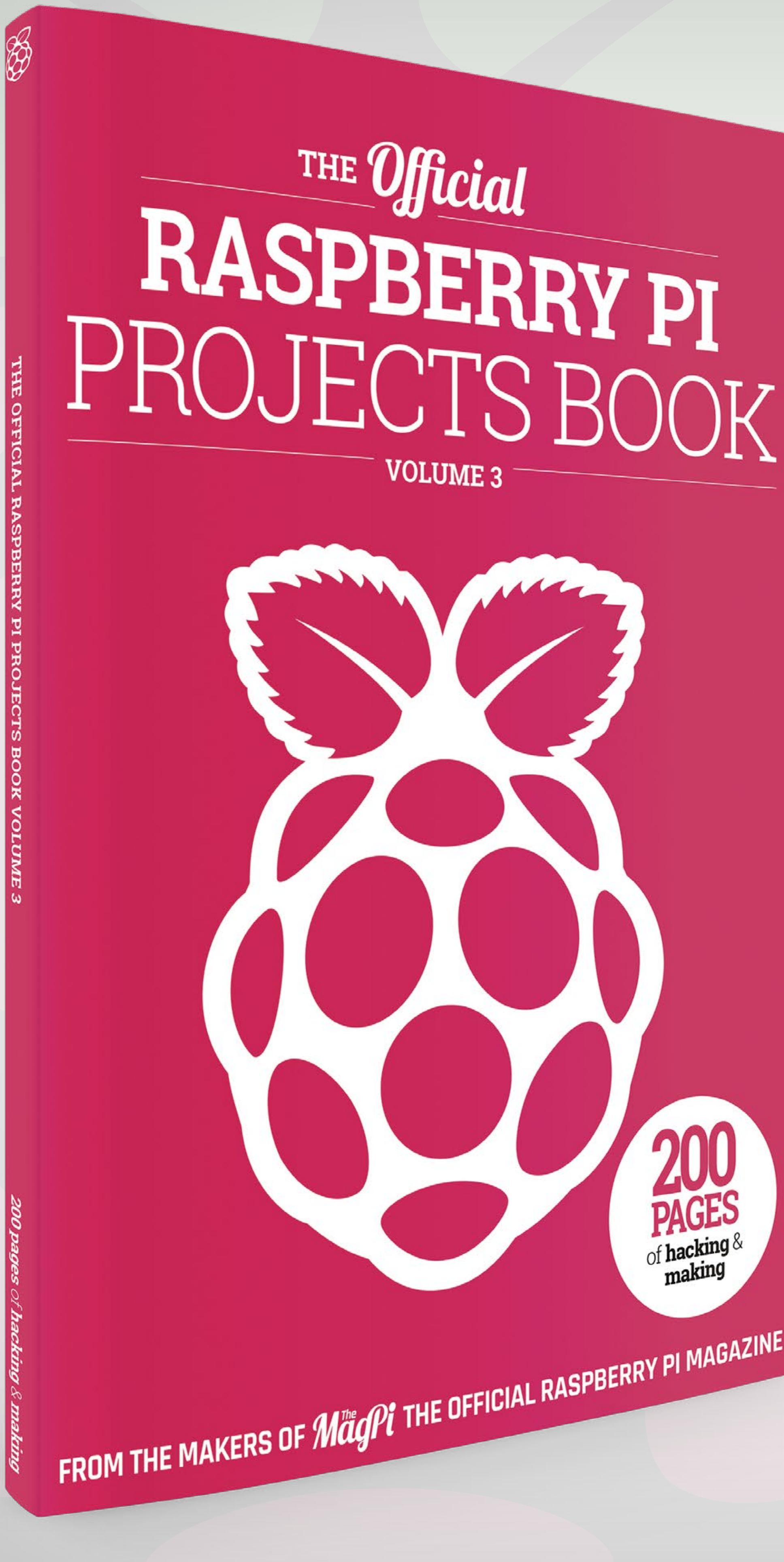
### FUTURE LEARN

Raspberry Pi has teamed up with FutureLearn to provide interactive courses, most of which are in Python (and there's a great Scratch to Python course). The courses are structured to last three to four weeks and only require a few hours a week to learn.
**rpf.io/online-training**



### CODECADEMY

One of the most popular coding sites around is Codecademy. Here you sign up and learn coding techniques in an interactive web environment. It's based on a 'gamification' approach, so you get points and win awards for completing code missions.
**magpi.cc/FUhUGy**

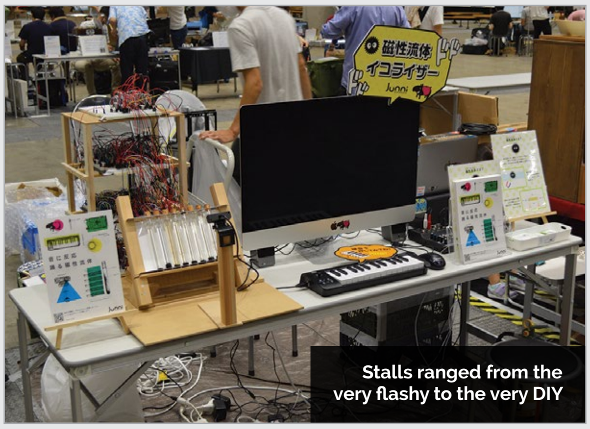

THE Official
RASPBERRY PI
PROJECTS BOOK
VOLUME 3
200 PAGES of hacking & making
FROM THE MAKERS OF The MagPi THE OFFICIAL RASPBERRY PI MAGAZINE
THE OFFICIAL RASPBERRY PI PROJECTS BOOK VOLUME 3
200 pages of hacking & making

