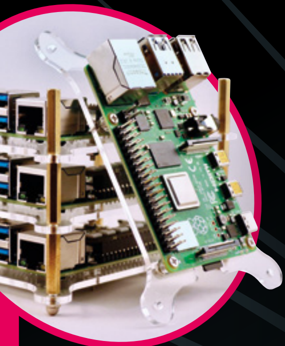
BUY IN PRINT **WORLDWIDE** MAGPI.CC/STORE

The MagPi

Issue 87 | November 2019 | magpi.cc

The official Raspberry Pi magazine



MAKE A CLUSTER COMPUTER

Learn to code a supercomputer

RASPBERRY PI 4K DIGITAL MEDIA HUB

High definition video streaming & playback made easy

TOP 10 AI PROJECTS

Build smarter kit with machine learning

HACK YOUR KITCHEN TOOLS

Precision cooking with Raspberry Pi

- BUILD A ROBOT WITH SENSORS
- **DESIGN RETRO GAME LEVELS**
- HACK A MARBLE RUN GAME

£5.99

Raspberry Pi PRESS

GLOBAL DELIVERY magpi.cc/store

40 PAGES OF PROJECTS & TUTORIALS

Over 10,000 Different Boards and Modules In Stock
NO ONE
KNOWS
BOARDS LIKE
DIGI-KEY!
Open Source Community Boards
Evaluation and Development Boards
Wireless Modules
Single Board Computers
Reference Designs
Programmers, Emulators and Debuggers
DESIGNS START HERE!
DIGIKEY.COM/BOARDS
Digi-Key® ELECTRONICS
MAKER.IO
POWERED BY DIGI-KEY
#MAKEWITHDIGIKEY
Digi-Key is a franchised distributor for all supplier partners. New products added daily. Digi-Key and Digi-Key Electronics are registered trademarks of Digi-Key Electronics in the U.S. and other countries. © 2019 Digi-Key Electronics, 701 Brooks Ave. South, Thief River Falls, MN 56701, USA
ECIA MEMBER
Supporting The Authorized Channel

WELCOME
to *The MagPi* 87

Thanks to Raspberry Pi, the home computer is once again a versatile thing. You can turn Raspberry Pi into a desktop computer, learn to code, play with electronics, and make handy household devices such as voice assistants and kitchen gadgets.

And you can build a media player (page 31). It's obvious when you think about it: Raspberry Pi is small, quiet and powerful, it connects to a screen using HDMI (perfect for televisions), and is packed with wireless LAN and Bluetooth connectivity – ideal for streaming media and channel surfing from a wireless controller.

In recent issues, we've looked a lot at Raspberry Pi 4. With its super-fast CPU and an amazing amount of RAM, it's the best computer we've ever used. But one small feature also makes it the best media playback device, and that's 4K. Hook a Raspberry Pi 4 up to a 4K monitor and you can immediately start streaming and playing your high-quality video. And that's on top of photo, music, and audiobook playback.

I love the media player project because it's so practical. Many projects are designed for learning (nothing wrong with that, mind). And media players tend to be built, and then used for the long run.

Our media player is a perfect example of something valuable you can build, with your own two hands, and make good use of. I hope you enjoy this project.

Lucy Hattersley Editor

EDITOR

Lucy Hattersley

Lucy is Editor of *The MagPi*. She once spent two weeks tuning school televisions for the council. This month she stressed out Raspberry Pi 4 computers.

magpi.cc

GET A
RASPBERRY ZERO W KIT
WITH A SUBSCRIPTION!
PAGE 38

Contents

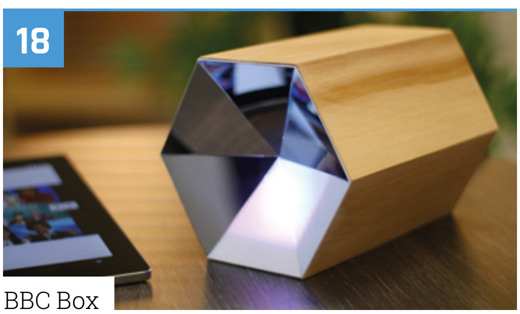
> Issue 87 > November 2019

Cover Feature

Regulars

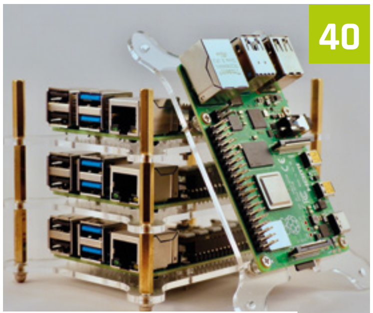
Project Showcases

Social Media without the Internet



BBC Box

40



Make a cluster computer

58

Create maps & levels in PICO-8

76

FLIRC Raspberry Pi 4

84

Alex Mous interview

Tutorials

The Big Feature

68

Power up your kitchen

Reviews

Community

WIN ONE OF FIVE TOUCHSCREEN CASES! 95

Code Club India collaboration success

Empowering students to improve their own futures is key, reports **Rosie Hattersley**

Over the last few months, Pratham Education Foundation and Code Club have successfully piloted a programme across 40 villages in rural areas of two Indian states, supporting children and young volunteers there to get hands-on with coding.

Pratham (**pratham.org.in**) is one of India's largest NGOs (non-governmental organisations). It was established in 1995 with the aim of providing educational opportunities for young people living in the slums of Mumbai.

To lay the groundwork for their collaboration with Code Club, Pratham first held a series of village meetings at which 16- to 25-year-olds could sign up to become Code Club volunteers. They attended a training session to build their confidence and learn how to set up a Raspberry Pi computer, use Code Club Scratch projects, and share their coding skills with young people attending their Code Clubs.

The kits needed for these Code Clubs each contained a Raspberry Pi computer, keyboard, monitor, and a mouse and were provided by Pratham.

Code Club members enjoyed learning as a group

Success story

The initiative was a remarkable success: 1109 Code Club members took part and 50 young adults

▲ Young Code Club members explore computer coding for themselves

> The aim was to introduce youngsters to coding and digital technology, while adults learned how to become Code Club leaders

trained as volunteers. The Pratham Code Club project has now funded 244 Code Clubs across 40 villages in Uttar Pradesh and Maharashtra.

The aim was to introduce youngsters to coding and digital technology, while adults learned how to become Code Club leaders.

One of the youth volunteers summed up the Code Club's importance: "It is only because of these sessions that I was introduced to this world of computers and I know what coding means."

To partner with Raspberry Pi in India, email **india@raspberrypi.org**. To help Code Club grow in other countries, email **hello@codeclubworld.org**.

▶ A youth volunteer demonstrates her newly acquired coding skills to friends

Sfera introduces Strato Pi CM Duo

Innovative Compute Module carrier can hold two microSD cards

Sfera Labs has released Strato Pi CM Duo, a Raspberry Pi Compute Module carrier module with two microSD card slots.

The Strato Pi CM Duo features a high-speed switch matrix that enables a Compute Module 3+ Lite to boot from either microSD card. With this architecture, Sfera claims Raspberry Pi can perform a full-image upgrade on one microSD card while the system runs from the other card.

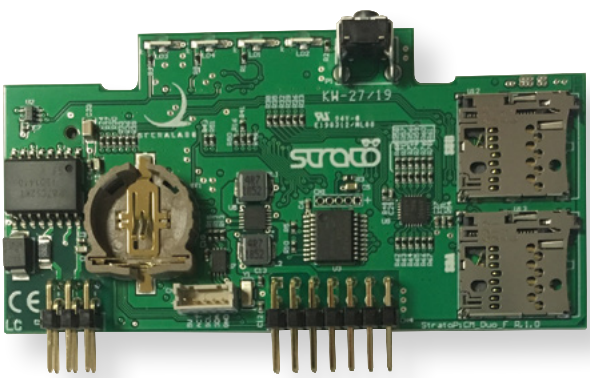
> A robust hardware platform for critical edge computing applications

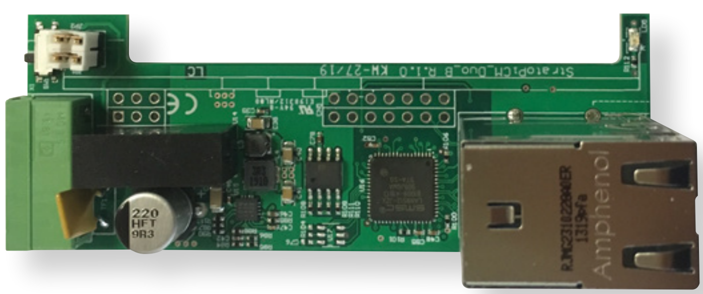
It's "specifically designed to create a robust hardware platform for critical edge computing applications," Sfera tells us.

"Edge computing solutions are often installed in remote locations, where physical access to the devices is limited and expensive, and at the same time are expected to have a very long service life."

▲ Strato Pi CM Duo, a Raspberry Pi Compute Module carrier designed for industrial environments

▼ The CM Duo has two microSD card slots, creating a platform that can be updated while still running





ALBATROS

Can you do long-term radio astronomy with a Raspberry Pi? Yes, and one maker is mapping out the universe and might find out how old it is. **Rob Zwetsloot** investigates

MAKER

Taj Dyson

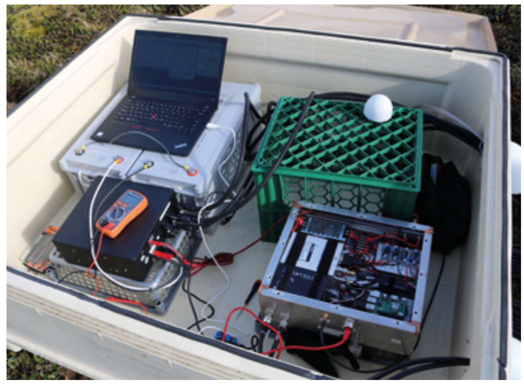
An honours physics undergrad at McGill University, high-tech maker, and programmer who also makes games.

mcgill.ca/mars

ALBATROS is one of our favourite daft things in science: a backronym. It was originally created to mean 'Array of Long Baseline Antennas for Taking Radio Observations from the Subantarctic', as it started on the island of Marion which has a lot of albatrosses.

"We want to make a map of the sky at low frequencies, to lay the groundwork for future observations of the cosmic 'dark ages' – the time before stars formed," Taj Dyson tells us. He's a physics student at McGill University in Canada, and recently took a trip to the McGill Arctic Research Station (MARS) to take radio astronomy measurements using equipment made in labs at McGill.

"What does frequency have to do with time?" Taj continues. "To understand, realise that the universe was mostly neutral hydrogen. This hydrogen naturally emits light, or 'glows', at a wavelength of 21 cm (or about 1400MHz in frequency). We know this frequency very precisely, and it's emitted at the same frequency for all time. Due to the expansion of the universe, though, light from hydrogen that is further away is redshifted – that is, its frequency is reduced (in the Earth's frame of reference). So, we can see light from hydrogen that is further away by 'tuning' our antenna lower in frequency."

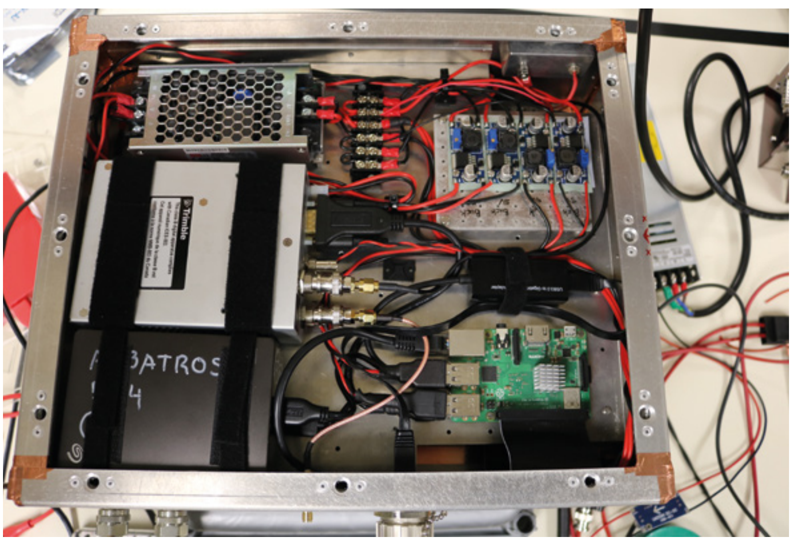


A full test kit to see how the interference was at specific points in the local area

Looking for hydrogen in this way allows us to look into the past of the universe. While this is a technique pioneered in the sixties, human-generated radio waves have created interference that make it harder to do unless you're in remote areas. There are other factors as well, such as solar activity, that make the polar regions very attractive.

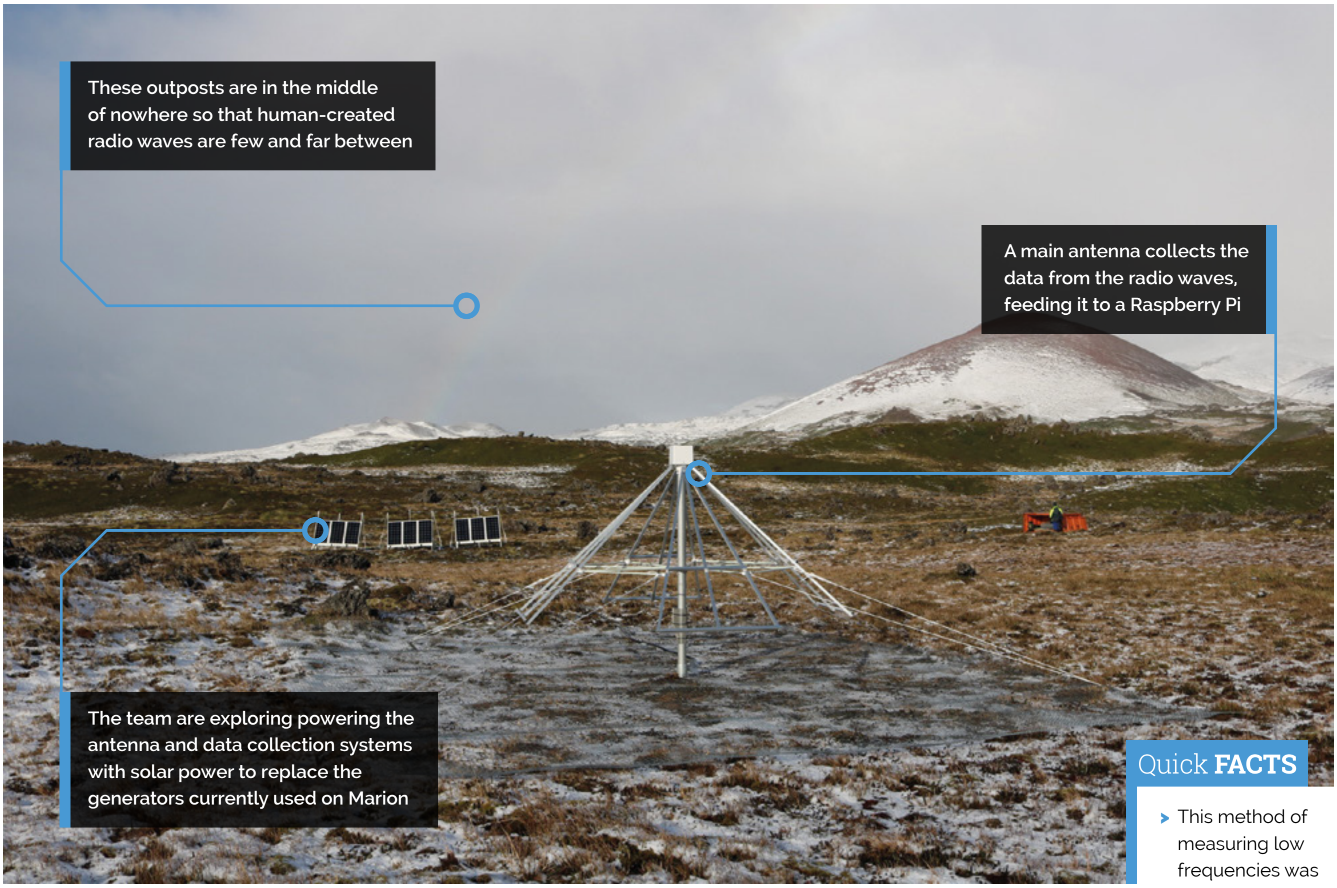
"Getting several maps of the sky at several low frequencies would be the first step towards understanding a whole era of the universe that hasn't been studied very extensively," Taj explains, "laying the groundwork for future measurements that could provide insight into cosmological mysteries like dark matter or dark energy."

One of the systems that collects data form the antenna – a Raspberry Pi gets the data which is then recorded on a hard drive



History in the stars

At Marion Island in the subantarctic Indian Ocean, several radio antennas are currently in use taking measurements of the sky. Taj and his crew are looking at expanding the operation to MARS on the other side of the planet to create another



▼ The team spend summers at MARS (McGill Arctic Research Station) preparing the site and retrieving data

Quick FACTS

- This method of measuring low frequencies was pioneered by Grote Reber
- One person stays at the Marion site each year to maintain the generators
- This kind of radio astronomy is nice and low-budget
- The McGill team were joined by a member of EDGES from MIT
- Taj has shown off other projects at Maker Faires in America

Big crates work well as Faraday cages if you need one

ALBATROS – 'Array of Long Baseline Antennas for Taking Radio Observations from the Seventy-ninth parallel.'

"MARS is new to this project," Taj says. "The first time we visited was last summer, and that was just to have a look at how much RFI [radio frequency interference] there is up there. A first look at the data suggests that MARS is very radio-quiet and will be an excellent place for future observations!"

> We want to make a map of the sky at low frequencies, to lay the groundwork for future observations of the cosmic 'dark ages'

Next summer they will return to set up antennas, the data from which is processed through a Raspberry Pi.

"Raspberry Pi has many desirable features for our application," Taj mentions. "It can communicate directly with our field-programmable gate array (FPGA)… the very expensive circuit board that turns analogue antenna signal into zeroes and ones according to rules we tell Raspberry Pi to send it… Configuring also prints out various bits of information useful for troubleshooting.

"Next, we run the actual data acquisition script (there are actually two types) on Raspberry Pi, which receives digitised signal from the FPGA through the Ethernet port. Raspberry Pi then writes that data to disk, either on the SD card for our small data volume mode, or on external SSDs for our huge ~10MB per second mode.

"Of course, our Raspberry Pi also saves logs from both of these programs to the SD, so they can be looked at later when something doesn't work or gets forgotten. Raspberry Pi consumes a relatively small amount of power, which is nice considering we want to make autonomously powered stations."

Cosmic patience

An experiment like this takes time, though – you can't just turn on the antennas and get an instant readout of the universe. It may take years.

"The short answer is I have no idea [how long it will take]." Taj admits. "We are funded to go to MARS for two more years, but we hope this is just

the beginning of a much longer observing program. It's going to take a lot of R&D, antennas, and time to reach our ultimate goal of mapping the universe during cosmic dawn and the dark ages."

The results will be worth the wait, however. "First [let's] talk about cosmic dawn," Taj continues. "When the first stars formed, their heat excited the hydrogen around them, causing it to absorb the cosmic microwave background (CMB) at a very specific wavelength. Since this event happened so long ago, the wavelength is now very long (again, due to redshift) and our low-frequency equipment can pick it up. So, we expect to see a slight dip in signal at a certain frequency (since we don't know exactly how long ago this happened, we don't know at exactly what frequency). [Other results] saw a dip that was about twice as deep as predicted, which means hydrogen absorbed more radiation than expected. This could have all sorts of cosmological implications. I'm not going to bet on whether [this] result was real or not.

"[Secondly] the mapping of the dark ages is not going to test any theory; it'll provide a baseline for future low-frequency measurements."

We look forward to seeing results in the not-so-far future to take a look into the very-distant past.

▲ Generators provide power at the moment, but the hope is to offload some of the power requirements on solar

Testing the site

01 "At MARS, we had a main antenna set up at the base camp, and that was out main data-gathering tool. However, we also took smaller, more portable antennas with us on a helicopter."

02 "[We took] a laptop and our data-gathering electronics – Raspberry Pi included – to take measurements in several different locations where we may put larger antennas in the future."

03 "We thought maybe local topography would shield us a little from RFI (it's strange: one typically thinks of an observatory on a hill, but really, for radio astronomy, we want to be in as steep a valley as possible!), but we didn't see a difference by eye, looking at the spectra, between valleys and crests of hills."

Cryptography ICE Cube experiment

With hackers posing a potential threat to the security of satellites and space missions, Raspberry Pi Zero is flying into action, as **David Crookes** explains

MAKER

Emmanuel Lesser

Emmanuel is a software product assurance engineer at the European Space Agency, where he provides software product assurance for the Copernicus Earth observation programme and the Biomass mission.

esa.int/ESA

In space, nobody can hear you scream, but there are some people who are determined to eavesdrop on communication being sent back and forth between Earth and orbiting satellites. As such, there is a danger that security and safety can become compromised, which is why a team of researchers at the European Space Agency (ESA) have sought to enhance cybersecurity for future space missions.

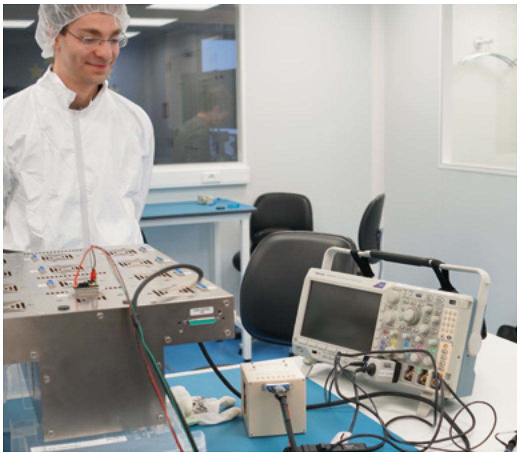
"There is a risk of satellites being intercepted and hacked, which leaves them susceptible to being controlled by a rogue third party," explains ESA software product assurance engineer, Emmanuel Lesser. "That presents a big risk for a mission and it's also a problem commercially since satellites are very expensive and the data that is transmitted is sensitive. It's important that we protect it."

The right shape

One of the cybersecurity solutions being explored by ESA involves Raspberry Pi Zero, and it is being worked on by a team headed by Emmanuel. Called the Cryptography ICE Cube experiment, or CryptIC for short, the aim is to make encryption-based secure communication feasible for even the smallest of space missions, and it is currently operational in the Columbus space laboratory module of the ISS.

> The aim is to make encryption-based secure communication feasible for even the smallest of space missions

"We wanted the experiment to have a small footprint and a relatively modest energy consumption," Emmanuel says. "We also wanted to achieve secure communication using the cheapest components. We knew that smaller missions, such as those which use CubeSats [miniaturised satellites for space research], utilise signals that are unencrypted. That leaves them vulnerable, so we looked at the possibilities."

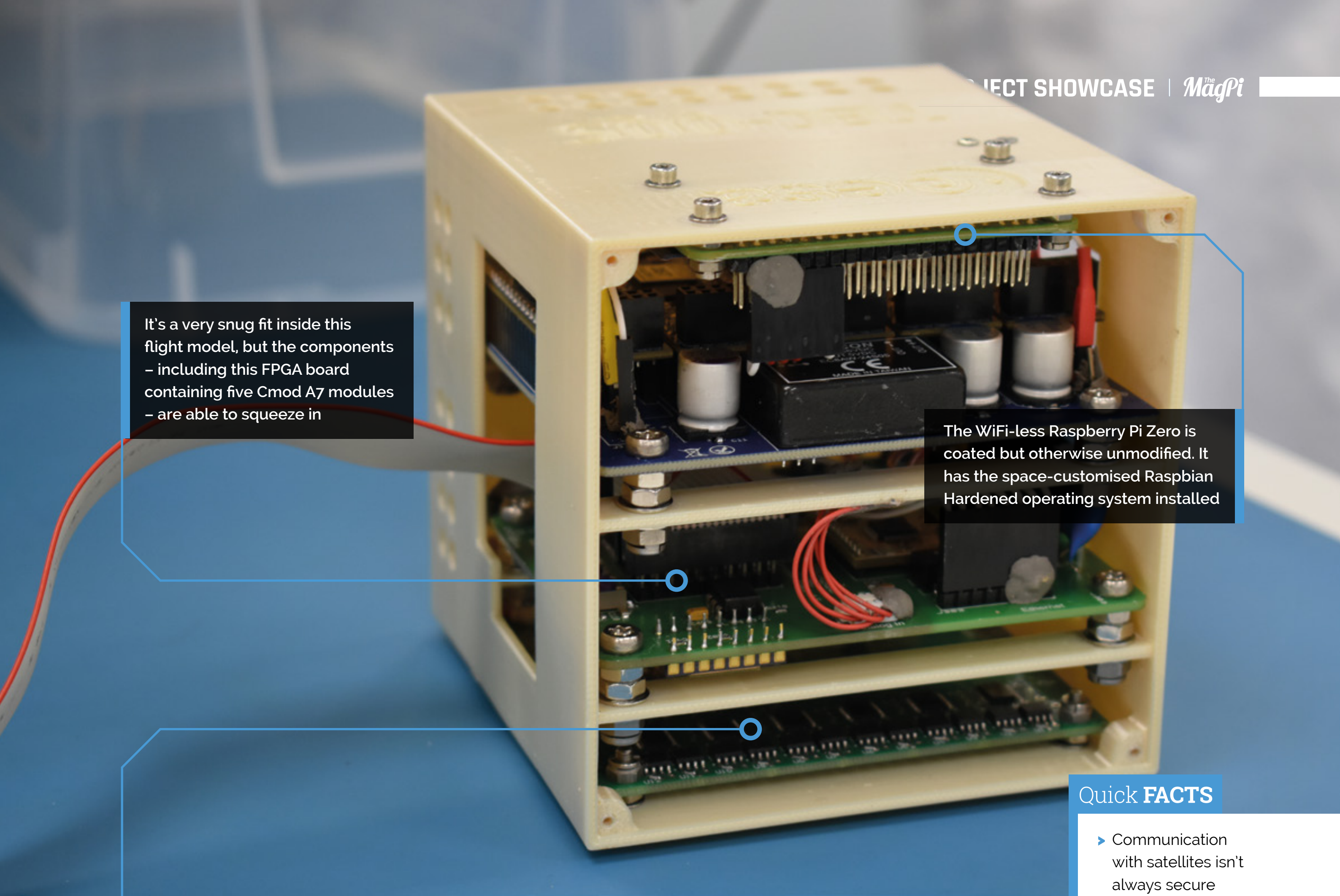


▲ Lukas Armborst, the main engineer working on the project besides Emmanuel Lesser, oversees the experiment
Image credit: ESA

Since the ICE Cubes facility in Columbus offers plug-and-play installation for cube-sized experiments, the idea was to fit CryptIC into a beige box measuring 10 cm on all sides. "We were able to make use of a Raspberry Pi Zero just as it comes out-of-the-box," Emmanuel says. "On that is a space-hardened version of Raspbian which has been previously commissioned by ESA. It removes the unnecessary parts of the system so it has fewer libraries, some of which we had to reinstall. But our main task after that was to write the software in Python using some of the existing libraries as well as our own."

Cryptic clues

The project's Raspberry Pi Zero also needed to be covered with a plastic 'conformal' coating. "This is an ISS requirement and it merely prevents fire hazards – you wouldn't know it's there unless



It's a very snug fit inside this flight model, but the components – including this FPGA board containing five Cmod A7 modules – are able to squeeze in

The WiFi-less Raspberry Pi Zero is coated but otherwise unmodified. It has the space-customised Raspbian Hardened operating system installed

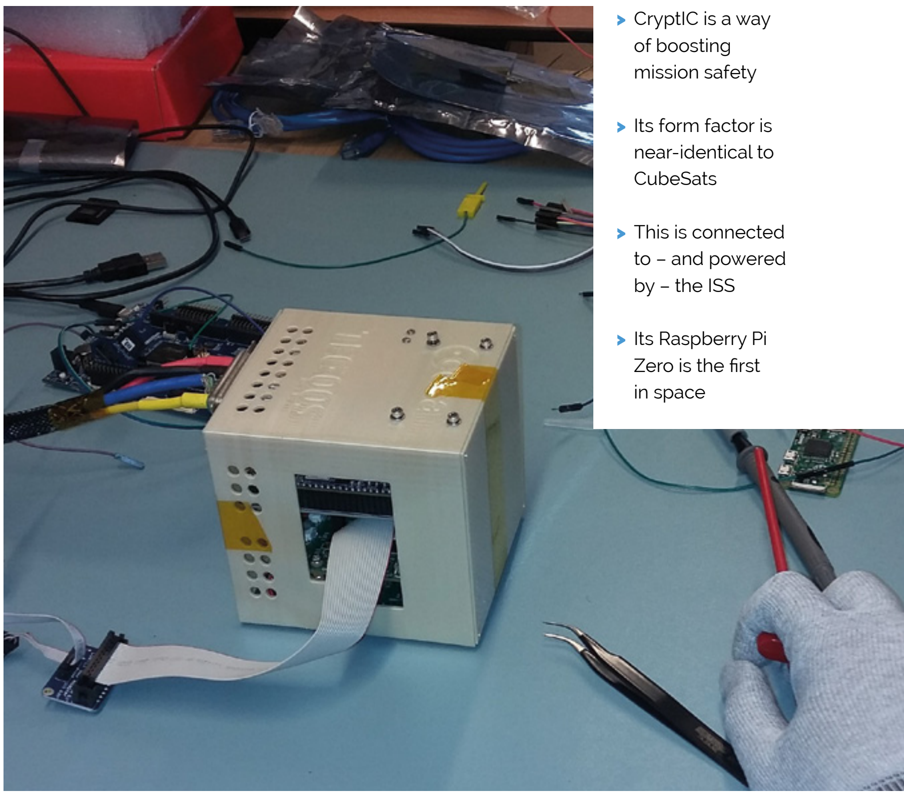
The module also contains computer flash memories which are being evaluated for their orbital performance, as well as a compact 'floating gate' dosimeter developed in cooperation with CERN to measure radiation levels

Quick FACTS

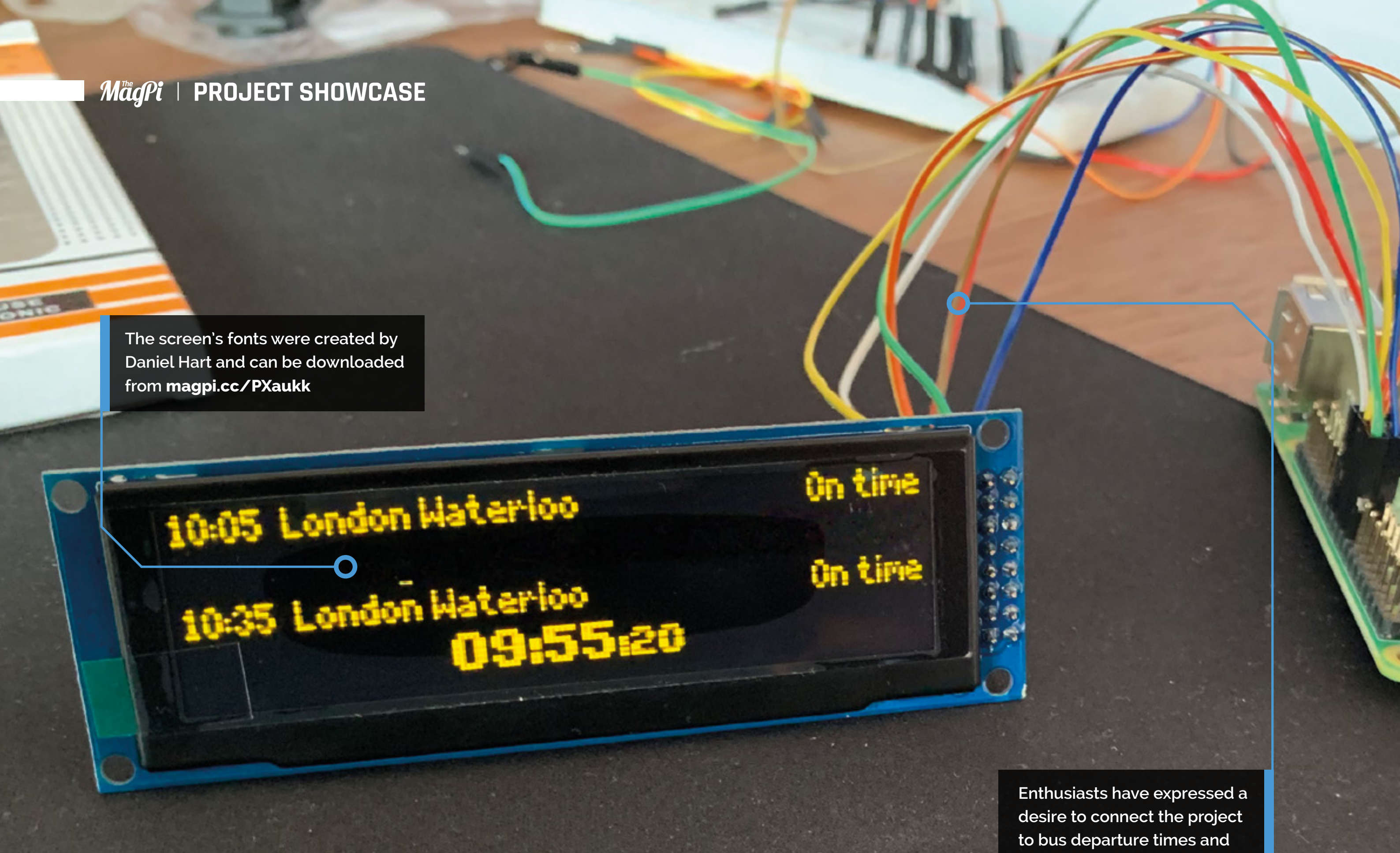
- Communication with satellites isn't always secure
- CryptIC is a way of boosting mission safety
- Its form factor is near-identical to CubeSats
- This is connected to – and powered by – the ISS
- Its Raspberry Pi Zero is the first in space

you looked really hard," Emmanuel says. The computer is controlled from a laptop based on Earth at ESA's ESTEC technical centre in the Netherlands, with data sent back and forth in near real time via the ICE Cubes operator Space Applications Services in Brussels. "We're not sending any real sensitive data – just strings of 'hello world', articles, and images," Emmanuel continues. It is testing the feasibility of using a backup key that can't be compromised from the ground, while studying whether microprocessor cores that are based around field-programmable gate arrays are able to offer redundancy if a core is affected by radiation."

