The MagPi

Issue 104 | April 2021 | magpi.cc

The official Raspberry Pi magazine

HOME OF THE FUTURE

Automate your home with Raspberry Pi

Upcycle iPod Classic with Spotify

Build an arcade machine

48 PAGES OF PROJECTS & TUTORIALS

WELCOME
to *The MagPi* 104

Welcome to the future. 2021 is a sci-fi year and science promised us hoverboards, jet packs, and robot butlers. We can't do much about the first two (although we're keeping an eye out). But help around the home is definitely our domain. In this issue we asked resident home automation expert PJ Evans to design our home of the future with Raspberry Pi (page 34). The result is a fabulous collection of widgets, projects, and gizmos to help around the house. I've already talked Rosie into installing some NeoPixel steps at home.

Meanwhile Rob has been looking how to take a Raspberry Pi project and turn it into a Pico project (page 72). KG has started to build an arcade machine (page 42). Nicola has been looking at an autonomous home robot (page 20). Me? I've been playing around with ARM assembly like the nerd I am (page 84).

This issue has been a huge amount of fun! And we particularly like the illustration adorning this month's cover. Thanks to Sam Alder for incredible drawing skills (and Sam Ribbits for laying it out).

Enjoy this month's edition and let us know what you do with your home.

Lucy Hattersley Editor

EDITOR

Lucy Hattersley

Lucy is editor of *The MagPi* magazine and continues to bash her south London home into shape.

@LucyHattersley

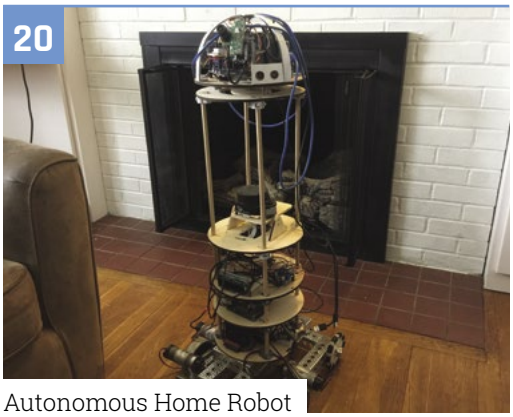
Contents

Issue 104 April 2021

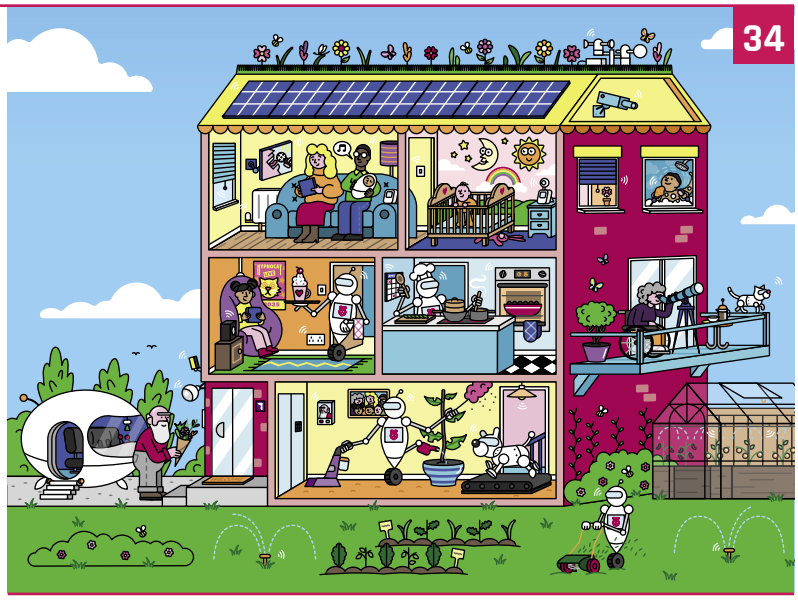
Cover Feature

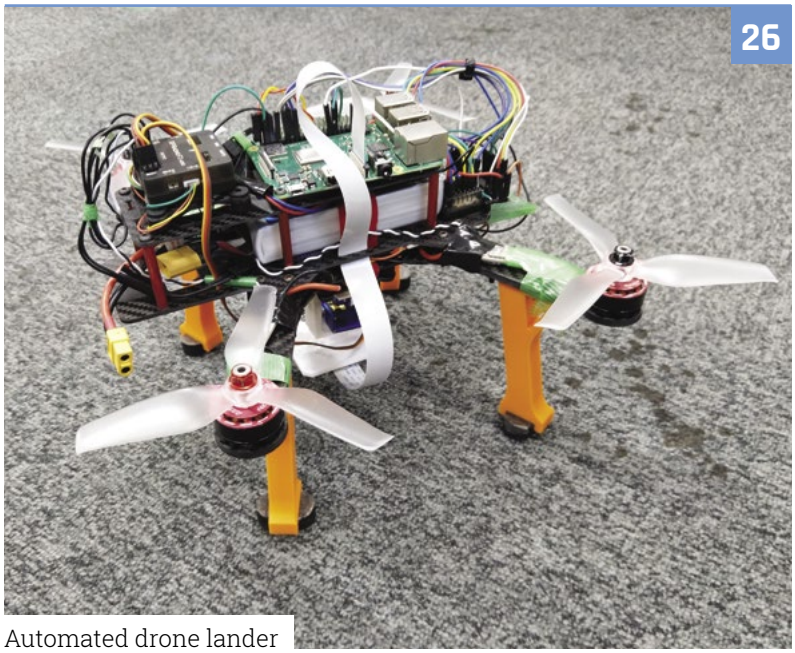
Regulars

Project Showcases



Autonomous Home Robot

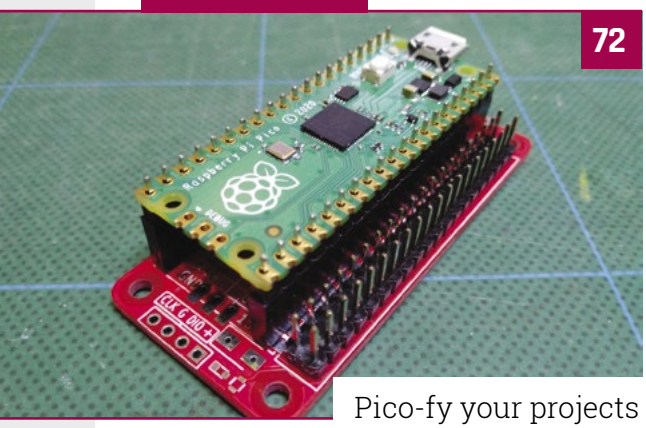




Automated drone lander

Tutorials

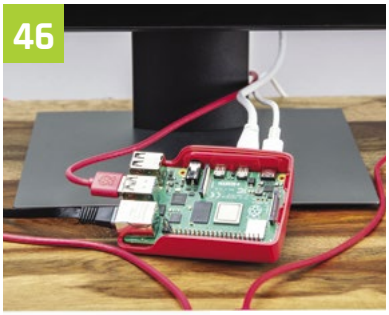
The Big Feature



Pico-fy your projects

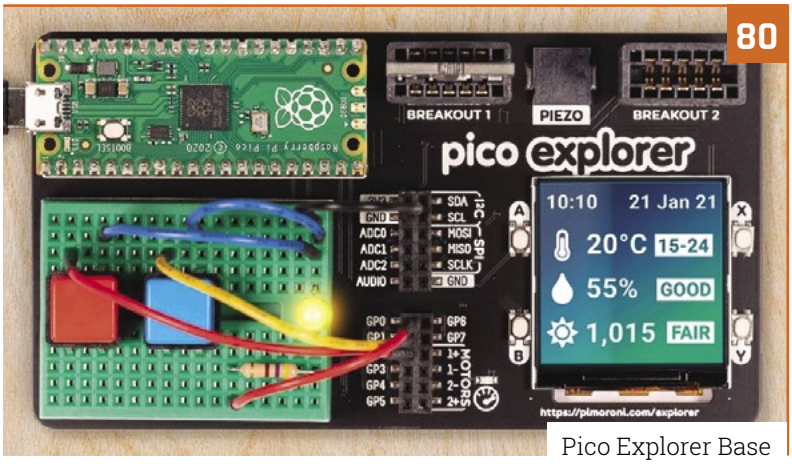
Reviews

Community



Set up Pi-hole

Use Visual Studio Code

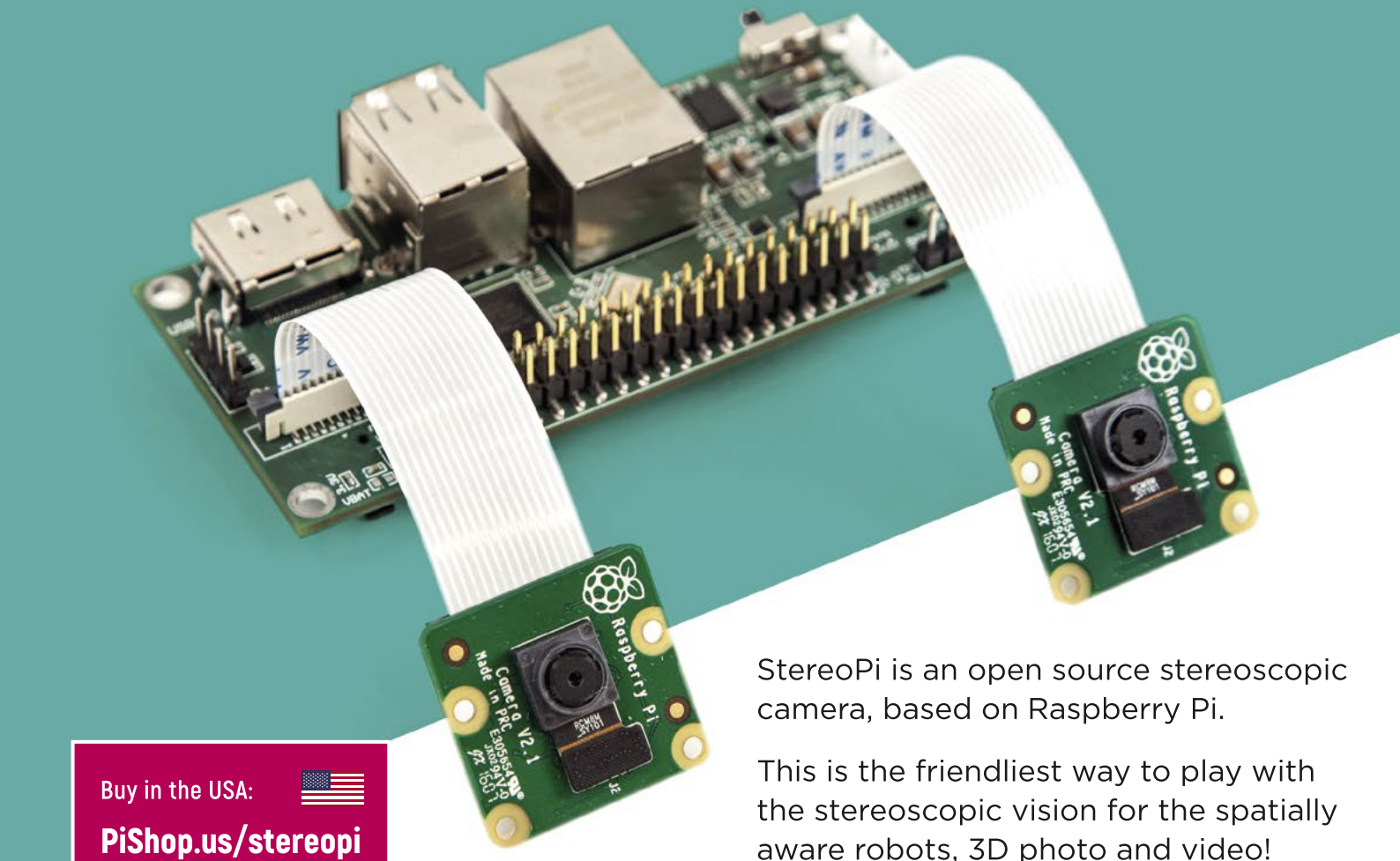


Pico Explorer Base

Kevin Johnson interview

WIN 1 OF 6 **SMARTIPI TOUCH PRO CASES**

Do you know
HOW ROBOTS SEE?



StereoPi is an open source stereoscopic camera, based on Raspberry Pi.

This is the friendliest way to play with the stereoscopic vision for the spatially aware robots, 3D photo and video!

Buy in the USA:
PiShop.us/stereopi

Buy in Canada:
BuyaPi.ca/stereopi

RASPBERRY PI INSIDE

Raspbian

STOCK RASPBIAN SUPPORT

OPEN SOURCE

CROWDFUNDED PROJECT

LinuxGizmos.com
"The StereoPi can capture, save, livestream, and process real-time stereoscopic video and images for robotics, AR/VR, computer vision, drone instrumentation, and panoramic video."

MickMake
"With it you can do things like, stream stereoscopic 3D video to YouTube, build real-time depth maps using OpenCV, create panoramics using Hugin and even a 3rd person view of real life. Cool."

Raspberry Pi Blog
"There are some excellent community efforts too, of which our current favourite is this nifty dual camera board."

Hackster News
"You can hook this up to YouTube, to Oculus Go, you can use it with OpenCV.. I cannot wait to start messing around with these because it's basically a dream come true."

Elettronica In · LinuxGizmos.com · Geeky Gadgets · OPENELECTRONICS · electronics-lab

iPod Classic Spotify player

Recreating Apple's iconic iPod Classic as a Spotify player may seem like sacrilege, but it works surprisingly well. **Rosie Hattersley** listens in

MAKER

Guy Dupont

A software and mobile phone developer, Guy enjoys acquiring new skills while tackling tech projects.

magpi.cc/ipodspotify

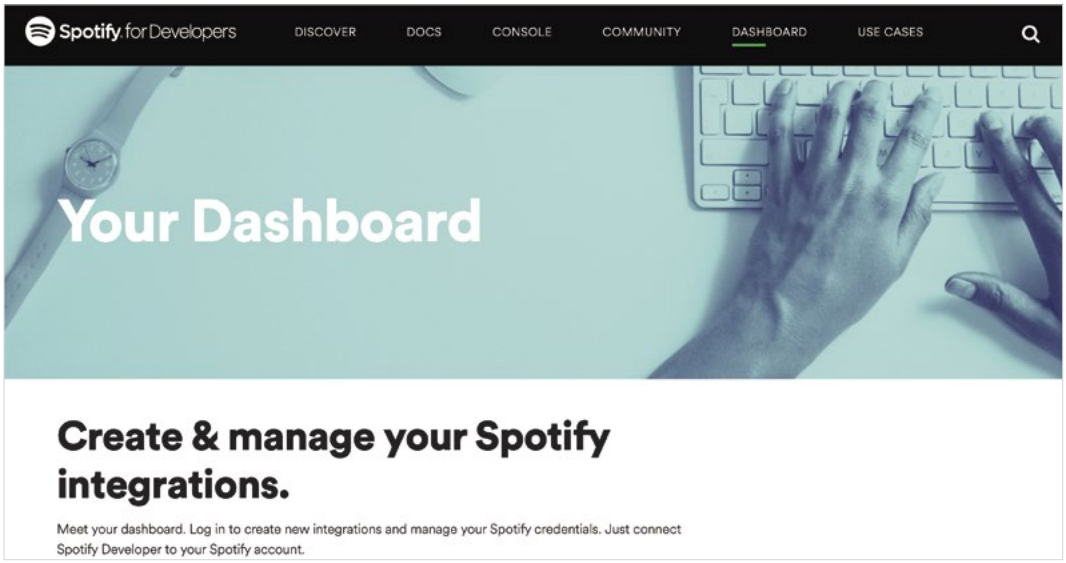
When the original iPod was launched, the idea of using it to run anything other than iTunes seemed almost blasphemous. The hardware remains a classic, but our loyalties are elsewhere with music services these days. If you still love the iPod but aren't wedded to Apple Music, Guy Dupont's Spotify hack makes a lot of sense. "It's empowering as a consumer to be able to make things work for me – no compromises," he tells us. His iPod Classic Spotify player project cost around $130, but you could cut costs with a different streaming option.

"I wanted to explore what Apple's (amazing) original iPod user experience would feel like in a world where we have instant access to tens of millions of songs," he says. "And, frankly, it was really fun to take products from two competitors and make them interact in an unnatural way."

Guy's career spans mobile phone app development, software engineering, and time in recording studios in Boston as an audio engineer, so a music tech hack makes sense. He first used Raspberry Pi for its static IP so he could log in remotely to his home network, and later as a means of monitoring his home during a renovation project. Guy likes using Raspberry Pi when planning a specific task because he can "program [it] to do one thing really well... and then I can leave it somewhere forever", in complete contrast to his day job.

Mighty micro

Guy seems amazed at having created a Spotify streaming client that lives inside, and can be controlled by, an old iPod case from 2004. He even recreated the iPod's user interface in software,



Warning!
Conductive electricity

If you're upcycling an MP3 player with a metal case, make sure you tape off the conductive components so you don't short-circuit them.

magpi.cc/electricalsafety

▶ This project streams music using Spotify using an API from the music service's development page

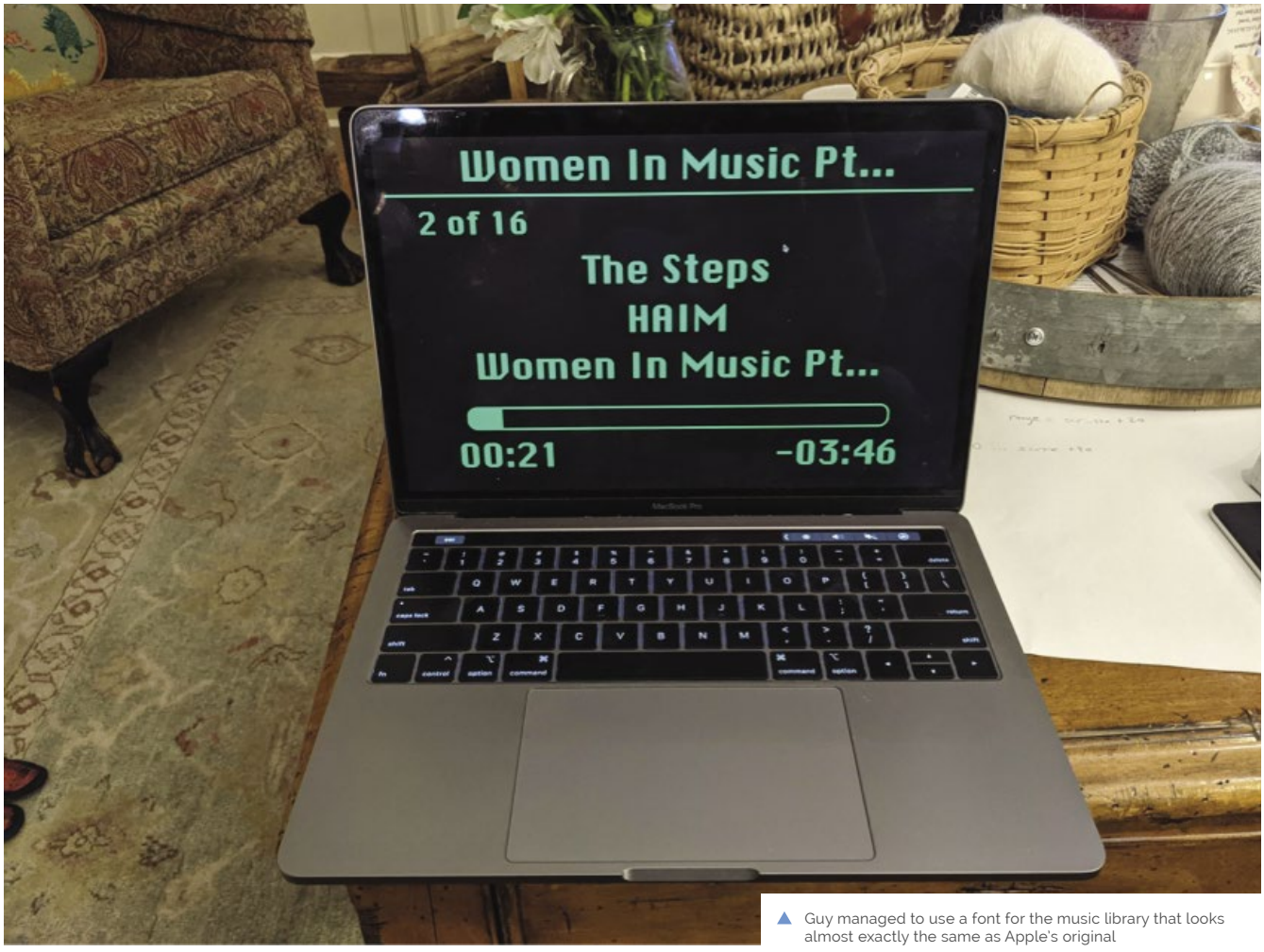
The 2004 iPod shell hides a Raspberry Pi Zero W connected to a Spotify Premium account

The display is a faithful facsimile of the original Apple iPod

Just like the iPod Classic, running a finger round the click wheel scrolls through the list of tracks or artists

Quick **FACTS**

- Guy hacked his RV by adding a Raspberry Pi media server...
- ...much to the surprise of its co-owner (his dad!)
- He's also Raspberry Pi-enabled his record player...
- ...it streams albums around the house
- iPod was one of the first devices to use the VideoCore processor found in Raspberry Pi



▲ Guy managed to use a font for the music library that looks almost exactly the same as Apple's original

right down to the font. A ten-year-old article about the click wheel provided some invaluable functionality insights (**magpi.cc/ipodclickwheel**) and allowed him to write code to control it in C. Guy was also delighted to discover an Adafruit display that's the right size for the case, doesn't expose the bezels, and uses composite video input so he could drive it directly from Raspberry Pi's composite out pins, using just two wires. "If you're not looking too closely, it's not immediately obvious that the device was physically modified," he grins.