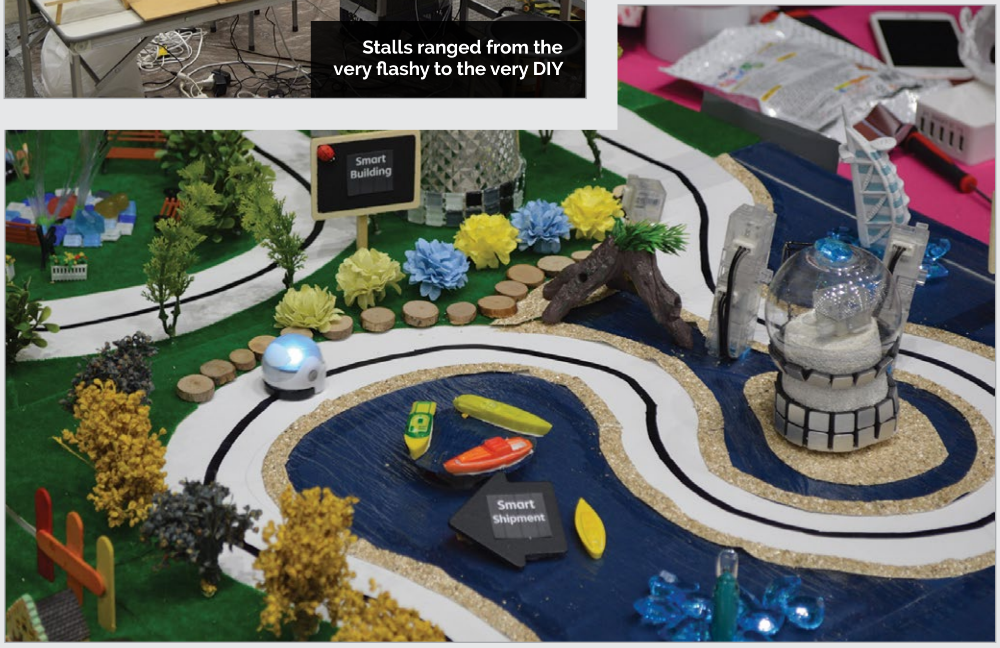
# MAKER FAIRE TOKYO

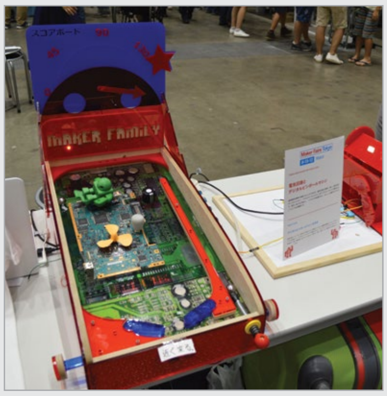
We head to Tokyo to see what the Japanese Raspberry Pi community has been getting up to lately

When we say the Raspberry Pi is used all around the globe, we truly mean it. There are thousands of Code Clubs and CoderDojos spread across the world and you'll find Raspberry Pis everywhere on (and in some cases orbiting around) the Earth. The Japanese Raspberry Pi community is huge and, as we found out during our visit to Tokyo for Maker Faire, incredibly inventive.