The experiments began to run continuously from September and Emmanuel says the team is looking to stick with a Raspberry Pi Zero in the future too. "Ideally, we'd like to have more RAM and a version without WiFi – we had to buy an older one on eBay because the ISS doesn't allow WiFi without a special procedure – but it's near-perfect for what we want," he concludes.



▲ The beige box is the ICE Cube, inside which fit the components – including a Raspberry Pi Zero which, because it runs Linux, must (under ISS rules) also run an antivirus app
Image credit: ESA



The screen's fonts were created by Daniel Hart and can be downloaded from **magpi.cc/PXaukk**

Enthusiasts have expressed a desire to connect the project to bus departure times and non-UK railway networks

UK train departure screen

When Chris Hutchinson had an idea to create a real-time train departure board, he got to work without delay, as **David Crookes** discovers

MAKER

Chris Hutchinson

Chris has had an interest in trains since he was a child and he is the principal engineer at The Times and The Sunday Times newspapers.

magpi.cc/duTXHX

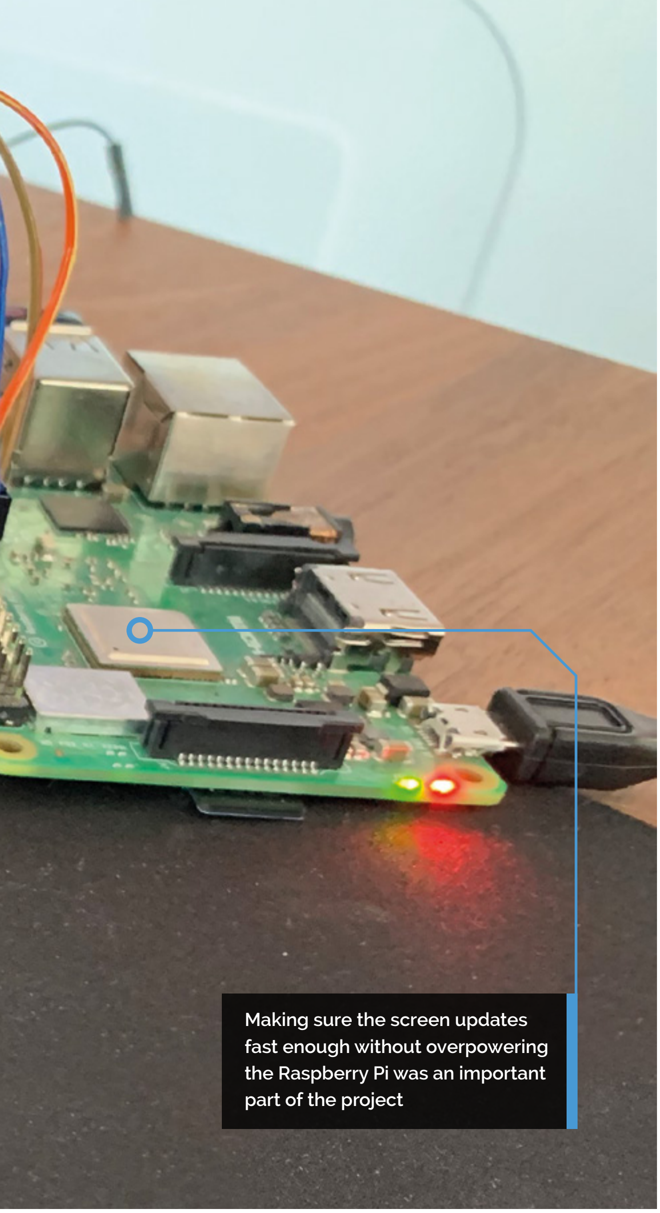
Given the time of year, we can perhaps expect some leaves on the line, slippery rain or maybe, if the weather takes a real turn for the worse, the wrong type of snow. But in any case, Chris Hutchinson will be as prepared as any commuter can be thanks to his Raspberry Pi computer.

The railway enthusiast has developed a miniature real-time train departure board that resembles the ones adorning stations across the world. "I wanted to be aware of any delays or cancellations on the trains before heading out of the house," he explains. "I knew I could use an app on my phone, but where's the fun in that?"

At first, Chris took videos of the dot matrix displays at the stations he visited, making a note of the quirks in their displays and paying attention to spacing and the wording they used. "I must have taken videos of ten or fifteen different departure board variations from across the UK," he tells us. "Based on those, I worked out the minimum dataset I'd need to display the next two services from my local station, and started looking into APIs that provided the data."

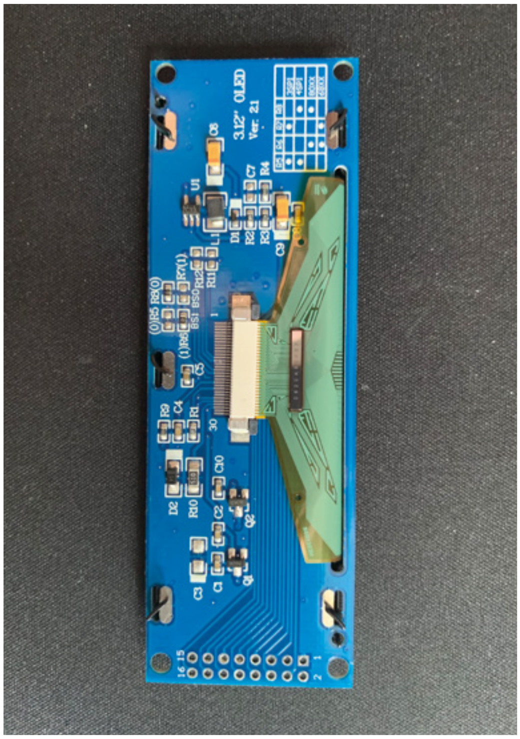
Tunnel vision

Chris decided to use TransportAPI (**transportapi.com**) and he soon got down to coding using Python 3. He's already found the perfect display for the project – a 256×64 SSD1322 OLED screen which he liked because it was affordable, low power, and comes in different colours. "The yellow one looked like a perfect match to the videos I had been taking," he says. "So I ordered one, excitedly waited for the post every morning and once it arrived, I wired it up, got my code running, and couldn't believe how brilliant it looked."



Making sure the screen updates fast enough without overpowering the Raspberry Pi was an important part of the project

Other makers have produced their own versions, including this one by Chris Crocker-White which incorporates a Raspberry Pi Zero W



The SSD1322 OLED screens allow for different-coloured displays, and are easily connected to a Raspberry Pi computer

In the meantime, Chris had found an open-source Python library for displaying graphics on OLED screens – one he says is typically used for small animations or displaying debugging data. It also had a software simulator using Pygame under the hood. "It allowed me to test my code before the real screen arrived," Chris recalls. He'd also stumbled across an open-source set of fonts that replicated real dot matrix departure boards and these only needed some tweaks to their size.

> I knew I could use an app on my phone, but where's the fun in that?

"Most of the heavy lifting is managed in the code and there are two key parts to it," Chris reveals. "The first is data loading and parsing, making the appropriate network requests for my station and transforming it into a useful data structure. Second is the rendering code where I take the data and turn it into pixels on the screen."

Now it's up and running, Chris keeps an eye on the display ahead of his daily London commute: "The board has prevented me from getting stuck out in the rain on more than one occasion."

On the rails

It also seems to have struck a chord with others. "After I first got it up and running, I shared a video of the build on Twitter and the response was phenomenal. It turns out the crossover between Raspberry Pi and train enthusiasts is pretty significant and there are already a number of forks. It's been a real honour to see so many people engage with the product and take it in their own directions."

Quick FACTS

- The screen can be bought for about £25
- Python 3.6+ is needed to run the code
- TransportAPI has a free tier for makers
- An API call is made every 1–2 minutes
- Chris's boss wants one for The Times office

BBC Box

Thanks to BBC Box, you might be able to enjoy personalised services without giving up all your data. **Sean McManus** reports

MAKER

Max Leonard

Project Tech Lead. Research Engineer at BBC R&D. Responsible for the technical aspects of the project.

MAKER

Jasmine Cox

Project User Experience (UX) Lead. Development Producer with BBC R&D. Responsible for producing the experiential and interaction aspects of prototype.

magpi.cc/gpkepj

One day, you could watch TV shows that are tailored to your interests, thanks to BBC Box. It pulls together personal data from different sources in a household device, and gives you control over which apps may access it.

"If we were to create a device like BBC Box and put it out there, it would allow us to create personalised services without holding personal data," says Max Leonard.

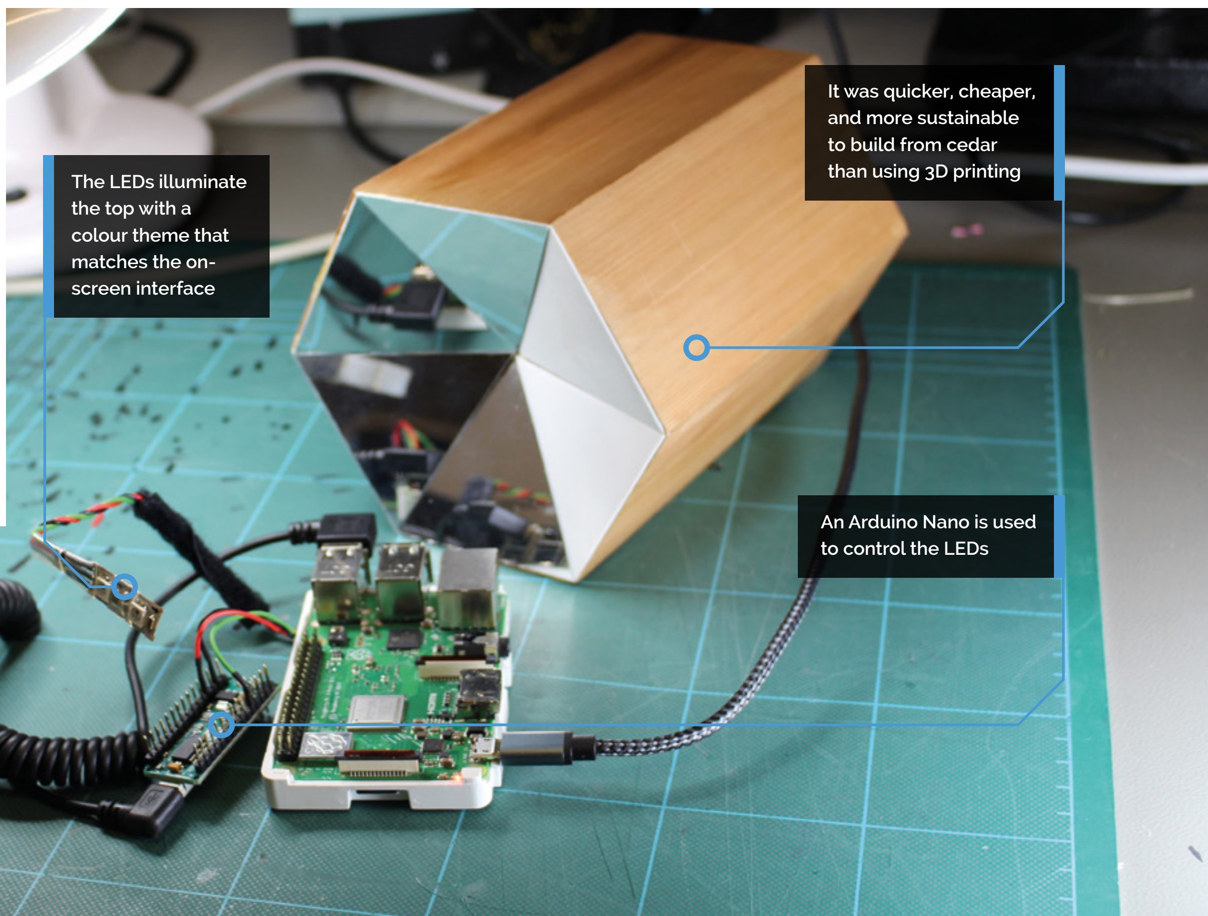
TV shows could be edited on the device to match the user's interests, without those interests being disclosed to the BBC. One user might see more tech news and less sport news, for example.

BBC Box was partly inspired by a change in the law that gives us all the right to reuse data that companies hold on us. "You can pull out data dumps, but it's difficult to do anything with them unless you're a data scientist," explains Max. "We're trying to create technologies to enable people to do interesting things with their data, and allow organisations to create services based on that data on your behalf."

Building the box

BBC Box is based on Raspberry Pi 3B+, the most powerful model available when this project began. "Raspberry Pi is an amazing prototyping platform," says Max. "Relatively powerful, inexpensive, with GPIO, and able to run a proper OS. Most importantly, it can fit inside a small box!"

That prototype box is a thing of beauty, a hexagonal tube made of cedar wood. "We created a set of principles for experience and interaction with BBC Box and themes of strength, protection,



The LEDs illuminate the top with a colour theme that matches the on-screen interface

It was quicker, cheaper, and more sustainable to build from cedar than using 3D printing

An Arduino Nano is used to control the LEDs

You manage your BBC Box apps using a companion device with a web browser, such as a tablet

and ownership came out very strongly," says Jasmine Cox. "We looked at shapes in nature and architecture that were evocative of these themes (beehives, castles, triangles) and played with how they could be a housing for Raspberry Pi."

The core software for collating and managing access to data is called Databox. Alpine Linux was chosen because it's "lightweight, speedy but most importantly secure", in Max's words. To get around problems making GPIO access work on Alpine Linux, an Arduino Nano is used to control the LEDs. Storage is a 64GB microSD card, and apps run inside Docker containers, which helps to isolate them from each other.

Combining data securely

The BBC has piloted two apps based on BBC Box. One collects your preferred type of TV programme from BBC iPlayer and your preferred music genre from Spotify. That unique combination of data can be used to recommend events you might like from Skiddle's database.

Another application helps two users to plan a holiday together. It takes their individual preferences and shows them the destinations they both want to visit, with information about them brought in from government and commercial sources. The app protects user privacy, because neither user has to reveal places they'd rather not visit to the other user, or the reason why.

The team is now testing these concepts with users and exploring future technology options for BBC Box.

The travel app enables people to collaborate on holiday planning without disclosing places they don't want to visit

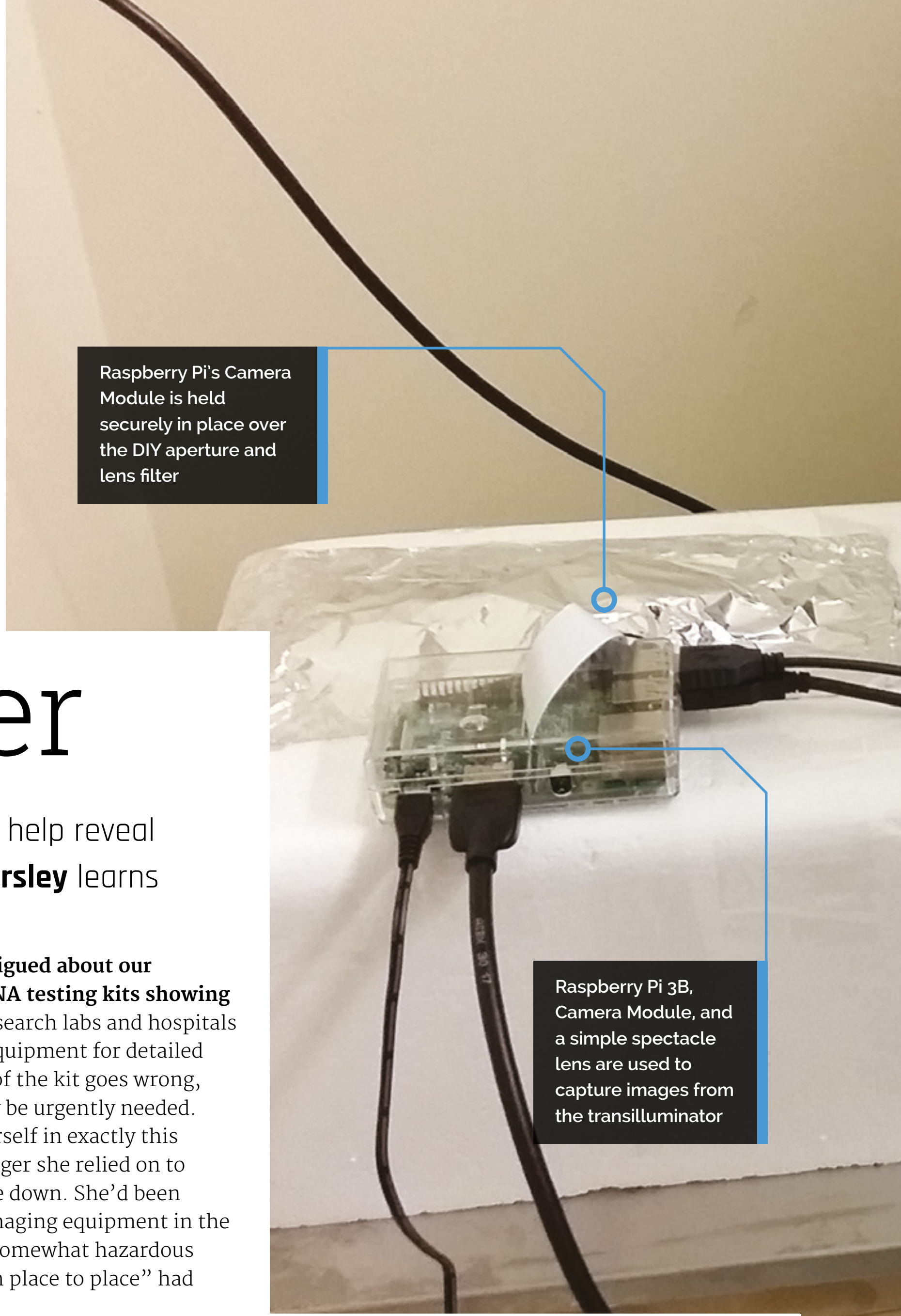
> Raspberry Pi is an amazing prototyping platform. Most importantly, it can fit inside a small box!

Your data is stored on a device in your home, rather than online. You control which apps can use it

Quick FACTS

- The pilot project took about eight weeks
- 17 people worked on the pilot in total
- Hook-and-loop tape holds everything together inside
- Apps can be developed in Node.js or Go
- Two WS2812B multicolour LEDs illuminate the hexagonal top

DNA Gel Imager



A Raspberry Pi DNA imaging setup can help reveal our individual make-up, as **Rosie Hattersley** learns

MAKER

Dr Lindsay Clark

Lindsay holds a PhD in Genetics. Her work on plant genetics focuses on environmental and agricultural issues. Her interest in Raspberry Pi began when she and her husband wanted a webcam that wouldn't steal their data.

magpi.cc/RFqRm4

Many of us are intrigued about our ancestry, with DNA testing kits showing how we fit in. Research labs and hospitals use multimillion-pound equipment for detailed DNA tests, but when part of the kit goes wrong, a low-cost alternative may be urgently needed. Dr Lindsay Clark found herself in exactly this situation when the gel imager she relied on to expose DNA samples broke down. She'd been relying on another lab's imaging equipment in the first place, and "carrying somewhat hazardous ethidium bromide gel from place to place" had never been ideal.

While we mainly think of DNA as useful for catching criminals and identifying possible blood relatives using ancestry testing kits, its main use is for genetic research. Cystic fibrosis researchers use it to test for patterns in DNA clusters, for example (see **magpi.cc/Cmkj9F**).

Lindsay studies DNA information about plants, animals, and microbes. In all cases, the DNA material extracted needs to be visualised and, in this case, she needed her own replacement imager, fast.

As a Raspberry Pi enthusiast, and Python and R programmer, she soon came up with a plan. Despite scant knowledge of optics, some online research soon gave her all the details she needed.

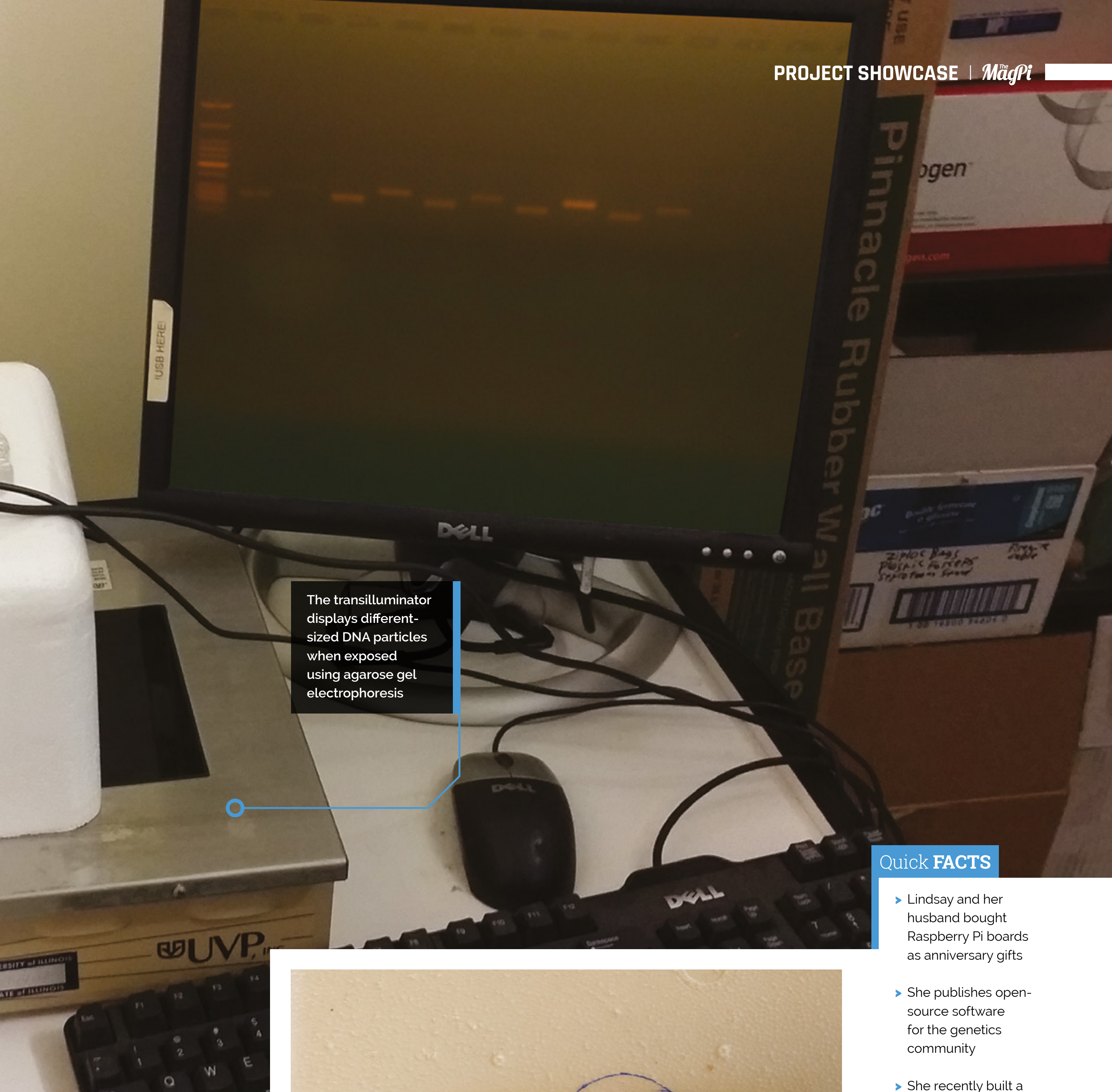
Sizing things up

"A typical gel imaging system costs tens of thousands of dollars, but it is essentially a camera mounted over a transilluminator, attached to a computer for taking digital images," she explains. "We had a transilluminator we'd inherited from an emeritus professor. The only components left then were a camera and a computer, which could be done very cheaply with a Raspberry Pi." Since Raspberry Pi uses an HDMI output, she also needed an HDMI to DVI converter costing a princely $8.

"We needed to be able to take close-up, reasonably focused images of an orange-fluorescing compound, filtering out other wavelengths of light. The images didn't need to be publication quality (we could always use another lab's transilluminator if we really needed to), but they needed to be good enough for us to document approximate DNA fragment sizes."

Optical collusion

Lindsay first set up her Raspberry Pi 3B camera kit with a fresh Raspbian installation. The only



The transilluminator displays different-sized DNA particles when exposed using agarose gel electrophoresis

Quick FACTS

- Lindsay and her husband bought Raspberry Pi boards as anniversary gifts
- She publishes open-source software for the genetics community
- She recently built a RetroPie for a friend
- Lindsay loves solving genetics' real-world logic problems…
- …such as DNA sequence alignments in allopolyploid organisms!

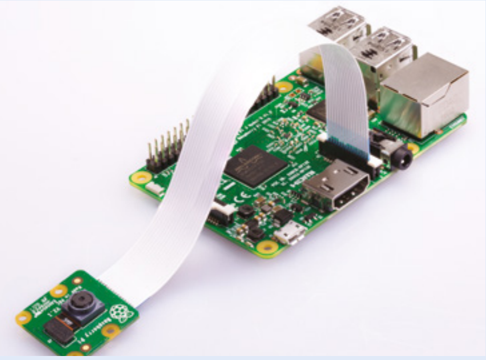
Tracing around the glasses helped get the position right

Warning!

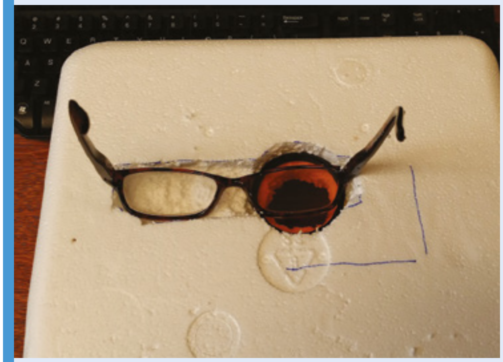
The ethidium bromide used in this project is a carcinogen. Please follow this project with care.

magpi.cc/MTtK3x

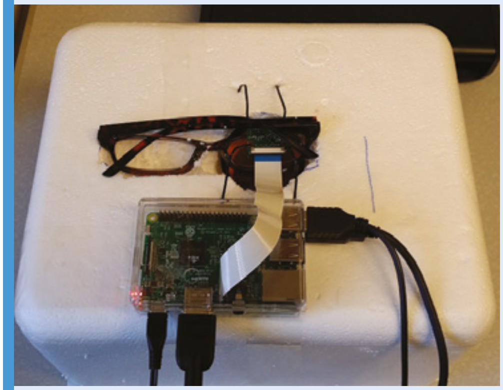
Self-assembly time



01 Set up the Raspberry Pi 3 B Camera Module and install Raspbian. You'll also need the raspistill command to trigger the Raspberry Pi camera.



02 Cut a hole in the bottom of the Styrofoam box to form an aperture and carve out one seating for the lens filter and a second to hold the glasses lens. Mount the Raspberry Pi camera and secure it in place using paper clips or other adjustable fastening.



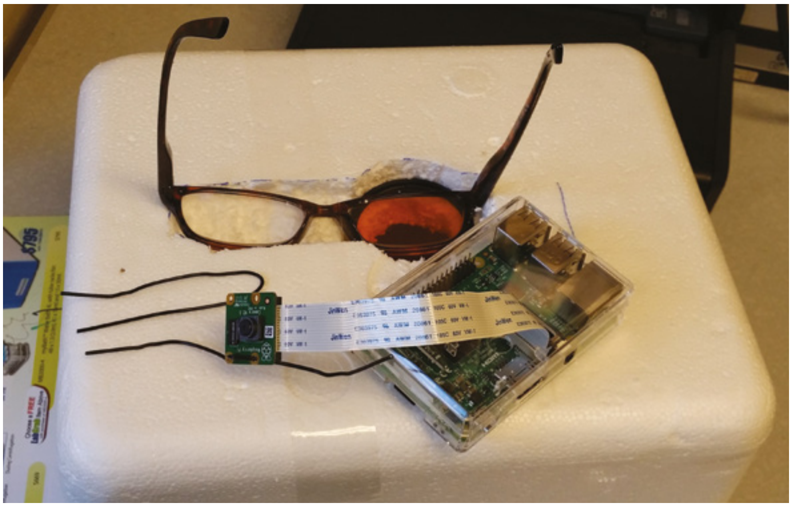
03 Trace around the camera filter, position one of the lenses in the middle of where the filter was, then trace around the reading glasses.

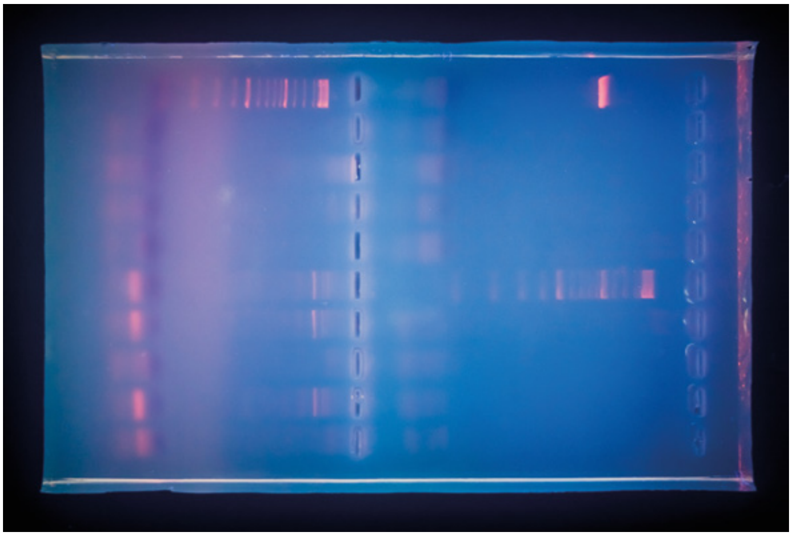
▲ The gel imager needed to sit 12 inches above the transilluminator

▼ Raspberry Pi 3 Camera Module was easy to position in place

software needed was the standard raspistill command, so she didn't need to write any custom code. The +2.00 lens used came from an old pair of spectacles. (A stronger lens may have worked even better, she thinks.)

Lindsay then added a lens filter which she bought cheaply online. This was simply to cut out any glare from the transilluminator lamp. The whole setup was housed in a cut-up Styrofoam box. In all, the device cost $150 including the Raspberry Pi 3 and Camera Module, monitor, keyboard, and mouse.





"The biggest challenge was making sure that it could focus on the gel up close," recalls Lindsay; it can focus from 1 m to infinity. "I tested it out by holding the reading glasses in front of the camera and taking pictures of some flyers." The imager she's built can't zoom or focus but, as she says, "the camera is positioned such that it can get a decent picture of any gel."