Guy's retro iPod features a Raspberry Pi Zero W. "I'm not sure there's another single-board computer this powerful that would have fit in this case, let alone one that's so affordable and readily available," he says. "Raspberry Pi did a miraculous amount of work in this project." The user interface is a Python app, while Raspberry Pi streams music from Spotify via Raspotify (**magpi.cc/raspotify**), reads user input from the iPod's click wheel, and drives a haptic motor – all at once.

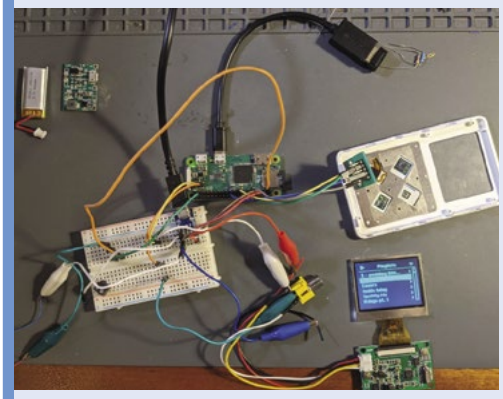
Most of the hardware for the project came from Guy's local electronics store, which has a good

> It's really empowering as a consumer to be able to make things work for me – no compromises

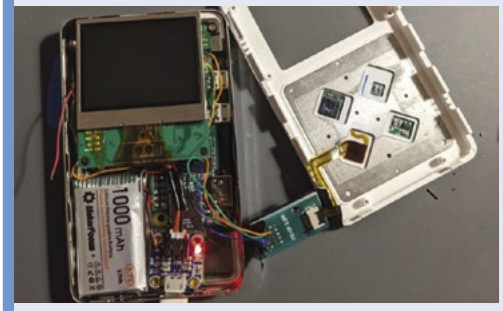
▲ Guy previously used Raspberry Pi to stream albums around his home

Music on the move

01 Carefully open up the MP3 player you're repurposing, remove its existing components, retaining anything you'll reuse, then check the replacement parts will fit within the chassis.



02 Install the C-based haptic code on Raspberry Pi Zero. Connect Raspberry Pi, display, headers, and leads. Then power on and check the click wheel data is recognised and drives the haptic motor.



03 Tape off any conductive elements in the case to prevent short-circuiting. Test Raspotify and install on Raspberry Pi Zero before mounting it in your new music player and powering it up.

▲ Impressively, the click wheel scrolling mechanism works just like the iPod Classic's

line in Raspberry Pi and Adafruit components. He had a couple of attempts to get the right size of haptic motor, but most things came together fairly easily after a bit of online research. Help, when he needed it, was freely given by the Raspberry Pi community, which Guy describes as "incredible".

Things just clicked

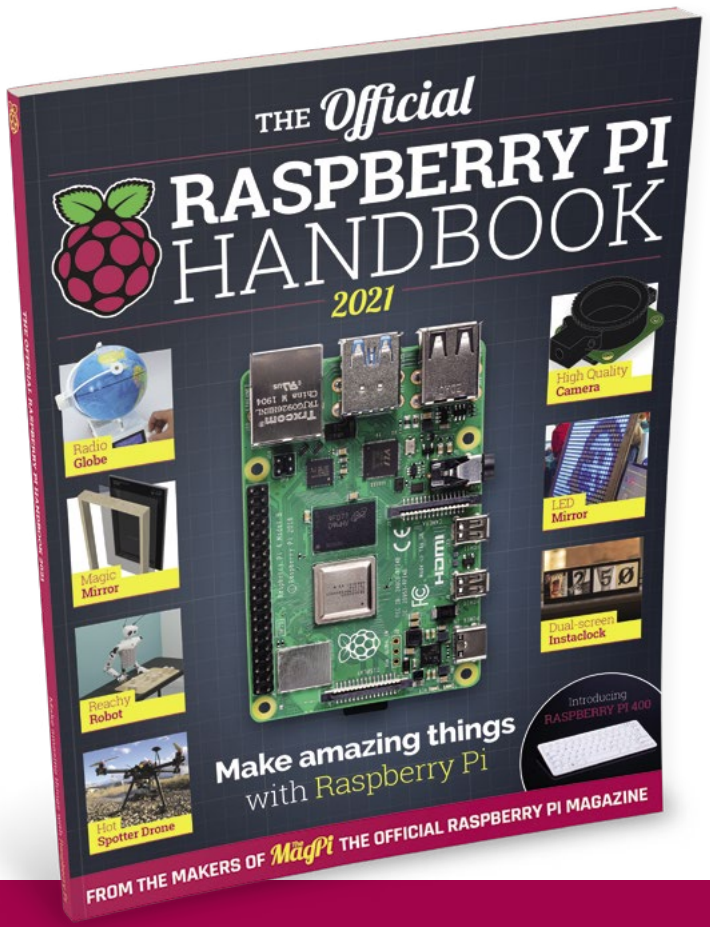
Part of the fun of this project was getting the iPod to run a non-Apple streaming service, so he'd also love to see versions of the iPod project using different media players. You can follow his instructions on GitHub (**magpi.cc/ipodspotify**).

Next, Guy intends to add a DAC (digital to analogue converter) for the headphone jack, but Bluetooth works for now, even connecting from inside his jacket pocket, and he plans to get an external USB DAC in time.

THE *Official* RASPBERRY PI HANDBOOK 2021

200 PAGES OF RASPBERRY PI

- Get started with Raspberry Pi, electronics, and more
- Be inspired by incredible projects made by other people
- Learn how to code and make with our step-by-step tutorials
- Find out about the top kits and accessories for your projects



Buy online: magpi.cc/store

Kay-Berlin Food Computer

Rob Zwetsloot talks to teacher Chris Regini about the incredible project that his students are working on

MAKER

Noah Kay and Noah Berlin

Eighth grade students at West Hollow Middle School in Long Island, New York that are the leads on this Food Computer.

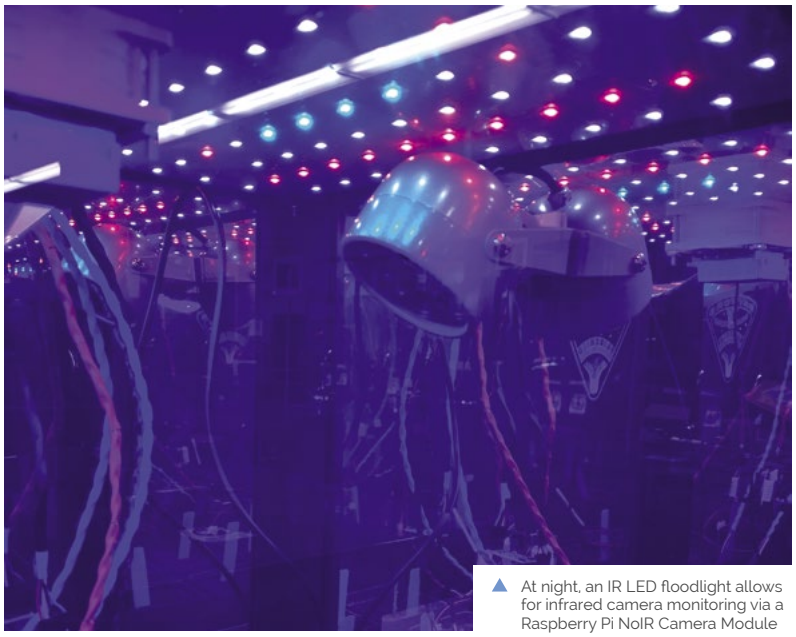
When we think of garden automation here at *The MagPi*, we often think of basic measures like checking soil moisture and temperature. The Kay-Berlin Food Computer, named after student creators Noah Kay and Noah Berlin, does a lot more than that. A lot more.

"It is a fully automated growth chamber that can monitor over a dozen atmospheric and root zone variables and post them to an online dashboard for remote viewing," Chris Regini tells us. He's supervising both Noahs in this project. "In addition to collecting data, it is capable of adjusting fan speeds based on air temperature and humidity, dosing hydroponic reservoirs with pH adjustment and nutrient solutions via peristaltic pumps, dosing soil with water based on moisture sensor readings, adjusting light spectra and photoperiods, and capturing real-time and time-lapsed footage using a [Raspberry Pi] NoIR Camera Module in both daylight and night-time growth periods."

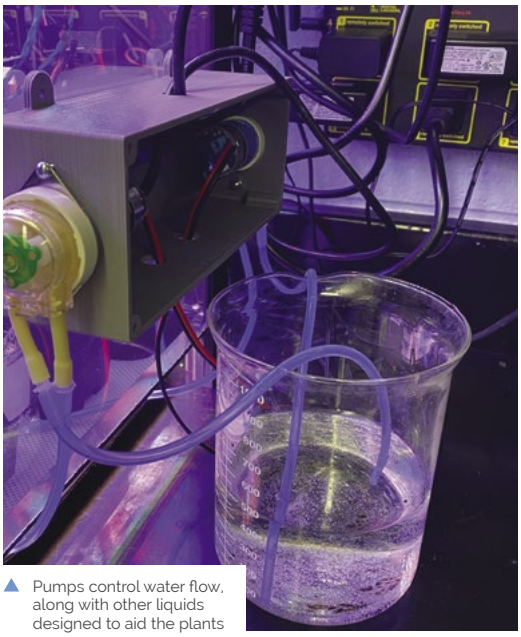
Everything can be controlled manually or set to be autonomous. This isn't just keeping your garden looking nice, this is the future of automated farming.

Seeds of knowledge

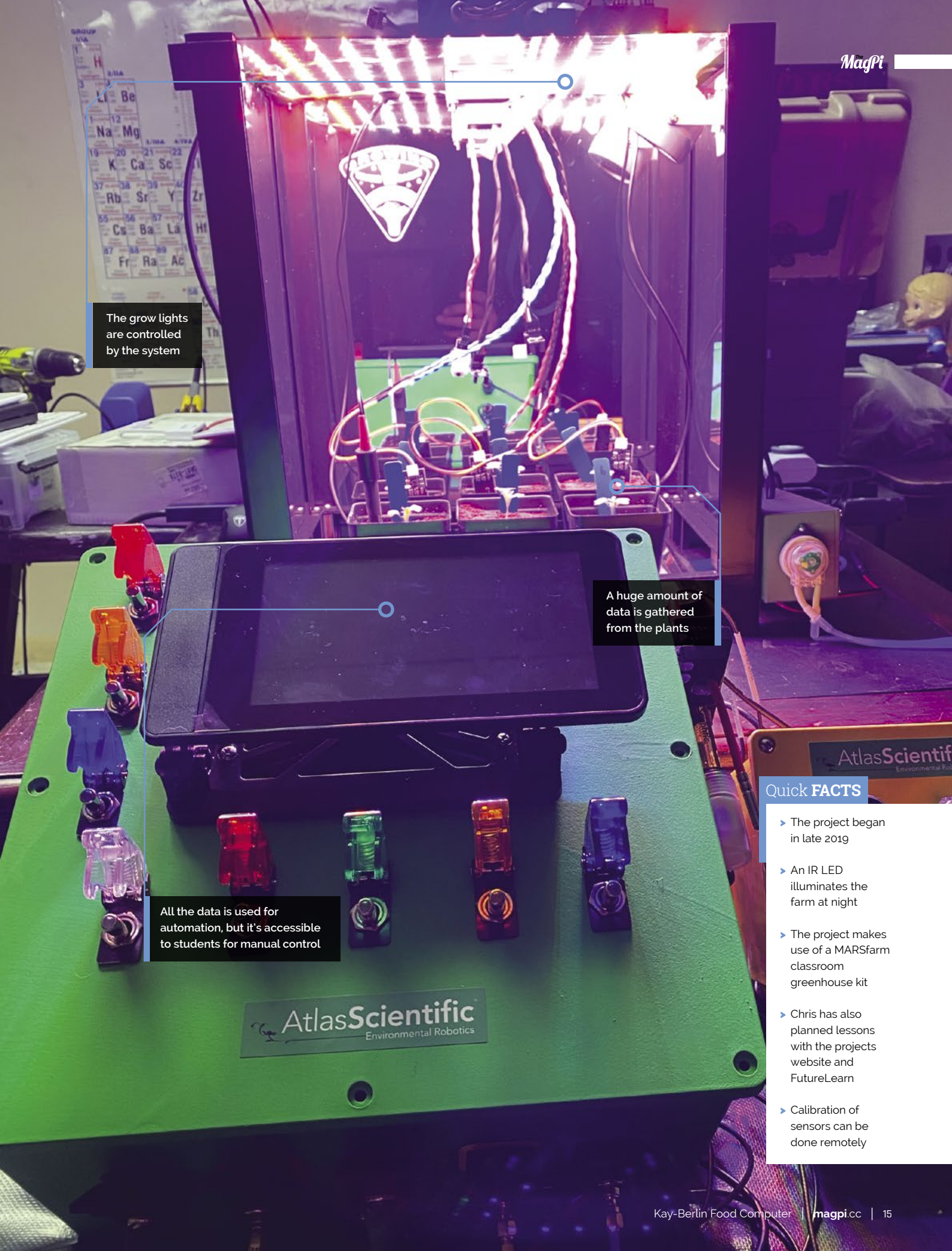
"The idea originated from the long-standing MIT food computer project and lots of open-source collaboration in both the agriculture and Raspberry Pi communities," Chris explains. "We've always had the hopes of creating an automated growing system that could collect long-term data for use in the ISS during space travel, or in terrestrial applications where urbanisation or climate concerns required the growth of food indoors."



▲ At night, an IR LED floodlight allows for infrared camera monitoring via a Raspberry Pi NoIR Camera Module

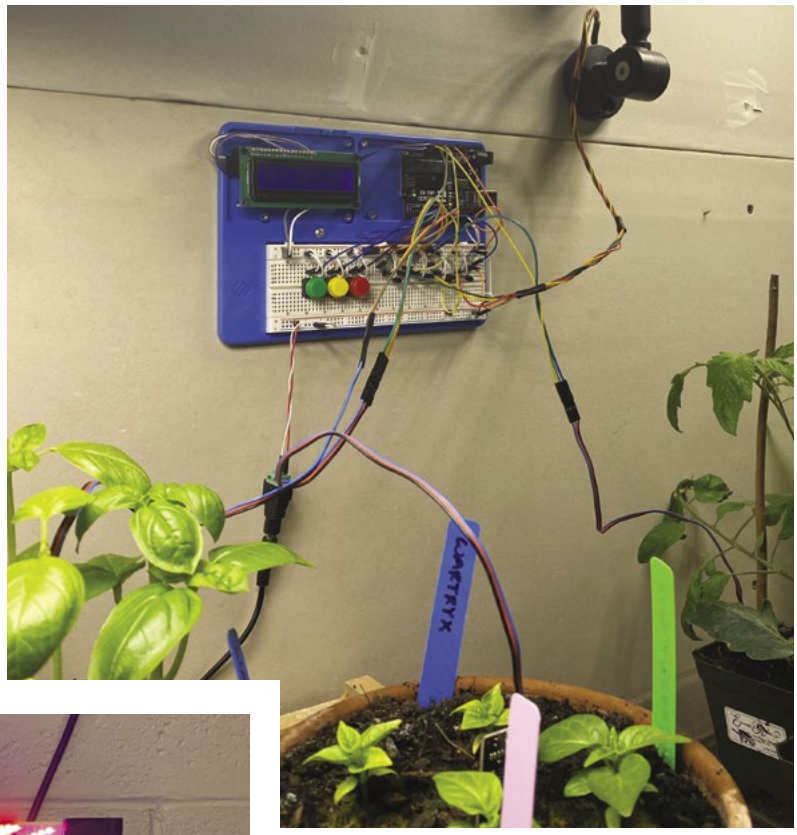


▲ Pumps control water flow, along with other liquids designed to aid the plants



Quick **FACTS**

- The project began in late 2019
- An IR LED illuminates the farm at night
- The project makes use of a MARSfarm classroom greenhouse kit
- Chris has also planned lessons with the projects website and FutureLearn
- Calibration of sensors can be done remotely



There are six plants in the box, allowing for a lot of data collection

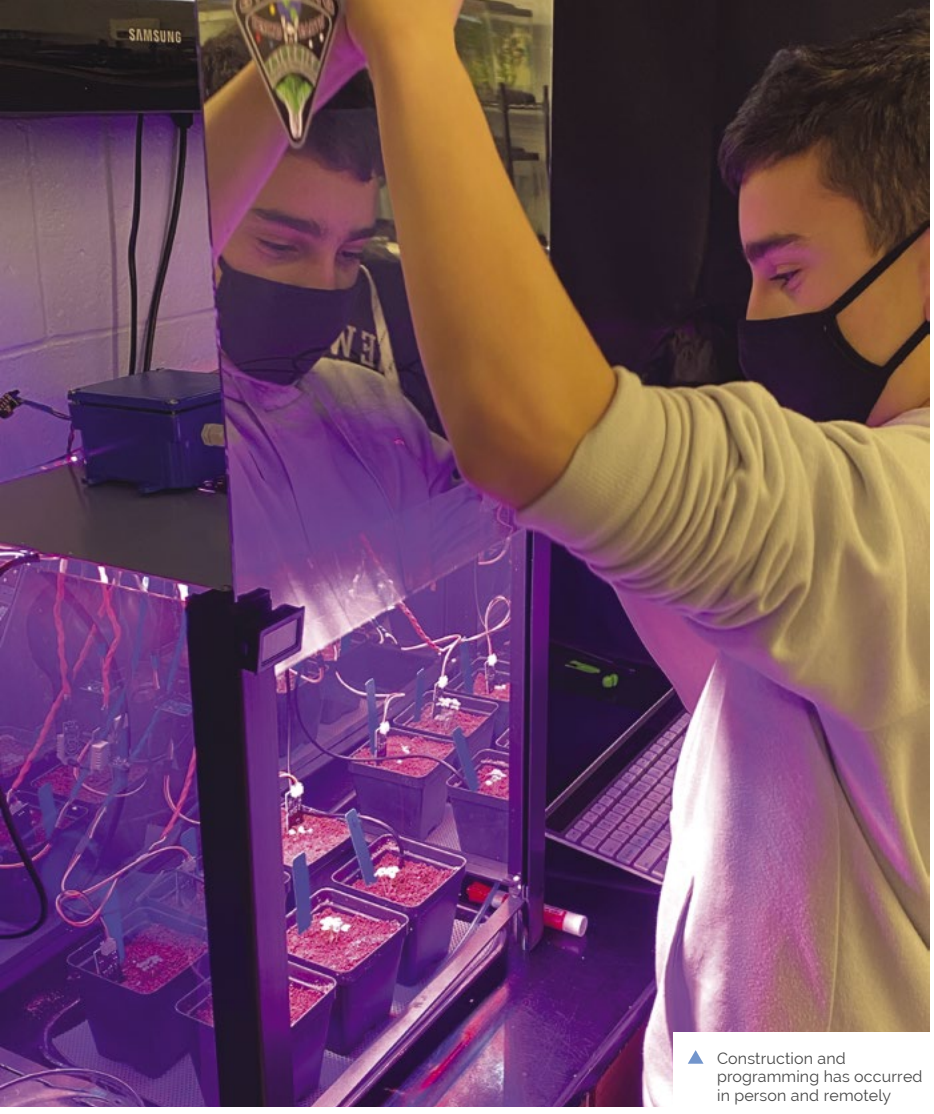
With students doing a lot of learning from home in the past year, having such a system accessible online for interaction was important for Chris: "Adding a layer that could keep students engaged in this endeavour during remote learning was the catalyst that truly spurred on our progress."