Robots wandering the show floor were common

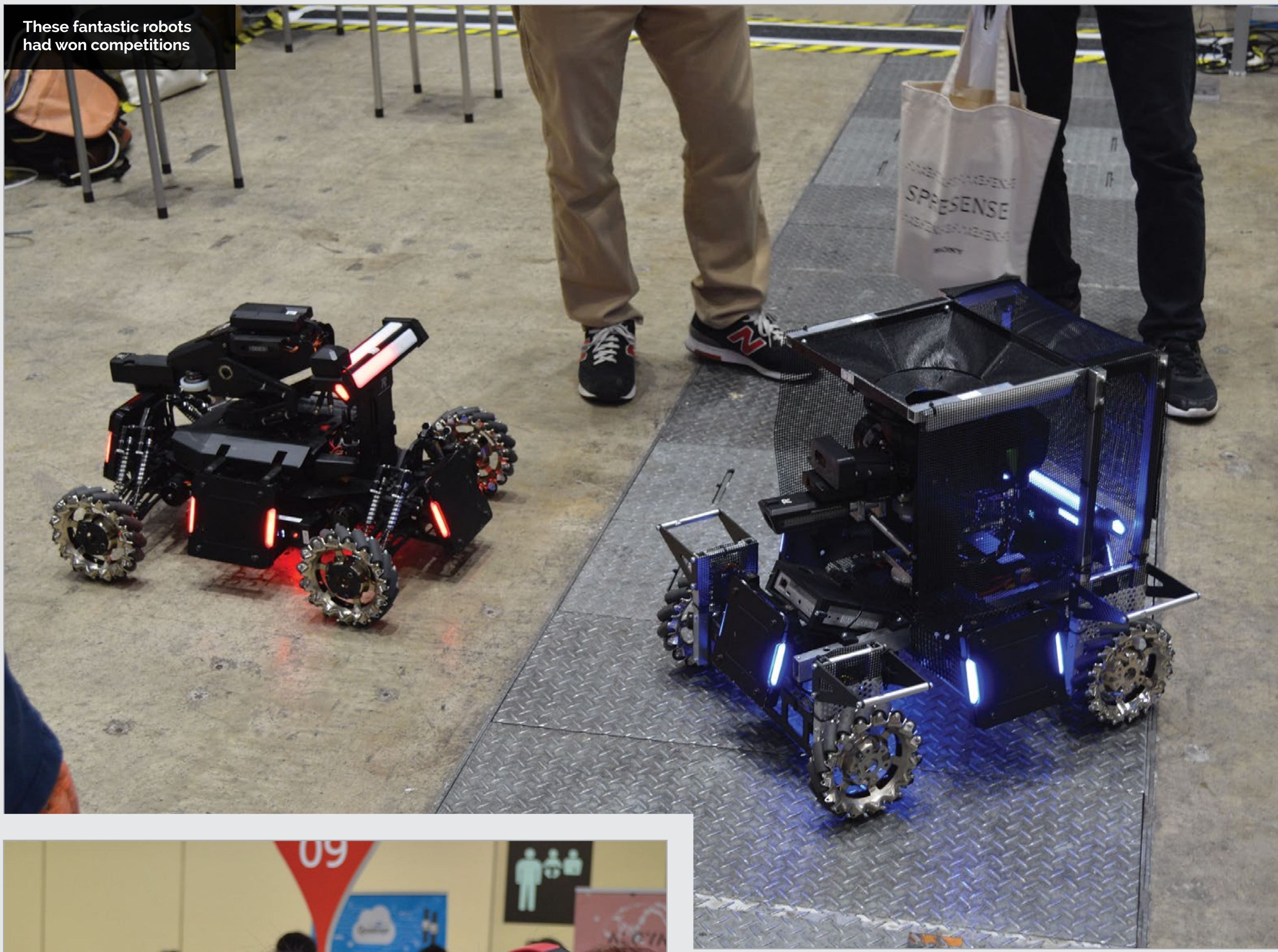

Stalls ranged from the very flashy to the very DIY





**Above** This custom pinball machine was made up of old computer parts

**Left** Korean projects also made an appearance, such as this cute line-following robot

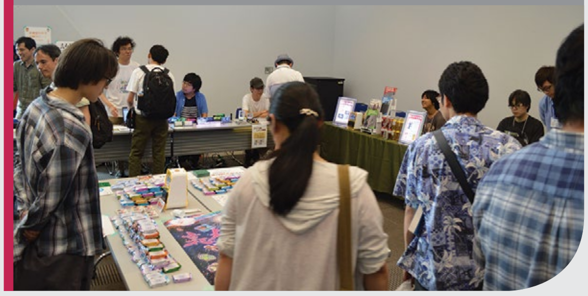


These fantastic robots had won competitions

Japanese pop culture was well represented

## UNDERGROUND MAKER FAIRE

Maker Faire Tokyo takes up a huge chunk of Tokyo Big Sight, a major international convention centre in the Odaiba area of Tokyo and also famous for housing Comiket. While plenty of makers were able to display projects, some were sadly unable to secure a spot. An unofficial Maker Faire was held for people still eager to show off their stuff.



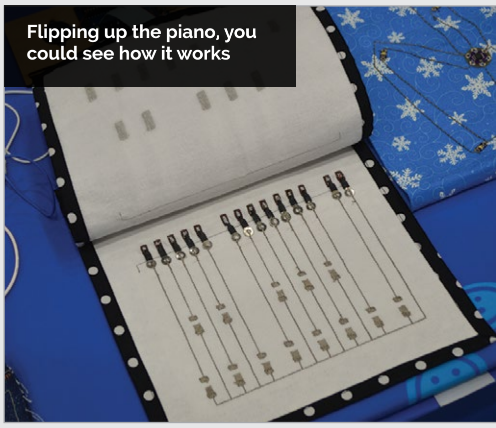
# MAKER FAIRE TOKYO

This LEGO loo roll tweeted whenever you… used it

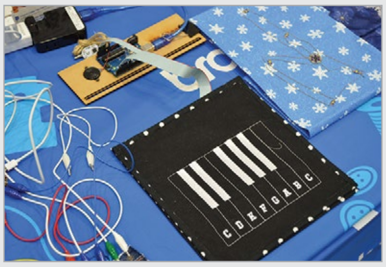
**Above** There were amazing selfie spots set up everywhere

**Right** Baking is a classic form of making, and we especially loved these Raspberry Pi-shaped cookies

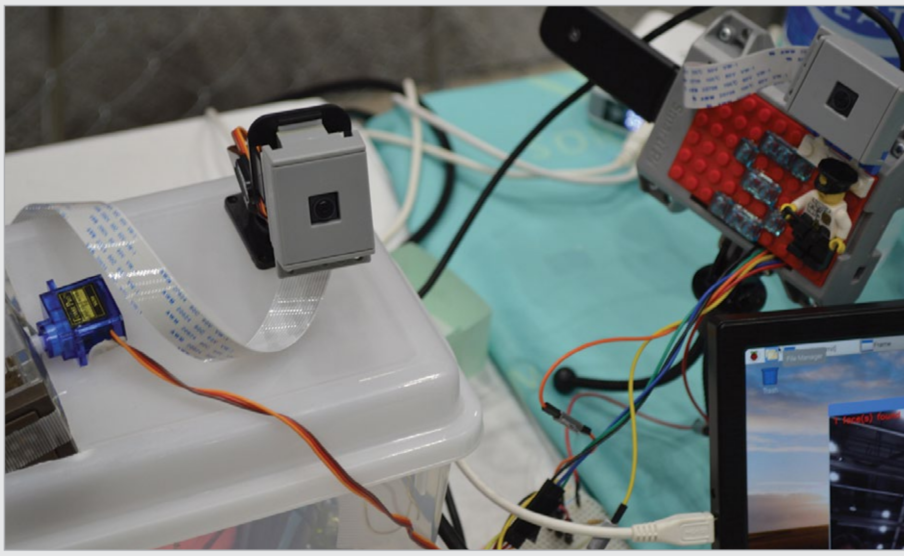
We saw more computer vision projects than ever