Such has been the project's success, Lindsay's colleagues without any Raspberry Pi knowledge have easily made use of it too. She taped instructions on the wall explaining how to trigger the camera using Raspbian, but rarely has anyone asked how to use her DIY setup.

> Carrying somewhat hazardous ethidium bromide gel from place to place had never been ideal!

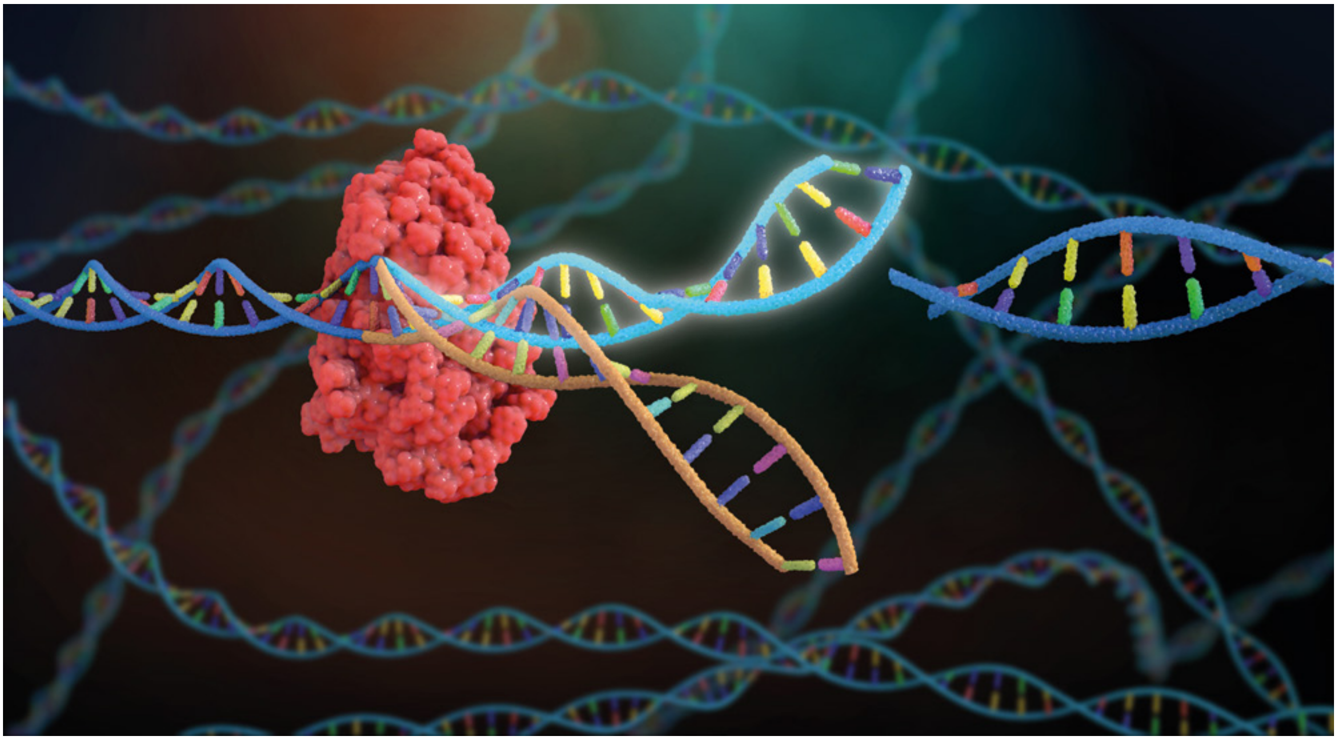
She also encourages others to create their own bespoke equipment. "There's a great tradition of scientists building their own equipment. Don't let the existence of a commercial version deter you from building your own!"

However, Lindsay warns, "Ethidium bromide is a carcinogen, not to mention that a lot of specialised equipment and reagents are needed to get DNA to the point where it could be visualised with my little imager. For example, the most common method of DNA extraction involves using chloroform in a fume hood, and then PCR is typically done after that before visualisation on a gel."

So the DNA gel imager isn't something you could easily replicate at home. Nonetheless, the idea of using a Raspberry Pi for medical applications has merit.

▲ Agarose gel helps expose DNA samples under UV light

▼ A visualisation of the CRISPR DNA editor



Social Media without the Internet

Could we interact with people using social media features in real life? **Phil King** hits the 'like' button

MAKER

Tuang Thongborisute

Raspberry Pi enthusiast, NCCE facilitator, teacher, and coder who enjoys creating new projects and hacks to inspire others to start learning.

tuangstudio.com

If you're anything like us, you probably spend many hours liking, following, and friending on social media. But have you ever pondered how this kind of digital interaction might transfer to the real world? It's a concept that interactive artist Tuang Thongborisute wanted to explore, leading her to create the 'Social Media without the Internet' interactive performance art project.

The original idea came from her research into a 'digital sense'. "Its hypothesis says that people nowadays might gradually develop an additional sense to perceive digital contents," she says.

Social Media without the Internet was created "to investigate people's familiarity of social media's data and interactions in the physical world, and to explore the digital sense by applying it to the sense of touch."

Part of the inspiration also came from a curiosity to examine society's sceptical thoughts about the lack of physical interaction in online communication and to see if physical interaction is significant to a meaningful connection.

Someone can press the shoulder pad to 'follow' the wearer; an LCD screen shows numbers followed and following

When worn in public, the interactive blazer attracted lots of attention from curious people

Social interactions

To explore these themes, Tuang created the 'Social Touch Suit', a blazer featuring numerous electronic elements – controlled by a Raspberry Pi, aided by an Arduino – which enable people the wearer meets to engage in six social interactions.

'Add friends' is achieved by a handshake, connecting two conductive rings on the wearer's fingers. "With some practice, it works naturally when interacting with people," says Tuang.

'Unfriending' is easy: just a push of a button located on the left side – "near the heart, which most people [have had] broken at least once in a lifetime anyway."

'Following' someone involves hand-holding, triggered by customised extendible strings with a microswitch. "Typically, people ask the wearer to follow them after they have followed the wearer for a while because at the end of the day, everybody also needs some attention back."

'Follow' is a tap on a Velostat sheet (pressure-conductive resistant) on the right shoulder. "A

Quick FACTS

- The electronics can be removed to wash the blazer
- A 1m strip of 144 RGB NeoPixels was used to light up the sleeves
- Batteries are used to power Raspberry Pi and Arduino
- The blazer's Raspberry Pi runs Python and Processing code
- Analogue input is read via an MCP3008 ADC

To 'like' the project/wearer, people can tap the touchscreen or do a high five

This double dot matrix display shows the number of friends gained by shaking hands

NeoPixel strips on the arms light up in different colours and patterns for each social interaction

▲ A high five is the more interesting way of adding a 'like', shown on the touchscreen

◀ Some of the social interactions possible by pressing various parts of the blazer or touching the hand of its wearer

good follower and friend would put their hand gently on the wearer's shoulder."

A 'Like' can be achieved via two interactions: a high-five, triggered by FSR pad attached to the edge of the right sleeve, or tapping a button on the 7-inch touchscreen.

'Dislike' is also done via the touchscreen – "I don't remember anyone intentionally disliking me... except my best friends, who did it several times."

In addition, three tiny cameras are attached to the blazer, to broadcast the interaction in real time using a local network for performing in a closed environment like an indoor gallery. "This feature mimics an action of social media users' observation on other people's interaction without any interference," says Tuang.

> Three tiny cameras are attached to the blazer, to broadcast the interaction in real time using a local network

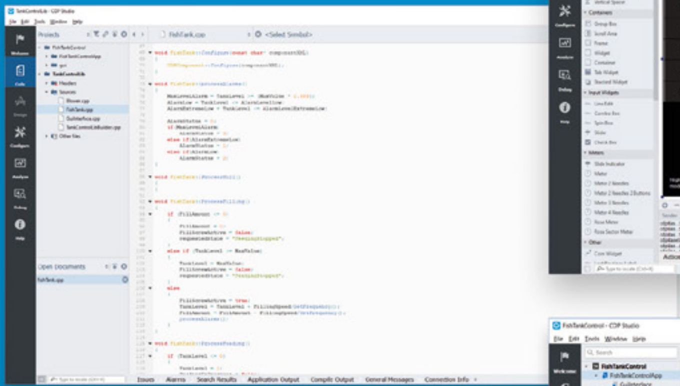
Out and about

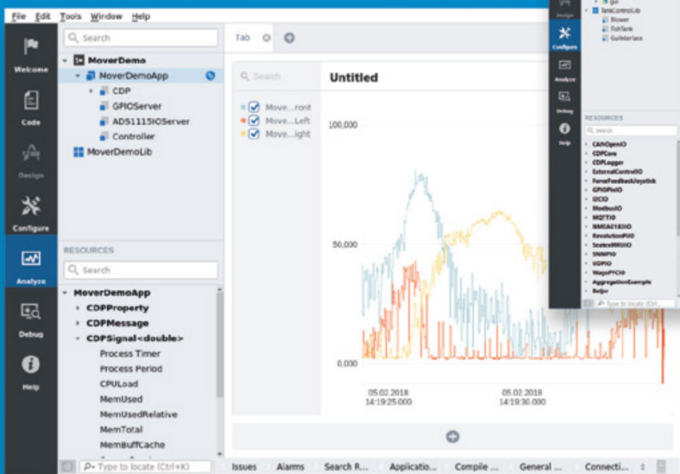
When wearing the blazer in public, Tuang found that people were curious or confused. "Some may hesitate to ask or interact, but some partake in face-to-face conversation," she tells us.

"From these experiences, I think what's interesting is that these data of physical interaction have more potential to come from a sincere feeling and determination after they understand how it works. Because no one can give this feedback remotely, there's some work that needs to be done to give and receive the numbers and to actually do it face-to-face with one another."

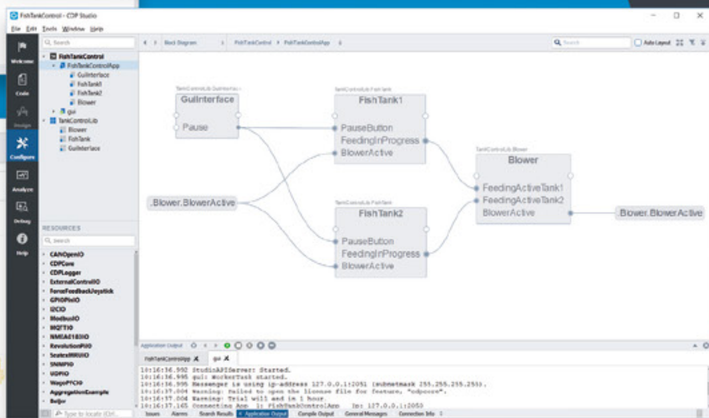
Design

Code





Configure



Analyze

NEW GRAPHICAL EDITOR!

CDP STUDIO

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.

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

Perpetual Chimes

Frazer Merrick has developed a set of interactive wind chimes to take you away from it all. **Nicola King** tunes in to find out more

When a chime hits the central disc, it completes a circuit and triggers a key press on the Raspberry Pi

As they're not outside, the chimes rely on human interaction to move them

The atmospheric soundscape generated can be enjoyed by the user via headphones

MAKER

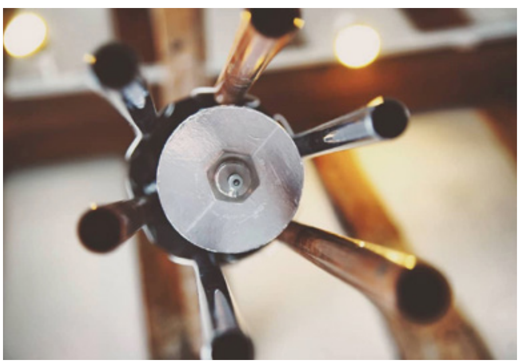
Frazer Merrick

A sound artist and educator, Frazer enjoys circuit bending, field recording, and physical computing.

magpi.cc/jtses3

The Perpetual Chimes project is very clever – a set of augmented wind chimes which, since they are indoors, require the user to interact with them in order to create an escapist soundscape played through headphones. Frazer Merrick came up with the idea for the chimes while experimenting with ways of creating interactive musical instruments, and exploring how sounds can transform a space. "At the time I was really interested in escapism," he recalls, "so I knew I wanted to make a headphone-based interactive installation that captivated the player/listener – something that didn't make much acoustic sound, but instead something just for the player."

Frazer initiated the idea back in 2016, creating a prototype where the chimes were suspended from a brake disc hanging from a mic stand. Keen to improve on this first version, Frazer and a collaborator from **@LimboEducation** began work again in 2018 to revive and improve the project: "We set to work making it far more robust and, importantly, standalone."

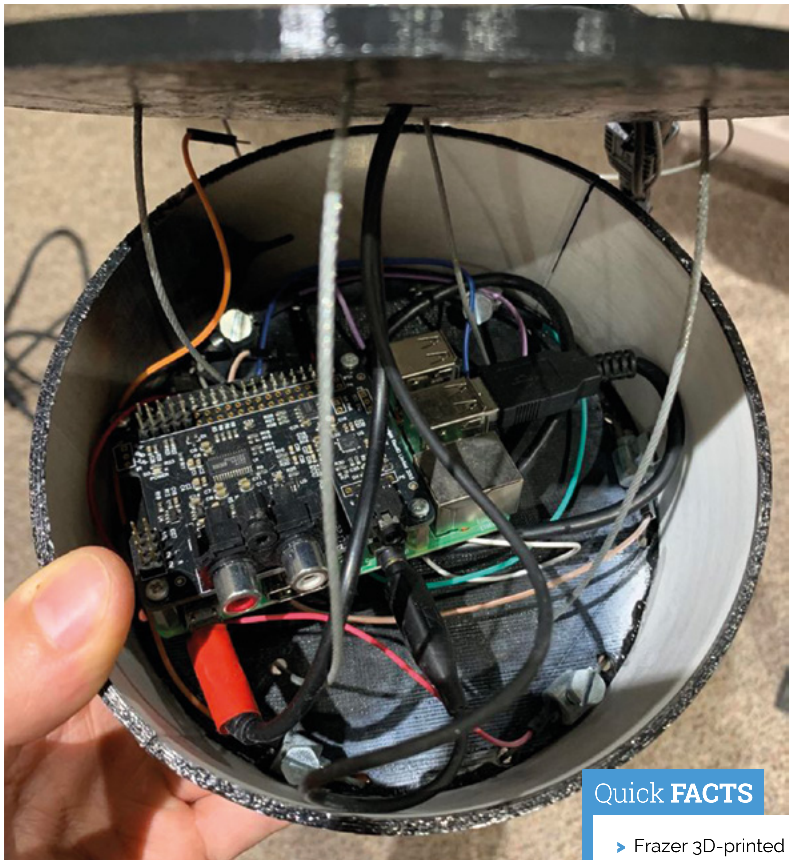


Viewed from below, you can clearly see the chimes around the central disc

The sound of music

So, how do the chimes work? Frazer explains, "When you hit one of the chimes, it strikes a disc in the middle. Both these elements are connected to a Makey Makey (the central disc being 'earth'), which then triggers a key press on [the project's]

▲ Someone enjoying the calming sounds triggered by the chimes



▲ The Makey Makey and Raspberry Pi are housed in a 3D-printed case

Raspberry Pi. I programmed a patch in Scratch to play audio files when it receives these key presses, which you then hear over the headphones. There is no acoustic sound other than the dull clunk of stainless steel and copper pipes hitting one another. However, in the headphones is an atmospheric soundscape of calming field recordings and synthesizer drones."

Frazer made the recordings using the Alchemy synthesizer in Logic Pro. "I used the amplitude of a waterfall recording (which I made in the Isle of Mull) to affect different parameters of the synthesizer," he says. "In Scratch, there's a variable counting every time the chimes 'strike' and when this is a modulus of 25, one of three large pulsating bass notes plays too, adding an element of surprise to the installation and encouraging you to keep playing and discovering more combinations of notes. Alongside this is a subtle field recording of the coastline from the same peaceful trip to the Isle of Mull, completing the escapist soundscape."

Heavenly harmonies

Sounds idyllic, and Frazer has clearly enjoyed seeing people explore the possibilities of the chimes: "It's so rewarding to see people playing and smiling with the chimes, creating their own soundscapes by activating them (some harder than others). My favourite comment was someone who called the work 'curious', as this summed up my work so much better than I ever could."

> It's so rewarding to see people playing and smiling with the chimes, creating their own soundscapes

Frazer admits that it was a challenge to fit all of the components inside the head unit of the chimes, but "after cutting down a few cables and shuffling things around, I managed to fit it all in. Thankfully the support system is designed so I can easily adjust the hanging height from the beam above, which is very useful when installing in different venues."

It was also his first project with a Raspberry Pi, which he used "because I wanted the chimes to be unmanned and to be installed for long periods of time, all without having to worry about securing a laptop somewhere behind the scenes."

Quick FACTS

- Frazer 3D-printed the case for the electronics
- Frazer exhibited the chimes at Colchester School of Art
- A lot of holes needed to be drilled in the project!
- The Scratch code can be found at **magpi.cc/gPpuw4**
- Frazer plans to switch from Scratch to Pure Data for better audio quality

RASPBERRY PI 4K DIGITAL MEDIA HUB

Build the ultimate 4K home theatre PC using a Raspberry Pi 4 and Kodi – By **Wes Archer**

We love Raspberry Pi for how it's helping a new generation of children learn to code, how it's resulted in an explosion of new makers of all ages, and how it's really easy to turn any TV into a smart TV.

While we always have a few Raspberry Pi computers at hand for making robots and cooking gadgets, or just simply coding a Scratch game, there's always at least one in the house powering a TV. With the release of the super-powered Raspberry Pi 4, it's time to fully upgrade our media centre to become a 4K-playing power house.

We asked Wes Archer (**@raspberrycoulis**) to take us through setting one up. Grab a Raspberry Pi 4 and a micro-HDMI cable, and let's get started.

Get the right hardware

Only Raspberry Pi 4 can output at 4K, so it's important to remember this when deciding on which Raspberry Pi to choose

Raspberry Pi has been a perfect choice for a home media centre ever since it was released in 2012, due to it being inexpensive and supported by an active community. Now that 4K content is fast becoming the new standard for digital media, the demand for devices that support 4K streaming is growing, and fortunately Raspberry Pi 4 can handle this with ease! There are three versions of Raspberry Pi 4, differentiated by the amount of RAM they have: 1GB, 2GB, or 4GB. So, which one should you go for? In our tests, all versions worked just fine, so go with the one you can afford.

Remote controlled

It's possible to plug in an IR receiver and program Kodi so that you can use your TV remote to control your media centre, but the Flirc USB IR receiver is perfectly designed to do just that. Simply plug the USB receiver into a PC and follow a few quick steps to be up and running in no time.

Cases

Flirc Raspberry Pi 4 case

Made of aluminium and designed to be its own heatsink, the Flirc case for Raspberry Pi 4 is a perfect choice and looks great as part of any home media entertainment setup. This will look at home in any home entertainment system.

magpi.cc/NnDZiA

Official Raspberry Pi 4 case (in black and grey)

The official Raspberry Pi 4 case is always a good choice, especially the black and grey edition as it blends in well within any home entertainment setup. If you're feeling adventurous, you can also hack the case to hold a small fan for extra cooling.

magpi.cc/frppYm

Aluminium Heatsink Case for Raspberry Pi 4

Another case made of aluminium, this is effectively a giant heatsink that helps keep your Raspberry Pi 4 cool when in use. It has a choice of three colours – black, gold, and gunmetal grey – so is a great option if you want something a little different.

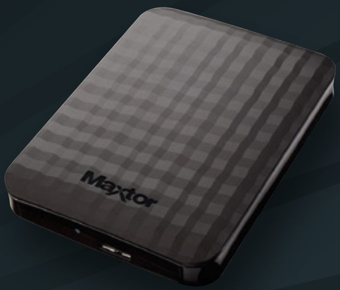
magpi.cc/knNohY

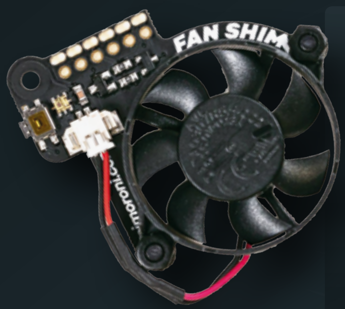
Optional add-ons

▶ Maxtor 2TB external USB 3.0 HDD

4K content can be quite large and your storage will run out quickly if you have a large collection. Having an external hard drive connected directly to your Raspberry Pi using the faster USB 3.0 connection will be extremely handy and avoids any streaming lag.

magpi.cc/hyDQvY





◀ Fan SHIM

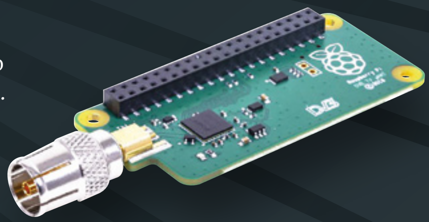
The extra power Raspberry Pi 4 brings means things can get quite hot, especially when decoding 4K media files, so having a fan can really help keep things cool. Pimoroni's Fan SHIM is ideal due to its size and noise (no loud buzzing here). There is a Python script available, but it also 'just works' with the power supplied by Raspberry Pi's GPIO pins.

magpi.cc/qiYBWd

▶ Raspberry Pi TV HAT

If you are feeling adventurous, you can add a Raspberry Pi TV HAT to your 4K media centre to enable the DVR feature in Kodi to watch live TV. You may want to connect your main aerial for the best reception. This will add a perfect finishing touch to your 4K media centre.

magpi.cc/imDdcw



▶ Rii i8+ Mini Wireless Keyboard

If your TV does not support HDMI-CEC, allowing you to use your TV remote to control Kodi, then this nifty wireless keyboard is extremely helpful. Plug the USB dongle into your Raspberry Pi, turn on the keyboard, and that's it. You now have a mini keyboard and mouse to navigate with.

magpi.cc/ApxYux

Get the right cables

Raspberry Pi 4 uses a micro-HDMI cable instead of the standard HDMI. In fact, it uses two as you can output to two 4K displays at once. We recommend buying the newer micro-HDMI cables, but you can also use micro-HDMI to HDMI adapters if needed. If you do buy new cables, be sure they are micro-HDMI to HDMI, as most TVs will have the standard-sized HDMI inputs.

Install & set up LibreELEC

LibreELEC is a lightweight OS designed to run Kodi, the home media centre software we'll be using in this guide

Installation steps

01 Download the LibreELEC USB-SD Creator app

LibreELEC has a lovely app that makes it really simple to get up and running. Download the version for your OS (it supports Windows, Linux, and macOS) from **magpi.cc/epmapU** – or if you prefer, you can download the image for your chosen Raspberry Pi instead.

02 Download the LibreELEC image

Once downloaded, insert your microSD card into your computer and fire up the LibreELEC app you just downloaded. Select 'Raspberry Pi 4' from the version drop-down and then hit Download. Choose where to download the image file to and wait for the app to download the image.

03 Create your microSD card

Once your LibreELEC image has finished downloading, insert your microSD card into your computer, then select the drive from the drop-down menu. Lastly, hit Write and then wait for that to finish. Once done, you should have a working microSD card to use in your Raspberry Pi 4.

First-time setup steps

Spending some time **organising your libraries** first is definitely worth it

01 The first boot of LibreELEC

Now you have your microSD card ready, connect the micro-HDMI cable to your Raspberry Pi 4 and TV and then hit the power. The first time LibreELEC boots, it will do some housekeeping, like checking the file system and expanding to fill your microSD card before rebooting automatically for the next step.

02 Choose your language and get online

LibreELEC will launch a welcome wizard to help get you started. You'll need to pick your language, give your device a name (so it's easy to find on your network), connect to WiFi, and configure SSH and/or Samba file sharing. You should be able to control this using your TV's remote control without any setup if it supports HDMI-CEC.

> "The first time LibreELEC boots, it will do some housekeeping, like checking the file system"

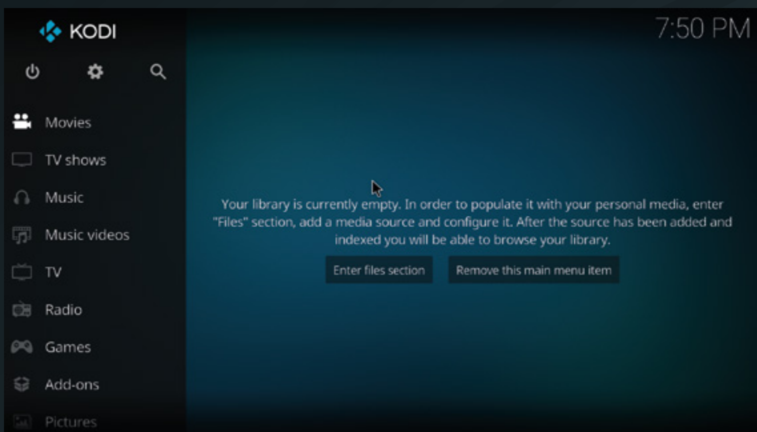
03 Welcome to Kodi on LibreELEC!

Once the wizard has been completed, you should be taken to the main home screen within Kodi. Congratulations! Right now, there's not much to do other than familiarise yourself with the menus, maybe make some settings changes (e.g. change the location to the correct one for you), and explore. We'll now show you how to add your media.

Add media libraries

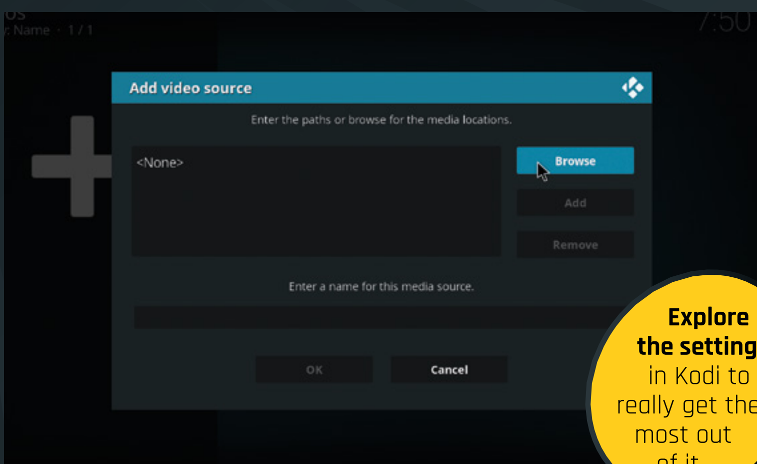
01 Organise before you begin!

Before you add your media to your new Raspberry Pi 4K media centre, a little organisation is recommended. This way, when you add the libraries to Kodi, scrapers (more on that later) will be able to download all the extra artwork to make your entertainment system look the business.



02 Add your media libraries

A media centre is nothing without media, so we'll need to add ours before we can play them on our Raspberry Pi. Kodi makes this really simple, so navigate to Videos in the menu, select Files, and finally 'Add videos'. You'll now be able to use the Browse option to locate your media files, depending where they are stored.

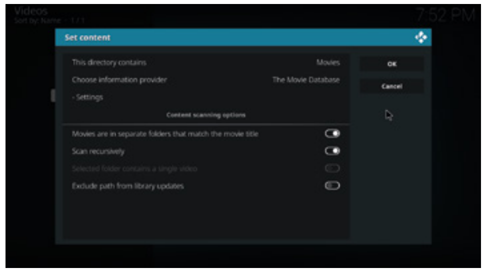


Explore the settings in Kodi to really get the most out of it

03 Give your libraries names

You can group your libraries, and typically Kodi uses Movies, TV, Music, and Photos. By adding your media into groups, Kodi will treat them accordingly so you can browse and watch very easily from the navigation menu. It is even possible to combine multiple sources (e.g. from a NAS and a USB HDD) for convenience.

Organising media and scraping info



01 Using scrapers to add artwork

Scrapers are essentially scripts within Kodi that can search online databases to pull information about each media file you have, and download the corresponding artwork, such as movie poster, disc art, cases, and wallpapers (aka fan art). Assuming your libraries are organised as per our steps earlier, this should be straightforward.

02 Set the content type when adding media

When adding media, you should be asked what the directory contains (it defaults to 'None'). If you pick Movies from the options, this tells Kodi to use the appropriate information provider (i.e. The Movie Database) to scrape and download the resulting information and artwork when scanning the library.

03 Let the scraping begin!

Once you have set the content and chosen the information provider, hit OK and then you'll be asked if you want to refresh information for all items – hitting Yes will start the scraping, and you'll soon have detailed information about each movie, including any artwork too.

Advanced configuration

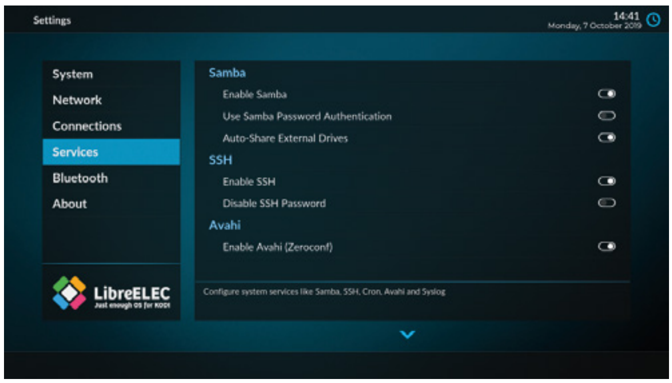
Now that the basics have been mastered in our Raspberry Pi 4K media centre, why not try something a little more advanced?

A network-attached storage device (NAS for short) is a hard drive (or drives!), usually served by a lightweight operating system, that is attached to a network. The beauty is that you can share files across your network so that other network attached devices (such as our media centre) can access them.

Using your phone as the remote

Kodi has an official app available (iOS: **magpi.cc/LbeLJp**, Android: **magpi.cc/RBNqRy**) that allows you to control your media centre directly from your phone. Download the app to your phone, then hit 'Add host' to pair it with your Raspberry Pi. You can manually enter your Raspberry Pi's IP address, or hit 'Find Kodi' which should automatically find it for you. Once paired, you can use the app to navigate around Kodi like a regular remote.

Add network storage



01 Ensure SMB has been enabled in Kodi

Remember at the welcome wizard when you were asked about SSH and Samba? Well, Samba is not just a Brazilian dance, but a form of file sharing too! If you did not enable Samba (abbreviated to SMB), then you can do so in System > LibreELEC > Services.

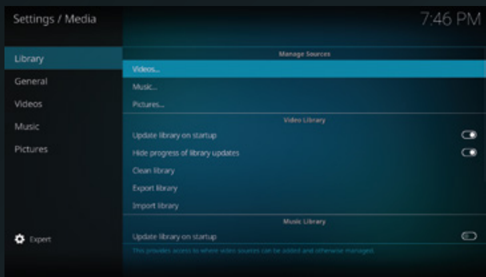
02 Add your network shared media

This process is very similar as before. Go to Videos > Files > Add videos, then hit Browse. This time, select 'Windows network (SMB)' and then you should see your shared files appear. This assumes you have already configured SMB on your NAS; you may be prompted to enter a username and password if required.

03 Set the content for your libraries

Again, be sure to pick the content type for your network shared files. Kodi will add the files to the appropriate place in the navigation menus accordingly. Sharing files over the network can cause some buffering, depending on how fast and reliable your home network is, so just keep this in mind.

Advanced settings



▲ Update and clean your library automatically

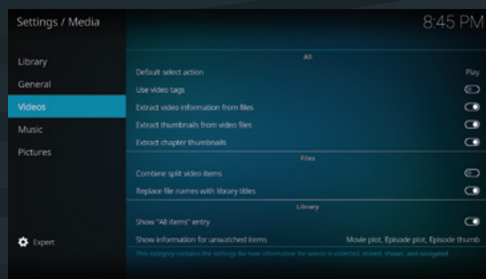
If you regularly add and remove content to and from your libraries, you will want to ensure that Kodi updates the actual libraries too. Enable the 'Update library on startup' option to ensure that your new files are added automatically. It is also worth 'cleaning' your library too, in order to remove deleted files.

Set the region to your own

By default, Kodi favours the US audience, so if you are one of our US readers then this probably won't apply to you! However, the regional settings – such as the date, time, and temperature format – can be changed to match your own. Head to Settings > Interface > Regional and set it to your liking.

Change the look and feel of Kodi

There's nothing wrong with the default Kodi skin (Estuary), but there are a number of different ones to install and try out. Head over to Settings > Interface, then you can change the skin, or 'Get more' with a few clicks. Our personal favourite is Aeon Nox: SiLVO because there are so many customisations available.