This level of control and web accessibility is perfect for Raspberry Pi, which Chris, his students, and his Code Club have been using for years.

"The fact that we had access to the GPIOs for sensors and actuators, as well as the ability to capture photo and video was great for our application," Chris says. "Being able to serve the collected data and images to the web, as well as schedule subroutines via systemd, made it the perfect fit for accessing our project remotely and having it run time-sensitive programs."

The computer has been in development for a while, but the students working on it have a wide range of skills that have made it possible.

"We have had a dedicated nucleus of students that have spent time learning plant science, electronic circuitry, Python, developing UIs, and creating housings in CAD," Chris explains. "They all started as complete beginners and have benefited greatly from the amazing tutorials available to them through the Raspberry Pi Foundation website, as well as the courses offered on FutureLearn."



Construction and programming has occurred in person and remotely

> The system does a fantastic job collecting data and allowing us to visualise it via our Adafruit IO+ dashboards

Grow beyond

The project is ongoing – although they're already getting a lot of data that is being used for citizen science.

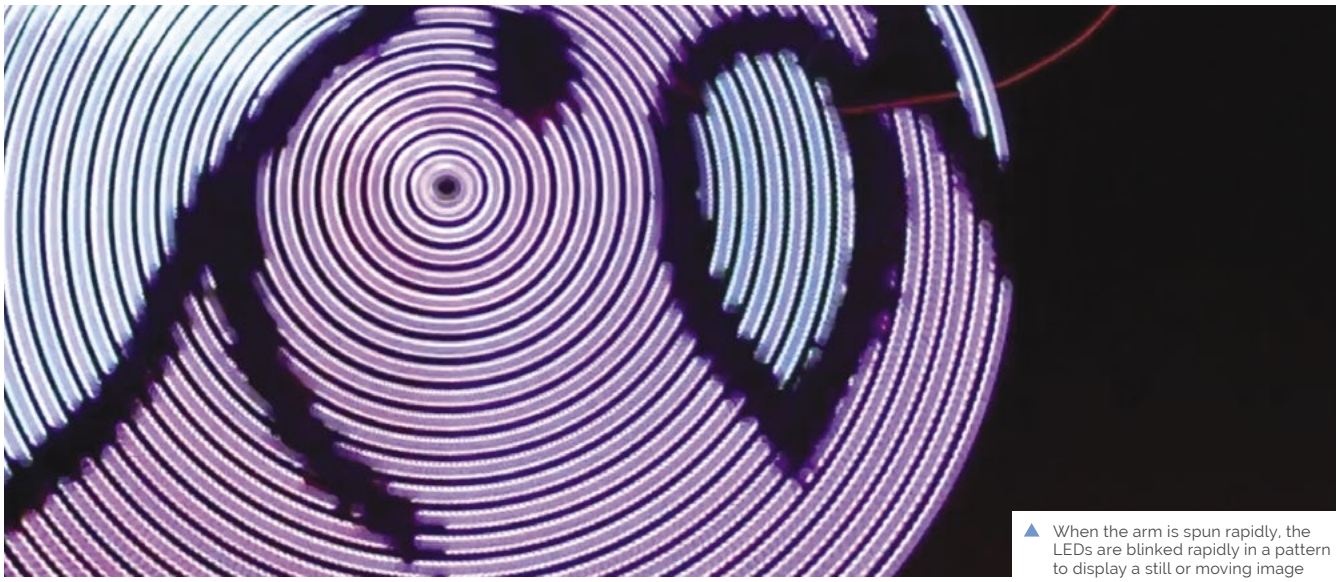
"The system does a fantastic job collecting data and allowing us to visualise it via our Adafruit IO+ dashboards," Chris says. "Upgrading our sensors and actuators to more reliable and accurate models has allowed the system to produce research level data that we are currently sharing in a citizen science project called Growing Beyond Earth. It is funded by NASA and is organised through Fairchild Botanical Gardens. We have been guided along the way by industry professionals in the field of hydroponics and have also collaborated with St Louis-based MARSfarm to upgrade the chamber housing, reflective acrylic panels, and adjustable RGBW LED panel. Linking our project with scientists, engineers, researchers, and entrepreneurs has allowed it to really take off."

Food science in action

01 "The entire system has a network of sensors... which monitor atmospheric variables of air temperature, humidity, CO_2, O_2, and air pressure. Root zone variables include soil moisture by way of capacitive resistance sensors, reservoir levels by way of float switches, water/soil temperature, water/soil pH, nutrient levels by way of EC, dissolved oxygen, and water quality/root health by way of ORP (oxidation reduction potential)."

02 With this data, the computer controls an array of pumps that provide water, nutrients, and pH adjustments. Grow lights are controlled and tweaked, fans are adjusted, heating mats activated, and it even switches between day and night vision for the camera.

03 "All data is viewable in real time and historically, while all actuators are controllable via our Adafruit IO dashboard. Adjustment parameters like RGBW light spectrum, photoperiod, fan speed, pH and temperature set points, and liquid dosing volumes can be chosen here as well."



When the arm is spun rapidly, the LEDs are blinked rapidly in a pattern to display a still or moving image

POV Display

Putting Raspberry Pi Pico in a spin, this rotating LED arm can show moving images. **Phil King** ponders whether seeing is believing

MAKER

HomeMade Garbage

A family team of two children and a couple. They live on a handmade basis. Dad is good at electronic work, and mom is good at music production and web technology. Children are good at growing up well.

magpi.cc/garbage

Persistence of vision (POV) is the optical phenomenon in which the illusion of motion is created because the human brain – which can only process 10–12 frames per second – interprets a rapid sequence of still pictures as a continuous moving image. It's the basis of how animation works, as well as film and TV in general.

By rotating a strip of LEDs at high speed and syncing their blinking patterns, it's possible to create the illusion of a still or moving image. This is just what Japan-based family team of makers HomeMadeGarbage have done for their POV Display (**magpi.cc/povdisplay**), even spinning a Raspberry Pi Pico around with the LED strips.

Power of PIO

The makers tell us that, after creating a similar project using Sony's considerably more expensive Spresense board, they were inspired to try it with Pico, as they were "very surprised at the parallel high-speed operation of PIO." The latter is the unique Programmable Input/Output feature of Pico's RP2040 chip that enables the use of custom communication protocols in addition to the built-in I2C and SPI. This offers a faster way of outputting bit-banged data (even video) to non-standard devices with a deeper level of control, without tying up the main processor.

The POV Display uses two different PIO state machines (from the eight available on the chip) to control, in parallel, a pair of super-bright 24-LED APA102 (aka DotStar) strips on its rotating arm. The arm is spun at high speed by a Mabuchi RS-540SH motor, as commonly used in remote-control cars.

Each rotation is detected using a reflectance sensor on the arm and a small white marker underneath. HomeMadeGarbage say that they improved detection reliability by inserting "a filter and a Schmitt trigger between the reflectance sensor and Pico to prevent chattering."

In the software, written in C, the time taken for one rotation is divided by 1000 to sync the

> They've managed to spin the arm at up to 960 rpm while displaying an image at 1000 frames per rotation

blinking of LED patterns stored in a graphics array. "In order to leave a beautiful after-image, a speed of ten revolutions or more per second is required," reveal HomeMadeGarbage. Discovering that "the I/O of Raspberry Pi Pico can run very

Two 24-LED APA102 strips blink a synchronised pattern to show an image as the arm rotates

Mounted on the other end of the rotating arm, Raspberry Pi Pico controls the LEDs

Power to Pico and the LEDs is supplied via a wireless charging module

fast", they've managed to spin the arm at up to 960 rpm while displaying an image at 1000 frames per rotation.

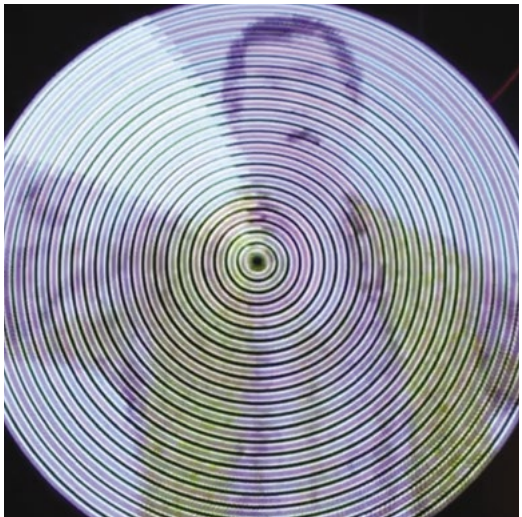
Spinning around

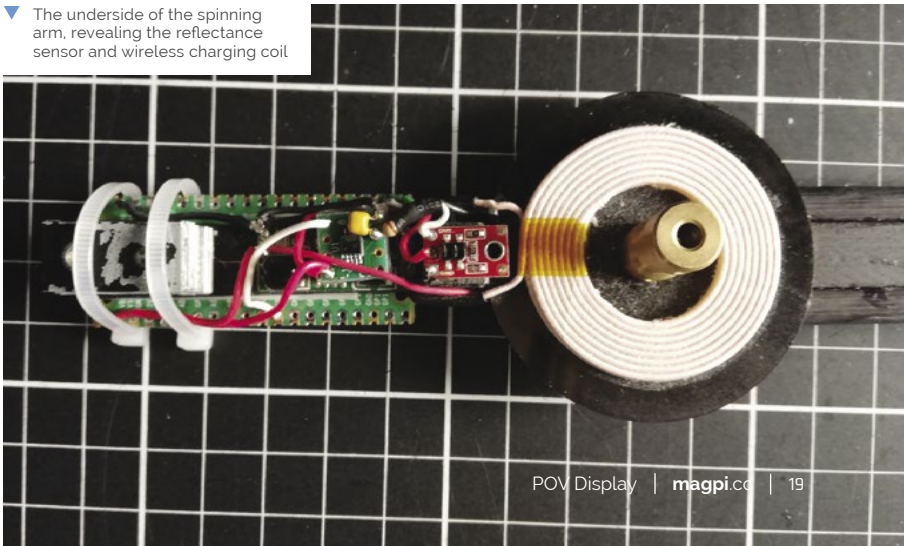
With everything, including Raspberry Pi Pico, whizzing around at high speed, you may well be wondering how it's powered without quickly causing a tangle of wires. The solution is the use of a wireless charging module, one of whose circular coils sits on top of the motor, the other in the underside of the arm centre. "5 V is supplied to the transmitting side of the wireless charging module, and the receiving side receives the voltage, [which] drives Pico, sensor, and LEDs."

After posting a video of the POV Display in action, HomeMadeGarbage found that people were "surprised at the high-speed operation." They now plan to make an improved version with a narrower-pitch LED bar. Also on the cards is a 3D POV Display using Pico – we can't wait to see that!

Quick FACTS

- The project took four days to complete
- The LED pattern is updated 1000 times per rotation
- The LED code is based on this example: **magpi.cc/picoapa102code**
- The arm spins at anything up to 960 rpm
- Everything is powered using a wireless charging module





The underside of the spinning arm, revealing the reflectance sensor and wireless charging coil

Autonomous Home Robot

When Nick Baddorf decided he wanted to make a robot to help around the house, he embarked on a very educational journey. **Nicola King** takes a look at a robotic work-in-progress

MAKER

Nick Baddorf

Nick loves to build autonomous robots, work on engines, and play with his dog Penny.

magpi.cc/nickbrobot

Making a robot from scratch, with no instructions, and teaching yourself all the necessary programming and various systems required is certainly no mean feat. So, let's applaud the tremendous efforts of Nick Baddorf, a US-based teenager who has created a unique robot completely under his own steam.

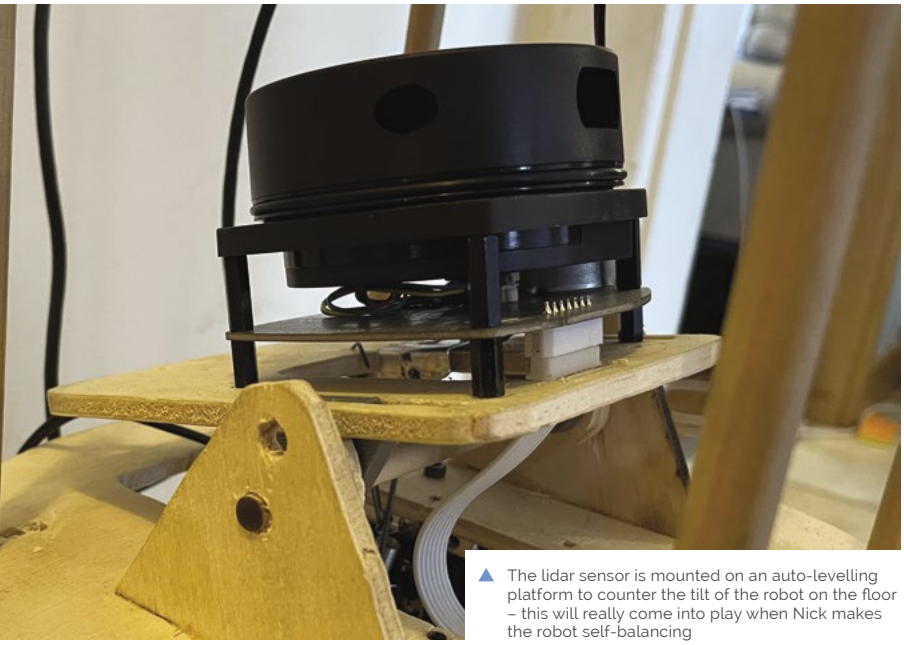
"I have always loved robots," Nick tells us. "Building my own autonomous robot has been a goal for me, as long as I can remember. I also wanted to make my robot useful. I am always making things but then end up taking them apart the next week because they didn't end up being useful. I challenged myself to make this robot useful, and help carry out tasks around the house."

And so, several years ago, work on the Autonomous Home Robot began and it's fair to say that the project has evolved over that time. "It started as a tiny remote-controlled, self-balancing robot," says Nick. "Now it is a big robot that navigates around the house!"

Nuts and bolts

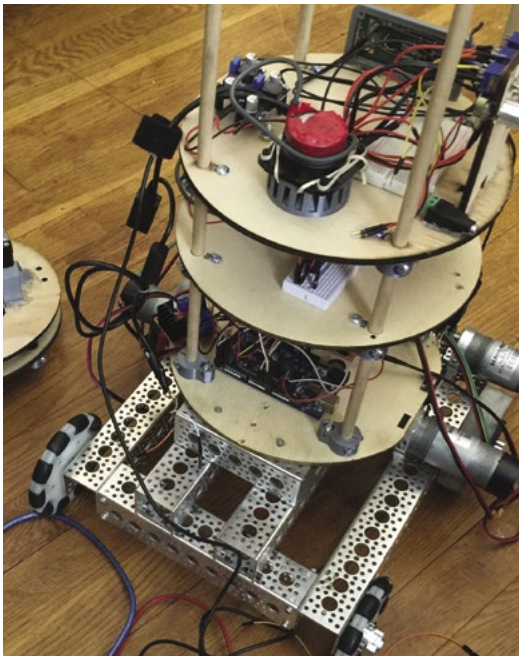
Nick's robot basically consists of two sections: the base and the head. "The base consists of a lidar laser scanner, Raspberry Pi, and Teensy," he reveals. "The lidar scanner, which is mounted on a small auto-levelling platform, sends its range data to Raspberry Pi. The [latter] is the brain which takes in the lidar data and streams it over to a master computer for processing. Raspberry Pi also sends drive messages to the Teensy, which handles motor control."