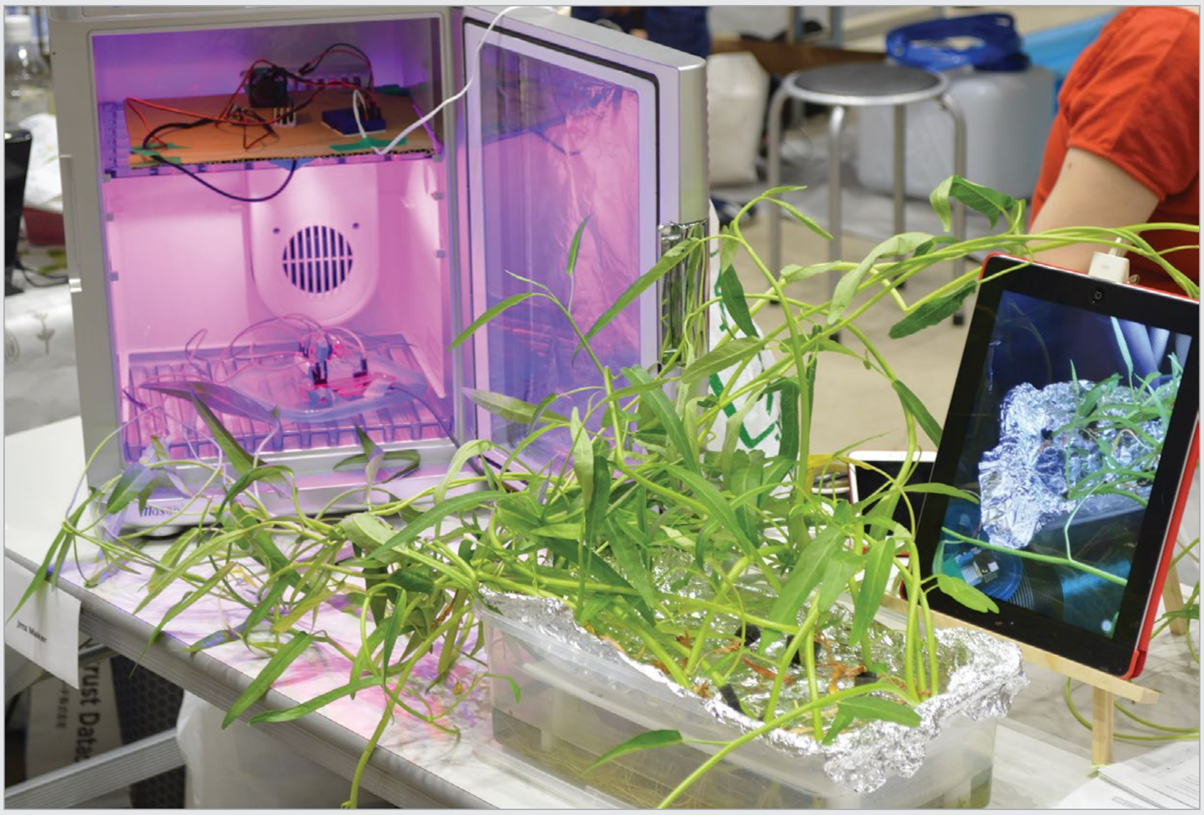

Flipping up the piano, you could see how it works

**Below** Brother was showing off projects using conductive thread, like this piano



Light-up balls for robot shooting helped with aiming





**Above** LEGO was rare here, but used very well with Pi

**Left** One of our favourite categories at Maker Faires is the agricultural products, and Tokyo did not disappoint

# COMPUTING EDUCATION AND PI IN JAPAN

In Tokyo's Electric Town, you can find a Pi in almost every store

**Above** Components are neatly arranged in every store, filling up as much space as possible

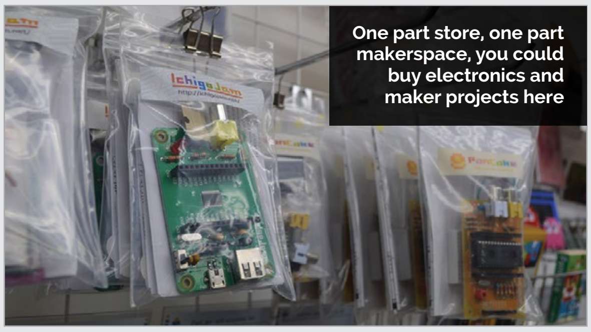
**Below** An old favourite of ours, Rapiro is still alive and kicking and you can buy them from Switch Science

Akihabara is one of the many areas that make up Tokyo, a loosely defined series of streets and shops in a sea of skyscrapers. Informally known as Electric Town, it's more famous these days to people in the West as a centre of Japanese pop culture and nerd stuff. But among the Animate and Super Potato stores, maid cafes, and UFO arcades still lie the electronics shops of the past.

Masafumi Ohta from the Japanese Raspberry Pi User Group showed us around Akihabara (or Akiba), pointing out all the interesting electronics shops. Row upon row of various components and electronics were lining the cramped shelves of the small stores, including sections dedicated to newer maker gear like the Raspberry Pi, Ichigo Jam, and Arduino. "You can also buy all of this online," he assured us.

## Make the switch

Earlier in the day we had taken a trip to Switch Science, an online supplier of Raspberry Pi and other maker goods in Japan. In Shinjuku, another area of Tokyo, Switch Science takes up a couple of offices where they ship out orders, make their own products, and create Japanese educational materials for young makers. "We saw a growing desire for educational materials among makers," Yoshichika Yasui of Switch Science told us. The office was a manic flurry of motion as orders were packed to be sent all over Japan, and even internationally.


One part store, one part makerspace, you could buy electronics and maker projects here

## Education spaces

Back in Akiba, Masafumi-san finally brought us to PCN's flagship location. One part store and one part makerspace, here you could buy the ever popular Ichigo Jam. It's a low-power, single-board PC in a similar vein to Raspberry Pi, albeit with a more basic control method. It uses a BASIC-esque programming interface, and school classes were free to come down and learn computing using it – in fact as we were talking, a retired teacher entered the store and proceeded to tell us how important this kind of education was.

It's also a general makerspace for people wanting to work on their projects in a communal setting – and there are plenty of projects to buy in the store and steps away in other electronics stores in the area. It's a hugely popular resource for the local community it seems, and really highlights one of the general themes we saw while with the Tokyo maker community: everyone enjoys helping each other, and wants to make sure there's a space to do so.

The office was very busy making sure orders were shipped

## CROWDFUND THIS!

Raspberry Pi projects you can crowdfund this month

kck.st/2zZulob

### AIR MONITOR & MULTIMEDIA KITS

Another innovative product idea from Nanomesher, this time a multipurpose case and Pi kit that can be used as a weather station, video streamer, and music player depending on which version you get. It uses the great HATs already available from Nanomesher and lets you build them all into a custom-made enclosure that offers more functionality than previous Nanomesher kits.

kck.st/2Mdm7TG

### PISHELL

The PiShell seeks to fix an issue that you may find with many Raspberry Pi cases – while they're great at protecting the Pi, you can't always access the camera port or even the GPIO pins. The PiShell comes in three parts: a base to set the Pi on; an intermediate step that lets you plug in the camera, use the DSI display port, and access the GPIO pins; and an easily removable lid that also has hole for the camera to peep through. It's a great idea, and SB Components is well known for its excellent-quality gear.