▲ Enhance the information Kodi sees

Media files often have additional 'tags' that contain extra information about the file itself, which can be quite useful when you have a large collection. Going into Settings > Media > Videos and turning on 'Use video tags' enables this. Whilst you're there, ensure the three 'Extract…' options are on too, in order to enhance your experience.

App recommendations

Official Kodi Remote for iOS

Available on the App Store for iOS, the Official Kodi Remote is a sturdy choice for Apple users.

Kore, Official Remote for Android

Android users can also use the Official Kodi Remote app, called Kore, available in the Play Store.

Yatse

An alternative for Android users, Yatse has fantastic reviews and also has support for Plex and Emby servers.

JOIN FOR 12 MONTHS AND GET A

FREE Raspberry Pi Zero W Starter Kit

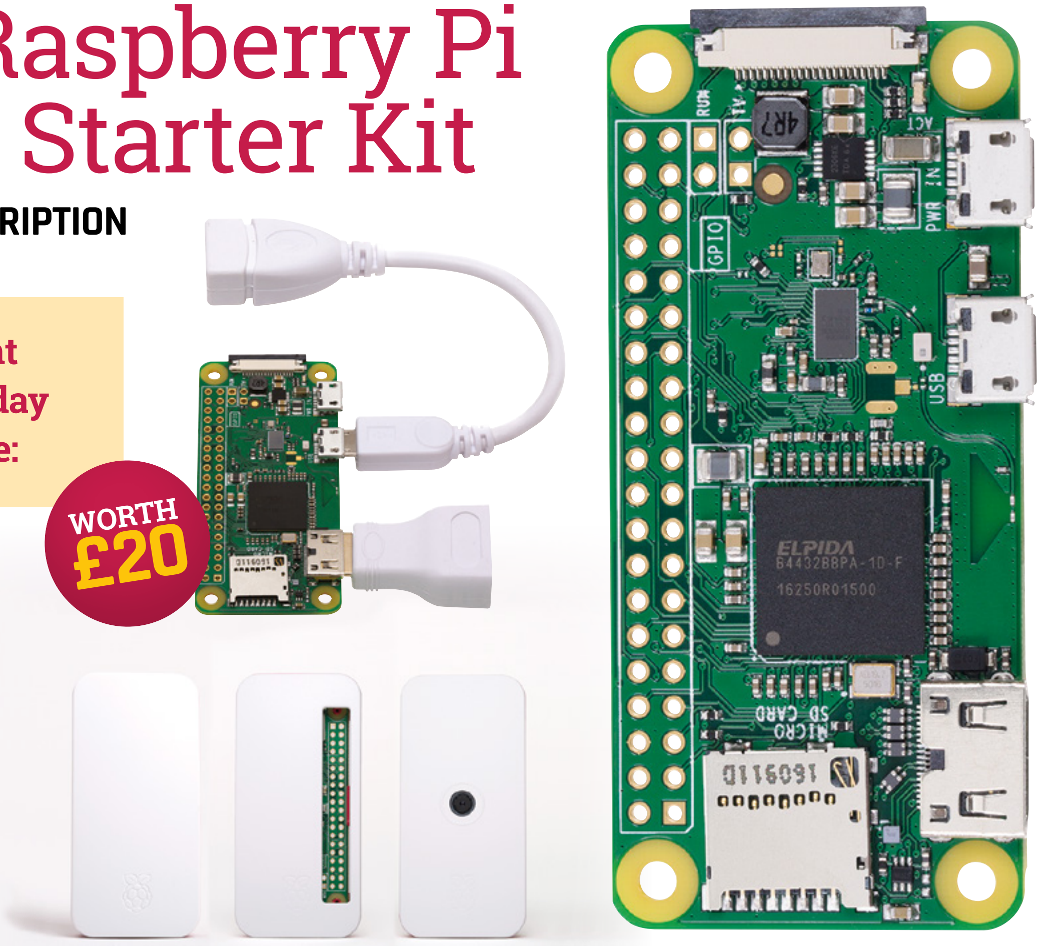
WITH YOUR SUBSCRIPTION

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

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

Offer subject to change or withdrawal at any time

WORTH £20



Buy now: magpi.cc/subscribe

SUBSCRIBE on app stores

From **£2.29**

Available on the App Store

GET IT ON Google Play

Build a Raspberry Pi cluster computer

Think Raspberry Pi computers are no match for their bigger cousins? Think again, because there's strength in numbers

MAKER

PJ Evans

PJ is a writer, developer, and Milton Keynes Raspberry Jam wrangler. Unlike a cluster, he can only do one thing at a time.

@mrpjevans

Raspberry Pi computers are famously cheap and cheerful. Great for playing around with and the odd little project, right? Well, sometimes.

However, our little friend is a surprisingly powerful computer and when you get lots of them working together, amazing things can happen.

The concept of computer 'clusters' (many computers working together as one) is nothing new, but when you have a device as affordable as Raspberry Pi, you can start to rival much more expensive systems by using several in parallel. Here, we'll learn how to make a cluster computer from a lot of little computers.

01 Cluster assemble!

A cluster of Raspberry Pi computers can start with as little as two and grow into hundreds. For our project, we're starting with a modest four. Each one, known as a 'node', will carry out part of our task for us and they all work in parallel to produce the result a lot quicker than a single node ever could. Some nice 'cluster cases' are available, and we start here by assembling our Raspberry Pi 4B computers into a four-berth chassis. Many different configurations are available, including fan cooling.

A dedicated inexpensive switch will speed up communications. Raspberry Pi 4 computers can take advantage of full-bandwidth Gigabit Ethernet

02 Power up

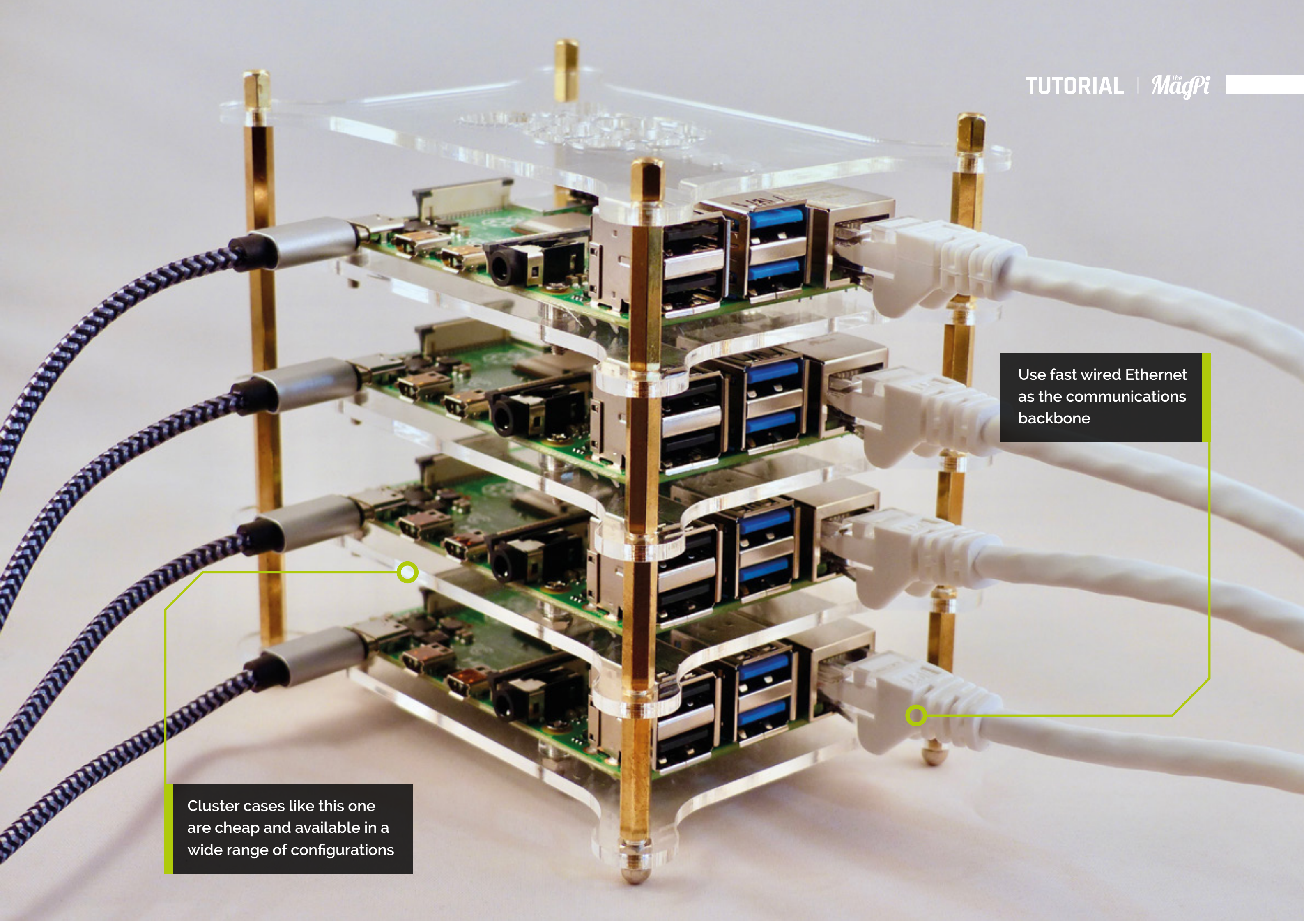
Consider the power requirements for your cluster. With our four nodes it's not going to be ideal to have four PSUs driving them. As well as being ugly, it's inefficient. Instead, track down a good-quality, powerful multi-port USB charger that is capable of powering your chosen number of computers. Then all you need are the cables to link them and you're using a single mains socket. USB units are available that can handle eight Raspberry Pi computers without breaking a sweat. Do be careful of the higher demands of Raspberry Pi 4.

03 Get talking

A cluster works by communication. A 'master' node is in charge of the cluster and the 'workers' are told what to do and to report back the results on demand. To achieve this we're using wired Ethernet on a dedicated network. It's not essential to do it this way, but for data-intensive applications it's advisable for the cluster to have its own private link-up so it can exchange instructions without being hampered by wireless LAN or other network traffic. So, in addition to wireless LAN, we're linking each node to an isolated Gigabit Ethernet switch.

You'll Need

- 4 × Raspberry Pi 4 computers **magpi.cc/rpi4**
- Cluster case **magpi.cc/XtUbVm**
- Ethernet switch **magpi.cc/fzcGEA**
- Multi-port USB PSU **magpi.cc/AekkAr**
- 4 × USB C cables **magpi.cc/UcmFAM**
- 4 × Ethernet cables **magpi.cc/LsAnrA**



Use fast wired Ethernet as the communications backbone

Cluster cases like this one are cheap and available in a wide range of configurations

04 Raspbian ripple

We're going to access each node using wireless LAN so the Ethernet port is available for cluster work. For each 'node', burn Raspbian Buster Lite (**magpi.cc/raspbian**) to a microSD card, boot it up, and make sure it's up to date with `sudo apt -y update && sudo apt -y upgrade`. Then run `sudo raspi-config` and perform the following steps:

- Change the 'pi' user password.
- Under 'Networking', change the hostname to nodeX, replacing X with a unique number (node1, node2 etc.). Node1 will be our 'master'.
- Enable WiFi if desired.
- Exit and reboot when prompted.

05 Get a backbone

The wired Ethernet link is known as the cluster's 'backbone'. You need to manually enable the backbone, as there is no DHCP server to help. We're going to use the 10.0.0.0 subnet. If your regular network uses this, choose something different like 192.168.10.0. For each node, from the command line, edit the network configuration:

```
sudo nano /etc/dhcpcd.conf
```

Go to the end of file and add the following:

```
interface eth0
static ip_address=10.0.0.1/24
```

For each node, replace the last digit of '10.0.0.1' with a new unique value: 10.0.0.2, 10.0.0.3, and so on. Reboot each node as you go. You should be able to ping each node – for example, from 10.0.0.1:

```
ping 10.0.0.2
```

06 Brand new key

For the cluster to work, each worker node needs to be able to talk to the master node without needing a password to log in. To do this, we use SSH keys. This can be a little laborious, but only needs to be done once. On each node, run the following:

```
ssh-keygen -t rsa
```

This creates a unique digital 'identity' (and key pairs) for the computer. You'll be asked a few

Top Tip

Load balancing

Clusters are also useful for acting as a single web server and sharing traffic, such as Mythic Beast's Raspberry Pi web servers.

Top Tip

Fault tolerance

Certain cluster types, such as Docker Swarm or Kubernetes, allow individual nodes to fail without disrupting service.

questions; just press **RETURN** for each one and do not create a passphrase when asked. Next, tell the master (node1, 10.0.0.1 in our setup) about the keys by running the following on every other node:

```
ssh-copy-id 10.0.0.1
```

Finally, do the same on the master node (node1, 10.0.0.1) and copy its key to every other node in the cluster.

07 Install MPI

The magic that makes our cluster work is MPI (Message Passing Interface). This protocol allows multiple computers to delegate tasks amongst themselves and respond with results. We'll install MPI on each node of our cluster and, at the same time, install the Python bindings that allow us to take advantage of its magical powers. On each node, run the following:

```
sudo apt install mpich python3-mpi4py
```

Once complete, test MPI is working on each node:

```
mpiexec -n 1 hostname
```

You should just get the name of the node echoed back at you. The `-n` means 'how many nodes to run this on'. If you say one, it's always the local machine.

Instead of a row of wall-warts, use a multi-port USB charger to power your cluster nodes, but make sure your choice of unit has enough amps

08 Let's get together

Time for our first cluster operation. From node1 (10.0.0.1), issue the following command:

```
mpiexec -n 4 --hosts 10.0.0.1,10.0.0.2,10.0.0.2,10.0.0.4 hostname
```

We're asking the master supervisor process, `mpiexec`, to start four processes (`-n 4`), one on each host. If you're not using four hosts, or are using different IP addresses, you'll need to change this as needed. The command `hostname` just echoes the node's name, so if all is well, you'll get a list of the four members of the cluster. You've just done a bit of parallel computing!

> "Now we've confirmed the cluster is operational, let's put it to work"

09 Is a cluster of one still a cluster?

Now we've confirmed the cluster is operational, let's put it to work. The **prime.py** program is a simple script that identifies prime numbers. Enter the code shown in the listing (or download from **magpi.cc/EWASJx**) and save it on node1 (10.0.0.1). The code takes a single argument, the maximum number to reach before stopping, and will return how many prime numbers were identified during the run. Start by testing it on the master node:

```
mpiexec -n 1 python3 prime.py 1000
```

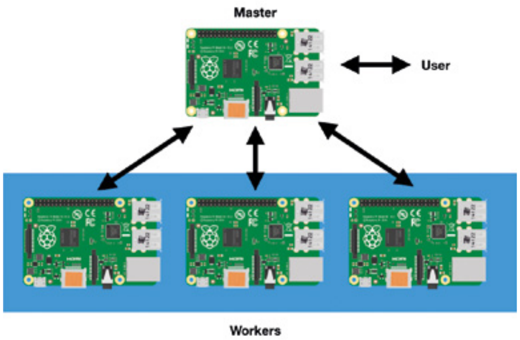
Translation: 'Run a single instance on the local node that runs **prime.py** testing for prime numbers up to 1000.'

This should run pretty quickly, probably well under a second, and find 168 primes.

10 Multiplicity

In order for the cluster to work, each node needs to have an identical copy of the script we need to run, and in the same place. So, copy the same script to each node. Assuming the file is in your home directory, the quick way to do this is (from node1):

```
scp ~/prime.py 10.0.0.x:
```



Our cluster works by assigning a master node. The master assigns tasks to its member nodes and waits for them to report their results.
Credit: *Raspberry Pi illustrations by Jonathan Rutheiser*

Replace `x` with the number of the node required: `scp` (secure copy) will copy the script to each node. You can check this has worked by going to each node and running the same command we did on node1. Once you are finished, we are ready to start some real cluster computing.

11 Compute!

To start the supercomputer, run this command from the master (node1):

```
mpiexec -n 4 --host 10.0.0.1,10.0.0.2,10.0.0.3,10.0.0.4 python3 prime.py 100000
```

Each node gets a 'rank': a unique ID. The master is always 0. This is used in the script to allocate which range of numbers each node processes, so no node checks the same number for 'primeness'. When complete, each node reports back to the master detailing the primes found. This is known as 'gathering'. Once complete, the master pulls all the data together and reports the result. In more advanced applications, different data sets can be allocated to the nodes by the master ('scattering').

12 Final scores

You may have noticed we asked for all the primes up to 1000 in the previous example. This isn't a great test as it is so quick to complete. 100,000 takes a little longer. In our tests, we saw that a single node took 238.35 seconds, but a four-node cluster managed it in 49.58 seconds – nearly five times faster!

Cluster computing isn't just about crunching numbers. Fault-tolerance and load-balancing are other concepts well worth investigating. Some cluster types act as single web servers and keep working, even if you unplug all the Raspberry Pi computers in the cluster bar one.

prime.py

DOWNLOAD THE FULL CODE:

magpi.cc/EWASJx

Language: **Python 3**

```
from mpi4py import MPI
import time
import sys

# Attach to the cluster and find out who I am and how big it is
comm = MPI.COMM_WORLD
my_rank = comm.Get_rank()
cluster_size = comm.Get_size()

# Number to start on, based on the node's rank
start_number = (my_rank * 2) + 1

# When to stop. Play around with this value!
end_number = int(sys.argv[1])

# Make a note of the start time
start = time.time()

# List of discovered primes for this node
primes = []

# Loop through the numbers using rank number to divide the work
for candidate_number in range(start_number,
                              end_number, cluster_size * 2):

    # Log progress in steps
    # print(candidate_number)

    # Assume this number is prime
    found_prime = True

    # Go through all previous numbers and see if any divide without remainder
    for div_number in range(2, candidate_number):
        if candidate_number % div_number == 0:
            found_prime = False
            break

    # If we get here, nothing divided, so it's a prime number
    if found_prime:
        # Uncomment the next line to see the primes as they are found (slower)
        # print('Node ' + str(my_rank) + ' found ' + str(candidate_number))
        primes.append(candidate_number)

# Once complete, send results to the governing node
results = comm.gather(primes, root=0)

# If I am the governing node, show the results
if my_rank == 0:

    # How long did it take?
    end = round(time.time() - start, 2)

    print('Find all primes up to: ' + str(end_number))
    print('Nodes: ' + str(cluster_size))
    print('Time elapsed: ' + str(end) + ' seconds')

    # Each process returned an array, so lets merge them
    merged_primes = [item for sublist in results for item in sublist]
    merged_primes.sort()
    print('Primes discovered: ' + str(len(merged_primes)))
    # Uncomment the next line to see all the prime numbers
    # print(merged_primes)
```

Part 04

Add sensors to a low-cost robot

Make a robot react to the world with sensors. Fascinating behaviours emerge with only a bit of code and electronics!

MAKER

Danny Staple

Danny makes robots with his kids as Orionrobots on YouTube, and is the author of *Learn Robotics Programming*.

orionrobots.co.uk

You'll Need

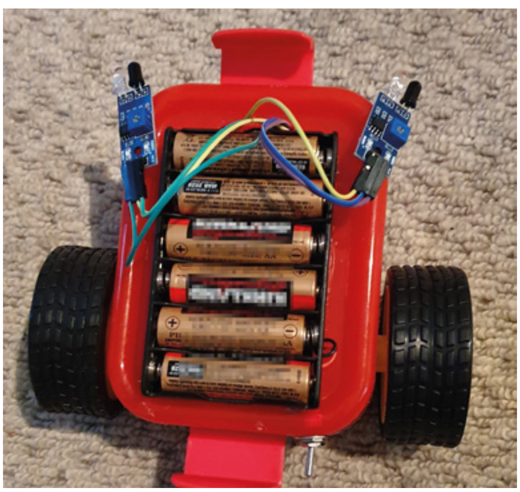
- 2 × Obstacle avoidance modules **magpi.cc/paTuQW**
- Mini breadboard **magpi.cc/CuBDyB**
- 2 × TCRT5000 sensor modules **magpi.cc/TtfhiF**
- Jumper wires **magpi.cc/pPnpZL**
- Plastic standoffs and screws **magpi.cc/Cixtpr**
- 5 mm pitch terminal block **magpi.cc/jeThnM**
- Black insulating tape
- Plain white paper

Over the last few issues we've built a low-cost wheeled robot. Without any sensors, it doesn't respond to the world and drives into walls.

The robot's Raspberry Pi has many GPIO pins left to add inexpensive sensors for it to interact with its surroundings. This tutorial shows how to use a couple of similar sensor types, put them in useful places, wire them in, and write the code. We'll touch on the trade-offs and limitations, learn how to calibrate the sensors, and make test tracks for line following.

01 Meet the sensors

The two types of sensor in this tutorial both detect reflected infrared light (IR). Obstacle sensors detect objects close enough by a brightness level. Line sensors detect how light/dark the floor is below them.



The robot with sensors part-fitted. See how the cables go through a port in front of the batteries

The sensor models chosen are cheap and easily found online, but being IR sensors, they can be dazzled and confused by bright sunlight and some fluorescent lights.

These sensors output digital (on/off) signals, with a dial adjusting the level to go from off to on. They come on carrier boards as 3.3 V compatible modules, making them easier to connect to Raspberry Pi.

02 How to use sensors

Obstacle sensors should be front-most, so they don't detect the robot itself. By using two sensors facing forward and slightly to either side, a robot can decide which way it needs to turn to avoid an obstacle.

Line sensors should be under the robot to detect if it has gone off track. A single sensor could sweep across and back over a line, but two sensors, wide enough to go either side of a line, make for a smoother system. They can sense when the line is not in the middle, and which way the robot needs to turn to correct it.

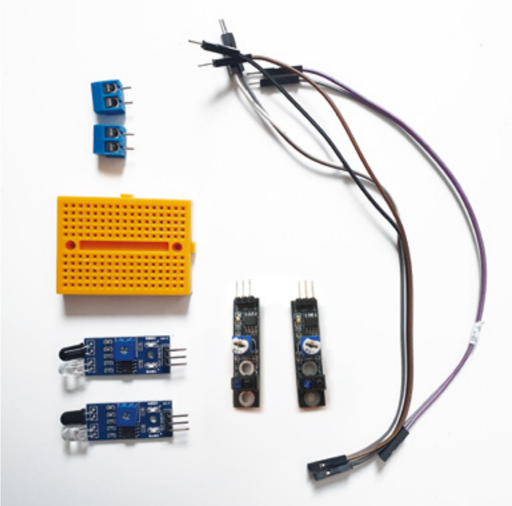
03 Planning and sketching

Test-fit the obstacle sensors on the top of the robot, facing forward and slightly outward.

Sketch the top of the robot showing where to attach the sensors (**Figure 1**). 2.5 mm bolts work for this. Add a 10 mm hole for wires to go through, clear of other features.

Next, test-fit line sensors under the robot, sticking out of the front and about 5 to 7 cm apart.

Finally, sketch the bottom of the robot with line sensors and another wire hole. You may be able to use the threads from the standoffs for Raspberry Pi to hold the line sensors if they are long enough.



The components needed include terminal blocks on a breadboard, obstacle sensors, line sensors, and jumper wires

04 Drilling holes

Take off the top of the robot, disconnecting the power wires from the motor controller. Use safety gear for all drilling.

Depending on the line sensor placement, you may need to detach the robot's Raspberry Pi and castor before drilling holes in the base of the robot.

Using your sketches, measure out and drill the holes for the sensors. For the cable holes, drill small holes and enlarge them. Remove any excess plastic. Replace your Raspberry Pi and castor, but don't bolt the sensors in or reconnect the batteries just yet.

05 Power distribution

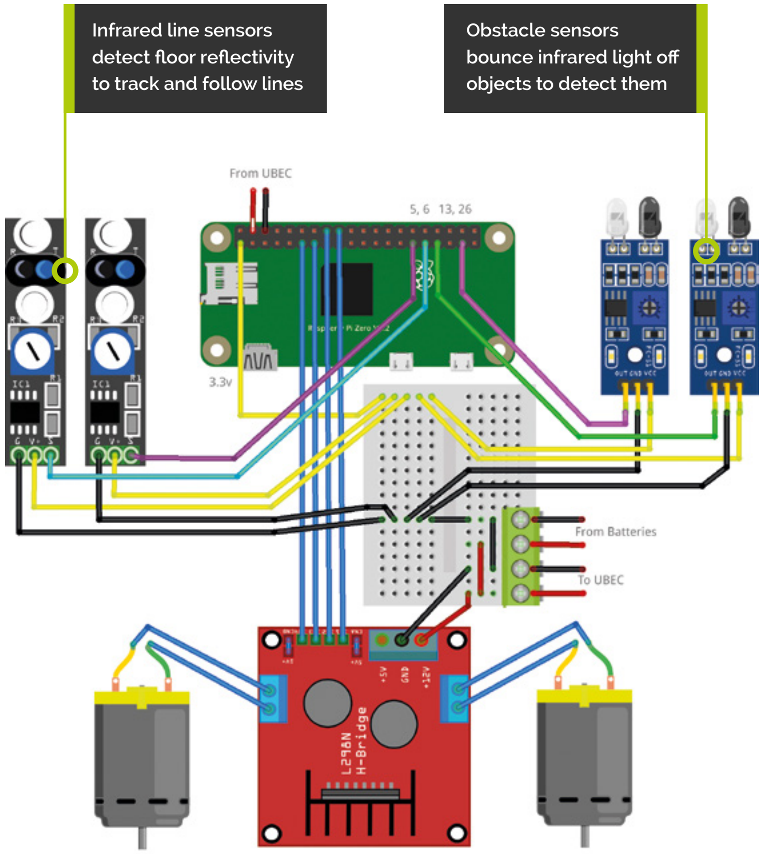
The breadboard gives the sensors access to power rails. 5 mm terminal blocks fit into the breadboard, simplifying connecting the batteries and UBEC. Slot two of the blocks together to make a block of four and pop it into the breadboard, as shown in **Figure 2** (overleaf).

Add breadboard internal wiring connections – pre-cut jumper wires do a tidy job of this. Popping the breadboard lightly into the robot, make the connections to Raspberry Pi 3.3 V and motor board.

The jumper coming from the battery red (positive) wire on the breadboard is a suitable place for an optional power switch.

06 Wiring the sensors

Push a four-way male-female jumper wire through each cable port on the top and bottom for power connections. Use these to wire the sensor VCC/V+ pins to the 3.3 V breadboard row and the GND/G pins to the ground row.



> 5 mm terminal blocks fit into the breadboard, simplifying connecting the batteries and UBEC

Figure 1

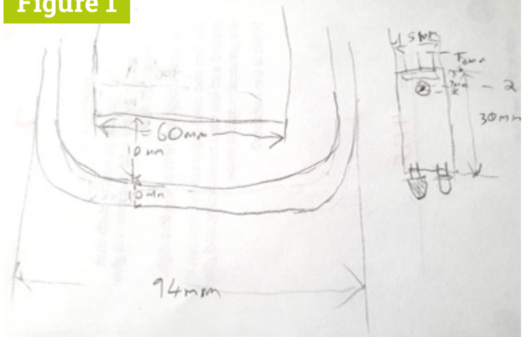
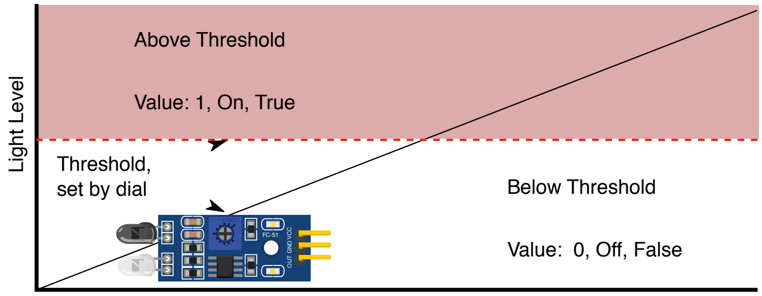


Figure 1 A start on a sketch of the top and the sensor dimensions. Be prepared to sketch a couple of times, adding more features

Warning!
Wear goggles

Please use safety goggles and a desk clamp for drilling.



As the light detected goes up and crosses a threshold set by the dial, the sensor will output a logic 1. Below this it's 0

Sensor digital output pins are labelled D0, DOUT, or S. Wire this pin to a free GPIO pin with a female-female jumper wire. The code uses pins 5 and 6 for the line sensors facing the floor and pins 13 and 26 for the obstacle sensors. Mount the sensors, reassemble the robot, and turn it on.

07 Calibrating the sensors

The sensors may have two LEDs: one is for power and always on, and the other for sensing. Take the robot out of direct sunlight. Ensure nothing is in front of the sensors for a couple of metres.

The sensors have a screwdriver dial on them to adjust the light level they switch on at, which translates to the distance.

obstacle_avoid.py

Language: **Python 3**

DOWNLOAD THE FULL CODE:

magpi.cc/QkkmqW

```
001. from signal import pause
002. import atexit
003. import gpiozero
004. from gpiozero.tools import scaled, negated
005.
006. robot = gpiozero.Robot(left=(27, 17), right=(24, 23))
007. left_obstacle_sensor = gpiozero.DigitalInputDevice(13)
008. right_obstacle_sensor = gpiozero.DigitalInputDevice(26)
009. # Ensure it will stop
010. atexit.register(robot.stop)
011.
012. robot.right_motor.source = scaled(
     left_obstacle_sensor, -1, 1)
013. robot.left_motor.source = scaled(
     right_obstacle_sensor, -1, 1)
014.
015. pause()
```

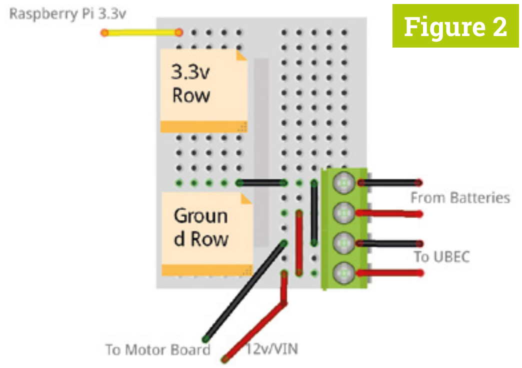


Figure 2 The power board wiring prepares a power rail for 3.3V and a rail for GND to connect the sensors to

Put a reflective obstacle about 10 cm in front of a sensor. Turn the dial slowly to a point the LED turns off. Turn it a tiny way back and it should be on. Wave the obstacle back and forth and watch the LED changing.

08 Sensors in GPIO Zero

GPIO Zero sensors have a value for code to read their state, which a loop could use to set motor values. However, there is a different way to use it.

GPIO Zero has a smart source/value system. A sensor input device can be a 'source' for an output device like a motor, sending a continuous stream of data. So sensor inputs can be virtually wired to affect output, eliminating the loop.

The source/value system has tools to manipulate the data like scaling, negating, or delaying it. Mathematical functions like a sine wave can also be sources. See **magpi.cc/hPmwrv** for more ways to use this.

> A sensor input device can be a 'source' for an output device like a motor, sending continuous data

09 Driving with sensors

The **obstacle_avoid.py** code makes a `DigitalInputDevice` for each sensor.

Line 10 registers `motor.stop` with the 'atexit' system, guaranteeing when the code stops, for any reason, the motors stop too.

Lines 12 and 13 wire the sensors to the opposite motors, so the robot will turn away from any detected object.

The code scales the sensor values 0 (obstacle detected) and 1 (clear) into motor speed values of −1 (reverse) and 1 (go forward).