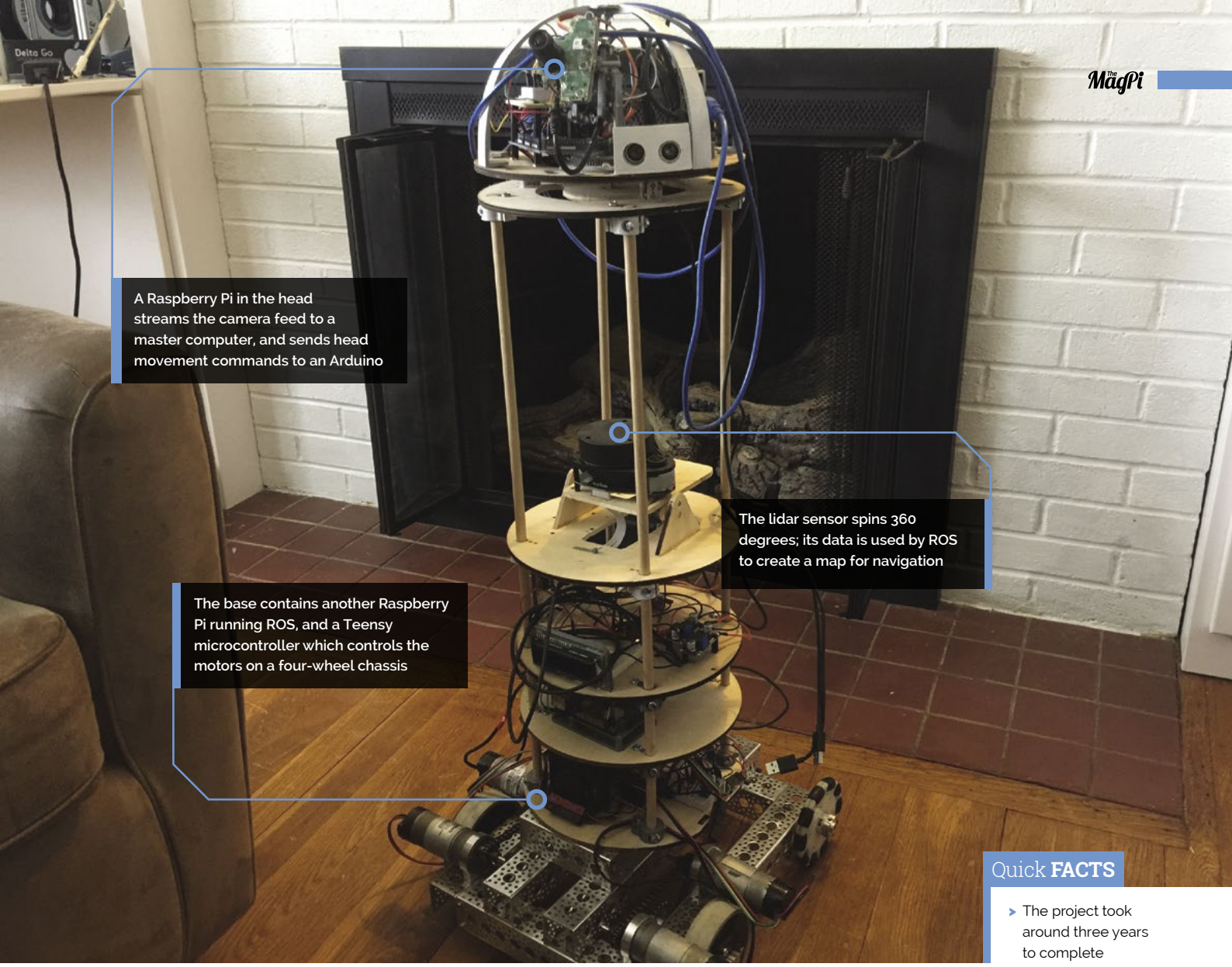
In the head of the robot, Nick has positioned a second Raspberry Pi, an Arduino, and a camera. "Similarly to the base, Raspberry Pi streams the camera feed over to the master computer, and sends head movement commands to the Arduino," he says.



▲ The lidar sensor is mounted on an auto-levelling platform to counter the tilt of the robot on the floor – this will really come into play when Nick makes the robot self-balancing

▼ On the lower part of the robot is a 12 V 10 A power system for the motors. A 5 V 16 A USB power brick is used for the two Raspberry Pi boards





> I challenged myself to make this robot useful, and help carry out tasks around the house

The robot navigates its way around the house by using various systems that Nick has installed, and it knows where it is in a room by using wheel rotations (odometry). Nick elaborates, "This odometry is then combined with the laser data to make a map. On the map, I can specify a point that the robot will drive to, and the robot will make its way there, while avoiding obstacles."

Robot research

There was a great deal of technical coding, along with languages and systems, to learn before Nick's robot came into being, including Python, C, HTML, and OpenCV. The hardest part was to learn Robot Operating System (ROS). "It took a lot of reading and studying to set up the robot. Once I was motivated enough to put in the time to really learn ROS, everything fell into place."

Nick is the first to admit that his robot is an ongoing project and is continually evolving, but the potential is there for it to perhaps carry things around the home, act as a robotic companion, and more. He's already made many finely tuned modifications: "I have added many things to the initial design, like tilting laser scanners, tilting cameras, a rotating head, and even the temporary four-wheel drive base,"

He also has plans for a lot more improvements, "including new motors, making the robot self-balancing, Raspberry Pi 4s, and a big battery so that the robot can completely navigate on its own."

Nick is clearly very self-motivated and has input many hours of hard work to get to this point. "I asked for feedback recently from my engineer friend who encouraged me to focus on getting the core of the robot working well and stop adding extraneous features. A combination of his help, along with many others', have helped make this robot work!"

Quick FACTS

- The project took around three years to complete (part-time)
- Nick won his first Raspberry Pi from a local group that started a coding program…
- …when he was ten years old!
- His other projects include a 'mind'-controlled smart light
- He's working on a Raspberry Pi-powered autonomous mobile robotic arm!

Haptic navigation

Satellite navigation may seem ubiquitous, but it doesn't work for everyone. **Rob Zwetsloot** finds out about a project that will make it truly accessible

MAKER

Sukriti Chadha

A product manager at Spotify who works on accessibility and also helps develop the WCAG group's guidelines for accessibility online and on mobile.

magpi.cc/sukriti

Accessibility is an oft overlooked aspect of modern technology development. Sukriti Chadha's job is to not overlook it, and she's become an advocate for accessibility in the process. This is exemplified by one of her recent projects.

"For people with hearing impairments, walking on a busy street while looking at a phone screen can be a challenge, and maybe even dangerous," she explains. In addition, "while driving, they might not be able to effectively use voice navigation and take their eyes off the street to look at directions. For those with visual impairments, navigating indoor spaces that are noisy, or experiences such as museums, can be less enjoyable because of voice navigation interruptions."

There are other ways to relay the information, though, such as how Sukriti has made use of touch: "This open-source solution uses a Raspberry Pi Zero W and a mobile phone to relay turn-by-turn instructions with haptic feedback, more commonly known as vibrations, over an SSH connection with the mobile device."

> "This open-source solution uses a Raspberry Pi Zero W and a mobile phone to relay turn-by-turn instructions with haptic feedback"

A running thread

Sukriti tells us that around 466 million people worldwide have disabling hearing loss, 34 million of whom are children. The WHO believes that this number will double in the next 30 years.

"Having worked in the distracted navigation space at a Tel Aviv startup in 2014, I was acutely aware of the limitations of voice navigation for those with hearing impairments," she says. "My interest in the space was reignited during a conversation with Pete Cossaboon, who runs obstacle races as a blind athlete. I learned that he wanted to be more independent in navigating the space he is in. Both of these problems could potentially be solved with a haptics-based solution, and this is the first version of it."

Using Raspberry Pi made sense in this context due to its low price and small size, and Sukriti found it easier to work with too. "For me as a developer,

The sensors vibrate to indicate the next direction

A Raspberry Pi Zero W is powered on by the phone and connects via SSH

Navigation is handled by the phone as it would be normally

Quick FACTS

- The entire setup costs $20
- Sukriti has used it attached to a leg while driving…
- …and in a small messenger bag for on foot
- The vibrations are provided by two 1.5 V motors
- It can be powered by a smartphone that has USB-C

▲ You can rest it on your person while driving to get directions

programming on Raspberry Pi is intuitive, especially since I was looking for an interface between a mobile phone and a physical device."

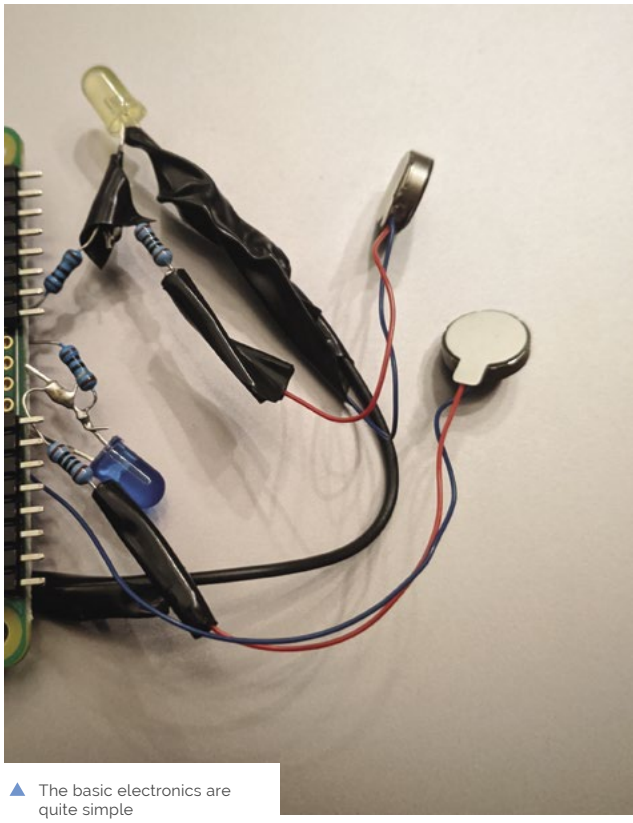
Touch and go

The system currently works using Mapbox, a third-party mapping and navigation service, which provides the directions that are then translated to vibrations on the device. It can be extended to other frameworks with an API that allows for HTTP requests to be sent to Raspberry Pi, such as iOS and mobile web.

"It works really well in terms of navigation … for people with hearing impairments," Sukriti reveals. "I can see its applications in VR navigation as well. The solution would be even more useful for visually impaired users with proximity sensors to help avoid obstacles, in addition to navigation outdoors."

Work is ongoing with the project, with Sukriti wanting to add more haptic sensors for different navigation commands, proximity sensors, PWM output for varying intensity, and more.

"I have tested the prototype with a couple of people, and the feedback has been really positive," she says. "I have personally been using it on runs, so my music is not interrupted if I go on unfamiliar paths."



▲ The basic electronics are quite simple

CamChess

Chess is in vogue, prompting a proficient player to finally create a version based on a club-standard playing board. **Rosie Hattersley** learns how

MAKER

Geoff Fergusson

Retired computing consultant Geoff's last major professional assignment was designing a battlefield logistics system, playing neatly to his chess-honed tactical strengths.

magpi.cc/camchess

Geoff Fergusson is the former captain of a county chess team, so he knew exactly what to build with his Raspberry Pi. "It was natural that I should build a chess project," explains Geoff. "*The Queen's Gambit* TV series has spurred an interest in the game, so the timing is right."

Playing against an on-screen computer opponent just isn't the same, says Geoff. "It loses the social aspect." Furthermore, it takes time to adapt from on-screen diagrams to playing with a physical board and pieces – something he imagines more people doing now that the TV show has given chess a moment in the spotlight. "The ideal practice is to train with the same board and pieces as you would use at the local club," he says, but it's prohibitively expensive for schools and many clubs, which is where his CamChess project fits in. "I wanted to build a system that would work with standard plastic pieces on a standard vinyl board," and also record the moves.

A canny move

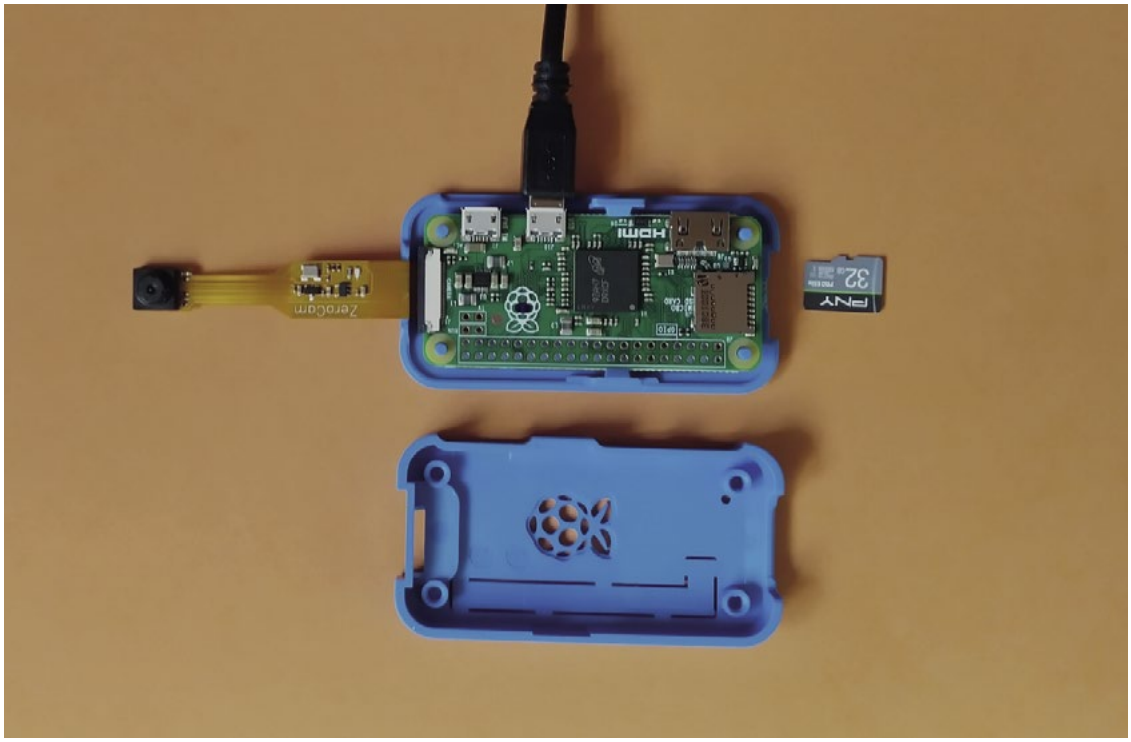
CamChess combines the best elements of an electronic board with the ability to record moves, but costs only around £50 ($70) excluding Raspberry Pi 4.

After retiring 20 years ago from his job as a consultant for large-scale computing systems, Geoff has racked upt experience of building small-scale projects. Despite a 30-year break from it, he can still write code. "Python was a bit of a culture shock," he reveals, "but I soon adapted." Geoff made full use of standard Python modules, along with OpenCV and NumPy for image processing, and Stockfish as the chess engine.

A Raspberry Pi Zero with an attached ZeroCam worked well for capturing images of the chess board. Mounting the ZeroCam on a door-frame, using some sticky tape high above the centre of the chess board, helped minimise lens distortion. Raspberry Pi Zero takes pictures on command and sends them back to another computer for analysis. Geoff used a Raspberry Pi 4 to do this, but says any "non-antique" Linux, Windows, or Mac computer would work.

Match ready

Due to congested WiFi channels where he lives, he opted for Ethernet to connect Raspberry Pi 4 and Raspberry Pi Zero and send images using the USB cable that also powers the latter. These images tell the computer the latest chess move that has



A Raspberry Pi Zero with ZeroCam is fixed to the ceiling above

> I wanted to build a system that would work with standard plastic pieces

been made, which python-chess then replicates on-screen. Having experimented with the image comparison method used by Realtime-OpenCV-Chess, Geoff thrashed out a method that determines whether each chess square is empty or occupied, and whether it's a white piece or a black piece.

It was a challenge to get the board to configure itself automatically, but Geoff eventually worked out how to do so using standard threshold optimisation techniques. CamChess now only needs to be shown the start position, and works out its internal settings from there. CamChess's difficulty level can be set to engage the hobbyist player and they can make moves for both sides until they reach a position that is of interest to them.

While he's largely happy with the existing build, Geoff hopes others will make their own versions of CamChess and let him know how they get on. We did ask him about his own next move, but he's keeping mum.

The ZeroCam picks up moves on the board and displays them on screen

Quick FACTS

- Geoff advises wannabe makers seek out similar projects to theirs
- Emulating others' code and methodology provides a good head start
- Geoff had admired Vatsal Parsaniya's Realtime-OpenCV-Chess (**magpi.cc/opencvchess**)
- ...which deduces chess moves by subtracting one image from another
- CamChess was also inspired by the Raspberry Turk chess robot (**magpi.cc/raspberryturk**)

Automated Camera-Based Drone Landing System

When Dr Chinthaka Premachandra and his team looked to autonomously land a drone using our favourite computer, it was certainly no pie in the sky idea, as **David Crookes** discovers

MAKER

Dr Chinthaka Premachandra

Chinthaka is an Associate Professor in the Department of Electronic Engineering at the Graduate School of Engineering in the Shibaura Institute of Technology.

magpi.cc/dronelander

Gravity dictates that what goes up must eventually come down but, when flying a drone, it helps if the landing is as smooth as possible.

That's doubly true for an unmanned drone since a crash will not only render it unusable, it could also potentially put lives at risk.

As such, researchers have been looking at ways to ensure autonomous aerial vehicles can identify and make use of a safe landing spot. "This is especially important when the drones are operating around disaster sites," says Dr Chinthaka Premachandra from the Shibaura Institute of Technology in Japan.

To do this, Chinthaka has been leading a team in using a standard radio-controlled quadrocopter drone fitted with a Raspberry Pi 3B+ and a mini camera. They've worked on an automatic landing system, allowing the drone to be brought down to the ground without mishap.

"Automatic landing is a kind of automatic flight towards a specific landing spot and to achieve this with a drone it must find the landing spot itself," Chinthaka explains. "I believe that landing spots can be recognised by processing the images from an on-drone camera, but this image processing needs to be implemented in real-time, generally less than 15 milliseconds."

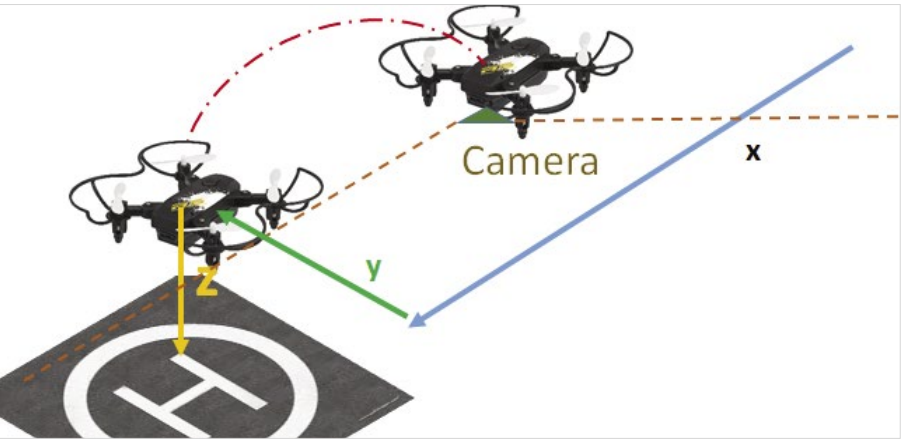
Be snappy

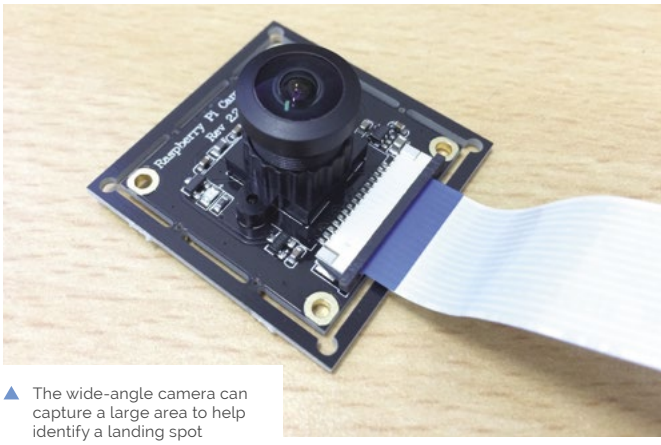
It's this need for fast on-board processing that led Chinthaka to consider using Raspberry Pi. "The idea is that flight control is conducted as soon as the images are processed, so we not only needed a good, lightweight camera but a lightweight on-drone computer too," he reveals. "Raspberry Pi 3B+ is obviously lightweight and it can be easily implemented within a drone. Raspberry Pi 4 can be used for this too."

The camera was chosen because it has a wide angle. "It's wider than the other cameras that are compatible with Raspberry Pi, so it can capture a large area and this capturing ability makes it easy to identify a landing spot," says Chinthaka. A depth camera was also important. "During the drone landing, the ground object information can be easily recognised with a depth camera rather than a 2D camera."