## BEST OF THE REST

Here are some other great things we saw this month

### WHERE'S AI WALLY?

Who hasn't spent minutes studying every face in a *Where's Wally?* photo looking for the elusive traveller? While it does strip the fun out of the process, Matt Reed has trained a Pi-powered robot to look for Wally using facial recognition tech. It's incredibly impressive. Now, we wonder if it could help us find Carmen Sandiego?

youtu.be/-i7HMPpxB-Y

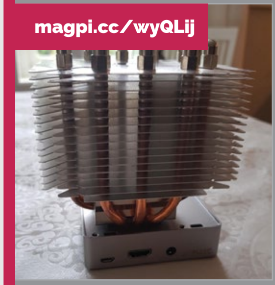
### TABLE FOOTBALL SCORING

Ever have to worry about cheating in your table football games? This ingenious Raspberry Pi system keeps track of the score digitally, so no one can sneakily move counters across during the heat of a showdown in the centre of the table.

magpi.cc/DqQLec

### PASSIVE COOLING

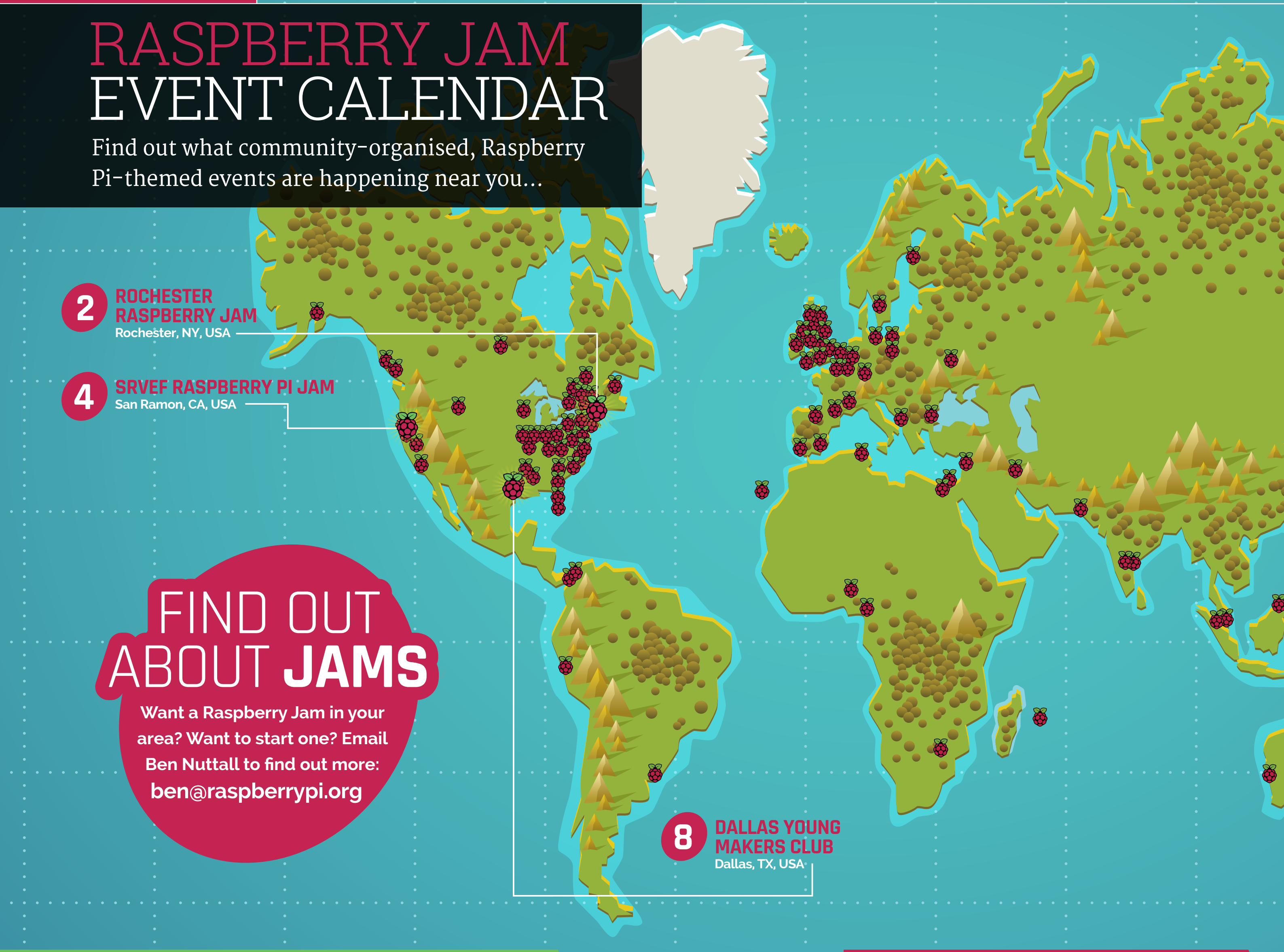
magpi.cc/wyQLij



Points for originality, we love the idea of a heatsink for the Raspberry Pi which is… bigger than the entire Pi itself. We bet with a nice breeze it would keep the BCM SoC at a lovely cool temperature however hard it was working.

# RASPBERRY JAM EVENT CALENDAR

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



## FIND OUT ABOUT JAMS

Want a Raspberry Jam in your area? Want to start one? Email Ben Nuttall to find out more:
**ben@raspberrypi.org**

## 1-4 HIGHLIGHTED EVENTS

### 1 RASPBERRY JAM CAMBRIDGE
**When:** Saturday 15 September
**Where:** Sancton Wood School, Cambridge, UK
**magpi.cc/WxgdfB**
Not the usual CamJam, but a new Raspberry Jam run by the Cambridge Technology Academy.

### 2 ROCHESTER RASPBERRY PI JAM
**When:** Saturday 22 September
**Where:** University of Rochester, Rochester, NY, USA
**magpi.cc/niySeR**
Although aimed at youngsters, anyone with an interest in programming is welcome.

### 3 FEN JAM
**When:** Sunday 23 September
**Where:** Bluntisham Village Hall, Bluntisham, UK
**fenjam.org**
The first Fen Jam, with plenty of exciting things planned for fans of Raspberry Pi.

### 4 SRVEF RASPBERRY PI JAM
**When:** Saturday 29 September
**Where:** BRIIA, San Ramon, CA, USA
**magpi.cc/TycEuL**
Run by (a different) Imagineering team and providing students with the technical expertise they need.

## 5-8 REGULAR EVENTS

### 5 CORNWALL TECH JAM
**When:** Saturday 8 September
**Where:** Bodmin Library, Bodmin, UK
**cornwalltechjam.uk**
Learn about programming on platforms including Arduino and Raspberry Pi, in a whole variety of languages.