STRATO PI CM DUO

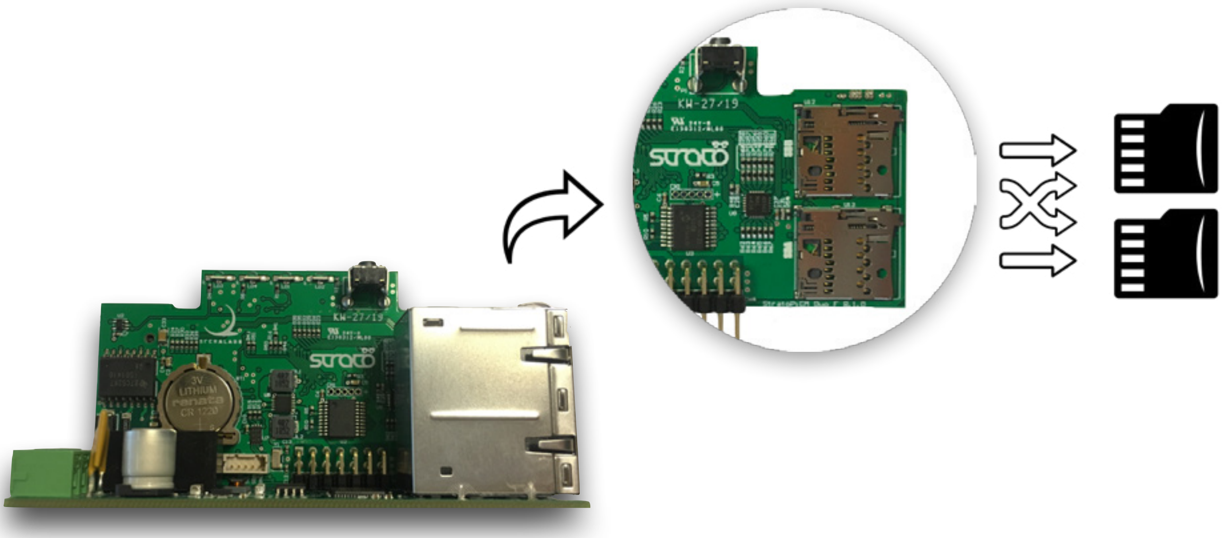
Dual SD card: for the first time on an industrial Raspberry Pi CM based device!

Strato Pi CM Duo is the perfect solution for high-reliability systems - featuring two distinct SD cards with high-speed switching matrix for separate OS/data storage, redundancy and in-field full-system upgrades.

Fault tolerance is further enhanced by:
- Regulated wide-range power supply
- Hardware watchdog with boot SD switch control
- Power-controlled USB ports

Additional features:
- Opto-isolated RS-485 serial interface
- Real-Time Clock with a replaceable backup battery
- Secure element

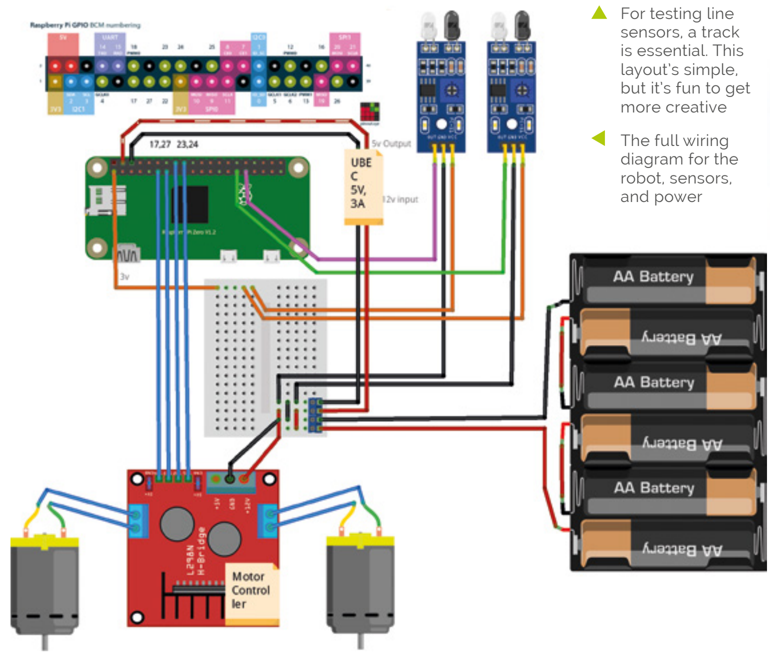
All of this in a compact DIN-rail case!



SFERALABS

www.sferalabs.cc

Run this to see the robot avoid walls and obstacles. The sensors miss obstacles above or below their fields of view, or those too dark and matte to reflect light.



For testing line sensors, a track is essential. This layout's simple, but it's fun to get more creative

The full wiring diagram for the robot, sensors, and power

follow_line.py

Language: **Python 3**

```
001. from signal import pause
002. import atexit
003. import gpiozero
004. from gpiozero.tools import scaled, negated
005.
006. robot = gpiozero.Robot(left=(27, 17), right=(24, 23))
007. left_line_sensor = gpiozero.LineSensor(5)
008. right_line_sensor = gpiozero.LineSensor(6)
009. # Ensure it will stop
010. atexit.register(robot.stop)
011.
012. robot.left_motor.source = scaled(negated(left_line_
     sensor), -0.3, 0.4)
013. robot.right_motor.source = scaled(negated(right_line_
     sensor), -0.3, 0.4)
014.
015. pause()
```

10 Line following

Using sheets of plain white paper (A4 to A1) and black tape (about 20 mm wide), make a single line along the middle of a sheet.

Create a small calibration square of about 40 mm, and put a strip of tape across one end.

Make some curved sections on other paper sheets; keep the turns to less than 45 degrees, and the lines no closer than a robot width apart.

The robot drives forward until the sensor encounters a line. You can use this to make a crossing, by leading tracks to a gap from both horizontal and vertical directions

11 Calibrating line following

Calibrate the line sensors in a similar way to the obstacle sensors. It may be easier to detach the sensor for calibration and reattach it afterwards depending on where the dial is.

Using the calibration square, hold a white area about 2 cm from a line sensor. Slowly turn the sensor dial until the LED changes – wave the paper between black and white to observe the LED changing, and adjust if needed.

You'll need to recalibrate for different lighting or surface conditions.

12 Line following code

The **follow_line.py** code is similar to obstacle avoiding.

Lines 7 and 8 set up GPIO Zero line sensors (based on digital input) on the correct pins.

The line sensors straddle the track. When the sensor detects white (sending 0), it's not crossing the line and so the motor goes forward.

Lines 12 and 13 connect the sensor output so a motor reverses when its sensor crossed the line onto black tape (sending 1). The input source is negated to make 0 the move forward condition.

So that the robot responds to the track before driving past it, scale the source data to go from −0.3 to 0.4.

Part 01

Hack GraviTrax with Raspberry Pi

Make spectacular marble runs by triggering sounds and animations from a Raspberry Pi

MAKER

Mike Cook

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

magpi.cc/TPaUfT

GraviTrax is a construction system to make your own marble runs of any complexity. From the famous toy maker Ravensburger, it is aimed at the STEM market as well as kids of all ages. It has lots of extension packs for more basic parts and new fun features. There is only one thing missing, until now: an interface to Raspberry Pi.

We have always wanted to do a marble run project, so when we saw the GraviTrax system we were excited because it had all the ingredients we needed. Also, it is built up on a grid of hexagon spacing, and we are suckers for hexagons. The standardised parts, and wide availability, meant we could make our own parts, or modify existing parts, to feed back events into the Raspberry Pi and have them trigger sounds or animations. We can also trigger lights to make our run more exciting. In this first part, we will concentrate on detecting the balls.

01 The GraviTrax system

When you get your starter set, you have to prepare the cardboard base by pushing out hexagons to leave holes to mount the tiles in. Do not discard these hexagons, because we are going to use them in our project. If you have already discarded them, then you'll need to cut out a hexagon from either cardboard of the same thickness or 3 mm plywood. We made two types of sensor: a standalone one, and ones where you have to modify an existing part. Just like the GraviTrax system, our approach is modular and you can make as many of each sensor type as you like.

02 How do they work?

All the sensors in this part use light to detect the presence of a ball – through breaking a beam, detecting reflected light, or simply the presence or absence of ambient light. As such, the circuits are all very similar as shown in **Figure 1**; the only difference is the physical sensor used, and the current limiting resistor value on the LED. They are connected to the Raspberry Pi by a length of three-wires-wide ribbon cable, with a pull-up resistor going on the board that connects to the GPIO pins. In later parts of this tutorial series, we will look at making a distribution board for them all.

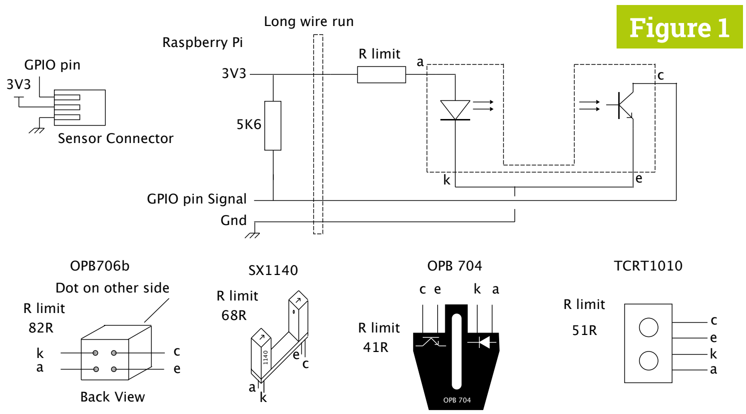


Figure 1 A universal schematic of the optical sensors

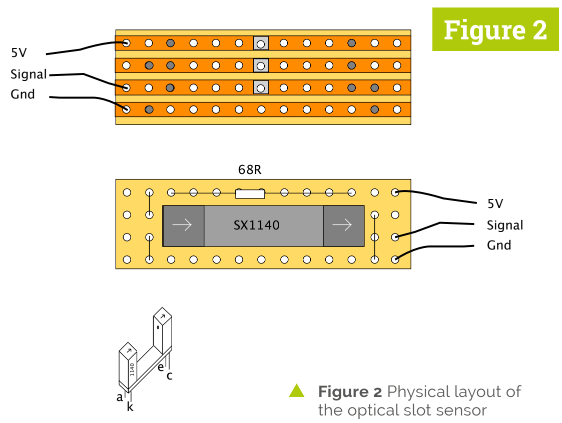
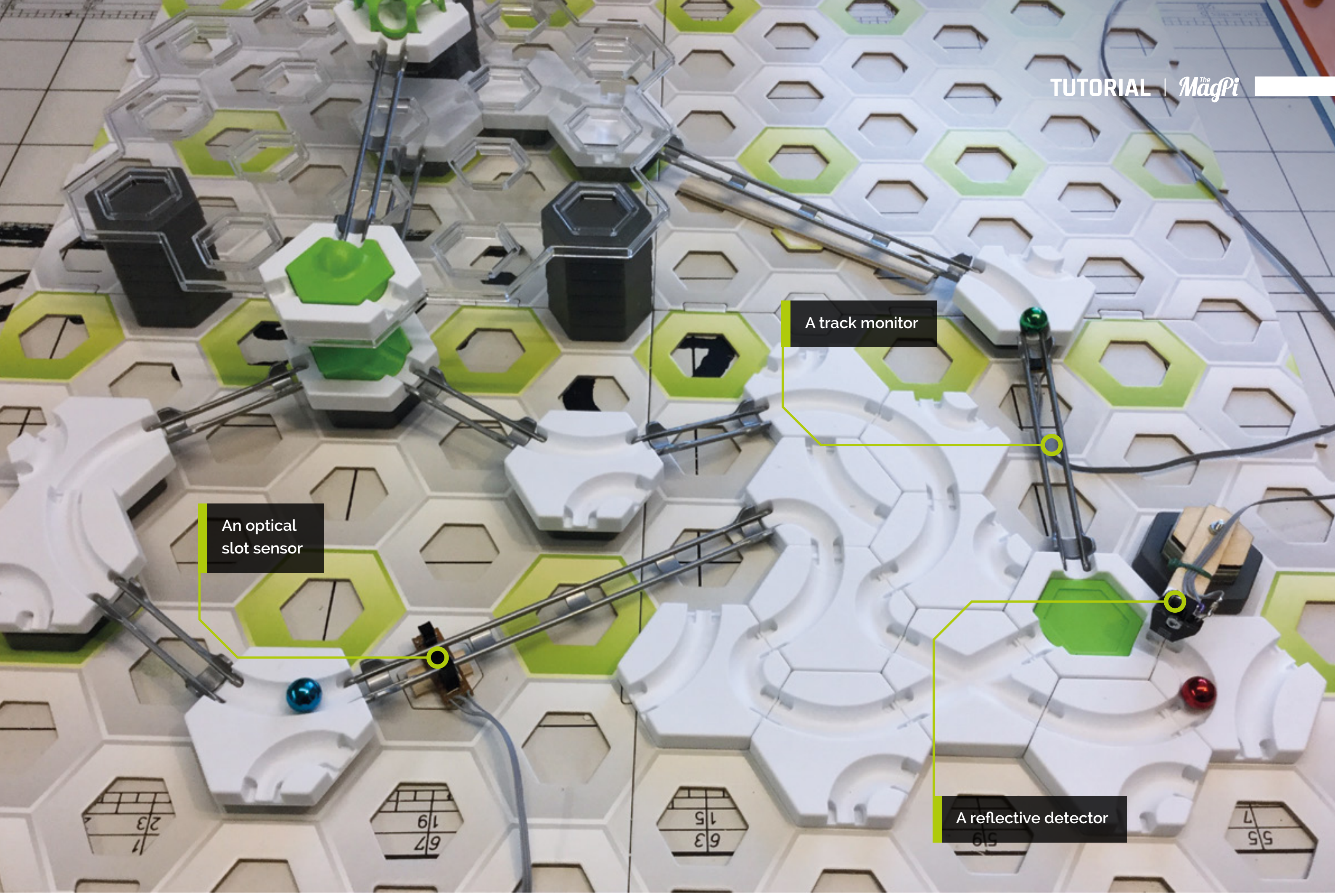
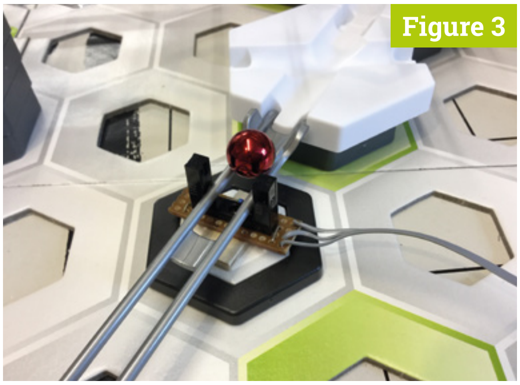
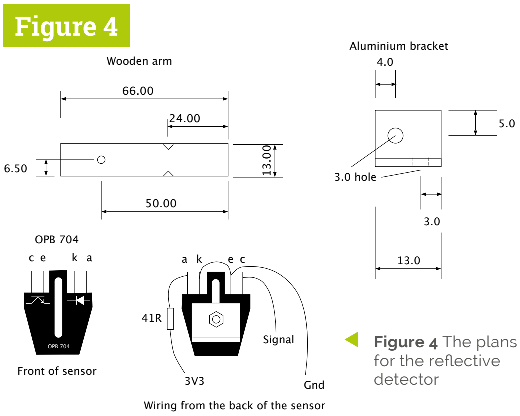


Figure 2 Physical layout of the optical slot sensor





▲ **Figure 3** The optical slot sensor in action



◀ **Figure 4** The plans for the reflective detector

03 Optical slot sensor

The first sensor we will look at is an optical slot sensor. This fits under a track and the height can be adjusted by using the small or large height tiles. Basically, this is a cardboard hexagon with a small piece of 3 mm plywood stuck on it and the electronics glued to that, going from point to opposite point on the hexagon. The physical construction of the electronics board is shown in **Figure 2**, with the whole assembly shown in **Figure 3**. It is best to glue the electronics in place with a track running through it, between two pieces, so you can get it aligned precisely.

04 Reflective detector

The previous sensor needs to be placed under a track, whereas this one can be placed in an adjacent space and it looks downwards. It is capable of detecting balls on the entrance or exit to any of the tiles, be it basic, curved, or launch pad. The sensor is easy to make, as the LED and transistor symbols are drawn on the part. We used a stack of five cardboard hexagons glued together, followed by a plywood hexagon with a hole in the centre and a 10 mm, M3 countersunk screw sticking through it. This was then placed on the arm, and could be rotated to the required point.

Top Tip

GraviTrax simulator

You can get a free GraviTrax simulator to run on your mobile device. You can make layouts and see the results with a variety of effects, even from a 'balls eye' view of the run.

You'll Need

- GraviTrax – Starter set **magpi.cc/LusGgA**
- ESSX1140 opto slot **magpi.cc/nQNQNW**
- OPB704 Reflective opto sensor **magpi.cc/fwUrdD**

Top Tip

Track monitor

When gluing the PCB to the track, glue the long edge first, then push the sensor slightly so it is not parallel to the track, and test that it detects the ball before gluing the sides to hold it into position.

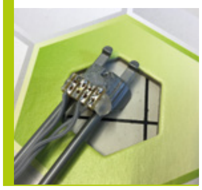


Figure 5

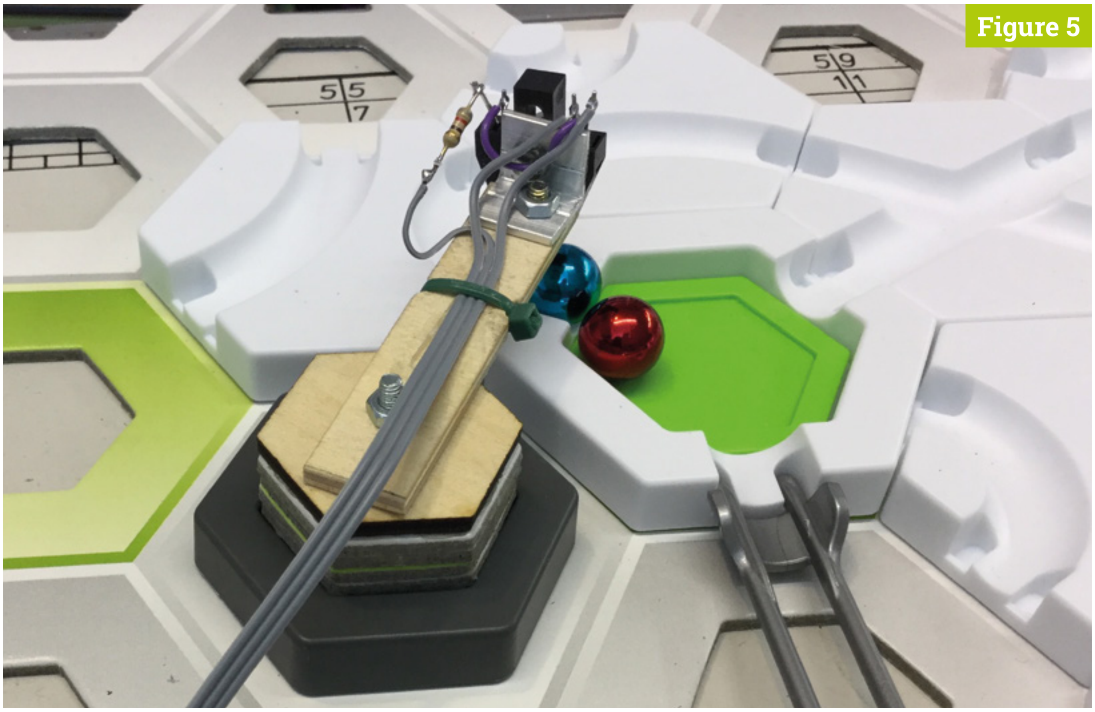


Figure 5 The reflective detector in action

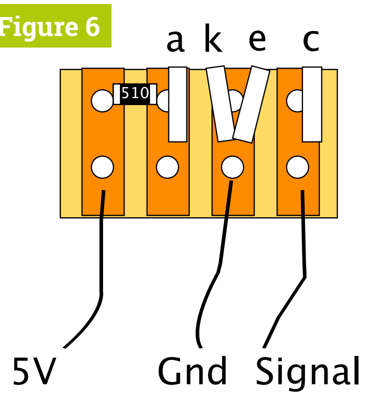
05 Making the reflective detector

Figure 4 shows the measurements of the plywood arm, and the mounting bracket made from a piece of 12×12 mm angle aluminium. The arm should be notched with the corner of a square file; this is so a cable tie can grip the ribbon cable without slipping. The height of the sensor is adjustable due to a long slot in it, allowing it to be slid up and down. You need a gap of about 2 mm between the sensor and the top of a ball to detect it reliably (**Figure 5**). You can mount two arms on a stack to get coverage of another position.

06 A track monitor

The last two sensors have been standalone; the next one modifies a GraviTrax part. The physical layout is shown in **Figure 6**. The TCRT1010 sensor comes with bent leads which neatly wrap round two holes of stripboard. On the strip side, you need to bend the connectors over again and cut them short. Then bend the middle two so they sit on one track and solder them up. You also need a surface-mount 51Ω resistor to make it small, although a 025 W through-hole resistor could be used. The assembly needs hot-melt gluing onto the track (**Figure 7**).

Figure 6



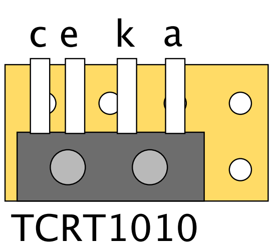


Figure 6 Physical layout of a track monitor

07 Launch pad monitor

Take an OPB706B sensor and wrap it round the centre post in the launch pad tile (**Figure 8**). Push it down and use a pencil to mark the outline on the wall. Then, using a Dremel, and 1 mm router bit, cut to about 1 mm short of the outline you drew, so the part does not go through the hole. Also, cut a slot in the opposite side to let the wires through when attached to a height tile. Paint black the area the sensor is pointing at, and glue a 10×2 mm piece of 1 mm thick styrene to the green plunger with polystyrene glue (**Figure 9**).

08 Switch monitor

This uses ambient light to detect which way the switch is set. Draw in pencil around the switching lever in both positions, showing the area being covered and uncovered. Then drill a 2 mm hole in the middle of this area (**Figure 10**). Next, paint

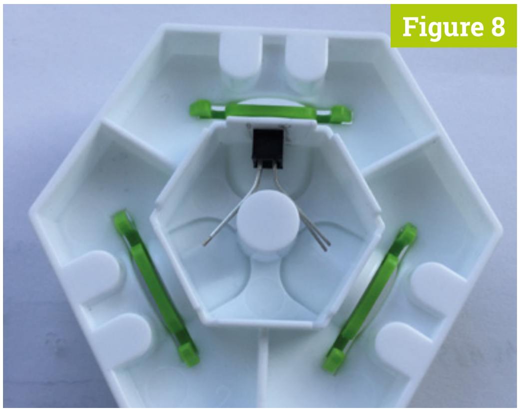


Figure 7 The track monitor in action

Figure 8 Marking the hole outline on the launch pad

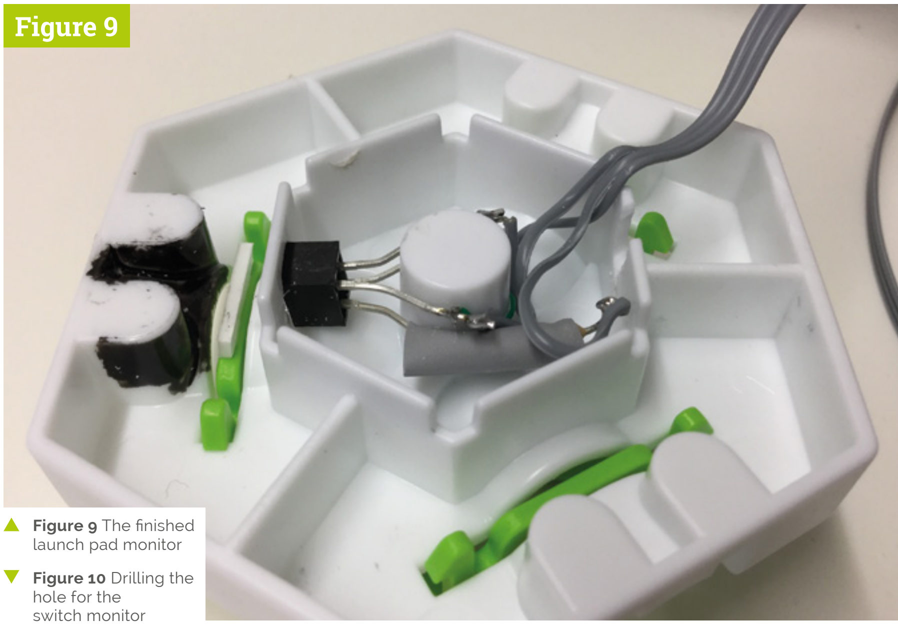


Figure 9 The finished launch pad monitor

Figure 10 Drilling the hole for the switch monitor

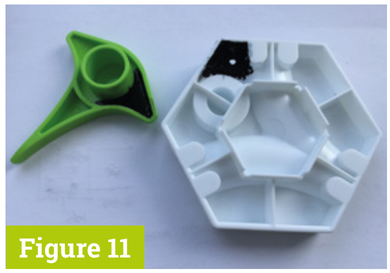
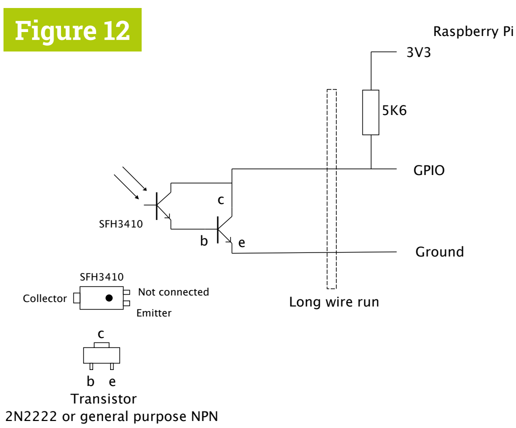


Figure 11 Painting parts of the switch tile black

Figure 12 Schematic of the switch monitor



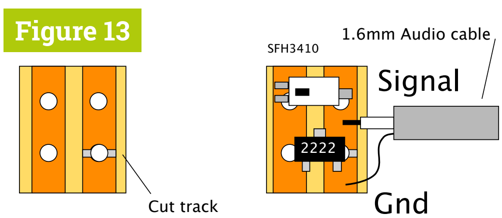


Figure 13 Physical layout of the switch monitor

the underside of the switch tile black, and make sure you paint on the inside of the hole (**Figure 11**). The schematic of this sensor is somewhat different from the others and is shown in **Figure 12**. Finally, **Figure 13** shows the physical layout of the parts. Position the board so you can see the white sensor through the hole and fix with Sugru.

> "The GPIO pins are scanned and when a trigger condition is met"

09 Software

We have written software, shown in the **sound_trigger.py** listing, that monitors these sensors, and triggers sounds, either immediately or after a delay. The software is modular: line 82 determines what GPIO pins you will use and it automatically generates a window size to accommodate the number of pins in this list. Note if you want to use GPIO 14, you should disable SPI before booting with the sensor attached. The sounds' names are in a list at line 88; by simply changing or adding to this list, different sounds can be used. The GPIO pins are scanned and when a trigger condition is met, the event is put in a pending list to be actioned at the correct time.

Figure 14 The software user interface

10 Using the software

The user interface is shown in **Figure 14**. For each line, you can set what the trigger action will be. These states are: disabled, when the signal goes high, when it goes low, or when it goes either high or low. They are changed by clicking on the trigger icon. The delay column, as you might expect, determines a delay between the trigger and the sound, whereas the sound sample played can be changed by the icons on the right. You can change the GPIO pin and note that one pin can trigger different actions; so, for example, you could have the switch tile generating a different sound depending if it is changed to the left or right.

11 Choosing sounds

We found short sounds were generally best, but longer sounds can be useful at the beginning or end of your run. We copied a lot of sounds from the Scratch media library from the path **/usr/share/scratch/Media/Sounds** into our **sounds** directory. Make sure that are all .wav files, because that suffix gets added automatically to the file names. Note that the slot sensor will read as a logic zero with no ball, whereas a reflective sensor will read high in the absence of a ball. There are lots of suitable sounds available online as well.

We have looked at adding optical sensors to detect where a ball is, be it on a track or a tile. Next month we will look at how to add different sorts of LED displays to enhance your GraviTrax layout. In the meantime, if GraviTrax is new to you, then have a play with the different layouts in the accompanying booklet.

sound_trigger.py

Language: **Python 3**

```python
#!/usr/bin/env python3
# GraviTrax Sound Trigger
# By Mike Cook September 2019

import time
import pygame
import os
import RPi.GPIO as io

pygame.init()
pygame.display.set_caption("GraviTrax Sound Trigger")
os.environ['SDL_VIDEO_WINDOW_POS'] = 'center'
pygame.mixer.quit()
pygame.mixer.init(frequency = 22050, size =- 16, channels = 2, buffer = 512)
pygame.event.set_allowed(None)
pygame.event.set_allowed([pygame.KEYDOWN, pygame.QUIT, pygame.MOUSEBUTTONDOWN,
                          pygame.MOUSEBUTTONUP]
                         )
textHeight=18
font = pygame.font.Font(None, textHeight)
backCol = (160, 160, 160) ; lineCol = (128, 128, 0)
hiCol = (0, 255, 255)

def main():
    global screen, lastIn, rows
    initIO()
    rows = len(inPins)
    screen = pygame.display.set_mode([390, 34 + 40*rows], 0, 32)
    init() ; pendPlay = [0]*rows
    nowIn = [0]*rows; pendTime = [0.0]*rows
    drawScreen()
    while True:  # repeat forever
        checkForEvent()
        for i in range(0, rows):
            nowIn[i] = io.input(inPins[inPin[i]])
            if lastIn[i] != nowIn[i]:
                lastIn[i] = nowIn[i]
                tmatch = trigNum[i]-1  # match trigger
                if tmatch == 2:
                    tmatch = nowIn[i]
                if trigNum[i] != 0 and nowIn[i] == tmatch:
                    pendPlay[i] = soundFX[soundNumber[i]]
                    pendTime[i] = time.time() + delayTime[i]
        for i in range(0, rows):  # check what to play now
            if pendTime[i] > 0.0 and time.time()>=pendTime[i]:
                pendPlay[i].play() ; pendTime[i] = 0.0
```

DOWNLOAD THE FULL CODE:

magpi.cc/dhaAam

```

def init():
    global incRect, decRect, icon, decRect, voiceRect
    global inPin, soundNumber, delayTime, triggerRect
    global lastIn, trigNum, trigIcon
    lastIn = [0]*rows
    loadResources()
    icon=[pygame.image.load("icons/"+str(i)+".png").convert_alpha()
                                for i in range(0,2)
          ]
    incRect = [pygame.Rect((0,0),(15,15))]*rows*3
    decRect = [pygame.Rect((0,0),(15,15))]*rows*3
    for j in range(0,3):
        for i in range(0, rows):
            incRect[i+j*rows] = pygame.Rect((76 + j*80, 30 + i*40),(15, 15))
            decRect[i+j*rows] = pygame.Rect((76 + j*80, 50 + i*40),(15, 15))
    triggerRect = [pygame.Rect((0, 0), (20, 20))]*rows
    trigNum = [0]*rows
    trigIcon = [pygame.image.load("icons/trig"+str(i)+".png").convert_alpha()
                                      for i in range(0,4)
               ]
    voiceRect = [pygame.Rect((0,0), (15,15))]*rows
    for i in range(0, rows):
        triggerRect[i] = pygame.Rect((10, 36 + 40*i,20, 20))
        voiceRect[i] = pygame.Rect((268, 39 + i*40),(100, 20))
    sounds = rows + len(soundNames)
    inPin = [1]*rows ; soundNumber = [0]*sounds
    for i in range(0, rows):
        inPin[i] = i
    for i in range(0, len(soundNames)):
        soundNumber[i] = i
    delayTime = [0.0]*rows

def initIO():
    global inPins
    inPins = [24, 23, 22, 27, 17, 4, 15, 14]
    io.setmode(io.BCM); io.setwarnings(False)
    io.setup(inPins, io.IN, pull_up_down = io.PUD_UP)

def loadResources():
    global soundFX, soundNames
    soundNames = ["owl", "Breaking Glass", "ComputerBeeps1",
                 "CymbalCrash", "Fairydust", "Dog1", "Zoop", "Ya", "Pop"
                  ]
    soundFX = [pygame.mixer.Sound("sounds/"+ soundNames[effect]+".wav")
                                  for effect in range(0,len(soundNames))
              ]

def drawScreen():
    screen.fill(backCol)
    for i in range(0,len(incRect)):  # increment / decrement icons
        screen.blit(icon[0], (incRect[i].left,incRect[i].top))
        pygame.draw.rect(screen, lineCol, incRect[i],1)
        screen.blit(icon[1], (decRect[i].left, decRect[i].top))
        pygame.draw.rect(screen, lineCol, decRect[i], 1)
    for i in range(0,rows):  # draw all triggers
        screen.blit(trigIcon[trigNum[i]], (triggerRect[i].left,
                                    triggerRect[i].top)
                    )
    drawWords("Trigger", 5, 8, (0, 0, 0), backCol)
    drawWords("GPIO", 70, 8, (0, 0, 0), backCol)
    drawWords("Delay", 138, 8, (0, 0, 0), backCol)
    drawWords("Sound", 218, 8, (0, 0, 0), backCol)
    updateValues()

def updateValues():
    for i in range(0,rows):
        drawWords(str(inPins[inPin[i]]) + "    ", 48, 39 + i*40, (0, 0, 0),
                  backCol
                  )
        drawWords("   " + str(round(delayTime[i], 1)) + "   ", 112, 39 + i*40,
                  (0, 0, 0), backCol
                  )
        pygame.draw.rect(screen, backCol, voiceRect[i], 0)
        drawWords(str(soundNames[soundNumber[i]]), 270, 39 + i*40, (0, 0, 0),
                  backCol
                  )
    pygame.display.update()

```

sound_trigger.py (continued)

▸ Language: **Python 3**

```python
def drawWords(words, x, y, col, backCol) :
    textSurface = font.render(words, True, col, backCol)
    textRect = textSurface.get_rect()
    textRect.left = x  # right for align right
    textRect.top = y
    screen.blit(textSurface, textRect)
    return textRect

def handleMouse(pos):  # look at mouse down
    global pramClick, pramInc, trigClick
    #print(pos)
    trigClick = -1
    for i in range(0, rows):
        if triggerRect[i].collidepoint(pos) :
            trigClick = i
            pygame.draw.rect(screen, hiCol, triggerRect[i], 0)
            pygame.display.update()
    pramClick = -1
    pramInc = 0
    for i in range(0, len(incRect)):
        if incRect[i].collidepoint(pos):
            pramClick = i ; pramInc = 1
            pygame.draw.rect(screen, hiCol, incRect[pramClick], 1)
            pygame.display.update()
    for i in range(0, len(decRect)):
        if decRect[i].collidepoint(pos):
            pramClick = i ; pramInc = -1
            pygame.draw.rect(screen, hiCol, decRect[pramClick], 1)
            pygame.display.update()

def handleMouseUp(pos):  # look at mouse up
    global soundNumber, delayTime, inPin
    if trigClick != -1:
        trigNum[trigClick] += 1
        if trigNum[trigClick] > 3:
            trigNum[trigClick] = 0
        pygame.draw.rect(screen, backCol, triggerRect[trigClick], 0)
        screen.blit(trigIcon[trigNum[trigClick]], (triggerRect[trigClick].left,
            triggerRect[trigClick].top))
        updateValues()
    if pramClick != -1:
        if pramClick < rows:  # GPIO Coloumn
            inPin[pramClick] += pramInc
            inPin[pramClick] = constrain(inPin[pramClick], 0, rows-1)
        elif pramClick < rows*2:  # Delay Coloumn
            delayTime[pramClick-rows] += (pramInc / 10)
            delayTime[pramClick-rows] = constrain(delayTime[pramClick - rows],
                                                  0, 5
                                                  )
            if delayTime[pramClick - rows] < 0.01:
                delayTime[pramClick - rows] = 0
        elif pramClick < rows*3:  # Sound coloum
            soundNumber[pramClick - rows*2] += pramInc
            soundNumber[pramClick - rows*2] = constrain(soundNumber[pramClick
                - rows*2], 0, len(soundNames)-1)
        if pramInc !=0:
            if pramInc < 0:
                screen.blit(icon[1], (decRect[pramClick].left,
                    decRect[pramClick].top))
                pygame.draw.rect(screen, lineCol, decRect[pramClick],1)
            else:
                screen.blit(icon[0], (incRect[pramClick].left,
                    incRect[pramClick].top))
                pygame.draw.rect(screen, lineCol, incRect[pramClick], 1)
            updateValues()

def constrain(val, min_val, max_val):
    return min(max_val, max(min_val, val))

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

def checkForEvent():  # see if we need to quit
    event = pygame.event.poll()
    if event.type == pygame.QUIT :
        terminate()
    if event.type == pygame.KEYDOWN :
        if event.key == pygame.K_ESCAPE :
            terminate()
    if event.type == pygame.MOUSEBUTTONDOWN :
        handleMouse(pygame.mouse.get_pos())
    if event.type == pygame.MOUSEBUTTONUP :
        handleMouseUp(pygame.mouse.get_pos())

if __name__ == '__main__':
    main()
```

Part 05

Design retro game maps and levels in PICO-8

Take your game design to the next level by learning to use PICO-8's map editor. Draw terrain and build your first level!