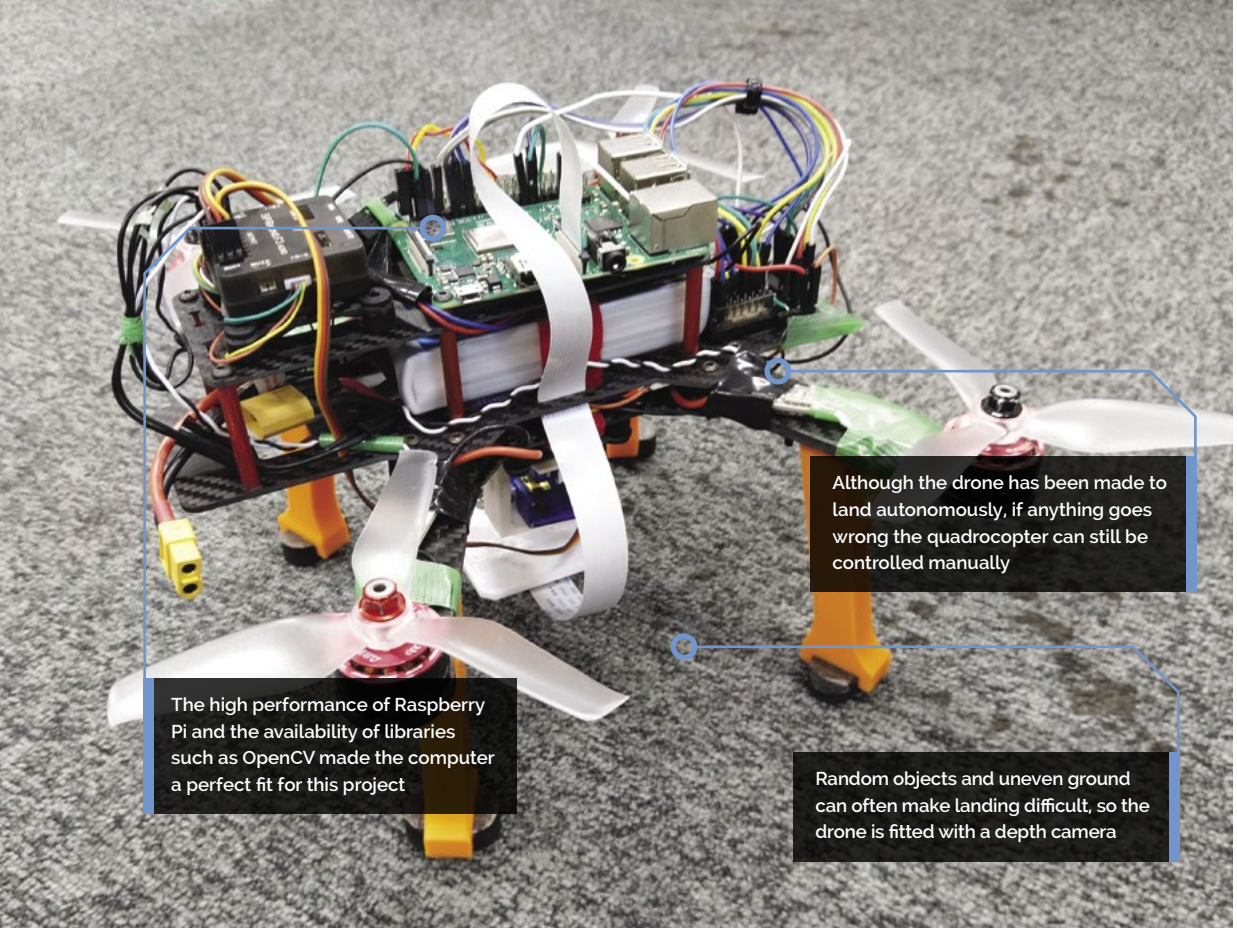
For optimal results, the camera lens is kept horizontal, even when the drone body is not in

▼ The drone looks for the landing pad by monitoring captured images before moving to the landing spot, hovering, and determining that it's safe to land





▲ The wide-angle camera can capture a large area to help identify a landing spot



Warning!

Be careful: The spinning blades on a drone can cause injuries. Make sure you fly your drone safely and legally.

magpi.cc/dronecode

Quick **FACTS**

- Raspberry Pi controls the drone's safe landing
- The drone is a standard radio control model
- It lands at the centre of a landing pad
- The height control operates at 20Hz
- Computing time is down to three milliseconds

this position during the flight. "If the lens was not always horizontal, then it would be difficult to smoothly capture the landing spot during flight because of the shaking of the drone," Chinthaka explains. "We made our own lightweight gimbal to keep the camera lens horizontal but a commercially available gimbal may also be used for this task."

> Some parts of the software include OpenCV library functions

H marks the spot

The idea is that the drone looks out for an H-shaped symbol placed on the ground at the landing location. Using software created by the team, the image is processed and converted into physical co-ordinates to generate a horizontal feedback.

"Some parts of the software include OpenCV library functions," says Chinthaka. "We also wrote software for the landing process, but developing the necessary algorithms to detect the landing spot in real-time was the greatest challenge."

Even so, the researchers were able to get the image processing time down to three milliseconds. This allows the drone to quickly fly over to the landing spot, hover over it, and land vertically, all controlled by Raspberry Pi. "It's been a big success and we expect it will have a wide number of future uses," Chinthaka concludes.

▲ A gimbal was built by combining two servos with 3D-printed parts and this ensures the camera, which sits at the very bottom of the setup, is stabilised

The Smart Crust-Cutting Robot

This innovative invention by Andrew DeGonge is arguably the best thing since sliced bread, as **David Crookes** discovers

MAKER

Andrew DeGonge

Andrew is a computer engineer in the med-device industry who has a passion for building unique and over-engineered robotics projects.

magpi.cc/smartcrust

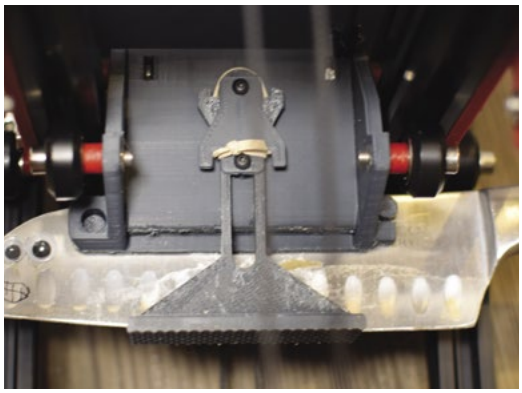
Warning!
Sharp edges

Be very careful when using knives and never work on a device such as this with the power switched on.

There's nothing quite like a good bread debate to get people talking. Do you cut sandwiches into rectangles or triangles? Are you calling it a roll, bap, barm, or muffin? And do you eat your crusts – thereby 'guaranteeing' your hair will grow curly – or slice them off and throw them away? For Andrew DeGonge, the latter question has a clear-cut answer: crusts are to be chopped. Indeed, he's so determined that the sides of bread are banished, he's used his loaf and created an automatic guillotine-like device to do the hard work for him. "Crust in general isn't my main problem. Cheap bagged bread is," he tells *The MagPi*. "It's all so mushy and the crust is even more of a burnt, soggy mush. So, like any reasonable adult, I prefer bagged bread without it to make it a bit more tolerable, although I do love the crust on a good sourdough or Italian loaf."

No crumby thought

The idea for a crust-cutter rolled into Andrew's head when he was considering building a robot with computer vision. "I wanted it to be original so I thought to myself, 'what's a problem many people have that doesn't have an automated solution?' I realised there was no way to automatically cut the crust off sandwiches so decided to create one, just because I can."



There are delays in key points of the final routines, and cutting board rotations are performed before a chop to give advanced warning before cutting

Andrew spent ten hours creating a CAD drawing of the project, ensuring enough room for the necessary motors and knife. "I went with stepper motors for the motion control and used V-slot extrusions and wheeled gantries for linear motion," he explains.

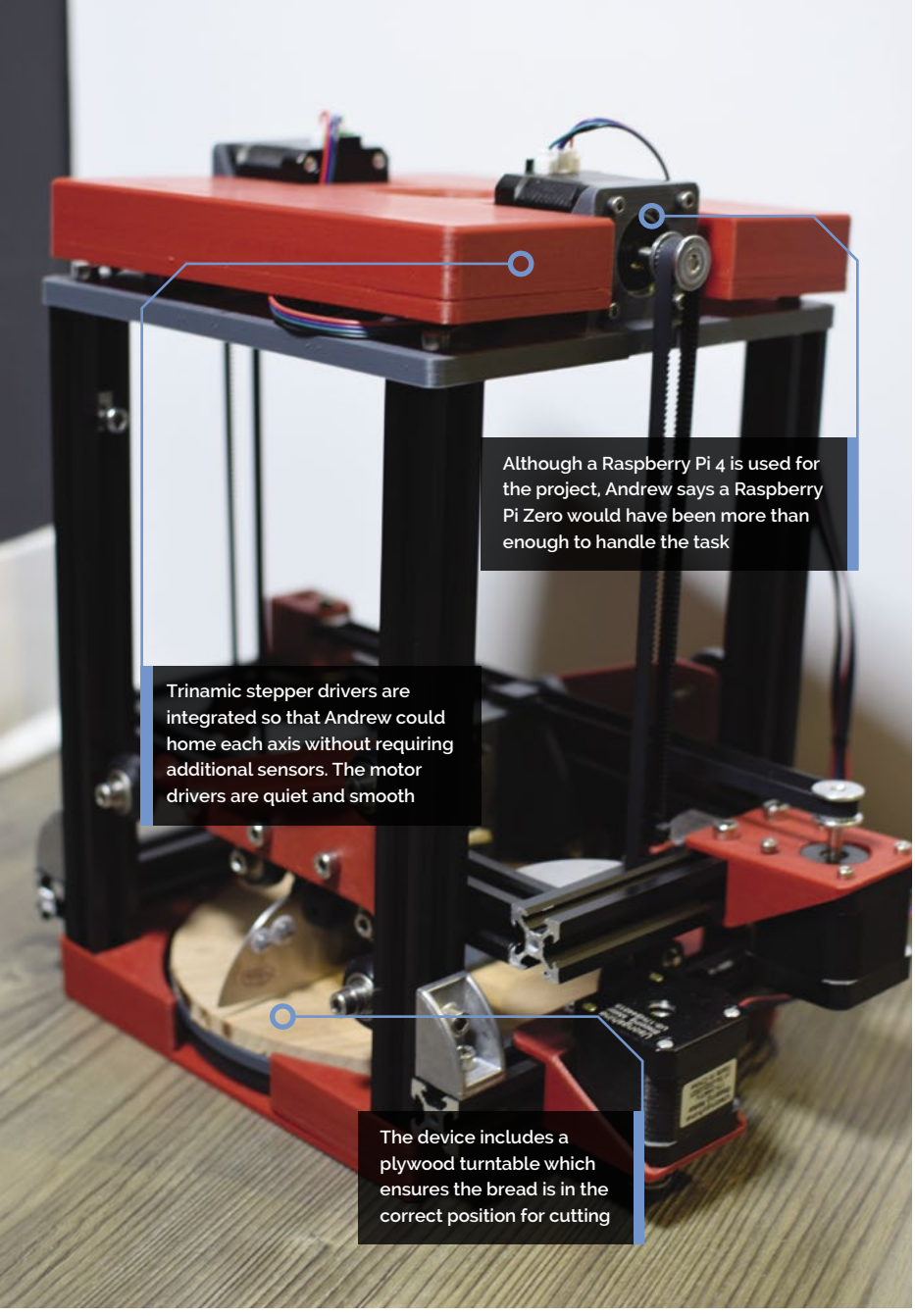
He also built a custom four-axis control board ("the OSR, which controls all of the stepper motors") and he used a Raspberry Pi 4 paired with a Camera Module V2. "Raspberry Pi is the easiest and cheapest way I know to integrate computer vision into my projects," he says.

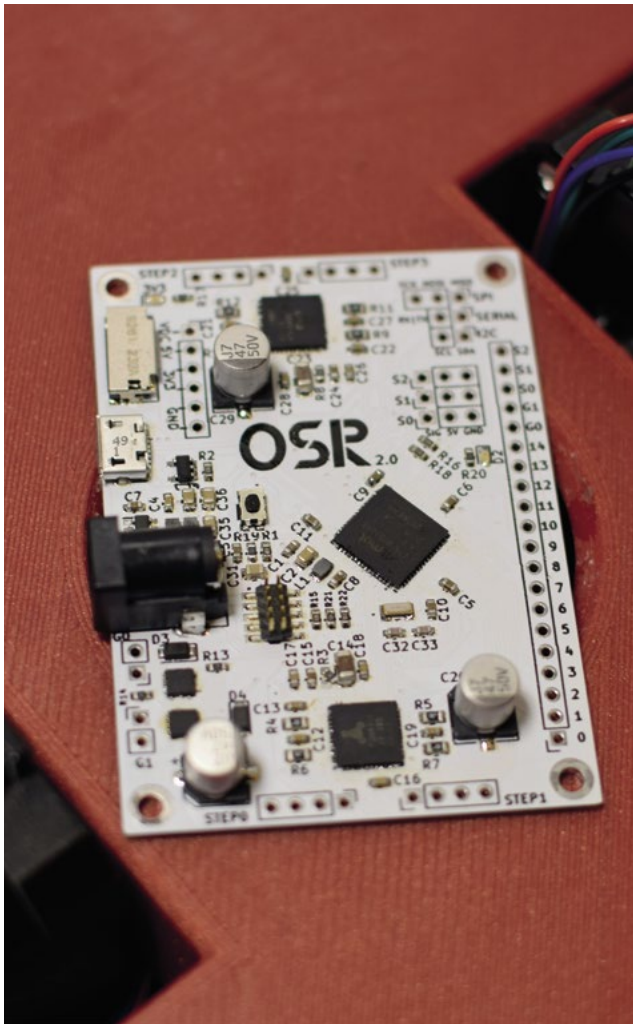
As such, with the components sitting within a 3D-printed frame, Andrew envisaged placing a sandwich on a rotating cutting board before having the camera take images for analysis using OpenCV. "I spent 30 to 40 hours on the code, which included all the OSR, computer vision, and Python segments," he reveals. It was time to get chopping.

Sticking the knife in

By comparing the current image frame to previous ones, the device determines the shape and size of the bread. "I then feed the rotation of the rectangle to the stepper controlling the turntable cutting area to align the sandwich with my knife," Andrew says. "Next, the knife moves horizontally in the X-axis to a position that is a small offset in from the edge of the bread, and finally the Z-stage comes down and makes a cut. Then it's a matter of doing three more rotations and three more cuts to remove the rest of the crust."

The main problem was figuring how to convert the camera's pixel measurements into real-world





▲ Andrew's own four-axis control board (dubbed the OSR control board) communicates with Raspberry Pi via USB and controls all the stepper motors

> It's now a seriously capable bot that cuts bread very cleanly and accurately

measurements. "I ended up uniformly converting the measurements because cutting food doesn't need to have sub-millimetre accuracy," he says. "I ran into some motor control issues which ended up being bugs in my stepper control loops, but it all works pretty well."

Now Andrew doesn't worry about crusts, especially now he's using a sharper knife. "People criticised the bot for not cutting well enough, so I redesigned the gantry to support a sharper knife while using a pivoting knife mechanism and adding a food-holding, spring-loaded platform. It's now a seriously capable bot that cuts bread very cleanly and accurately. It's cool, if not a little scary."

▲ And cut... the knife plunges down towards the bread, slicing the crusts clean off

Quick FACTS

- The device uses an aluminium frame
- It also involves 3D-printed parts
- Belts and stepper motors move the cutting board
- Computer vision figures the position of the cuts
- It cost a lot of dough: around $200

Giant Hornet Detector

Massive marauding insects threaten native bee populations, so one techie apiarist used Raspberry Pi to sniff them out. **Rosie Hattersley** hears how

MAKER

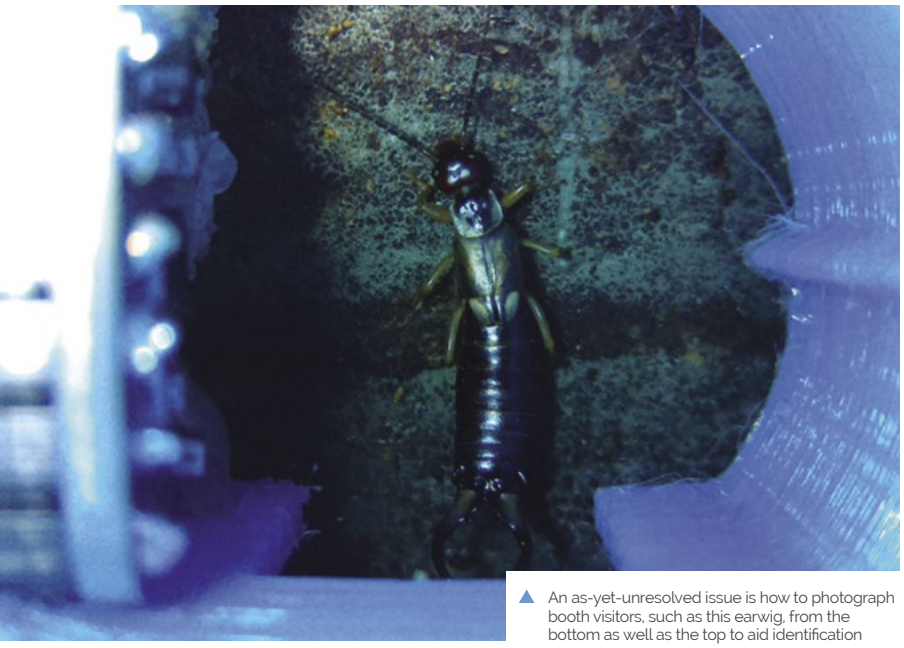
Sean Cusack

Bee-keeper, Microsoft engineer, and Raspberry Pi admirer Sean enjoys finding "elegant technical solutions" for monitoring the health of his beehives.

magpi.cc/beetrackergit

Murder hornets have become the stuff of tabloid hyperbole and alarming YouTube video clips, but Asian giant hornets (as they're more properly termed) aren't all that common. However, they've been making inroads in the US state of Washington where maker and long-term apiarist Sean Cusack lives. Keen to avoid seeing local bees' nests destroyed, he set about creating the Murder Hornet Detector, a citizen science project that identifies the invasive species and demonstrates a rather good use of Raspberry Pi.

Sean has been keeping bees for around seven years and began developing a small photo booth – the HoneyBee Booth – that would use artificial intelligence to detect and count Varroa mites, and notify him of the findings. He adapted it to detect other species on hearing about the murder hornets' arrival.