### 6 HULL RASPBERRY PI JAM
**When:** Saturday 15 September
**Where:** University of Hull, Hull, UK
**magpi.cc/gRYMsi**
Share, learn, and tinker with digital making projects using Raspberry Pi.

WE'VE HIGHLIGHTED SOME OF THE AREAS IN NEED OF A JAM! CAN YOU HELP OUT?

## 7 NORTHERN IRELAND RASPBERRY JAM

**When:** Saturday 8 September
**Where:** School of Maths and Physics Teaching, Belfast, UK
**magpi.cc/yNUGdg**
This free event involves tinkering, coding, electronics, and generally having fun making!

## 8 DALLAS YOUNG MAKERS CLUB

**When:** Saturday 22 September
**Where:** J. Erik Jonsson Central Library, Dallas, TX, USA
**dallasyoungmakers.org**
Build a robot, make a computer, and create comics and books at the Dallas Young Makers Club.

## RASPBERRY JAM ADVICE

## PROMOTING YOUR EVENT

"We searched online for caterers in a 30-mile radius of the CamJam venue, and managed to find someone who would attend free of charge. The venue is very convenient, for us and the attendees, and we know we can build food etc. into the ticket price."

**Michael Horne**
**Cambridge Raspberry Jam**

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

RASPBERRY JAM GUIDEBOOK
STARTING AND RUNNING A RASPBERRY JAM:
ADVICE GATHERED FROM THE RASPBERRY PI COMMUNITY

# YOUR LETTERS

## INTERNATIONAL MAGPI

**I loved following your exploits in Japan for the Maker Faire Tokyo [on *The MagPi* Twitter feed] – a lot of cool projects! I hope I get to see more from it in the magazine as I might have to go next time it's on!**

**You seem to sporadically cover other events outside the UK as well – are there any plans for you to go to more places and show what people around the world are doing?**

**Guy I**

You're in luck – we have a full five pages dedicated to Maker Faire Tokyo and the Japanese Raspberry Pi community in our bumper The Month in Raspberry Pi section this issue! There were many more amazing projects we didn't even have a chance to post on Twitter.

As for other international events, we're usually a little limited to where and what we can cover, but if there's demand for it we can figure out a way to get more stuff like that in the magazine!

## NEW SUBSCRIPTIONS

**At the beginning of the year I had to cancel my subscription as I couldn't quite afford it. However, I'm now able to subscribe again! I couldn't find your three- and six-month subscriptions though; are they no longer available?**

**Also, I was on a twelve-month subscription before and got my free Pi Zero W bundle with that, and I was wondering if I would get another bundle if I re-subscribed? I'd understand if it was just one per customer, though.**

**Finally, do you get anything free with the rolling subscription?**

**Lin**

That's correct, we currently only offer the monthly rolling subscription and the year-long subscription. Any savings you'd have got with three or six months you get with the monthly subscription, and you can cancel that at any time! It also means you never miss an issue.

As for free stuff, while you don't get any special extra bundles like with the year-long sub, you still get all the free stuff that you'd get with the magazines anyway – such as any kits or posters we include in future issues.

Also, the Pi Zero W bundle is not once per lifetime! Every time you take out or renew a twelve-month sub, you'll get the bundle we offer. That may change at some point down the line, but currently there is no end date on that.

Hope this helps!

## TOYS TO LIFE

**I was wondering if it was possible to use a Raspberry Pi to 'hack' a toy so that you can then control it or make it more functional with the Pi? There are some with simple electronics that must be controllable with a Pi, and even then I do wonder if I could just use the empty space in a toy to add electronics.**

**Has anyone ever done this?**

**Louis T**

There are a few limitations, but we've seen plenty of people do something like this with a wide variety of toys and the like. Usually the limitations involve whether or not you can actually interface with parts of the device (and some people just tend to replace them, such as old screens, anyway). If you can find space for parts, however, you can do whatever you want.

Some of our favourites include turning a toy car dashboard into a fully functional Out Run arcade machine, and turning a toucan toy for very small kids into a Google voice assistant.

The Tomy Turnin' Turbo Dashboard arcade mod is one of our favourite recent Pi projects

## FROM THE FORUM:

## OFFLINE VOICE

The Raspberry Pi Forum is a hotbed of conversations and problem-solving for the community – join in via **raspberrypi.org/forums**

**I'm stuck trying to add the repository for the Zamia speech stuff from the new *MagPi* [Ed's note: issue 72] and can't get it to work.**

**I've typed the commands in and when I press ENTER it says OK, but when I do an update and try to install the software it says 'not found'.**

**Please help!**

**Lee**

You need to make sure you start the entire sequence with **`sudo -i`** before trying to add the repository and then install anything. Make sure to use this exact series of commands:

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

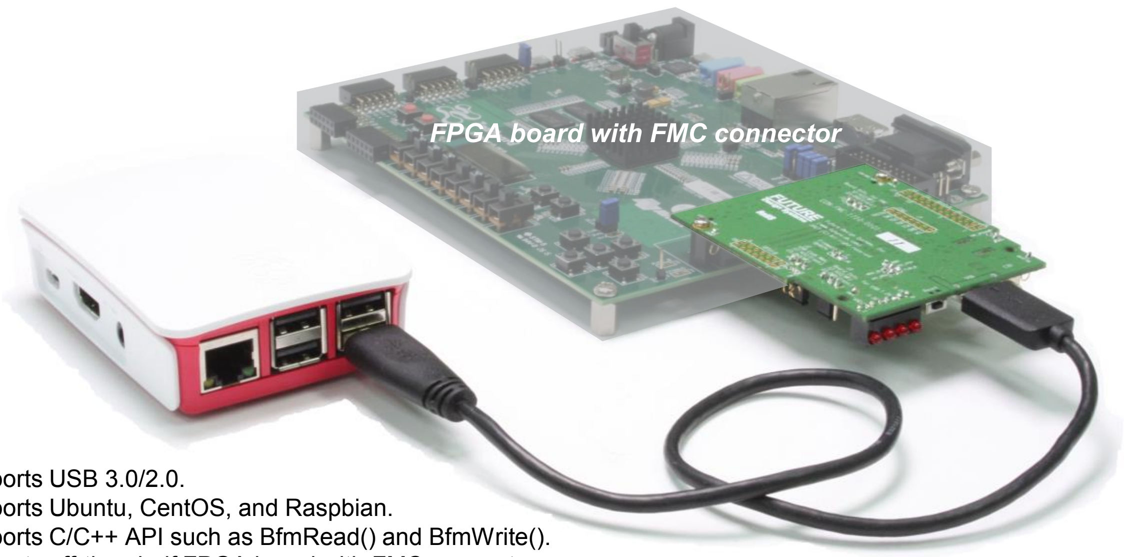
If you're still having some problems, double-check your internet connection or use a fresh install of Raspbian.