MAKER

Dan Lambton-Howard

Dan is an independent game designer based in Newcastle upon Tyne, where he is lucky enough to make games for his PhD.

@danhowardgames

You'll Need

- PICO-8 **magpi.cc/pico8**
- Raspberry Pi
- Keyboard and mouse

In this tutorial we will be turning our simple one-screen space shooter into a scrolling shoot-'em-up! You'll learn how to use PICO-8's handy map editor to quickly and easily draw out levels, and how to use the sprite editor to create terrain tiles. We'll talk about using sprite flags to distinguish between background and foreground and how to spawn enemies. Speaking of which, we'll also talk about level design basics and introduce a new turret enemy type to add extra spice and challenge to your game. There's lots to get through, so let's get started!

01 A blank canvas

Much like every other aspect of game development, PICO-8 has a quick and easy solution for designing levels. Switch to the map editor by selecting it from the editor menu at the top right. At first glance, it looks a lot like the sprite editor, with the same sprite sheets and drawing tools at the bottom of the screen. The difference is that, instead of plotting coloured pixels, the map editor paints with our finished sprites. Try this out by selecting a sprite and drawing on the canvas above.

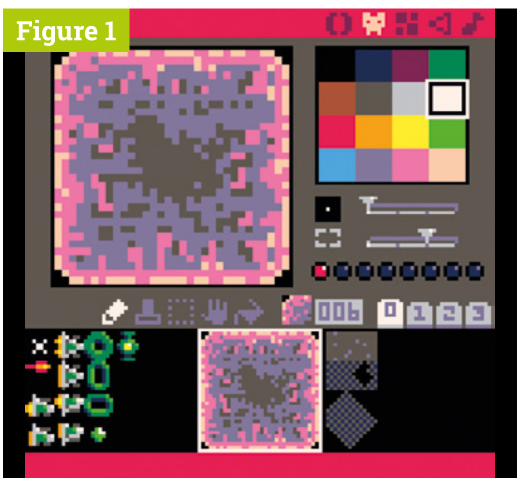


Figure 1 A big block of terrain tiles makes a good palette with which to paint your levels. Distinguish between background and foreground with sprite flags

02 Chunks of dirt

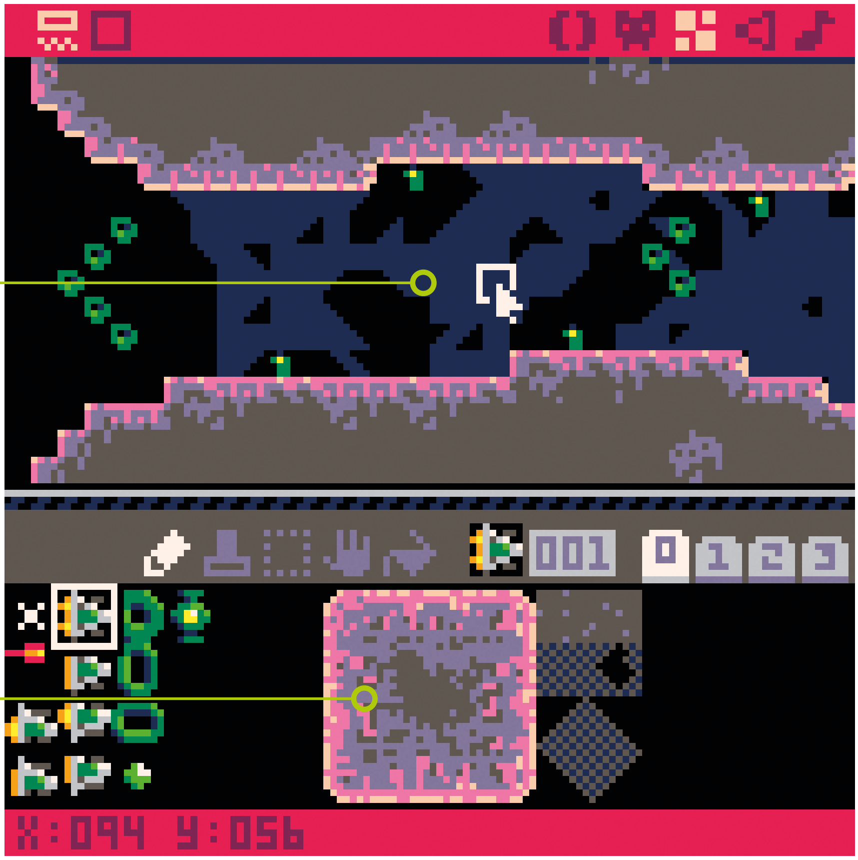
We can't build a level out of enemy and player sprites – that would be sheer insanity! PICO-8's map editor is grid based, so we'll need to create some new terrain sprites that we put together as tiles. **Figure 1** shows a 3×3 square of sprites that can be tiled easily, with a couple of variations along the side. Switch to the sprite editor and create something similar. We've chosen suitably weird-looking purple asteroids for our terrain, and we've also created simple background sprites out of a chequer-board 'dither' pattern that we can use to imply depth.

03 Tiles for miles

Now we have our raw level-making material, let's start working with it. Switch back to the map editor. You can zoom the canvas with the mouse wheel and pan with the pan tool. Hit the **SPACE** bar to view gridlines, and you'll see that your canvas is 128×64 tiles, with grid reference (0, 0) being the top-left tile. As PICO-8's screen resolution is 128×128 pixels, and each tile is 8×8

Draw your level in PICO-8's map editor for enemy and terrain placement

Create terrain tiles that can be combined for lots of possibilities



Top Tip

Sprite flags

Sprite flags are extraordinarily useful for lots of things, such as distinguishing between drawing layers or marking objects that collide.

pixels, a single screen in PICO-8 is 16×16 tiles. Use your terrain sprites to draw some asteroids in the top-left 16×16 tiles of the canvas.

04 Mapping it all out

Let's see what this looks like in game. First of all, comment out the enemy wave code, so that we can explore our level without being rudely interrupted by space blobs. You can use `--[[..]]` for block comments. Next, add `map(0,0,0,0,128,64)` to `_draw()` just after where we draw the background stars. This function tells PICO-8 to draw a 128×64 block of tiles starting from tile reference (0,0) on the map to coordinates (0,0) on the screen. Run your game and you should now see your asteroids. Great work, but it's all rather static – let's get this level scrolling!

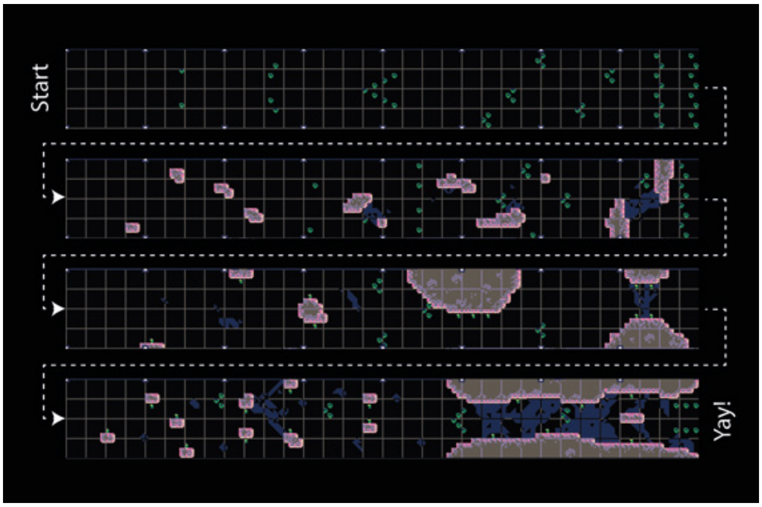
05 Look into the camera

To turn our game into a scrolling shoot-'em-up, we will need to use a scrolling camera. Declare new variables `camx,camy = 0,0` in `_init()` for the camera's coordinates. Next, add `camx+=1` to the start of `_draw()`, followed by `camera(camx,camy)` which sets the top left of PICO-8's built-in camera to these coordinates. We've modified our player, laser, and draw background, score, and game-over message code to be locked to the new camera coordinates. As the map is only eight screens long, we've also written a cheeky bit of code to move the camera and player to the next row on the map when it reaches the end.

> Modifying the code is mainly a matter of changing boundaries to be set to camx and camy instead of arbitrary values

06 A red flag

Modifying the code is mainly a matter of changing boundaries to be set to `camx` and `camy` instead of arbitrary values; we'll also add `player.x+=1` to `_update()` so that the player scrolls with the camera. See the code listing for more details. You'll have probably noticed that we can fly straight through the terrain unimpeded, so let's add terrain collision detection. We'll use sprite flags to do this. Set the sprite flags (those radial buttons above the sprite sheet tabs in the sprite editor) of each of your terrain tiles so that flag 0 is on. It should light up red.



Our level introduces new elements one by one and increases in difficulty at the end of each segment

07 Deep impacts

Sprite flags are a way of 'marking' sprites. In this case, we will treat any sprite with flag 0 as solid terrain that our player can crash into. To actually detect the collision, we'll create a new function `player_terrain_collision()` which will check four points of a square around the player's coordinates, retrieve whatever sprite is there, and return `true` if that sprite has flag 0 activated. Then we'll add few lines in our update loop that'll call that new function and kill the player if it returns true. We nearly have everything in place!

> It's a good idea to start simple and gradually increase the challenge

08 Enemy placement

Next, we want to slightly modify our enemy code so that instead of spawning in endless waves, we can place them in our level and they will attack when they appear on camera. See the code listing for the changes. To place enemies in the level, we will use one of our existing enemy sprites in the map editor. Then we will add a few lines to `_init()` that will check every map tile for enemy sprites and spawn enemies when it finds them – simple! Now that we can place terrain and enemies, we can begin the level design proper.

09 Flow state

Level design is as much an art as it is a science. For every rule of good level design, there are a hundred examples to prove it wrong. That being said, for your first few levels there are certainly some guiding principles you can follow. It's a good idea to start simple and gradually increase the challenge as your players become better at the game. This is to keep players in a satisfying state of 'flow' where a level is not too easy as to be boring, or too hard to be frustrating.

10 Difficulty curve

In our space shooter, difficulty is determined by the number and location of enemies and the placement of terrain. Modifying these factors allows us to control the challenge and ideally create a smooth 'difficulty curve'. In our level, enemies are introduced singularly at first, then in increasing numbers. Terrain is then introduced, then enemies and terrain, and lastly challenging combinations of both. You can see how new elements are introduced one at a time and in situations that allow the player to learn their behaviour before the difficulty is increased.

11 Reinforcements

Variety is the spice of life and although our green blobs from space have a certain appeal, it is the introduction of new elements, or new combinations, that keeps a level entertaining. That's why we've created a new enemy type, the turret. You can see the code, but essentially it is a malignant mutant that fires a mucus projectile at the player every few seconds. How delightful! This gives us more possibilities for interesting combinations with the other elements in our game; for example, turrets in an asteroid field or amongst waves of enemies.

12 A happy ever after?

So, your player has defeated every wave of enemy, dodged every asteroid, and made it to the end of your level. What now? Well, the polite thing to do would be to reward them in some way, or give them one final gigantic boss battle. Either way, we will need a congratulations message to tell the player that they are the saviours of mankind. As a final touch, we've added a message that will show when the player makes it all the way to the end. Well done space fighters, the galactic federation thanks you!

Top Tip

Top tile tips

Creating good-looking terrain tiles that work well with each other isn't easy; thankfully, a quick web search will reveal hundreds of examples to learn from.

part5code.p8

DOWNLOAD THE FULL CODE:

magpi.cc/eDDppk

Language: **Lua**

```
--new code reference
--***in _init()***
camx,camy = 0,0--camera coordinates
--check each map tile and spawn enemies
for mapx=0,127 do
    for mapy=0,63 do
        local sprite = mget(mapx,mapy) --get sprite
        if sprite==2 then
            create_enemy(mapx*8,mapy*8)
        --set map tile to blank so we dont draw it as background
            mset(mapx,mapy,0)
        end
        if sprite==3 then --turret
            create_turret(mapx*8,mapy*8)
            mset(mapx,mapy,0)
        end
    end
end
--***in _update()***
--make player scroll with screen
        player.x+=1
--change player boundary locking to camera
        player.x=mid(camx,player.x,camx+120)
        player.y=mid(camy,player.y,camy+120)
--kill player if hits terrain
if player_terrain_collision() then
    gameover = true
    create_explosion(player.x,player.y,20)
    sfx(1)
    music(-1)
end
--change to laser boundary
laser.x+=5 --speed up laser
if laser.x>camx+130 then --delete laser if off camera
    del(lasers,laser)
end
--change so enemies only attack when on camera
for enemy in all(enemies) do --enemy update loop
    if enemy.name=='blob' then enemy.x-=1 end
    if enemy.x<camx+130 and enemy.y<camy+128 then
        --[[rest of code here]]
        if enemy.name=='turret' then
            if enemy.animtimer%60==0 then
                create_mucus(enemy.x,enemy.y)
            end
        end
        if enemy.name=='mucus' then
            enemy.x+=enemy.vx
            enemy.y+=enemy.vy
        end
        if enemy.x<camx-8 then
            del(enemies,enemy)
        end
    end
end
--***in _draw()***
if not gameover then camx+=1 end --increment camera x value
if camx>=1024 then --if cam off right edge of map
    camy+=128 --move down one row
    camx=-128 --move to left of map
    player.x-=1152 --full width of map +1 screen(1024+128)
    player.y+=128--down one screen
end

camera(camx,camy)--set camera
map(0,0,0,0,128,64)--draw map onto screen
--update to background to lock to camera
rectfill(camx,camy,camx+128,camy+128,1)
for star in all(stars)do
    star.x -= star.speed
    pset(star.x+camx,star.y+camy,7)
    if star.x < 0 then
        star.x = 130
        star.y=rnd(128)
    end
end
--update to only animate blob enemy
if enemy.name=='blob' then
    enemy.sprite = enemy.sprites[flr--[[etc]]]
end
--update to gameover text to lock to camera
if gameover then
    print('game over',camx+50,camy+64,7)
end
--show score on screen
print('score: '..score,camx+2,camy+2,7)
--***other functions***
function player_terrain_collision()
    for newx=0,6,6 do --nested for loops generate 4 points
        for newy=0,7,7 do
            --divide by 8 to convert pixels to map coords
            local sprite = mget((player.x+newx)/8,(player.y+newy)/8)
            if fget(sprite,0) then return true end
        end
    end
    return false
end
--new turret enemy
function create_turret(x,y)
    local turret={x=x,y=y,sprite=3,name='turret'}
    turret.animtimer=0
    add(enemies,turret)
end
--new mucus projectile
function create_mucus(x,y)
    --maths that sets mucus velocity towards player
    local angle=atan2(player.x+32-x,player.y-y)
    local velocityx=cos(angle)*1.5
    local velocityy=sin(angle)*1.5
    local mucus={x=x,y=y,vx=velocityx,vy=velocityy,sprite=50}
    mucus.name='mucus'
    mucus.animtimer=0
    add(enemies,mucus)
end
--add name to blob enemy
function create_enemy(x,y)
    enemy.name='blob'
end
```

Part 07

Making menus with C and GTK

Create menu bars with drop-down menus, and also pop-up menus

MAKER

Simon Long

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

rpf.io

Many applications have a menu bar at the top of the main window. GTK provides a number of widgets which can be used to create either menu bars or pop-up menus.

The building block of menus is the GtkMenuItem widget. Each entry in a menu is a GtkMenuItem, which has a text label associated with it. A GtkMenu widget is used to hold one or more GtkMenuItem widgets, creating a single menu of the sort seen as a pop-up or when an item on an application menu bar is selected.

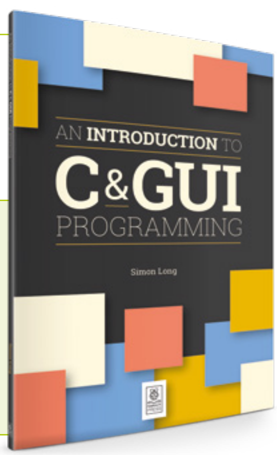
Menu bars

A GtkMenuBar can be displayed at the top of an application's window; this contains a number of GtkMenuItems, each of which provides the name for a GtkMenu, as described above.

It can be slightly confusing to consider that a GtkMenuItem is both a member of a menu and the name of the entire menu, but hopefully an example will make things a bit clearer. Here's the code for an application with a menu bar:

An Introduction to C & GUI Programming

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




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

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

  GtkWidget *mbar = gtk_menu_bar_new ();
  GtkWidget *vbox = gtk_vbox_new (FALSE, 5);
  gtk_box_pack_start (GTK_BOX (vbox), mbar,
TRUE, TRUE, 0);
  gtk_container_add (GTK_CONTAINER (win),
vbox);
  GtkWidget *file_mi =
gtk_menu_item_new_with_label ("File");
  gtk_menu_shell_append (
GTK_MENU_SHELL (mbar), file_mi);

  GtkWidget *f_menu = gtk_menu_new ();
  gtk_menu_item_set_submenu (GTK_MENU_ITEM (
file_mi), f_menu);

  GtkWidget *quit_mi =
gtk_menu_item_new_with_label ("Quit");
  gtk_menu_shell_append (GTK_MENU_SHELL (
f_menu), quit_mi);
  g_signal_connect (quit_mi, "activate",
G_CALLBACK (end_program), NULL);

  gtk_box_pack_start (GTK_BOX (vbox), btn,
TRUE, TRUE, 0);
  gtk_widget_show_all (win);
```

Figure 1a

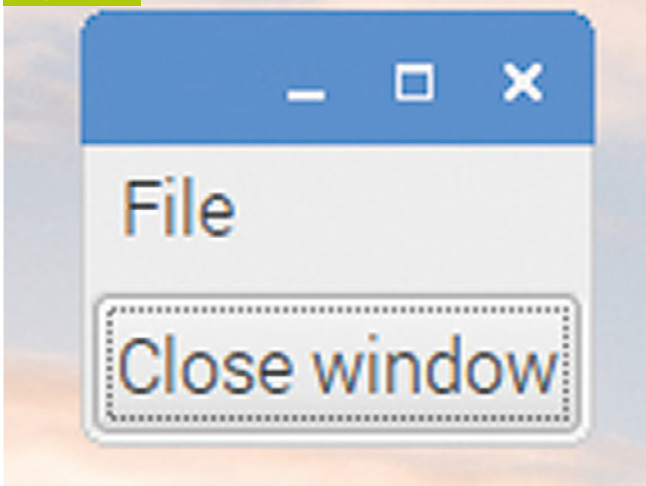
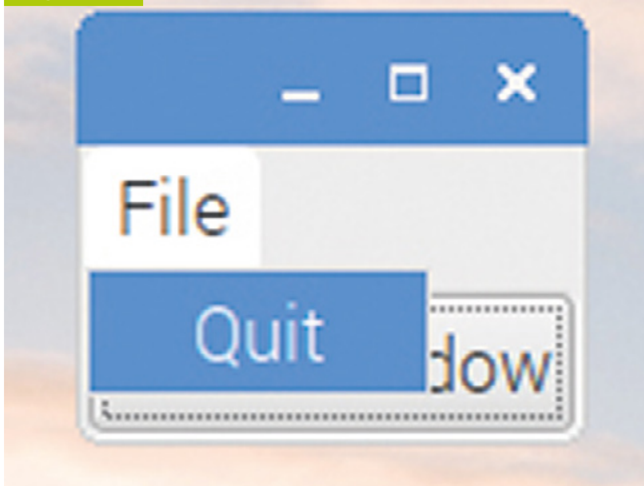


Figure 1b



▲ **Figure 1** A GtkMenuBar with a single GtkMenu ('File'), which contains a single GtkMenuItem ('Quit')

```
    gtk_main ();
}
```

First, we create a menu bar to hold the application menus.

```
    GtkWidget *mbar = gtk_menu_bar_new ();
```

We need to add it to the window; as with any other widget, to put it at the top of the window, we need to create a vertical box, pack the menu bar at the top and the rest of the window contents beneath it, and then put the vertical box into the window's container.

```
    GtkWidget *vbox = gtk_vbox_new (FALSE, 5);
    gtk_box_pack_start (GTK_BOX (vbox), mbar, TRUE,
TRUE, 0);
    gtk_container_add (GTK_CONTAINER (win), vbox);
```

We then create a menu item to hold the 'File' menu, and add it to the menu bar using `gtk_menu_shell_append` – a menu shell is anything which can hold menu items; in practical terms, this is either a menu or a menu bar.

```
    GtkWidget *file_mi =
gtk_menu_item_new_with_label ("File");
    gtk_menu_shell_append (GTK_MENU_SHELL (mbar),
file_mi);
```

At this point we have a menu bar with the single item 'File' on it – we now need to create a menu to associate with the menu item.

```
    GtkWidget *f_menu = gtk_menu_new ();
```

We then use `gtk_menu_item_set_submenu` to set our created menu as a submenu of the 'File' menu item.

```
    gtk_menu_item_set_submenu (GTK_MENU_ITEM (
file_mi), f_menu);
```

At this point, we have a menu bar with the 'File' item on it, and an empty menu as the submenu of that menu item. We now create a menu item to hold the 'Quit' option, and we use `gtk_menu_shell_append` to add this to the 'File' menu.

> "GTK provides a number of widgets which can be used to create either menu bars or pop-up menus"

```
    GtkWidget *quit_mi =
gtk_menu_item_new_with_label ("Quit");
    gtk_menu_shell_append (GTK_MENU_SHELL (f_menu),
quit_mi);
```

We now have a menu bar with a single menu ('File'), which contains a single option ('Quit'), but as yet that option doesn't do anything. To make the menu item do something, we connect a handler callback to its `activate` signal with `g_signal_connect`, just as with a button.

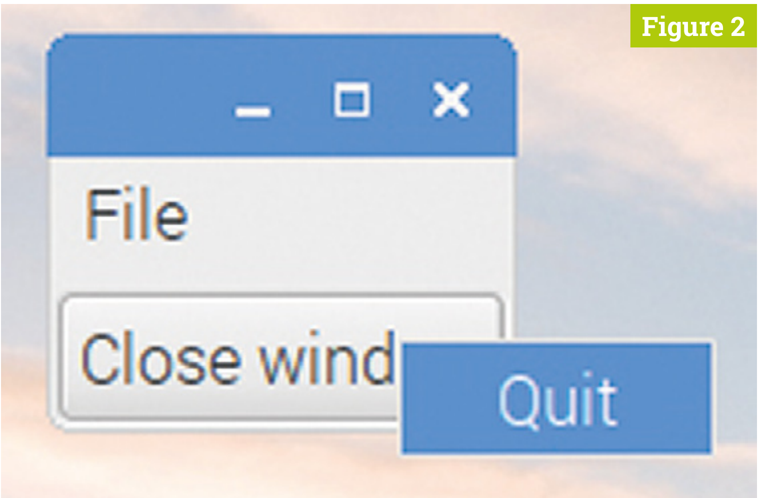


Figure 2 A pop-up menu, launched from a button

The code to connect our existing `end_program` handler to the 'Quit' menu item is:

```
    g_signal_connect (quit_mi, "activate",
G_CALLBACK (end_program), NULL);
```

That's it – you now have a working 'Quit' menu option in your application (**Figure 1**). You can use the same process to add as many menus and menu items as you want to a menu bar.

Note that the function we used to add a menu to the item in the menu bar was called `gtk_menu_item_set_submenu` – in effect, the menu is regarded as a submenu of the top-level menu item in the menu bar. You can use exactly the same function to create an actual sub-menu from a menu item in a menu, and can nest these calls as deeply as you want to create the sort of hierarchical menu structure that more complex applications use. (From a usability point of view, it's wise to stick to no more than one additional level of submenu – having a menu create a submenu from some items is fine, but if you are creating even more submenus from the submenus themselves, the users may get confused!)

Pop-up menus

The menu bar is the most common way of adding a menu to an application, but it is also possible to use very similar code to add pop-up menus that are produced when you click buttons, tree views, or various other widgets.

> "The menu bar is the most common way of adding a menu, but it is possible to use very similar code to add pop-up menus"

Create the following handler, and connect it to the `clicked` signal on a button.

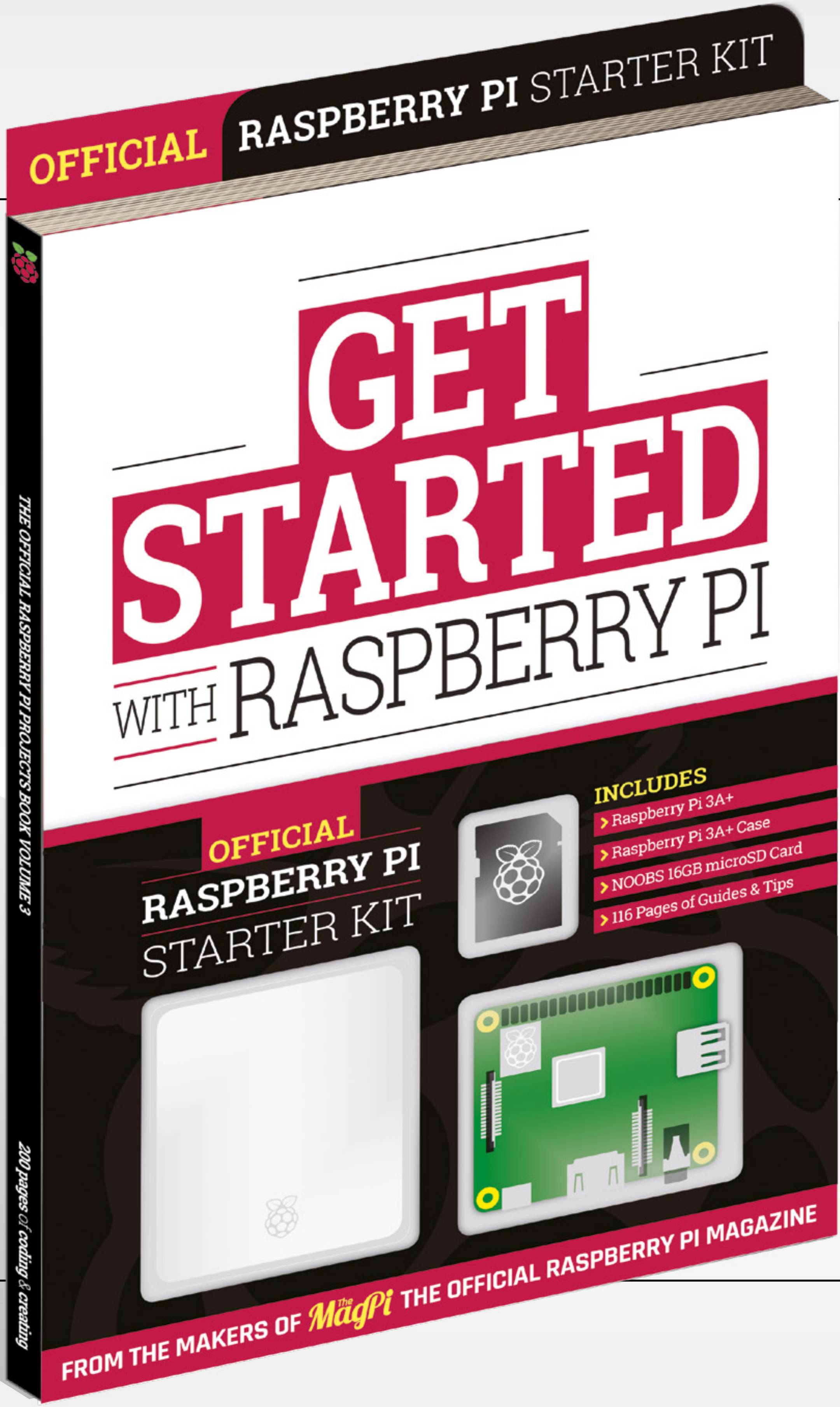
```
  void button_popup (GtkWidget *wid, gpointer ptr)
  {
    GtkWidget *f_menu = gtk_menu_new ();
    GtkWidget *quit_mi =
gtk_menu_item_new_with_label ("Quit");
    gtk_menu_shell_append (GTK_MENU_SHELL (f_menu),
quit_mi);
    g_signal_connect (quit_mi, "activate",
G_CALLBACK (end_program), NULL);

    gtk_widget_show_all (f_menu);
    gtk_menu_popup (GTK_MENU (f_menu), NULL, NULL,
NULL, NULL, 1,
        gtk_get_current_event_time ());
  }
```

This creates a menu with the single item 'Quit', as before, but instead of putting it into a menu bar, it uses the `gtk_menu_popup` function to display it at the mouse cursor position. When you press the button to which this handler is connected, the menu will be displayed over the button, and it can then be selected from there (**Figure 2**)

The `gtk_menu_popup` function is designed for use from very low-level system mouse events – when called from a button handler, most of the event information isn't needed, which is why most of its arguments are set to `NULL`. The only ones which matter are the first argument – the menu to pop up, and the last two, which are the code for the button which was pressed (`1` for the left mouse button) and the time at which the button was pressed (which is obtained from the `gtk_get_current_event_time` function).

This example shows a pop-up being generated by a button press, but it can also be linked to mouse events on many other widgets.


OFFICIAL RASPBERRY PI STARTER KIT
GET STARTED WITH RASPBERRY PI
OFFICIAL
RASPBERRY PI
STARTER KIT
INCLUDES
Raspberry Pi 3A+
Raspberry Pi 3A+ Case
NOOBS 16GB microSD Card
116 Pages of Guides & Tips
FROM THE MAKERS OF The MagPi THE OFFICIAL RASPBERRY PI MAGAZINE
THE OFFICIAL RASPBERRY PI PROJECTS BOOK VOLUME 3
200 pages of coding & creating

POWER UP your Kitchen with RASPBERRY PI

Get ready for cooking this holiday season with these kitchen tips and projects!

For those of us who enjoy cooking, it can be a lot of fun to get creative in the kitchen. For others, it can be a chore. Here in the pages of *The MagPi* magazine, though, we all overlap on the Venn diagram (or Arnold Schwarzenegger/Carl Weathers handshake) as makers and tinkerers and hackers.

We can make cooking better. Easier. Perhaps a little more fun for everyone. So, we've put together a tutorial for a little Raspberry Pi–powered kitchen aid, and found a selection of fun projects that might inspire you to hack your kitchen. Just in time for the upcoming holiday feasts as well.

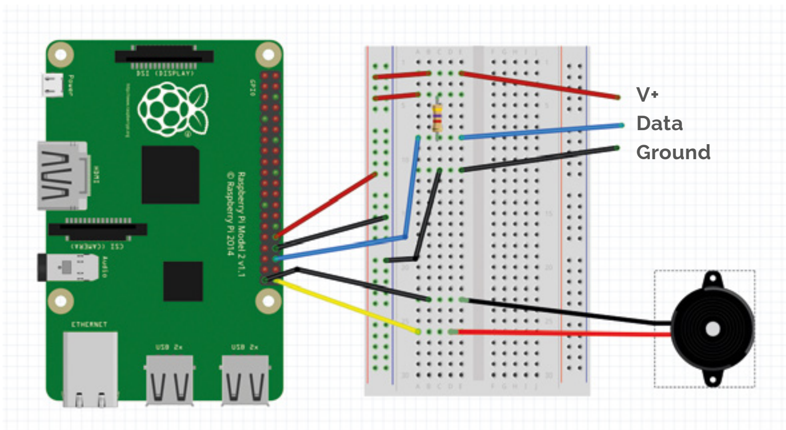
Warning!
Mains electricity!

Some of these projects involve mains electricity. We recommend substituting an Energenie where you can, if you plan to follow them.

magpi.cc/energenie

Make a Raspberry Pi cooking aid

Measure temperature precisely with alarms with our cooking aid



▲ The wiring diagram for the temperature sensor and buzzer

01 Wire it up

With your Raspberry Pi turned off, follow the wiring diagram (above) to set up our temperature sensor and buzzer. The sensor we're using is a digital 1-Wire device and will require extra software to be set up for it to work. Once it's all wired up, though, you can boot up your Raspberry Pi and install the Display-O-Tron software from the Terminal with:

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

02 Probe setup

Enter the following:

```
sudo nano /boot/config.txt
```

Add this line to the end of the file:

```
dtoverlay=w1-gpio,gpiopin=16
```

Reboot the Raspberry Pi and the open up the Terminal again. We need to activate the relevant kernel modules with:

```
sudo modprobe w1-gpio
sudo modprobe w1-therm
```

Move into the **w1** devices folder with `cd /sys/bus/w1/devices/` and use `ls` to list out what's there. It might take a few seconds to show up, but a folder with something like 28-000006d85491 will appear. Make a note of this folder and number.

03 Get the code

Download the code file **temperature.py**. Remember the number we noted down earlier? This is the serial for the thermal sensor and you'll need to edit line 19 in the Python file to replace `28-000006d85491`. While you're editing the file, you can also change the default temperature in `set_temp`.

Save and go back to the Terminal and enter `sudo nano /etc/rc.local`. At the end of the file, add the following so this script runs at boot (pointing towards where you saved the file):

```
python /home/pi/temperature.py &
```

Reboot, and test it out!

▲ You can take the temperature of water in a kettle and be alerted by a buzzer when it has boiled

You'll Need

- Display-O-Tron 3000 **magpi.cc/ihx5PF**
- DS18B20 temperature probe (waterproof)
- Buzzer
- Breadboard, wires

Hidden kitchen computer

MAKER: designforhire

The humble kitchen computer is an excellent tool for having a bit more info than TV or radio at your disposal. If you have a lack of counter space, you might be trying to figure out where to put one. Well, how about hiding it behind a calendar?

This particular build uses a sixties or seventies spy drama way of revealing the screen by rolling down the calendar at the touch of a button. A hidden keyboard is also revealed, allowing you control the computer a bit more manually. Genius. A bit more fun than installing a screen under a cabinet at least.

magpi.cc/byaPGR

▼ A nondescript larder door that you wouldn't look twice at

Raspberry Pi microwave

MAKER: Nathan Broadbent

"I was inspired by this post on Reddit, titled 'Food items should have QR codes that instruct the microwave exactly what to do. Like high for 2 minutes, let stand 1 minute, medium 1 minutes'," says Nathan on his blog. So he did just that.

This build is about six years old now; however, smart microwaves are only just starting to hit the shelves for consumers. Instead of QR codes, Nathan opted for the standard UPC barcodes you find on food, and so do the consumer models. For Nathan's build, we like the extra level of detail on the new touchpad, including a lovely little Raspberry Pi logo.

magpi.cc/q3iWaQ

▲ Nathan also supplies a recipe to make a raspberry pie in the microwave

HeaterMeter Pit Probe

MAKER: **Bryan Mayland**

Smoking meat can be a lot of work, and not very easy. One of the keys to success is keeping the temperature inside your smoker as consistent as possible. As you're likely going to have to add fuel (such as more charcoal) to your smoker to keep it going, it only adds to the complexity.

This is where something like the HeaterMeter Pit Probe comes in. It measures the temperature in your smoker/BBQ, the internal temperature of the meat, and ambient temperature outside. It uses all this information to control airflow into the smoker, allowing for the temperature to be more easily maintained. It also includes a handy web interface. Maybe you should try a smoked turkey?

magpi.cc/Hy7quP

▲ If this is your idea of a festive feast, then we'd happily accept an invitation

◄ You can buy a kit like this, or make one from scratch yourself

Internal meat temperatures

Roast, grill, or smoke your meat to perfection

Chicken TEMPERATURE **74°C**	**Steak** (rare) TEMPERATURE **52°C**	**Beef** (slow cooked) TEMPERATURE **63°C**
Turkey TEMPERATURE **74°C**	**Steak** (medium rare) TEMPERATURE **60°C**	**Lamb** (medium) TEMPERATURE **71°C**
Pork TEMPERATURE **63°C**	**Steak** (medium) TEMPERATURE **66°C**	**Fish** TEMPERATURE **63°C**

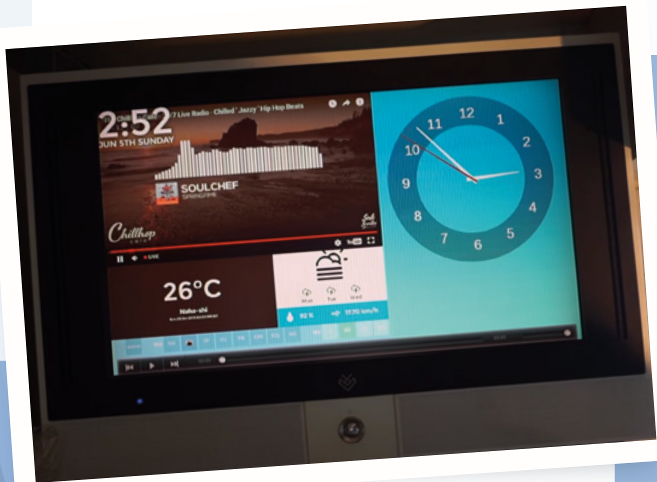
Kitchen TV

MAKER: **RiquezJP**

This project is more software-based, so it's something you can definitely look into before you start sawing holes into a door. All you need is a spare monitor, a Raspberry Pi to power it, and a suitable space in your kitchen.

It includes two distinct modes: TV mode and web browser mode. TV mode makes use of live YouTube channels for world news or music, while also displaying a clock, the weather, and a BBC News feed. From the web browser, you can search for different recipes. While a little more work to set up than an iPad with a smart cover, it's more useful in our opinion.

magpi.cc/KVNaXv



▲ TV mode is simple and gives you the info you need, and some tunes to boot

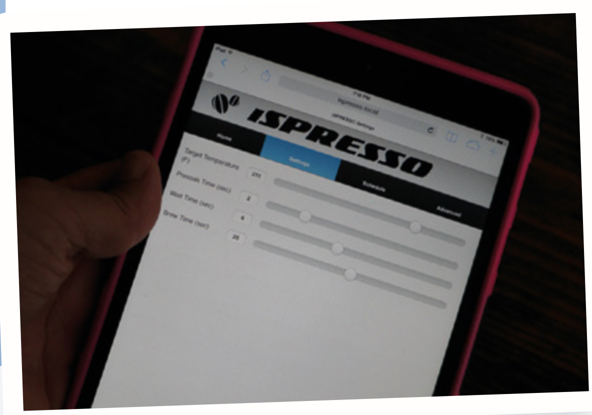
iSPRESSO

MAKER: **Chris Synan**

As coffee machines become more complex, the more you can hack them to brew at your bidding. There are a few 'smart' coffee machine hacks out there, but here we're going to talk about iSPRESSO. Not only does it have a pretty good tutorial that you can follow if you wish (with a supervising adult, of course), it also has pretty advanced remote control functions.

As it's a smart machine, you can also set some timers to preheat the water and such. A good way to make sure your morning cup of joe is freshly waiting for you when you wake up.

ispresso.net



▲ The remote control app gives you a lot of advanced control

BrewPi

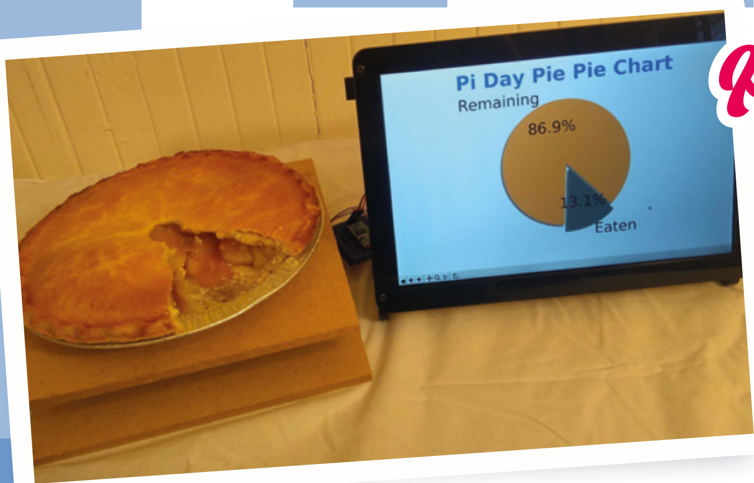
MAKER: **Elco Jacobs**

Brewing beer in your house may be a bit over-the-top for some folk. However, the method of doing so has many other applications, including sous vide-style cooking. While BrewPi can be bought as a standalone unit, the website has full instructions on how to design and create your own – which is a big of an undertaking!

Scaling it down for a rice cooker to make a water bath is a little simpler, although trading in beer for pulled pork might not be everyone's cup of tea. Oh yeah, you can make a perfect tea-brewing system with it as well.

brewpi.com

▲ Make your own fridge-based brewery!



▲ 86.9% of a pie is an excellent amount of pie to have

Pie Pie Chart

MAKER: **Mike MacHenry**

We've all had this problem, and so far there's never been a solution for it: how do you know how much pie is left? Thankfully, Mike has a solution in the Pie Pie Chart, powered by a Raspberry Pi.

This project weighs the pie at the start of its life, pre the first cut, and then gives an accurate representation of just how much pie is left once slices are taken. It's accurate to 0.1 percent, perfect for planning any late pie snacks in case you still want some for breakfast.

magpi.cc/FUuepB

Beverage temperatures

Brew the perfect cup of whatever you desire

Beverage	Temperature
White tea	65-75°C
Coffee	80-85°C
Black/normal tea	90-95°C
Green tea	75-85°C
Oolong tea	85°C
Herbal	95°C

PiArm

SPECS

SERVOS:
6 × smart servos – 0.24° accuracy, 0–240° angle, 17 kg/cm torque

BOARD:
PiArm Shield – on/off switch, multiple breakout pins

WEIGHT:
1.1 kg

MAXIMUM ARM EXTENSION:
375 mm

▶ SB Components ▶ **magpi.cc/aCndRx** ▶ From £279 / $369

A full-metal lightweight robotic arm with six smart servos. By **Phil King**

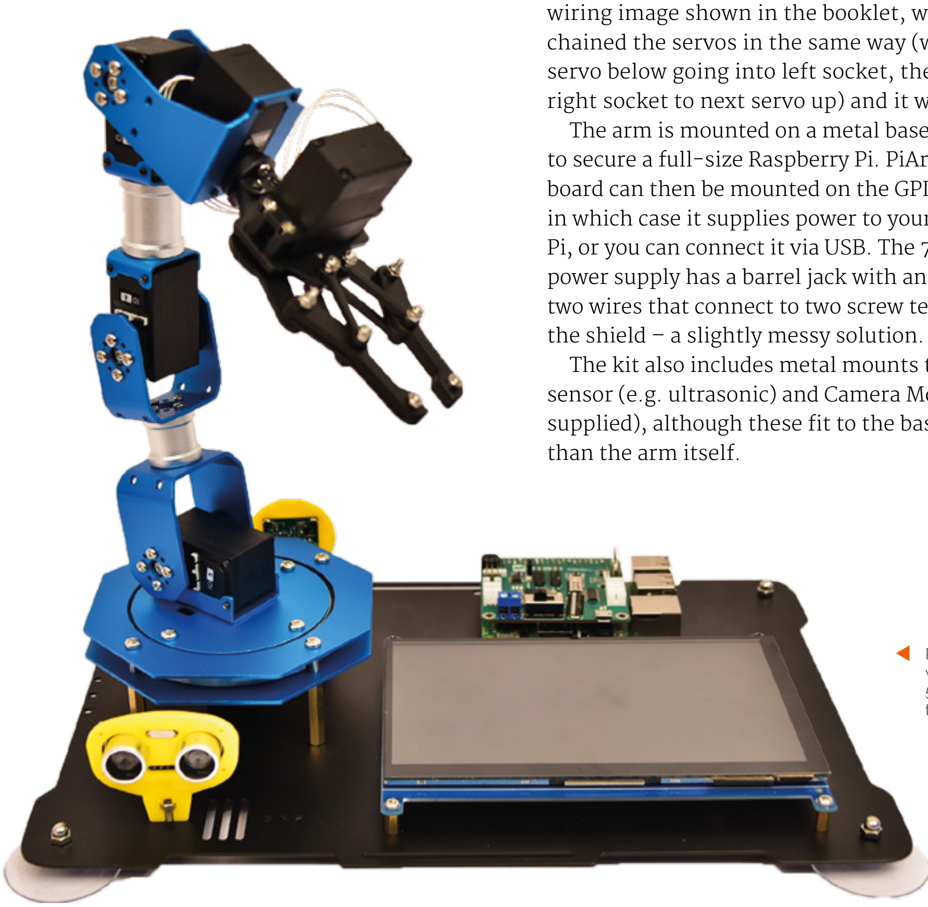
We've seen entry-level robotic arms for Raspberry Pi before, but PiArm is in a different league when it comes to build quality and precision movement. All the mechanical parts are solid metal, giving the assembled arm a very robust structure. Meanwhile, its six servos are smart ones, giving digital feedback on their angle, temperature, and voltage. As well as the Basic Kit reviewed here, versions are available with a 5- or 7-inch LCD touchscreen and speakers.

You'll need a spare hour or two to assemble the arm, using a large array of parts and different-sized screws. While the assembly guide booklet is well illustrated (and there's a video guide online), we found a few confusing discrepancies, including an annoying bit where we assembled a section, only to have to dismantle it again to wire up the servos (as revealed on the next page).

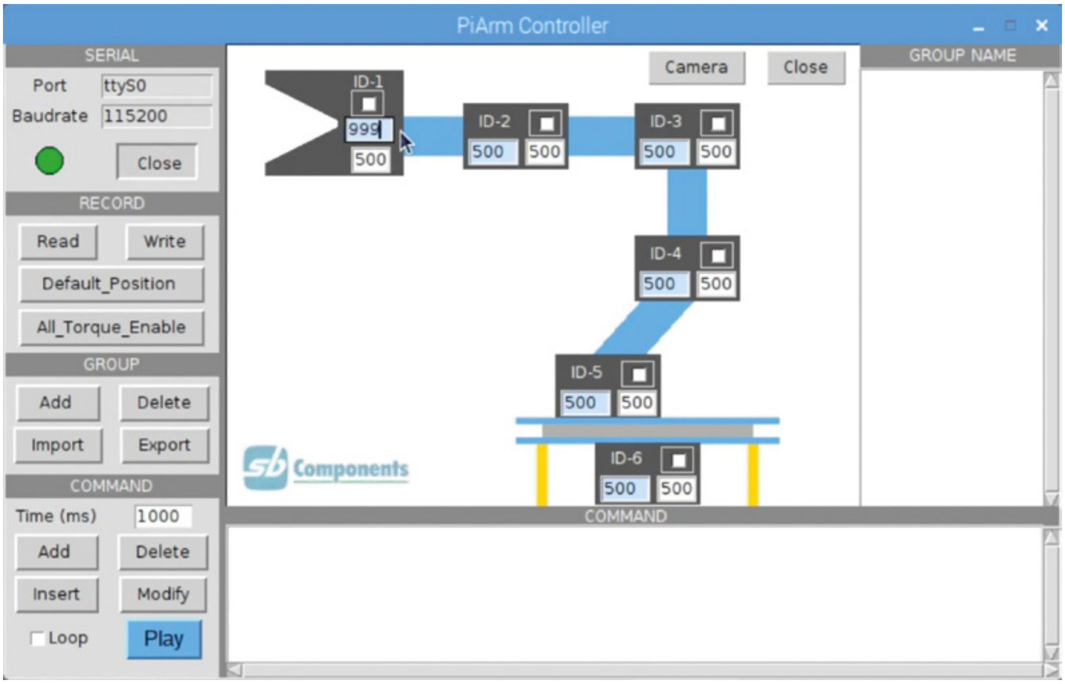
Nor was any servo wiring information supplied – SB Components says a video should be uploaded to the product page soon. Based on the single wiring image shown in the booklet, we daisy-chained the servos in the same way (wire from servo below going into left socket, then wire from right socket to next servo up) and it worked.

The arm is mounted on a metal base with holes to secure a full-size Raspberry Pi. PiArm's 'shield' board can then be mounted on the GPIO header, in which case it supplies power to your Raspberry Pi, or you can connect it via USB. The 7.5 V 5 A DC power supply has a barrel jack with an adapter with two wires that connect to two screw terminals on the shield – a slightly messy solution.

The kit also includes metal mounts to add a sensor (e.g. ultrasonic) and Camera Module (not supplied), although these fit to the base rather than the arm itself.



◀ PiArm is available with an optional 5- or 7-inch LCD touchscreen



A GUI interface enables you to create sequences of commands; by disabling the torque, you can manually position the arm and read its position

With the arm assembled, you can insert the preloaded 16GB microSD card supplied into the Raspberry Pi to get started. Our card was blank, however, so we needed to install Raspbian and clone the PiArm GitHub repo (**magpi.cc/TNSLVW**).

Graphical interface

While the software is based around a PiArm Python library, a GUI interface makes getting started much easier and lets you program command sequences. An image of the arm is shown on screen, with two number fields for each of the six servos. First, you need to input an address in the Port field to open up a serial connection to the arm: ttyS0 if Raspberry Pi is connected via GPIO; ttyUSB0 if via USB.

One way to program the arm is to type in numbers for each servo to set an arm position. A far simpler way, however, is to disable the torque and then manually position the arm to the desired position with your hands and read in the numerical servo data. This enables you to quickly store a sequence of commands (called a 'group') which you can then play back; sequences can be saved as text files for future reuse.

Its claw is good at gripping objects and picking them up

> A GUI interface makes getting started much easier and lets you quickly program command sequences

Picking it up

The arm rotates smoothly on its base, thanks to ball bearings, and moves fairly quietly. We soon managed to get the arm to pick up a keyring with its claw and then put it back down again elsewhere. We did find the default speed a bit too much, though, with the arm's more sudden movements sometimes being powerful enough to lift the base suckers off the table! Fortunately, we were able to reduce servo speed levels to a preferable level, counter-intuitively by raising the setting to 800. Other servo parameters, such as angle and voltage range, may also be altered using another GUI program, although it's not advisable to do so with the arm assembled.

A PlayStation-style wireless joypad is also supplied, enabling you to control the arm manually; in this case we found it slow to rotate and the arm automatically curled up while rising, but you can always alter the Python code to customise control. Indeed, you could use the PiArm Python library with your own programs.

Verdict

Excellent metal components and smart servos raise this robotic arm well above the level of cheaper entry-level rivals. The GUI interface makes it easy to program sequences, while advanced users could create their own programs based on the Python library.

9/10

Flirc Raspberry Pi 4 case is a stylish affair, with an aluminium case topped and tailed with soft-touch plastic

Flirc Raspberry Pi 4

SPECS

DIMENSIONS:
66×91×25 mm

CASE:
Aluminium housing

FEATURES:
Built-in heat sink

▶ Flirc ▶ **flirc.tv** ▶ £16 / $16

Stylish and clever case with a body that becomes a heat sink. By **Lucy Hattersley**

We see a lot of cases here at *The MagPi*, but the Flirc Raspberry Pi 4 case came recommended to us via Raspberry Pi's engineering team, so we had to take a look.

The Flirc Raspberry Pi 4 case comes in two halves: the bottom is a soft-touch plastic shell, and the top half is milled aluminium coated with more soft-touch. The result is a stylish and lovely-to-hold case made from solid metal.

Here's where things get clever: inside the case is a protruding heat sink that reaches down to Raspberry Pi 4's CPU. This turns the whole of the aluminium case into a giant heat sink, cooling down your Raspberry Pi 4.

Inside the pack is a square thermal pad (similar in substance to Blu Tack). You use this to squidge the Raspberry Pi to the heat sink. Putting together the case is ludicrously simple: you simply drop a Raspberry Pi in the bottom half, attach the thermal paste, squidge down the lid, and use four screws to hold everything together.

Flirc claims that this is "the most beautifully crafted Raspberry Pi 4 case" and it's not a wholly unwarranted claim. It certainly has a sense of style. Everything is neatly constructed from high-quality materials and there's considerable charm to the heat sink. The microSD card slot is easily accessible, and a small cut-out on the enclosure enables the LEDs to shine through.

On the box

One downside to the sealed approach is that the GPIO pins are hidden away inside the case. Unlike the official case, the lid cannot be quickly removed to provide access to the pins.

To Flirc's credit, it has addressed this issue via a small gap on the underneath of the Flirc Raspberry Pi 4 case, which could be used with a breakout I/O cable. But it's an ungainly addition to such a lovely looking case.

If you plan to use Raspberry Pi as a desktop computer, then this might be a valid trade-off.

Warning!

Using stress should only be done for short periods. Watch your Raspberry Pi and restart afterwards.

However, for many of us, GPIO pins are the very essence of Raspberry Pi.

Under pressure

We stress-tested a Raspberry Pi 4 board on its own vs a Raspberry Pi 4 inside the Flirc case to see what temperatures it reported.

We used stress (`apt install stress`) and the following script from Core Electronics (**magpi.cc/7Q0oX3**) to test our Raspberry Pi 4 while measuring temperature:

```
while true; do vcgencmd measure_clock arm;
vcgencmd measure_temp; sleep 10; done& stress
-c 4 -t 300s
```

This puts all four cores of Raspberry Pi 4 under stress. For each test, we left the Raspberry Pi to run for five minutes. Warning! Don't do this at home without doing your research first.

Unsurprisingly, the Raspberry Pi with no heat sink attached quickly went up to 79°C and hovered at that level for the rest of the test, nudging up against (but not pushing over) the level where Raspbian starts to throttle the CPU.

> We're going to experiment some more with overclocking, which makes this a fun case

Next, we put a Raspberry Pi into the Flirc Raspberry Pi 4 case and ran the same test. This time it idled at a mere 28°C and our five-minute stress test took it up to a mere 46°C.

Because this is comfortably below the threshold, it opens up a world of overclocking (something that has been reintroduced on Raspberry Pi 4 – see **magpi.cc/sED3id**).

We took the CPU clock speed up to 1.75GHz. The overclocked Raspberry Pi (inside the Flirc case) idled at 41°C, and running our five-minute stress test took it up to 67°C. Again, comfortably within a threshold. We also played around with CPU clock speeds up to 2.0GHz, which idled at 48°C and maxed out at 69°C.

We're going to experiment some more with overclocking, which makes this a fun case. Expect a tutorial shortly.

▲ The case cunningly doubles as a heat sink, enabling you to run your Raspberry Pi 4 more coolly (and overclock to faster speeds)

Verdict

We love the style of the Flirc case, and its heat sink opens up a world of overclocking. If only the GPIO pins remained accessible. If that's not a deal-breaker for you, though, then this is a great case to get.

8/10

Steam **Link**

▶ Valve ▶ magpi.cc/ZVyieK ▶ Free

With a change in hardware, does Steam Link streaming suffer or improve on Raspberry Pi 4? **Rob Zwetloot** tests it out

▶ Big Picture mode is used via the Steam Link, which is perfect for gamepad navigation

If you've been sleeping on the Raspberry Pi Steam Link app, then maybe the promise of new hardware to use it on will be enough to get you on board. Essentially, it's a local network streaming service that allows you to stream games from a gaming PC to another computer hooked up to a screen. There used to be dedicated hardware for this, but it has been available for Raspberry Pi (and other hardware) for a while now.

With the release of Raspberry Pi 4 and Raspbian Buster, it's taken some time to get a new version of the Steam Link app which works as well as it should. Over the last month or so, more stable versions have been released, so we thought it was time to give it a test.

Steam Link

Installing Steam Link is easy – it's available from the Raspbian software repository so can be installed from the Terminal with a simple `sudo apt install steamlink`. We highly recommend a wired connection for this – and thanks to the Gigabit Ethernet on Raspberry Pi 4, it's going to make a huge difference.

> Gameplay experience was only hampered by the computer to which we connected

So much so that our gameplay experience was only hampered by the computer to which we connected. A WiFi-connected laptop stuttered a little, and had some frame tearing; however, similar performance is experienced on an actual Steam Link. From a dedicated gaming PC hooked up over Ethernet, it was a different story.

While the experience is not seamless and one to one, it was extremely good – definitely good enough if you have the odd PC game you'd rather play on your TV without lugging a massive tower around.

Verdict

Not perfect, but near enough that if you have a spare Raspberry Pi 4 and want to play some PC games on your TV, there's no reason not to give it a try!

8/10

10 Best: AI projects

Bring your Raspberry Pi to artificial life with these machine learning projects

Machine learning and AI are just a normal part of the world now, which in some ways is kind of hard to process. On the plus side, it means we can have computers do really fun, useful (and useless) stuff for us. Here are ten ways to get your Raspberry Pi to learn and do.

▲ There's Waldo!

Robot cartoon-hunter

Waldo – or Wally as we call him in the UK – is a very elusive man who likes to travel around the world. The puzzle books asking young folks to find Wally in a busy crowd of people are very popular and can be tricky to solve; that is, unless you're an AI.

▶ magpi.cc/gApiTp

▼ Formula Pi

Self-driving racers

A lot of Raspberry Pi robots aren't autonomous – the Formula Pi racers are, though: using computer vision and your own bits of code, the aim is to make your robot the fastest and most accurate racer.

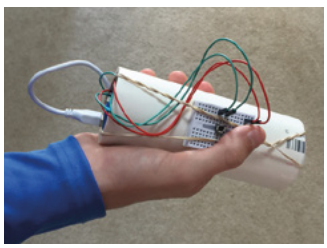
▶ formulapi.com

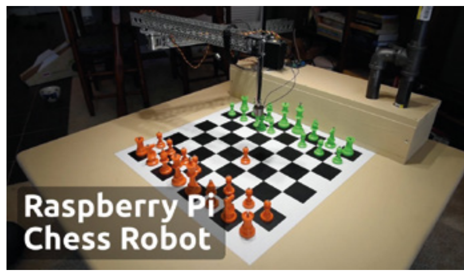
▼ Seeing wand

Magical item identifier

This project uses Microsoft's Cognitive Services to look at a picture for identification. When it works, it's pretty magical; however, it doesn't always work. Still, it will then use text-to-speech software to tell you what's in front of you. A future product for blind people, maybe?

▶ magpi.cc/pfpPwB





▲ Raspberry Turk

Computer chess IRL

The 'Mechanical Turk' was a magic trick where chess players would manipulate mechanical arms to make it look like people were facing a machine that could play chess. The Raspberry Turk is no magic trick – it does it for real.

▶ raspberryturk.com

◀ Cucumber sorter

Computer-aided vegetable categorisation

One of the promises of AI is that it can help people out with more mundane parts of work. The cucumber sorter allows a farmer to quickly and efficiently categorise his cucumber harvest. We've seen it in action and it is fun.

▶ magpi.cc/EWpGAp

▲ UBC Sailbot

Self-driving boat

Using GPS and a series of sensors and motor controllers, the Sailbot is one of a few autonomous sailing-boats that makes use of Raspberry Pi to control itself in races around the world.

▶ **sailbot.org**

◀ Just Keep Swimming

Fish-controlled robot tank

Living in a fish bowl must feel a bit limiting. So Alex Kent decided to allow his goldfish to move with the help of a computer vision project that senses where the fish is swimming, and moves its tank accordingly. Does it notice? Or just forget?

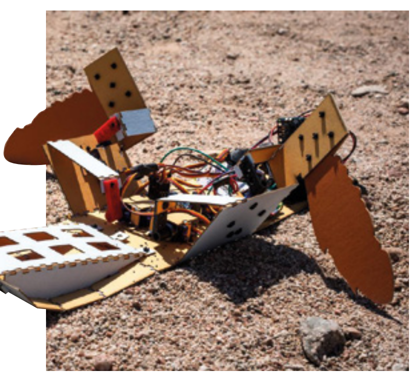
▶ **magpi.cc/ihFKyk**

▼ C-Turtle

Land-mine clearing project

This incredible project uses a low-cost robot design to probe abandoned (and extremely dangerous) minefields by sniffing out the mines and then detonating them. While this does result in each robot's heroic demise, it's much more cost-effective than other solutions.

▶ **magpi.cc/rKHQmo**

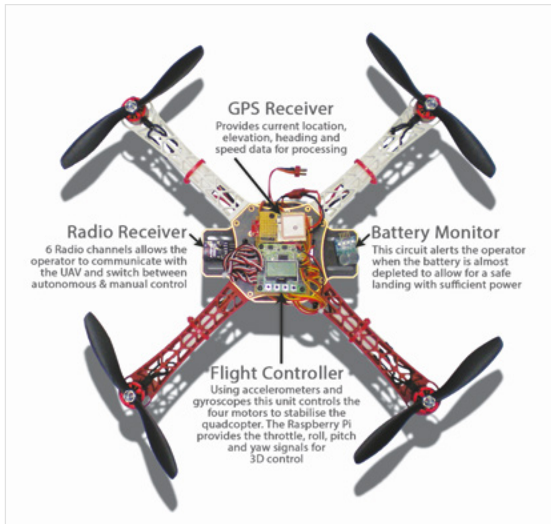


▼ Autonomous Quadcopter

Self-driving drone

This project didn't quite achieve full autonomy for a quadcopter/drone, but it got pretty close. Maybe you can build upon this design and create incredible aerial spectacles with a few drones?

▶ **magpi.cc/ysuieR**

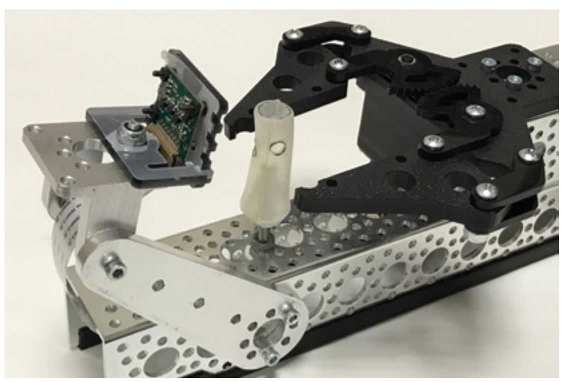


▼ Stent-testing robot

Testing breathable tubes

Stents are little tubes used to keep a patient's airway open. As they are vital, they need to be tested to extremes – this robot is able to control clamps that squish and compress the stent hundreds of thousands of times and monitor if and when it breaks.

▶ **magpi.cc/nivpET**



Learn to code with toys

Play your way to programming prowess. By **Lucy Hattersley**

Turing Tumble

CREATOR
Alyssa and Paul Boswell

Price:
£55 / $70

turingtumble.com

Turing Tumble is a vertical marble game, like a Japanese pachinko machine. Blue and red marbles are loaded to the top of the board and roll down the pins, hitting your switches as they go.

The direction of balls is changed via crossovers, ramps, and interceptors. Meanwhile, logic is simulated via bits (arrows that move left or right) and gears (which make the board Turing-complete). An accompanying book has 60 puzzles to solve. This board game has a lot of fans at Raspberry Pi Towers, and you can even pick it up at the Raspberry Pi Store in Cambridge.

Low-cost toys

Low-tech learning that doesn't cost the earth

TOWERS OF HANOI

The Towers of Hanoi puzzle is widely available (or you can easily make your own). It's a great way to think about solving a problem with an algorithm, and is a classic computer program, as shown in this Geeks For Geeks course (**magpi.cc/DXt231**).
magpi.cc/aQesb9

A DECK OF CARDS

You don't need expensive tools to teach coding. A deck of cards can be used to discover conditionals and other programming concepts. Check out this Code.org teaching resource.
magpi.cc/sNMBiq

MATCHBOXES

In the early 1960s Donald Michie, a pioneering British computer scientist, came up with Menace (the Machine Educable Noughts And Crosses Engine). It's a great way to learn ultra-modern AI techniques in a low-cost manner.
magpi.cc/6pw32R

Robo Rally

CREATOR

Richard Garfield

Price:
£40 / $44

magpi.cc/bwKcig

Designed by Richard Garfield of Magic: The Gathering fame and first published in 1994, Robo Rally is a board game where you program robots by lining up cards.

The rules are simple to learn, and it's highly entertaining. The aim is to keep your robot alive in a dangerous factory while trying to ram or shoot other players. Rather than a computer simulating a board game, it's a board game simulating a computer. Each player plays up to five cards per turn, and needs to plan and sequence ahead if they want to win.

Parrot Mambo FLY

CREATOR

Parrot

Price:
£100 / $130

magpi.cc/ARmPhL

Drones are a great toy for learning coding skills, although picking the right one is tricky. Go too high and you'll spend a lot of money with a risk of being 'broken by Boxing Day', but go too low and you'll get a toy without any coding nuances.

Parrot Mambo FLY is our pick of the bunch. It's got an advanced flight controller and sensors to keep it stable when the controls are lost and automatically cuts out the motors on collision.

There's an Android and iPhone app, but pah! Check out the pyparrot API to get coding your drone in Python on a Raspberry Pi (**magpi.cc/xuaKpT**).

For younger ones

Teach them to code from an early age

ROBOT TURTLES

Robot Turtles sneakily teaches little kids to program computers by moving turtles around a board game. Players use 'move' and 'rotate' cards to pick up jewels on the board.
robotturtles.com

BEE-BOT

Bee-Bot is a programmable floor robot controlled via a keypad on its back. It rolls around the floor and can record and play back audio. It moves in 15 cm steps and rotates by 90° turns.
magpi.cc/EGDtQt

CODE-A-PILLAR

This caterpillar robot is reprogrammed by pulling apart and rearranging the eight segments of its body. It's a fun robot that encourages thinking and experimentation in toddlers.
magpi.cc/tiefS6

Alex Mous

A 15-year-old university engineering student, who also co-organises a Raspberry Jam in his spare time

Day job **Student** | Community role **Raspberry Jam organiser** | URL **magpi.cc/viDVTM**

One of the original purposes of Raspberry Pi was to help ignite a passion for computing in young people today, and Alex is definitely one of these people.

"In 2014, I received my first Raspberry Pi (a Model B+) for Christmas," he tells us. "I started out by teaching myself Python using the projects in *Adventures in Raspberry Pi*. Using this knowledge of Python, I created several apps (both for Raspberry Pi and Windows). After that, I became interested in web design. I still do a lot of web design today, as shown in the recently completed coded-from-scratch website, **eikyutsuho.com**. The next language I tackled was C/C++, and I am still learning the ins and outs of it."

Five years later and at the age of 15, Alex is already working on an Associates Degree in Electrical and Computer Engineering, and he's also giving back to the community by helping out with the Seattle Raspberry Jam.

Why did you start co-organising the Seattle Raspberry Jam?

I began co-organising the Seattle Raspberry Jam in May because only a few people were showing up to each meeting. I thought that with some time and effort, I could increase the

It may be a small Raspberry Jam at the moment, but the Seattle Raspberry Jam accepts anyone

A thermal printing instant camera-style project, complete with a little screen for a viewfinder

Alex shows off his self-balancing robot to the Jam members

membership to something more respectable, such as ten members, for example.

How long has the Jam been running?
A makerspace called Jigsaw Renaissance started the Seattle Raspberry Jam in August 2013. Sadly, in July 2015, the makerspace decided that they no longer wanted to run the Jam. Stephen (my co-organiser) started up meetings again in August of 2015. I first joined the club in mid-2016 (I found out about it through the Jam Map). [Ed's note: you can find the map on page 90, or at **rpf.io/jam**.]

What kind of attendees do you get at the Jam?
We get just about everyone – from seasoned programmers who began coding during the days of punch cards to first-time programmers and Raspberry Pi users. People often come in confused about how to get started with Raspberry Pi, and we try to point them in the right direction.

The people who do come seem to enjoy it. We have just joined up with the ideaX Makerspace; they are happy to support our Raspberry Jam. We're hoping that ideaX Makerspace will give our Jam more visibility.

What's your favourite Raspberry Pi project you've made?
My favourite completed project is an instant camera, christened the PolarPiBerry. It uses a thermal printer, touchscreen, arcade-style button, and multiple battery packs (a future improvement is to combine the power sources) for the hardware, and a custom Python WX GUI as the interface with a live stream from the camera.

I am currently working on a self-balancing robot for under \$50. It uses the cheap yellow geared DC motors, an L298 dual H-bridge, an MPU6050 IMU, a DC-DC boost converter, and a 6×AA battery pack. I have designed and 3D-printed the chassis, but I am still working on the code.

> " I am currently working on a self-balancing robot for under \$50 "

Seattle Raspberry Jam

The Seattle Raspberry Jam is a monthly event that happens every third Wednesday of the month at the Bellevue Library. Entry is free, and it's a great way to learn something about Raspberry Pi and geek out with your fellow makers and coders!

This Month in Raspberry Pi

A Raspberry Pi Spookfest 2019

People are preparing for Halloween with some excellent projects

Due to the nature of the way we make *The MagPi*, we're writing this about two weeks before the magazine comes out on Halloween. Even this far into October, though, people have got their projects ready and Halloween fever is here. Prepare yourselves…

01. CoderDojos and Raspberry Jams look like they're getting into the spooky spirit

02. We love these Scratch-powered scary pumpkin lights!

03. There are a lot of costumes with eyes this year and we're not sure why. They're very cool, though

04. You might be a little late to get one of these this Halloween, but maybe now is a good idea to stock up for next year. Buy one here: **magpi.cc/sKNihf**

05. Taking store-bought products and hacking them with Raspberry Pi is an excellent idea

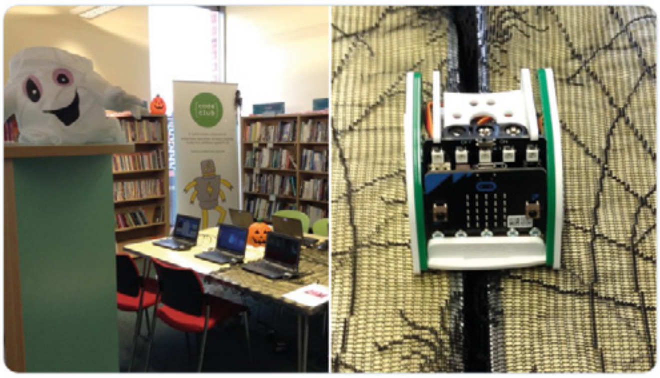
06. What's cooler than a spooky skull? A spooky dinosaur skull… with moving eyes!

07. Another scary eye, this time based on Sauron, from *Lord of the Rings*

01

failsworth.uk@coderdojo.com
@FailsworthUk

Looking at last years photo's we're pulling together some projects ready for next week's Halloween Session at @FailsworthUk - We'll have Spooky Scratch, Pumpkin Pi (Raspberry Pi really) and some Night Mare MicroBits - So come along to @FailsworthLib to try some scary coding



1:10 PM · Oct 19, 2019 · TweetDeck

02

Lacaze Sarah
@slmediation

Tuto pour #halloween en préparation avec du #RaspberryPi et du #scratch

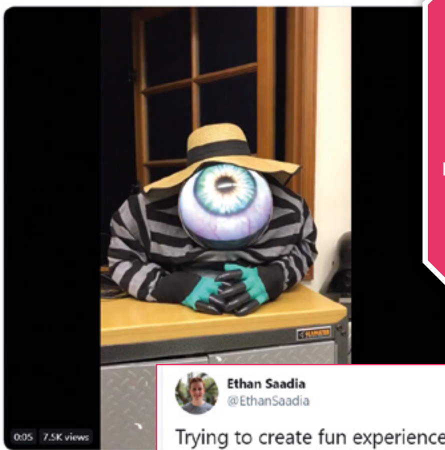
Translate Tweet

10:23 AM · Oct 17, 2019 · Twitter for Android

03

mfletcherchristian
@mfletcherchristian

My dad's birthday is in October, so built him a Halloween Mike Wazowski... @PaintYourDragon @adafruit @Raspberry_Pi @Hacksterio



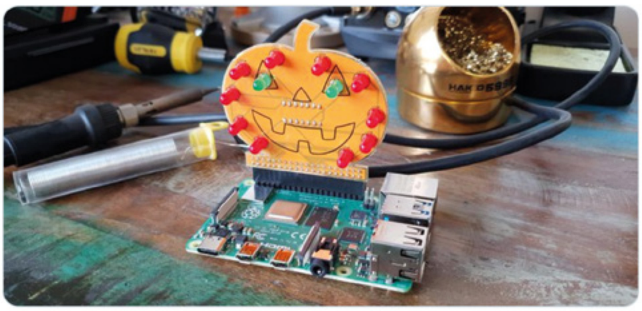
SHARE YOUR SCARY PROJECTS!

Got a spooky, scary project you're making for Halloween? Share pictures of it with us on Twitter **@TheMagPi**, or via email: **magpi@raspberrypi.org**

04

The Pi Hut
@ThePiHut

Spooky! Introducing the Halloween PumpkinPi 🎃 for Raspberry Pi! - mailchi.mp/thepihut/06092...



M · Oct 18, 2019 · Mailchimp

05

Ethan Saadia
@EthanSaadia

Trying to create fun experiences for the engineering club a friend and I started at school. We're doing a tear down of this @Target Halloween decoration. It would be a great candidate for a Raspberry Pi project! #Halloween #RaspberryPi #Hacktober

06

mfletcherchristian
@mfletcherchristian

T-Rex Halloween Candy bowl. Shout out to @PaintYourDragon for inspiration (and code). This uses the same graphics I made for my Eye Of Sauron; sadly, the lovely orange color gets washed out in video. @Raspberry_Pi @Hacksterio

07

mfletcherchristian
@mfletcherchristian

Progress on the Eye Of Sauron... Halloween seems to be coming more quickly this year. Not finished yet, it still needs more structure and surrounding fire. @Raspberry_Pi @Hacksterio @PaintYourDragon

MagPi Monday

Amazing projects direct from our Twitter

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

01. The tortoise fridge is back and being shown off at Pi and More in Germany next month!

02. Remote-controlled theatre lights that have been 3D-printed? Now that's amazing!

03. A robot that will boldly go around your living room floor

04. This full-on weather station is one heck of a big weekend project! Well done!

05. We love a good Pip-Boy build here, and this one is coming along nicely

01

PumbatheTurtle
@pumbatheturtle

Replying to @TheMagPi

This weekend i created a presentation for PI and More 12 in Trier (Germany). The event opens the doors at 16.11.2019. I will talk about my Tortoise Fridge.

@PiAndMore

02

S Organ
@makercupboard

Replying to @TheMagPi

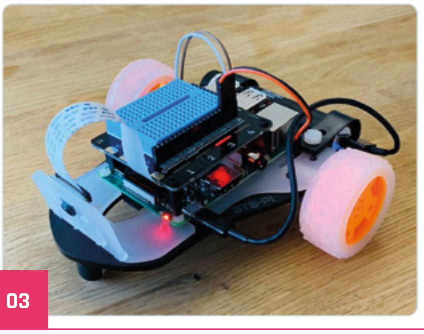
Raspberry Pi powered 3d printed theatre light

03

Ciprian Manea
@ovipic

Replying to @TheMagPi

The mini Space Shuttle #unPi

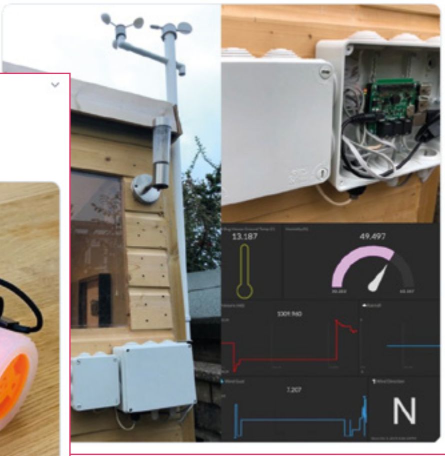


04

Allen Heard
@AllenHeard

Replying to @TheMagPi

Fully indulged in the @Raspberry_Pi projects making a weather station and creating an IoT dashboard



05

Justin Hart
@jshart

Replying to @TheMagPi

WIP fallout pipboy... Got everything running, but a long way to go yet!

Crowdfund this!

Raspberry Pi projects you can crowdfund this month

FAST RFID

We've seen a couple of RFID kits available for Raspberry Pi, but they're still not very common. So this USB-connected RFID reader that works with Raspberry Pi is a welcome addition. You can register up to five RFID devices with it.

▶ kck.st/2VqZ8H1

COEX Clover

The COEX Clover is an educational drone kit – you build it yourself and it's completely programmable as well. It can also be used for racing – drone racing is an amazing sport that you'll find at many maker events.

▶ igg.me/at/coex

CROWDFUNDING A PROJECT?

If you've launched a Raspberry Pi-related project, let us know!

magpi@raspberrypi.org

Best of the rest!

Here are some other great things we saw this month

WALL-MOUNTED ARCADE MACHINE

While we assume this is definitely not the first arcade cabinet that is mounted to a wall, it's the first we've seen. The wood panelling and stools give it an excellent and different aesthetic.

▶ magpi.cc/ifbyMS

INTERNET RADIO

Internet radio projects with Raspberry Pi aren't too uncommon, but this is definitely one of the better builds we've seen. We like the partial retrofuturistic vibe.

▶ magpi.cc/Usfrbg

IRL STREAMING BACKPACK

While not something everyone would like to build, we appreciate the effort that's gone into this setup. Ever wanted to follow someone's Pokémon GO capturing?

▶ magpi.cc/BCucWJ

Raspberry Jam Event Calendar

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

01. Ibagué Science Raspberry Jam

- Thursday 7 November to Friday 8 November
- **Universidad Antonio Nariño, Ibagué, Colombia**
- **magpi.cc/XkRPBp**

A Jam in Ibagué to bring students of different levels of expertise together to learn about Raspberry Pi.

02. Cornwall Tech Jam

- Saturday 9 November
- **Cornwall College Camborne Campus, Redruth, UK**
- **cornwalltechjam.uk**

For anyone interested in technology, of all ages and abilities. Ask questions and learn about programming.

03. Dallas Young Makers Club

- Saturday 9 November
- **Dallas Public Library, Dallas, TX, USA**
- **magpi.cc/WTGUep**

Free, mentor-led, hands-on projects for kids with Raspberry Pi, Lego Mindstorm robots, and more.

04. Raspberry Pi Jam DC

- Saturday 9 November
- **Chevy Chase Library, Washington, DC, USA**
- **magpi.cc/wwJyDt**

If your children are seven or older, bring them to this one-hour STEM after-school programme.

05. South Devon Tech Jam

- Saturday 9 November
- **Paignton Library and Information Centre, Paignton, UK**
- **magpi.cc/9vhGQ5**

A monthly informal and friendly session for anyone interested in technology, regardless of age or ability..

06. Stafford Raspberry Jam

- Tuesday 12 November
- **Stafford Library, Stafford, UK**
- **magpi.cc/sWMiLe**

A meet-up for folks who have a Raspberry Pi computer and want to learn more about it.

07. Seattle Raspberry Jam

- Wednesday 20 November
- **ideaX Makerspace, Bellevue, WA, USA**
- **magpi.cc/bSB7kY**

A free monthly Raspberry Pi meet-up for beginners and experts. Come and participate in the monthly project.

08. Raspberry Jam Halifax

- Tuesday 10 December
- **Halifax West High School, Halifax, NS, Canada**
- **magpi.cc/fpsxLA**

An event centred around learning about the Raspberry Pi and sharing projects based on it

FULL CALENDAR

Get a full list of upcoming events for November and beyond here:

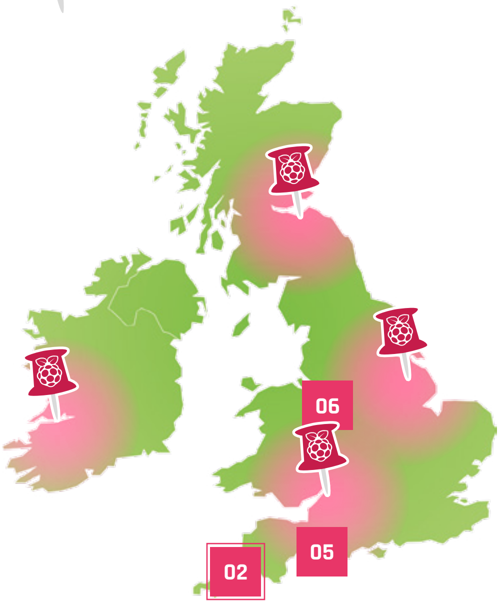
rpf.io/jam

FIND OUT ABOUT JAMS

Want a Raspberry Jam in your area? Want to start one?

jam@raspberrypi.org

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



Raspberry Jam advice: Fundraising

"In order to avoid charging for tickets, we always hold a raffle and raise between £60 and £200, and everyone's happy whether they win or not. You can use Twitter to ask for swag from the community, to give away as prizes. It's so nice to be able to run community events which are free to attend."

Grace Owolade-Coombes
South London Raspberry Jam

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

Your Letters

Jumping into Christmas

I notice we're getting towards that time of year again… and you did have a Halloween special… will you be doing more Christmas projects this year? I do love using my Christmas break to make my place a lot more festive with a Raspberry Pi or two.

Kit via email

However much we try, we cannot stop our Features Editor, Rob, from pitching and writing a Christmas feature or tutorial each year. We think there's something wrong with him. He has some more lighting plans for Christmas this year, though, so you may need to get ready to stock up on LEDs.

You can always still print off the issue 52 cover and make your own Christmas card

Contact us!

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

Website updates

I've been an avid reader of *The MagPi* for a long time now, and also enjoy other magazines that you publish, like *HackSpace* and *Custom PC*. However, I have noticed that they use very different styles of website than *The MagPi*. Is this intentional? Will you be getting a website update to match them? I prefer the way they navigate, I have to say! I'll still keep reading the website, though, so keep up the good work!

Jon via email

We've been beavering away on this for a while now and we're finally happy to say that, at the time this magazine comes out, we have a brand new website! We've done a rejig of how we present and organise the articles we post, and overhauled the 'contact us' form so that you can more quickly get in contact with the team that will best help you. Give it a look at the same address as always: **magpi.cc**.

We like our new website, and we hope you do too!

Here is Igalia's Continuous Integration system that's used for running regression tests

Open graphic drivers

Has there been any recent news about the open-source versions of the graphics drivers for Raspberry Pi's VideoCore? Also, is there any word on whether the new VideoCore in Raspberry Pi 4 will have open-source drivers as well? I assume VideoCore VI uses a different set of drivers.

Christine via email

You're in luck Christine, as there was recently a big update on the VideoCore drivers you can find on the Raspberry Pi blog here: **magpi.cc/CyxynC**.

A brief summary is that a new team from Igalia is working on the open-source drivers. You're right in thinking that VideoCore IV and VI have different drivers – they're referred to as VC4 and V3D respectively. The team have been working on getting to know the hardware, and have made some advances in the last few months including shader optimisations, support for OpenGL Logic Operations, as well as support for OpenGL ES 3.1. Work isn't finished yet, but it is progressing!

New learning

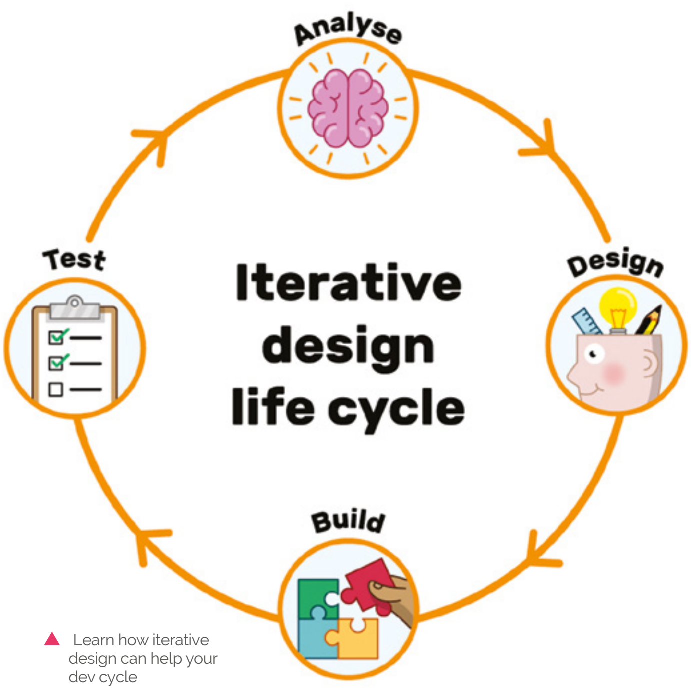
Will there be new course on FutureLearn from Raspberry Pi any time soon? I like the courses already up there and would like to learn more!

Lee via Twitter

Yes, there are three brand new, free courses being launched by the Raspberry Pi Foundation on FutureLearn starting now! Here are the new courses...

- **Design and Prototype Embedded Computer Systems:** in this course, you will discover the product design life cycle as you design your own embedded system.
- **Programming 103: Saving and Structuring Data:** this explores how to use data across multiple runs of your program.
- **Introduction to Encryption and Cryptography:** In this course, you'll learn what encryption is and how it was used in the past, and you'll use the Caesar and Vigenère ciphers.

The last course is still under development, so check back for a release date.



Learn how iterative design can help your dev cycle

WIN ONE OF FIVE SMARTIPI TOUCH 2 TOUCHSCREEN CASES!

The original SmartiPi was a great case that made use of the official Raspberry Pi Touch Display, and this new version includes a specific mount for a camera, a Lego-compatible faceplate, and a little cooling fan.

In association with **The Pi Hut**

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

Terms & Conditions

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

RETRO COMPUTING SPECIAL

SET UP A RASPBERRY PI 4 RETRO GAMES CONSOLE

MODERNISE A ZX SPECTRUM+ WITH RASPBERRY PI

BUILD A CLASSIC ARCADE MACHINE FOR YOUR HOME

THE MAGPI **#88**
ON SALE **28 NOVEMBER**

Plus!

Hack smart Christmas lights

Make an RPI232 bulletin board

Construct a camera-enabled robot

DON'T MISS OUT!

magpi.cc/subscribe

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

EDITORIAL

Editor
Lucy Hattersley
lucy@raspberrypi.org

Features Editor
Rob Zwetsloot
rob.zwetsloot@raspberrypi.org

Sub Editors
Phil King and Nicola King

ADVERTISING

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

DESIGN

criticalmedia.co.uk

Head of Design
Lee Allen

Designers
Sam Ribbits

Illustrator
Sam Alder

CONTRIBUTORS

Wes Archer, Mike Cook, David Crookes, PJ Evans, Rosie Hattersley, Nicola King, Daniel Lambton-Howard, Simon Long, Sean McManus, Danny Staple

PUBLISHING

Publishing Director
Russell Barnes
russell@raspberrypi.org

Director of Communications
Liz Upton

CEO
Eben Upton

DISTRIBUTION

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

SUBSCRIPTIONS

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

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

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

ISSN: 2051-9982.

A new golden age

The arrival of Raspberry Pi sparked a renewed interest in computing for **Phil King**

It was in the early 1980s when I had my first encounter with a computer. A teacher friend of my mum's – Miss Horsburgh, who also made superb shortbread biscuits – had kindly offered to show me her new Commodore PET. Compared to today's computers, it now looks incredibly primitive with its small green screen and floppy disk drive, but back then it was cutting-edge technology and I was totally enthralled by what it could do. I recall playing simple games, like Nightmare Park (**magpi.cc/QwujhD**), which had ASCII characters for graphics.

I was hooked and soon wanted a computer of my own. After persuading my parents that it would help me with my school homework (that old chestnut), I was incredibly excited to unwrap a Sinclair Spectrum on Christmas Day 1982. While I never did use it for homework, it was my dream machine, complete with colour graphics, beepy sound, a rubber keyboard, and a massive 16kB of RAM.

Electric dreams

As well as playing lots of games on my Spectrum – including those laboriously typed in, line by line, from magazines – I did get round to doing some BASIC programming. I was heavily into *Dungeons & Dragons* at the time and decided to code a computerised character generator. Even though it was fairly simple – simulating the throwing of various-shaped dice to generate elves, clerics and so on, with their alignments and abilities – I felt a great sense of accomplishment at having created it.

I dreamt of becoming one of the 'bedroom programmers' of the time, who were single-handedly creating commercial games. While I never managed to fathom the complexities of Z80 assembly language, I did eventually write a basic shoot-'em-up – in BASIC, sped up by using a compiler. As a puerile teenager, I even did a version with rude graphics.

> "I now have numerous Raspberry Pi devices scattered around the home"

Back to the future

After that, I forgot all about programming for years... until the arrival of Raspberry Pi. What grabbed me in particular was the ability to hook up electronic components and control them with code. Flashing my first LED was a genuine thrill and I've since dabbled with all sorts of lights, sensors, and add-on boards. I'm certainly no expert, but I'm learning all the time. I now have numerous Raspberry Pi devices scattered around the home, including an air quality monitor, TV streamer, wildlife camera, and several wheeled robots.

It's not just the ability to do physical computing with Raspberry Pi that's impressed me: I've discovered that slick-looking games can be created with just a few hundred lines of code using Pygame Zero – as demonstrated in Mark Vanstone's recent tutorial series for *The MagPi*.

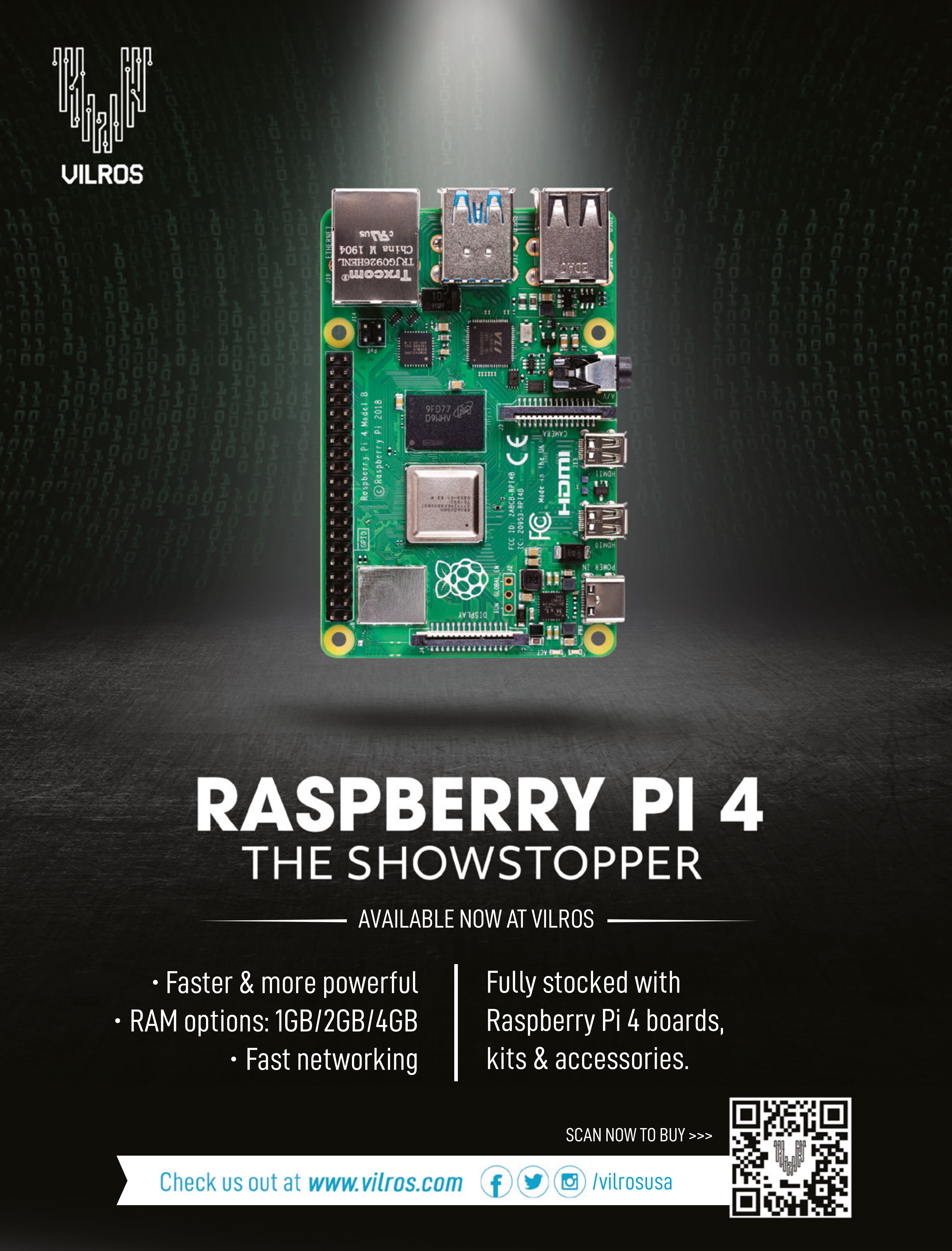
While it's all a far cry from the primitive technology of those early computing days, I feel it's recapturing the spirit of that era. Once again, amateur programmers and makers are beavering away and being creative, often sharing their code and advice freely with others thanks to the open-source ethos. It really is a new golden age.

AUTHOR

Phil King

When not editing books for Raspberry Pi Press, Phil enjoys dabbling with coding and attempting to do the Listener crossword.

magpi.cc


VILROS
RASPBERRY PI 4
THE SHOWSTOPPER
AVAILABLE NOW AT VILROS
• Faster & more powerful
• RAM options: 1GB/2GB/4GB
• Fast networking
Fully stocked with
Raspberry Pi 4 boards,
kits & accessories.
SCAN NOW TO BUY >>>
Check us out at www.vilros.com
/vilrosusa