▲ An as-yet-unresolved issue is how to photograph booth visitors, such as this earwig, from the bottom as well as the top to aid identification

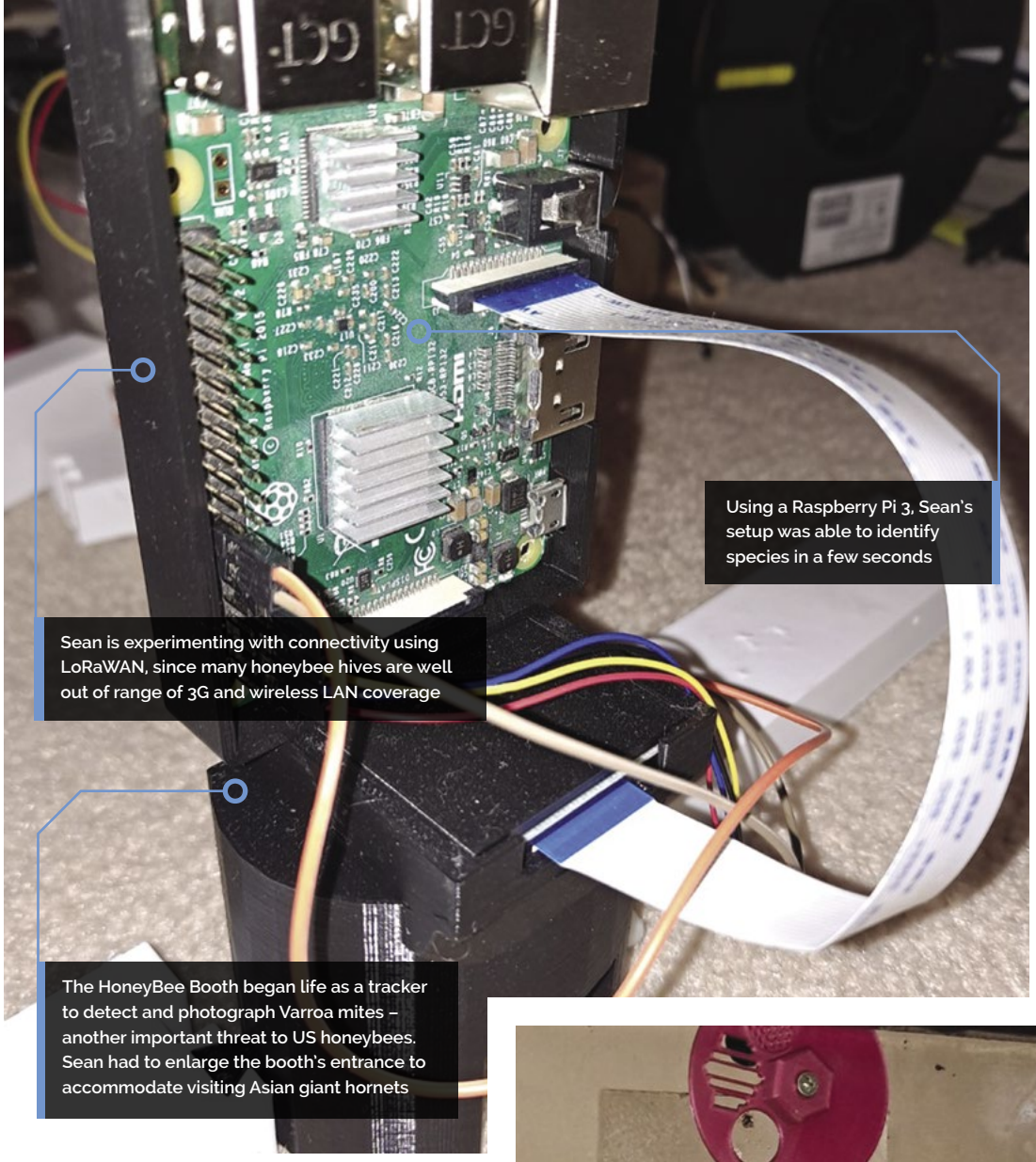
Geolocation data of any affected hives is sent to a Microsoft Azure IoT Central dashboard. Washington's Department of Agriculture can use this in their efforts to eradicate the invasive species. "Using Raspberry Pi 3, Arducam [camera], small motion sensor, and 3D-printable case, I'm able to classify an image in about two seconds," says Sean. However, keen to make this an easily and cheaply replicable project, he suggests a $5 Raspberry Pi Zero will work almost as well, with the caveat that identification takes closer to a minute.

Sharing the resulting images poses a problem, since many beehives are remotely located, but Sean is hoping LoRaWAN (Long Range Wide Area Network) will help. LoRaWAN is designed to work with low-power devices over vast areas and will work well in locations not covered by 3G or wireless LAN. "All the image classification takes place on Raspberry Pi, leaving only the need to transmit telemetry data, which should fit nicely within packet size limitations that exist with LoRaWAN."

Track and trace

Sean is keen to ensure others can replicate his project, envisaging "a non-tech savvy bee-keeper" putting one together and watching for Asian giant hornets. Cost, open-source code, and a simple build were therefore priorities. "I love the idea of a community contributing to a large pool of images that can be used to further train and improve the learning models," he says. It also means Raspberry Pi Zero can be used, keeping all-important costs low.

He sketched and 3D-printed his own basic, but weatherproof, case as the camera booth and to house Raspberry Pi. There's a hole on the top for the LED, while the camera and wires for the motion sensor were inserted via the booth door and attached at the end. Sean enabled SSH and I2C using the command prompt (see the setup instructions at **magpi.cc/beetrackergit**) so Raspberry Pi can be accessed remotely. A funnel added at the end helps



Quick FACTS

- Sean needed an Asian giant hornet to train his detector…
- …So he got Washington's Department of Agriculture to send him one!
- Friends at a hackathon helped Sean refine the HoneyBee Booth design
- Overcoming motion blur was a particular challenge
- Sean's setup now works really well, "even on blurry images"

> Murder hornets have become the stuff of tabloid hyperbole and alarming YouTube video clips

ensure insects are hustled into the booth to be photographed. Getting the camera focus right and clear images of booth visitors was time-consuming but, as Sean's enthusiasm shows, the results have proved rewarding.

Exciting sightings

Sean is excited about the possibilities of the project based around a camera, motion sensor, and a learning model. "Maybe you want to spot an incoming locust migration, get up-close pictures of a very rare insect species, [or] spot elephants on a train track and alert the conductor or train station to stop?" he enthuses. "I can't wait to see what people come up with!"

▲ The booth can be fitted with a weather protection cover

SUBSCRIBE TODAY FROM ONLY £5

SAVE UP TO 35%

Subscriber Benefits

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

Rolling Monthly Subscription

- **Low monthly cost** (from £5)
- **Cancel at any time**
- **Free delivery to your door**
- **Available worldwide**

Subscribe for 12 Months

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

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

Subscribe by phone: **01293 312193**

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

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

HOME OF THE FUTURE

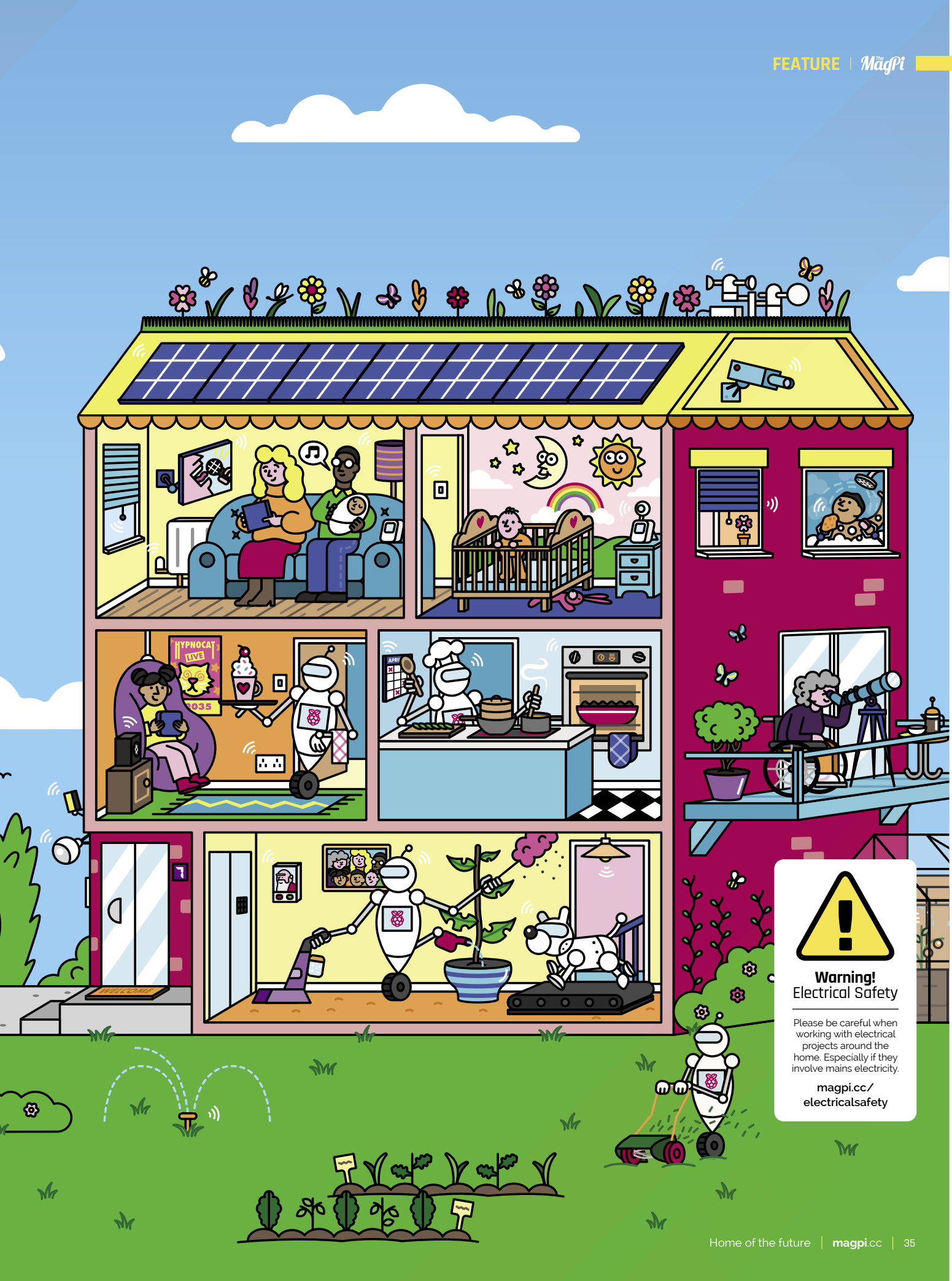
Automate your home with Raspberry Pi

By **PJ Evans**

We love home improvement here at *The MagPi*, and if there's any excuse to fit a new piece of kit around the house we'll jump at it. Raspberry Pi is the perfect tool for smart home setups. It is easy to program, small and unobtrusive, and the wireless LAN networking and GPIO pins make it perfect for chatting to, and controlling, smart devices around the house.

In this feature we will look at how to install smart lights, control central heating, pump internet music throughout the home, keep tabs on the critters in your garden with smart cameras, cook with robotics, and much more.

It's time to pay a visit to the home of the future. So let's get building.



Warning!
Electrical Safety

Please be careful when working with electrical projects around the home. Especially if they involve mains electricity.

magpi.cc/electricalsafety

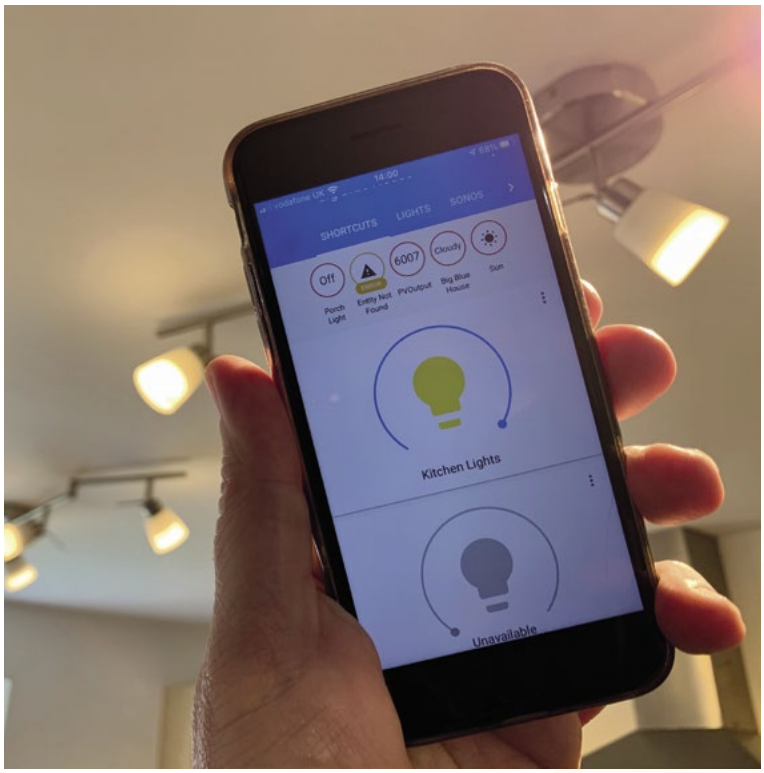
GETTING STARTED WITH HOME AUTOMATION

Put a Raspberry Pi computer at the heart of your home

Starting out in home automation can be a little overwhelming. There are many competing standards, some devices are harder to use than others, and there's the issue of locked-down services demanding subscriptions and access to your data. The good news is there's a massive community dedicated to open-source and self-hosted solutions where you're in control.

We would struggle to find a better place to start than with Home Assistant (**magpi.cc/homeassistant**). If you view all the various switches and sensors that make up your home's Internet of Things (IoT) as instruments in an orchestra, then Home Assistant (HA) is the conductor. It supports hundreds of different devices and allows you to control, group, and create rule sets that govern their behaviour. Best of all, it has a dedicated Raspberry Pi image available that makes setting it up and managing the server very straightforward. Alternatives include OpenHAB and cloud-based services like IFTTT. Check out our three-part guide to Home Assistant in *The MagPi* issues 99, 100, and 101 (**magpi.cc/issues**).

▼ Home Assistant is compatible with a huge range of smart devices



> "There's a massive community dedicated to open-source and self-hosted solutions"

Once you have your 'conductor' in place, it's time to add the musicians. In the next few pages we're showcasing projects that have been carefully selected to make the most of your home without compromising your privacy.

▲ Home Assistant can be used on mobiles, touchscreens, or your desktop – from anywhere in the world

LIVEN UP YOUR **LIVING ROOM**

Whether it's music, TV or game time, tech can help

▼ Install smart lighting

Most HA projects start with lighting. Being able to switch lights on throughout the house, wherever you are, or have them react to different sensors or events is a great way to get started in this field. There are a lot of options and the prices have plummeted. Check out Ikea's Trådfri range, or the popular Philips Hue system.

▼ Build a home cinema

How about the ability to convert your living room into a cinema at the click of a button? Logitech's Harmony Hub (**magpi.cc/harmonyhub**) allows you to control your TV, amp, and game consoles from your phone or Home Assistant. You can create complex scripts and also set lighting or lower blinds. Popcorn optional.

▲ Set up a smart sound system

If you're after some sounds, check out our recent tutorials on building the ultimate music server using Raspberry Pi and Mopidy. Create your own Sonos-rivalling system and have synced music throughout the home. A Raspberry Pi Zero with an inexpensive DAC (digital-to-analogue converter) HAT will give you excellent audio reproduction for a fraction of the commercial price. See issues 96, 97, and 98 (**magpi.cc/issues**).

Control your central heating

Need to be cosy? Need to control exactly how cosy you are? Many solutions exist for remote control of central heating. We particularly like the motorised thermostats (**magpi.cc/smartradiator**) that can be quickly added to radiators as these give you precise control over every room and HA can use individual temperature sensors to get every room just right.

Build a magic mirror

A magic mirror places a screen behind a semi-transparent sheet. Once framed, it gives the impression of text floating in the air that can display your calendar, the weather, or anything else. To find out how to make one, pick up a copy of *The MagPi* issue 90 (**magpi.cc/90**).

AUTOMATE YOUR **KITCHEN**

Improve your kitchen gadgets with even more gadgets

▼ Wire up a coffee machine

If you really want that fresh coffee in the morning to be just right, how about automating your coffee machine? If your machine can start when power is applied, then a simple WiFi power switch will suffice. There are some 'smart' coffee machines available such as Smarter Coffee (**magpi.cc/smartercoffee**), but another option is to add a button-presser such as MicroBot Push (**magpi.cc/microbot**) that's controllable with HA.

Get smart about safety

After lighting, a logical first step for the home automation fan is monitoring of the environment. Smart smoke alarms and carbon monoxide detectors are available, but you may be interested in making open-source versions so you can trigger alerts when things don't seem right. The Pi Hut sells an MQ-135 Gas Sensor (**magpi.cc/mq135**) which is great for experimenting with home-built smoke detectors. Please note that no DIY smart device is ever any replacement for a proper certified smoke alarm, such as those made by FireAngel (**fireangel.co.uk**).

▶ Install smart meters

The kitchen is the heart of the home, so it pays to keep an eye on how much blood it's pumping. As smart meters roll out, it's good to know that many models such as Honeywell's AS302P advertise their usage to supplied displays. With a bit of help, these transmissions can be captured by HA so you can create your own dashboards or set alarms when the cooker gets too greedy. Take a look at this tutorial by Erik Schrama (**magpi.cc/smartmeter**).

> A logical first step for the home automation fan is monitoring of the environment

▼ Discover sous-vide

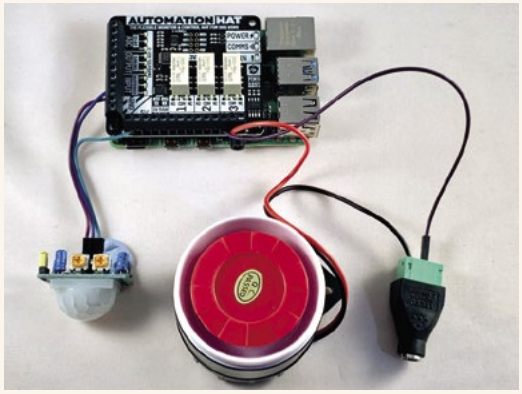
This amazing technique cooks plastic-wrapped food in a water bath using precise temperatures. The result is tender and flavoursome meals that cook over hours not minutes. Sous-vides are typically expensive, but you can build your own and monitor it using HA, such as this project (**magpi.cc/sousvide**). As ever, always be careful mixing electricity and water.

CREATE A COOL KID'S ROOM

Use tech to brighten up your bedroom and keep unwanteds away

▼ Set up a sibling alarm

Brothers! Sisters! Boo! If your sibling is driving you up the wall by messing with your stuff, then catch them in the act. Using a Raspberry Pi HQ Camera and a cheap magnetic door sensor, you can take a photo or record video whenever the door opens, then send it straight to your phone. Busted! Take a look at this Room Guard project: **magpi.cc/roomguardproject**.



▼ Fit some fun lighting

If you really want to make an impression on your friends, get some LED strips and power them with a Raspberry Pi. You can start with single-colour 5050-type LED strips or move up to individually controllable NeoPixel sets (**magpi.cc/neopixels**). A project like this can just build and build; how about adding a microphone to create dancing lights? The Pi Hut has a great NeoPixels tutorial (**magpi.cc/usingneopixels**).

▶ Fit a dinner klaxon

A really popular project from *The MagPi* issue 73 (**magpi.cc/73**), the teenage klaxon is the ultimate solution to yelling up the stairs in frustration at a headphone-clamped offspring. Simply use the web interface on your phone to set the candle light to green, amber, or 'right, now you're really in trouble'.