### WRITE TO US

**Have you got something you'd like to say?**

Get in touch via **magpi@raspberrypi.org** or on The MagPi section of the forum at: **raspberrypi.org/forums**

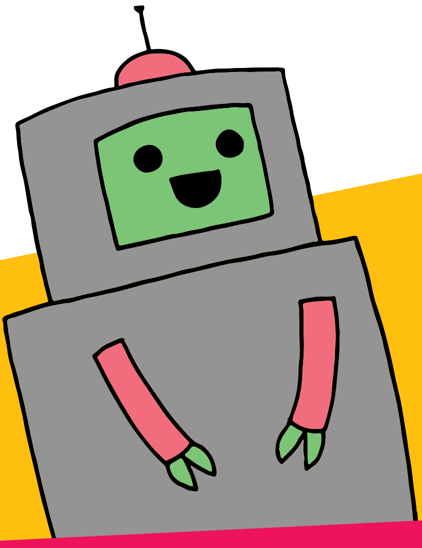
## Connecting *Raspberry Pi* to *FPGA* through *USB*

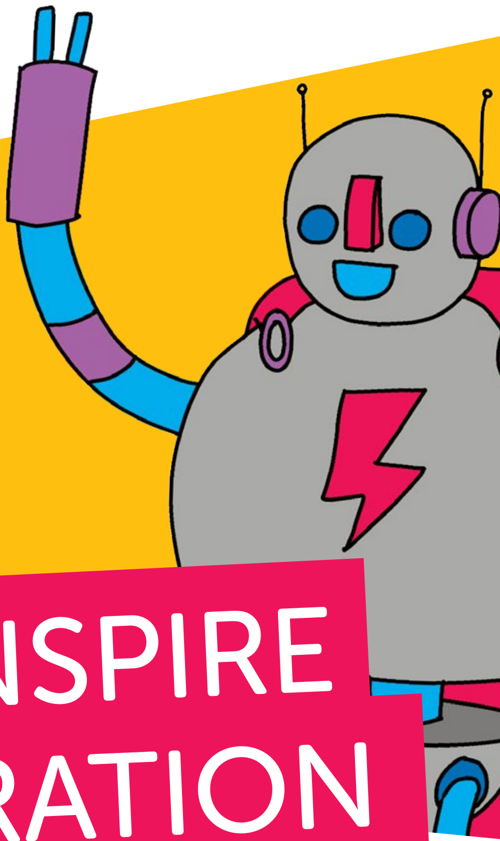


- CON-FMC™ supports USB 3.0/2.0.
- CON-FMC™ supports Ubuntu, CentOS, and Raspbian.
- CON-FMC™ supports C/C++ API such as BfmRead() and BfmWrite().
- CON-FMC™ supports off-the-shelf FPGA board with FMC connector.
- CON-FMC™ supports FIFO interface and AMBA AXI/AHB/APB buses.
- CON-FMC™ supports CPU-to-FPGA offloading.

For additional information contact at **contact@future-ds.com** or visit **www.future-ds.com/CON-FMC**

FUTURE Design Systems



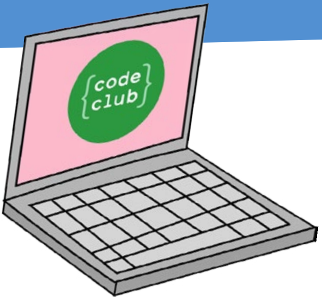


# CAN YOU HELP INSPIRE THE NEXT GENERATION OF CODERS?

Code Club is a network of volunteers and educators who run free coding clubs for young people aged 9-13.

We're always looking for people with coding skills to volunteer to run a club at their local school, library, or community centre.

You can team up with friends or colleagues, you will be supported by someone from the venue, and we provide all the materials you'll need to help children get excited about digital making.



To find out more, join us at

**www.codeclubworld.org**

# WIN! ONE OF FIVE PI-HOLE NETWORK SECURITY BOXES!

In association with Pi Supply

We reviewed the Pi-hole in issue 71 of *The MagPi* and gave it a four-star rating due to its ability to handle even the heaviest ad-laden websites, so it can keep your network very secure from bad ads.

"Pi-hole is a device that allows you to block all of those annoying internet advertisements on any device and even in apps"
**Pi Supply**

**Pi Supply** has five of the **Pi-hole** boxes to give away, **so enter today for your chance to win one!**

**Enter now at magpi.cc/win**

**Learn more: magpi.cc/ZeOTqW**

## Terms & Conditions

Competition opens on **29 August 2018** and closes on **26 September 2018**. Prize is offered to participants worldwide aged 13 or over, except employees of the Raspberry Pi Foundation, the prize supplier, their families or friends. Winners will be notified by email no more than 30 days after the competition closes. By entering the competition, the winner consents to any publicity generated from the competition, in print and online. Participants agree to receive occasional newsletters from The MagPi magazine. We don't like spam: participants' details will remain strictly confidential and won't be shared with third parties. Prizes are non-negotiable and no cash alternative will be offered. Winners will be contacted by email to arrange delivery. Any winners who have not responded 60 days after the initial email is sent will have their prize revoked. This promotion is in no way sponsored, endorsed or administered by, or associated with, Instagram or Facebook.

**SIMON LONG**

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

# THE LAST 10 PERCENT

**Simon Long** talks about his experience of revamping the Raspberry Pi Desktop

It was almost exactly four years ago when I was offered the chance to work at Raspberry Pi. I knew all the team very well, but I'd had hardly any involvement with the Pi itself, and wasn't all that sure what they would want me to do; at that time, I was working as the manager of a software team, with no experience of hardware design.

Fortunately, this was when software had started to move up the list of priorities at Raspberry Pi. Eben and I sat down on my first day and played with the vanilla LXDE desktop environment in Raspbian for 15 minutes or so, and he then asked me the fateful question: "So – do you think you can make it better?"