▼ Listen to internet radio

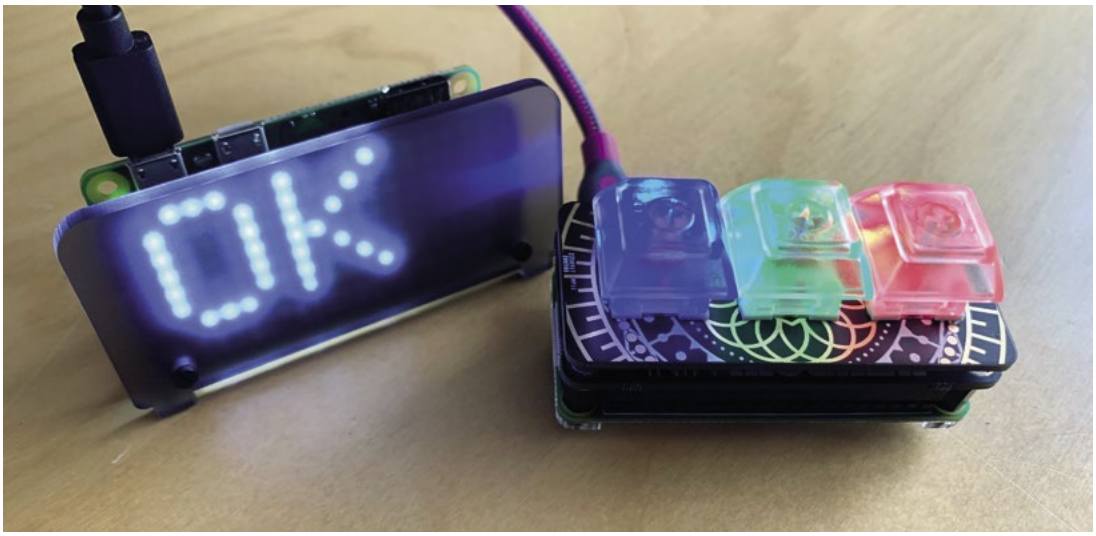
No kid's sanctuary is complete without some sounds. Why not add to your home audio system (see 'Liven up your living room', page 37) with another device or build an internet radio, capable of reaching thousands of stations across the globe. We really like Pimoroni's Pirate Radio (**magpi.cc/pirateradio**) as a great starter kit.

Back issues

Don't miss out on our earlier issues. See our At Home with the Internet of Things feature in *The MagPi* issue 93 (**magpi.cc/93**) and Home Halloween Hacks (**magpi.cc/98**). Our Home Assistant with Raspberry Pi tutorial series can be found in issues 99, 100, and 101 (**magpi.cc/issues**).

DESIGN A HIGH-TECH HOME OFFICE

Don't work hard at home, work smart at home



Build a do-not-disturb sign for your office

In *The MagPi* issue 103 (magpi.cc/103) we addressed one of the burning issues of the day: how do I let my family know I'm on a Zoom call? We invented a 'digital do-not-disturb sign' that can be remotely controlled by some buttons or an event detected by Home Assistant. Display any message you want right outside your door and avoid those moments that could send you viral on social media.

> "Display any message you want right outside your door"

This miniature railway sign can make sure you are on time

If you're working at home and have your own working space, particles and gases (now, now) can build up without proper ventilation. Consider adding a Pimoroni Enviro+ to your arsenal (**magpi.cc/enviroplus**). This HAT with optional add-on particle sensor monitors dust, temperature, and other concerns, sending the data straight into Home Assistant.

Home automation isn't just about switching lights on and off. There's a wealth of information provided free-of-charge from the internet that you can use to affect your environment. A great example is the data that powers all the railway station displays in the UK. This is freely available and this great project by Chris Crocker-White (**magpi.cc/trainoled**) allows you to make your own home display.

Our newest member of the Raspberry Pi family, Pico, can act as a keyboard over USB. Pairing one with Pimoroni's RGB Keypad Base (**magpi.cc/rgbkeypad**) creates an inexpensive macro keyboard. Create short cuts for your common operations and program them into your Pico to speed up your day. No more fighting with the mouse to reach 'Leave Meeting'.

TAKE YOUR SMART TOYS OUTSIDE

Track wildlife, tend to the flowers, and keep an eye on the world with these projects

▲ Install solar panels

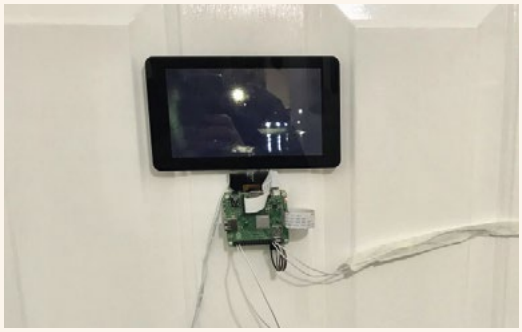
If you've got solar panels, chances are the inverter (the device that converts DC electricity to mains AC) is producing vast amounts of information that HA and other services such as PVOutput (**pvoutput.org**) can consume. Some have USB connectors, but many more have Bluetooth connections, making it easy and safe to link a Raspberry Pi to keep track of all that lovely sunlight powering your house.

▲ Plant monitoring

A smart garden is a better garden. Keep track of moisture levels using sensors coupled with devices like the IoT Cricket (reviewed this issue) to relay our soil condition to HA. You can then create rules to alert you when your favourite plants need a drink. More advanced tinkerers can add irrigation systems controlled by HA. Check out Pimoroni's Grow range (**magpi.cc/grow**).

▼ Ring the smart doorbell

Another stalwart of the home automation scene, smart doorbells are becoming commonplace on our roads. Systems like Ring's video doorbell (**ring.com**) provide video to your smartphone and even the ability to talk with the person at your door from anywhere in the world. If you would prefer a more DIY approach, check out our smart door feature (**magpi.cc/smartdoor**).



Watch the wildlife

A Raspberry Pi wildlife camera (such as Naturebytes', **naturebytes.org**) is just the thing for keeping an eye on various visitors to your garden or window box. This can be a great evolving project. Start with a camera trained on your garden (consider a Raspberry Pi NoIR camera for night vision), then add motion triggering and even machine learning to identify different birds. Take a look at this Watch Wildlife tutorial that first appeared in HackSpace magazine issue 33: **magpi.cc/watchwildlife**.

The MagPi 70

We featured a range of smart home hacks in *The MagPi* issue 70 (**magpi.cc/70**).

Power sockets
Control cheap power sockets using radio commands with the Energenie HAT. Full house coverage and simple to set up.

Doors and doorbells
Creating internet-connected door sensors and doorbells can be surprisingly simple and inexpensive. Get a heads-up when the door opens behind you.

Smart fish tank
When an aquarium heater's thermistor fails, it tends to heat the tank too much. Get an alert with this straightforward monitoring project.

Part 01

Build an arcade machine: Get the parts

If you've ever wanted to build your own arcade machine, here's your guide. This month: the parts you'll need, how to choose them, and where to buy them

MAKER

K.G. Orphanides

K.G. is a writer, maker of odd games, and software preservation enthusiast. Their family fully supports the idea of an arcade machine in the living room.

@KGOrphanides

Over the coming months, we'll go through the process of sourcing, building, connecting, and installing a Raspberry Pi-based arcade cabinet.

While you can restore and convert a former JAMMA cabinet for use with Raspberry Pi, or build a cab entirely from scratch, we'll be taking the flat-pack route. This lets you build the cabinet of your dreams relatively easily, somewhat cheaply, and without recourse to full-on home woodworking.

This tutorial series will use an LCD screen due to the inconvenience of sourcing and potential issues with installing a CRT model, which carries the risk of a dangerous electric shock if not correctly discharged.

01 Choose your cabinet style

If you're after a classic upright one- or two-player cabinet, then you'll want either an all-in-one model or a 'bartop' cabinet with a pedestal or stand. Bartop cabinets can also be bought without the optional stand and placed on a table.

Flat 'cocktail' or 'coffee table' style cabinets are available in models for between one and four seated players and often use a vertically oriented screen, which can be split by software into two horizontal views for multiplayer games.

Other models include seated upright cabinets (often designed to take very large screens), angular tabletop models, and mini-bartops with 10-inch displays for those short on space.

02 Big screen glamour?

The size of your screen dictates the size of your cabinet, and vice versa. Before you start shopping, work out where you want the cabinet to live, and take height, width, and depth measurements.

If you're working with a 19-inch monitor, you'll likely get a bartop cab that's a little under 50 cm wide. This is the most practical choice if available space is limited. A 22-inch screen translates to a cabinet of a little under 60 cm, and a 24- or 25-inch screen means a cabinet width of a bit under 65 cm. You're generally fine fitting a smaller screen in a larger cabinet, but the end result won't look quite so polished.

Check the internal measurements of the cabinet against those of the monitor, including its bezel.

Top Tip

Button positioning

We're going with a six-button Japanese-style layout. Check out **magpi.cc/joysticklayout** to see some alternatives.

03 A good fit

Depending on the era of games you want to play, a large 1920×1080 widescreen display may not be the most authentic choice, but it is the most flexible, and modern emulators handle HD displays well.

Most cabinets have a VESA mount, usually in the form of a monitor support bar drilled for 75×75 and 100×100 mount points. Make sure your monitor has mounting points that match.

Finally, ensure that your monitor will work with Raspberry Pi: anything with a standard HDMI input should be fine, but older DVI and VGA displays require inconvenient adapter arrangements.

04 Materials

Self-assembly cabinets are usually made in MDF, but laminate, melamine, and veneer finishes are also widely available.

MDF swells badly if exposed to water, so if you're going to have drinks anywhere near your cabinet, a water-resistant finish is strongly recommended. If you buy an untreated MDF kit, apply and sand down between multiple coats of an MDF-specific solvent-based primer, then paint it to your heart's content, ideally with oil-based paint.

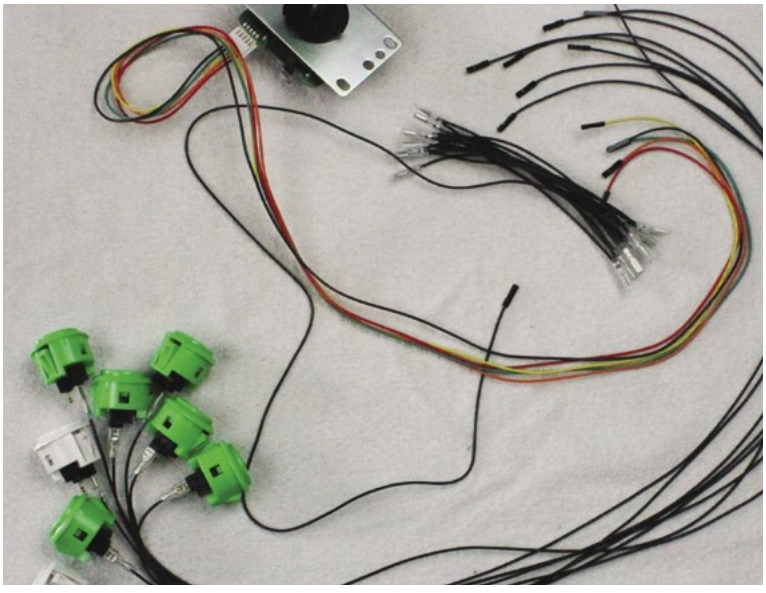
18 mm MDF is common, but you'll find cabinets in anything down to 10 mm for budget models. 18 mm or thicker construction materials may require a longer shaft or extender for your joystick. If in doubt, talk to the kit's supplier.

05 Finish and decoration

Regardless of the materials used, you'll probably want some plastic edging strip. This plastic trim helps to protect the edges of your cabinet, makes it easier to clean, and looks a lot more professional than exposed MDF edges.

Two types are popular. T-Molding is more secure but requires a slot to be cut for it to clip into – some DIY kits have ready-cut slots for this purpose, but budget models frequently do not.

U-Molding just clips over the edge. Cabinet makers will usually tell you how much moulding



You can get kits containing all the joysticks, buttons, and connectors you'll need; just make sure your button and cabinet hole sizes match

▲ A variety of compact bus- and mains-powered amp and speaker kits are available: this one takes power from the USB port and audio from the 3.5mm port

their kit will need and can usually supply the required quantity and type of edging.

Many arcade cabinet suppliers also sell a range of decorative and protective graphical vinyl sticker wraps. These should be applied with care to an appropriately finished surface (check with the sticker manufacturer for any finish requirements).

06 A giant screen protector

To protect your screen and create a flush finish, you can – and should – opt for an acrylic (polymethyl methacrylate, also known as plexiglass) screen protector. Again, this is something most self-assembly kits are designed to take and the majority of retailers will happily sell you one as either a standard part of the kit or an optional extra. Make sure you do opt in, as cutting your own plexiglass to precise dimensions can be a pain. Toughened glass and UV-resistant polycarbonate can also be used. You may need to add some standoffs to stop front monitor buttons being pressed by the screen protector.

07 The marquee club

Also included in kits as a matter of routine is a strip of acrylic for your cabinet's top marquee. You'll probably want to get a backlight-ready vinyl marquee (available from print shops, arcade suppliers, and on Etsy) to stick to this, but you could also decorate your own.

Sample shopping list

Here is an illustrative price list. The prices include VAT but not shipping or additional costs.

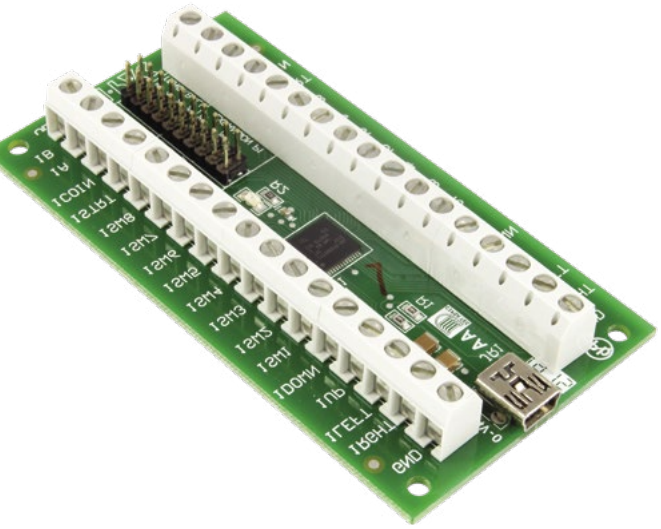
Item	Price
24-inch LCD monitor	£125.00
Bartop cabinet	£170.00
Bartop stand	£100.00
10 m T-Molding	£25.00
Acrylic control panel guard	£25.00
Two-player USB joystick + button kit	£70.00
Amp, speaker & cover kit	£25.00
Amp power supply	£12.00
Printed marquee	£6.00
LED strip lighting	£15.00
Molex power adapter for LEDs	£15.00
5-way plug bar	£15.00
TOTAL	**£603.00**

While you're at it, you may wish to get acrylic or metal panels to surround your buttons and joystick. These can be decorated, and protect your cabinet's surface, as well as providing a smoother feel. Button layouts tend to be standard, but these should ideally be bought from the same supplier as your kit for the best fit.

08 Raid the button tin

We'll be building a cabinet with an eight-way joystick and six 30 mm buttons, plus Start and Select buttons, for each player. A variety of alternative sizes and brands are available, with Sanwa perhaps being the most recognisable. You can order a cabinet with holes for extra side buttons if you're into digital pinball.

An easy cross-platform connection solution is a USB arcade encoder. Models by Zero Delay and Xin-Mo are popular, but the I-PAC 2 keyboard encoder has slightly lower latency.



If you want to use USB, the Ultimarc I-PAC 2 encoder is a popular choice that'll work with most computers. Check out **magpi.cc/ultimarcgit** for advanced configuration

09 Pick a driver

You can connect controls to Raspberry Pi's GPIO, using either the Adafruit Retrogame (**magpi.cc/adaretrogame**) or mk_arcade_joystick_rpi (**magpi.cc/mkjoystick**) drivers – we'll be using the latter.

Arcade joysticks generally use a five-pin JST connector, while non-illuminated buttons each have a pair of quick-connect spade connector fittings, one of which must go to ground. Spade to DuPont GPIO cables are uncommon, but can be bought either individually or as part of a kit from specialist retailers such as SmallCab. Illuminated button kits are available with an extra external PSU.

> LED strip lighting is a popular choice for marquee panels

10 The sound of success

It's a good idea to order your cabinet with a couple of pre-drilled speaker holes and covers to go over them. The most common option for audio is an externally powered stereo amp, connected to Raspberry Pi's 3.5mm port, and 10cm/4-inch speakers, but USB-powered kits are also available. If you have one lying around, you could also consider mounting a compact USB sound bar behind your speaker grilles.

11 More power, Igor!

A major advantage of this kind of arcade machine build is that there are no internal power supplies to bother with. There's enough space to mount a plug bar inside most cabinets, and you can use this to power the monitor, Raspberry Pi, and any extra transformers required for lights or speakers.

Where to buy

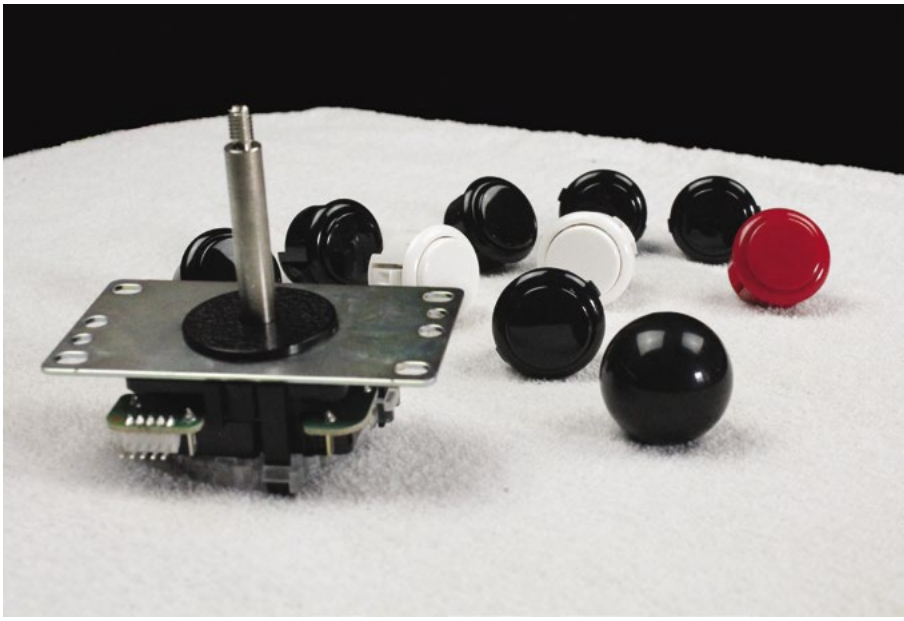
There are a number of UK and EU retailers specialising in self-assembly arcade cabinets and components. While it's easiest to get everything in one place, you have to mix and match for specialist components such as GPIO-compatible wiring looms.

- **Arcade World UK** – arcadeworlduk.com – supplies a wide range of kits and components; discount codes available for most non-furniture items
- **Bitcade** – magpi.cc/bitcadekits – UK arcade machine maker that also supplies kits
- **Omnireto** – omniretro.com – Spanish firm with a notable budget range
- **Rockstar Print** – rockstarprint.co.uk – custom marquee and wrap printer
- **SmallCab** – smallcab.net – French supplier of arcade kits and hardware including GPIO-friendly wiring

LED strip lighting is a popular choice for marquee panels, but you'll need to buy a Molex power adapter to go with it, or repurpose a PC power supply. You can run a plug lead out of the back or optionally install an external power socket and switch, if you're comfortable with simple electrical wiring.

12 Room to build

Before you start ordering, consider not only the space you have to house your cabinet, but also how much room you have to build in. Don't get an untreated MDF cabinet unless you have a large, ventilated (and paint-resistant!) space where you can apply primer to each part, as well as appropriate eye and breathing protection.