With rather more confidence than I felt, I replied "of course!" I then spent the next week wondering just how long it was going to take before I was found out to be an impostor and shown the door.

## UI experience

To be fair, user interface design was something of which I had a lot of experience – I spent the first ten years of my career designing and implementing the user interfaces for a wide range of products, from mobile phones to medical equipment, so I knew what a good user interface was like. I could even see what changes needed to be made to transform the LXDE environment into one. But I didn't have a clue how to do it – I'd barely used Linux, never mind programmed for it...

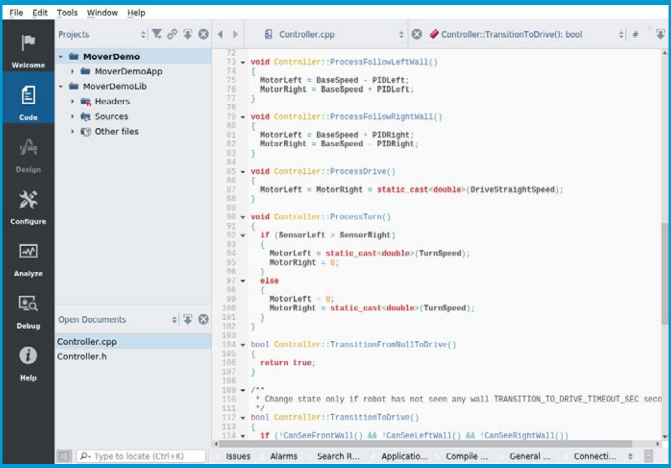
As I said above, that was four years ago, and I've been hacking the Pi desktop about from that day on. Not all the changes I've made have been popular with everyone, but I think most people who use the desktop feel it has improved over that time. My one overriding aim has been to try to make the Pi desktop into a product that I actually want to use myself; one that takes the good user interface design principles which we are used to in environments like macOS and Windows – ideas like consistency, attractive fonts and icons, intuitive operation, everything behaving the way you expect without having to read the instructions – and sculpting the interface around them.

> "So – do you think you can make it better?"

## Final polish

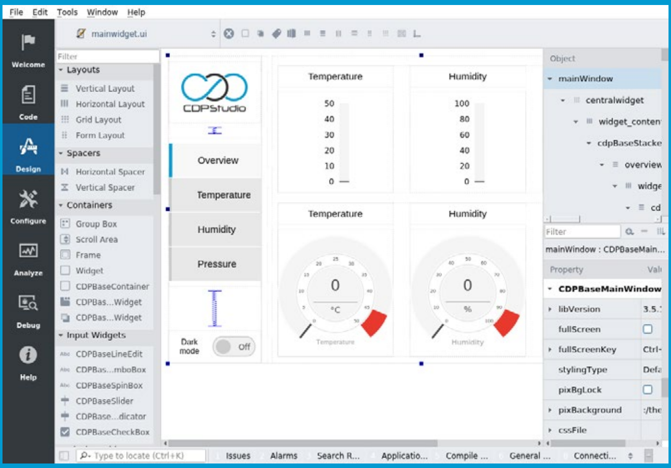
In my experience, the main difference between the Linux desktop environment and those of its commercial competitors is the last 10%; the polishing you do once everything works. It's not easy making something that works, and a lot of people, once they have created something and got it working, leave it and move onto creating something else. I'm really not great at creating things from scratch – and have nothing but admiration for those who are – but what I do enjoy doing is adding that last 10%; going from something that works, to something that works well and is a pleasure to use.

Being at Raspberry Pi means I get to do that every day when I come to work. Every time I see a photo of a Pi running at a Jam, or in a classroom, anywhere in the world, and it's using my desktop – the thrill from that never goes away.

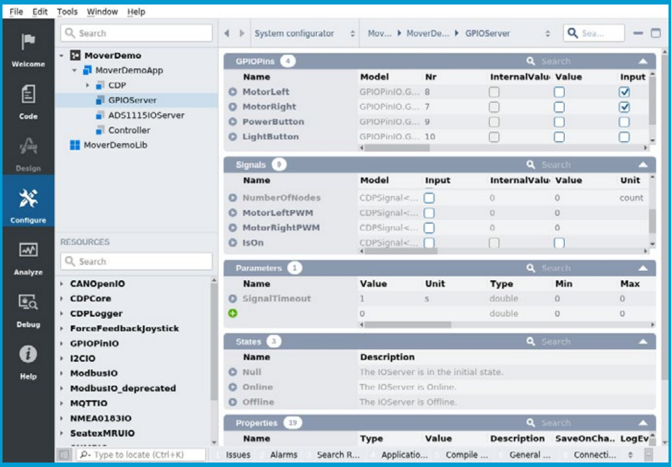


```
void Controller::ProcessFollowLeftWall()
{
  MotorLeft = BaseSpeed - PIDLeft;
  MotorRight = BaseSpeed + PIDLeft;
}

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

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

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

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

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

Code



Design



GPIOPins 4

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

Signals 9

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

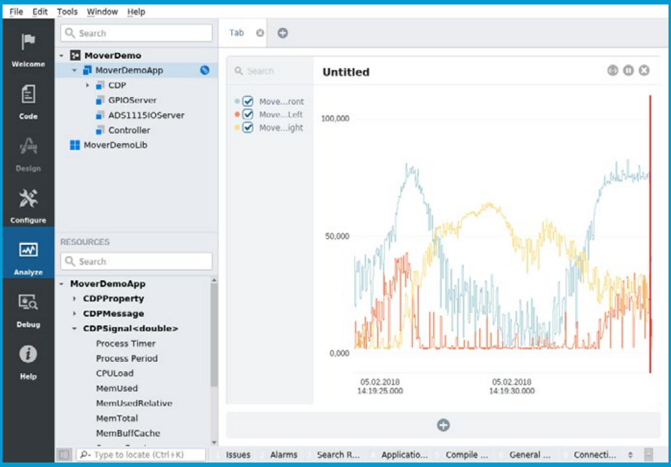
Parameters 1

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

States 3

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

Configure



Analyze

# Now free for home projects

## A professional control system development tool

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

Free download on www.cdpstudio.com

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

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

CDPStudio