You'll want to source durable joysticks and buttons for your arcade machine

Warning!
Paint and dust

When sanding, sawing, or painting, be sure to use appropriate eye and breathing protection in a well-ventilated space.

magpi.cc/diysafety

Part 01

Set up Pi-hole with Raspberry Pi

Pi-hole blocks ads, cookies – and whole sites if you choose – from your home network so you can browse more quickly and securely

MAKER

Nik Rawlinson

Esperanto-speaking, pencil-wielding, single-board computing fan who likes hyphens and remembers what that icon on the save button depicts.

nikrawlinson.com

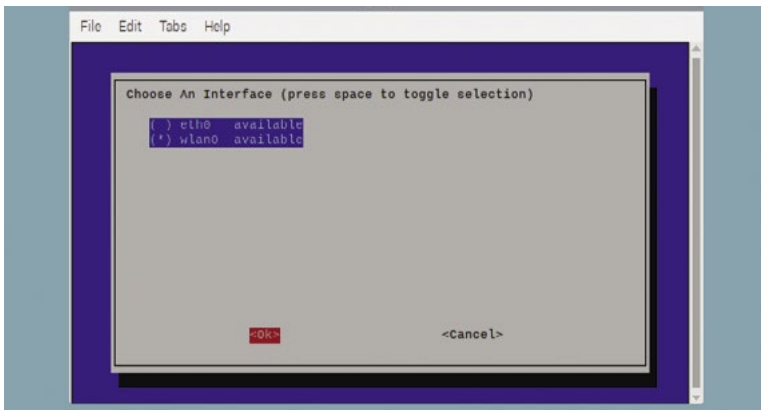
Pi-hole is a free web filtering tool that runs on a Raspberry Pi on your network. By connecting your other computers, tablets – and even your smartphone – to Raspberry Pi, rather than directly to your router, Pi-hole will interrogate their internet traffic and strip out unwanted content. With built-in lists for ad servers, its primary use is to block advertising, but you can just as easily bar social media and other sites you find distracting or objectionable. In this tutorial, we'll show you how to set up Pi-hole on a Raspberry Pi and connect to it from another device on your network.

01 Locate Raspberry Pi

Start with a fresh installation of Raspberry Pi OS connected to your local network (via an Ethernet cable or wireless LAN). Pi-hole only works if the other computers on your network know where to find it. Most routers assign IP (internet protocol) addresses dynamically, and there's a chance your Raspberry Pi might move around the network and get a new IP address. To ensure your other devices can always find it, we're going to give your Raspberry Pi a static IP address. Open a Terminal window by clicking the icon on the Raspberry Pi menu bar and enter:

```
hostname -I
```

This will tell you which IP address is currently assigned to your Raspberry Pi. Ours is 192.168.1.148. Make a note of this number.



▼ Make sure Pi-hole is set to filter content on the interface through which it's connected to your network

02 Get your router's IP address

Now do the same for your router's IP address (the 'default gateway'). Enter this command in Terminal:

```
ip r
```

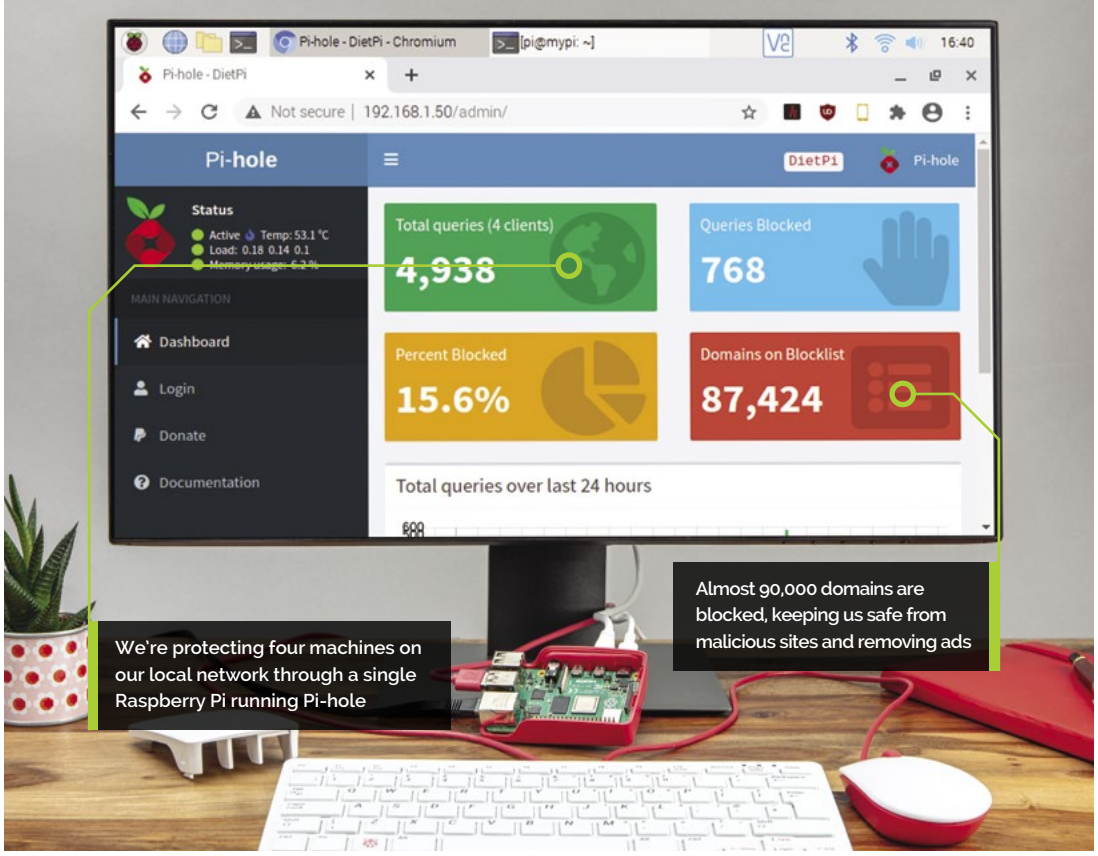
Press **ENTER**, and note the first four sets of digits after 'default via'. Ours is '192.168.1.1'. This is your router's address.

03 Update your configuration file

We're going to add this info to our configuration file so it never changes. Open Terminal and enter:

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

At the bottom of the **dhcpcd.conf** file you will see an example of a static IP configuration. Delete the hashtags ('#') at the start to uncomment the 'static IP address' and 'static IP routers' lines. Replace the example IP address with the number from Step 1 and your router address from Step 2. See the **dhcpcd.conf**



code listing for an example of how our configuration looks (but don't forget to use your own IP and router addresses).

If your Raspberry Pi is connected to the router via an Ethernet cable, remove the comment '#' before `interface eth0`. If you are connecting Raspberry Pi to your network with wireless LAN, replace it with `interface wlan0`.

Press **CTRL+O** to save your file ('O' stands for output) and **CTRL+X** to quit Nano, then enter `sudo reboot` and press **ENTER** to restart.

04 Download the Pi-hole installer

When it's up and running again, you're ready to install Pi-hole. First, download a copy of the latest build. Open a new Terminal window and enter:

```
wget -O basic-install.sh
https://install.pi-hole.net
sudo bash ./basic-install.sh
```

This will start the Pi-hole installation script. It will check which packages are already installed, and install the ones you need. Follow the instructions in the installation while reading through the following steps.

05 Specify your interface

After some preliminary configuration, the splash screen for the Pi-hole automated installer will appear, explaining that it's about to "transform your device into a network-wide ad blocker". Press **ENTER** on each of the first three screens. On the 'Choose an interface' screen, use the arrows and **SPACE** bar to select either etho or wlano for a wired or wireless connection, as appropriate to your setup. Press **TAB** to select OK and hit **ENTER**.

> "Pi-hole uses an external DNS provider to locate authorised resources"

06 Choose a DNS provider

Any internet resources, like images, text and code, that don't trip Pi-hole's safeguards will be retrieved from the servers where they're stored and fed back to the device that requested them. Pi-hole uses an external DNS provider to locate these authorised resources, and gives you a choice of nine to pick from, plus a custom option for business users running their own DNS server. If you're happy to stick with the default, which is Google, press **TAB** to select OK, then press **ENTER**. If not, select an alternative from the list, press **TAB** to select OK, then **ENTER** to move to the next page.

07 Confirm your block list

To save you specifying every server that should be blocked, Pi-hole is configured to use a pre-compiled list ('StevenBlack') to which you can add your own entries once it's up and running. Leave the

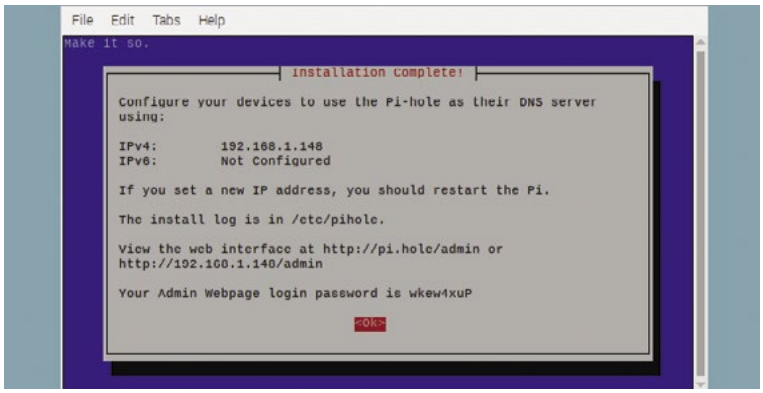
Top Tip

Change password

You can change your password at the command prompt with the command: `sudo pihole -a -p`

You'll Need

- Raspberry Pi
- Pi-hole installer **pi-hole.net**
- Your router details



▲ When you've finished setting up Pi-hole, make a note of the IPv4 address and password for later use

existing list selected by pressing **TAB** to select OK and press **ENTER**. Ensure both IPv4 and IPv6 are selected on the 'Select Protocols' screen, and select OK.

At this point, the installer checks that you're happy with the numeric (IP) address Raspberry Pi is using. Check that the IP address and Gateway match your IP address and router's address. As we configured this in Step 3, and set it to remain fixed, step through the next two screens without worrying too much over the warnings that static addresses might cause conflicts.

08 Install the web interface

You can use Pi-hole's web interface to monitor your web traffic and temporarily deactivate web filtering when you need to access resources that would otherwise have been blocked. It's also where you'll add entries to and remove them from the block list. Make sure 'On' is selected when asked if you want to install the web admin interface, then press **TAB** to select OK and press **ENTER**. Do the same on any following screens to install the lighttpd web server that will host the web interface, and the PHP modules that it relies on.

09 To log or not to log

When you reach the 'log queries' screen, decide whether you want to log queries or not.

Top Tip

Permanently powered

Keep the Raspberry Pi running Pi-hole switched on. If not, your connected devices will resort to the fallback DNS server.

dhcpcd.conf

DOWNLOAD THE FULL CODE:
magpi.cc/github

› Language: **Bash**

```
001.  interface wlan0 # use eth0 for wired Ethernet
002.  static ip_address=192.168.1.148/24 # Use your Raspberry Pi's IP address
003.  static routers=192.168.1.254 # Use your Router's IP address
004.  static domain_name_servers=192.168.1.254 8.8.8.8
```

They're useful if you want to look back and see what's been requested and blocked on your network, but not critical to the effective running of Pi-hole. If you want to save space on your Raspberry Pi's microSD card, or reduce the number of times your Raspberry Pi writes to it (media cards can sustain a generous but limited number of write and wipe operations before they start to fail), you can turn off this option. We're going to leave it on.

10 Select a privacy mode

How much do you want to know about what's happening on your network? Pi-hole next asks what level of detail it should record in its stats. Level 0, the default, logs everything, with higher numbers anonymising more and more data. If you're never going to need to know what other users on your network are up to, you can set it to 3, but sticking with level 0 can be useful if you suspect processes installed somewhere on your network might be sending your personal information to remote servers, as you can trawl the logs for unfamiliar hosts.

11 Specify your interface

Pi-hole has all the information it needs to complete the setup process, the remainder of which is automated.

When it's completed, take a note of the IPv4 address displayed on the final screen, and the Admin login password. Quit the installer and reboot Raspberry Pi with:

```
sudo reboot
```

At this point you might want to remove the screen, keyboard, and mouse. Locate your Raspberry Pi near your router if using wireless LAN or keep it connected via the Ethernet cable. From now on, we will access it via the local network.

12 Using Pi-hole

Now switch to another computer on your network. It can be another Raspberry Pi, or a Mac or PC (or even a smartphone or tablet). You will need to change the network settings on every device you want to filter through Pi-hole so it accesses your Raspberry Pi rather than your router directly.

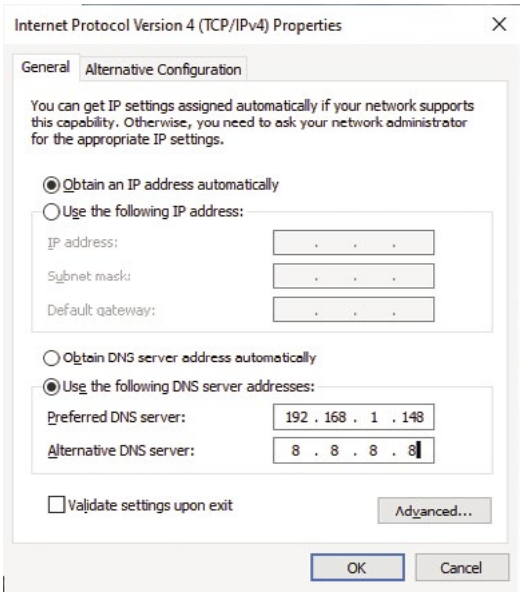
Manually configure each device for Pi-hole

Configure a Windows device

On a Windows computer, open Control Panel and click 'View network status and tasks'. Click the hyperlink of your network connection (Ethernet or WiFi, followed by Properties. Double-click 'Internet Protocol Version 4 (TCP/IPv4)'. Click the radio button beside 'Use the following DNS server address' and, in the box beside 'Preferred DNS server', enter the IP address of your Raspberry Pi running Pi-hole.

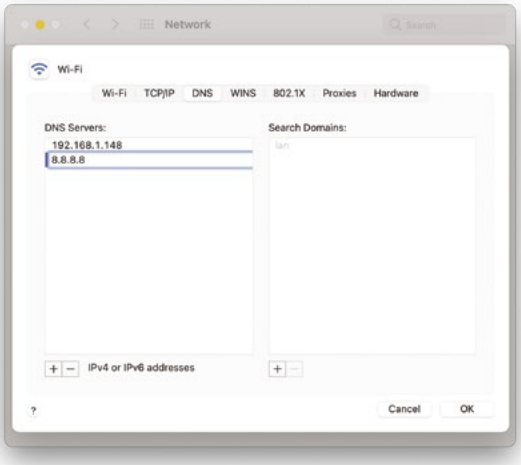
In the 'Alternative DNS server' box, enter either your router address, if you want to fall back on your ISP's DNS server should there be any problem with Pi-hole, or 8.8.8.8 if you want to use Google's DNS server instead.

▼ Make sure Pi-hole is set to filter content on the interface through which it's connected to your network



Set up macOS

Open System Preferences and choose Network. Choose the network interface from the sidebar (typically Wi-Fi or Ethernet) and click 'Advanced'. Choose the DNS tab and click the '+' (Add a DNS Server icon). Type the address of your Raspberry Pi running Pi-hole in the line that appears. You can optionally add another server, like your router's IP address or 8.8.8.8 to use Google's DNS server



◀ Specifying a second DNS address gives your computer a fallback that it will use if it can't reach your Pi-hole device

as a fallback in the case of problems. Select the addresses of any existing DNS servers in the left-hand box, and click '–' to delete them.

Set up Linux

To set up another Raspberry Pi to use Pi-hole as its DNS server, switch to that Raspberry Pi, open Terminal and type:

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

Move your cursor to the bottom of the file and add the following, replacing `[ip address]` with the numeric address of the device running Pi-hole:

```
static domain_name_servers=[ip address]
```

Press **CTRL+X** to exit and confirm that you want to save the file when asked. Now restart the dhcpcd service by typing:

```
sudo service dhcpcd restart
```

On a Linux PC running the Gnome interface, launch the Settings app and click Wi-Fi in the sidebar, followed by the cog beside your active network name. Click the IPv4 tab, then turn off the switch beside Automatic to the right of DNS. Enter the numeric address of your Pi-hole device in the field below, then click Apply.

◀ Run Pi-hole on your Raspberry Pi and enjoy the worldwide web without all those annoying ads

Top Tip

Beware changing networks

When you move your computer to a different network, you'll bypass Pi-hole. Change your DNS settings for every network you use.

Part 04

Create GUIs with Python:

World's worst GUI

Learn good GUI design by doing it all wrong first!

MAKER

Laura Sach

Laura leads the A Level team at the Raspberry Pi Foundation, creating resources for students to learn about Computer Science.

@CodeBoom

MAKER

Martin O'Hanlon

Martin works in the learning team at the Raspberry Pi Foundation, where he creates online courses, projects, and learning resources.

@martinohanlon

It's time to really go to town with your GUIs and experiment with different widgets, colours, fonts, and features. Like most experiments, it's likely that you won't get it right first time! In fact, you are going to explore the wrong way to approach creating your GUI.

It's hard to read

The right choice of GUI colour and font are important. It's important that the contrast between background and text colour ensure that your GUI is easily readable. What you shouldn't do is use two very similar colours.

Import the widgets at the top of the code:

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

Create an app with a title:

```
app = App("it's all gone wrong")
title = Text(app, text="Some hard to read text")

app.display()
```

Experiment by changing the colours, font, and text size (see **worst1.py** listing). Our choices are not the best!

```
app = App("it's all gone wrong", bg="dark green")
title = Text(app, text="Some hard-to-read text", size="14", font="Comic Sans", color="green")
```

It's important that text on a GUI also stays around long enough to be read. It certainly shouldn't disappear or start flashing.

All widgets in guizero can be made invisible (or visible again) using the `hide()` and `show()` functions. Using the `repeat` function in guizero to run a function every second, you can make your text hide and show itself and appear to flash.

Create a function which will hide the text if it's visible and show it if it's not:

```
def flash_text():
    if title.visible:
        title.hide()
    else:
        title.show()
```

Before the app is displayed, use `repeat` to make the `flash_text` function run every 1000 milliseconds (1 second).

```
app.repeat(1000, flash_text)

app.display()
```

Your code should now look like **worst2.py**. Test your app: the title text should flash, appearing and disappearing once every second.

The wrong widget

Using an appropriate widget can be the difference between a great GUI and one which is completely unusable.

Figure 1

▶ **Figure 1** A slider to set date and time

Which widget would you use to enter a date? A TextBox? Multiple Combos? A TextBox would be more flexible but would require validation and formatting. Multiple Combos for year, month, and day wouldn't require validating but would be slower to use.

Using a Slider to set a date and time (**Figure 1**), as in the **worst3.py** code example, is not a great idea, though.

The Slider widget returns a number between 0 and 999,999,999. This is the number of seconds since 1 January 1970. The function `ctime()` is used to turn this number into a date and time.

Getting text from your user is simple: a TextBox or a multi-line TextBox should fulfil all your needs. Is it too simple, though? Does this require too much typing?

What about the user who just wants to use a mouse? Perhaps a series of Combos each containing all the letters in the alphabet would be better (**Figure 2**)? Start by importing the guizero widgets and `ascii_letters`.

```
from guizero import App, Combo
from string import ascii_letters
```

`ascii_letters` is a list containing all the 'printable' ASCII characters which you can use as the options for the Combo.

Create a single Combo which contains all the letters and displays the app.

```
a_letter = Combo(app, options=" " + ascii_
letters, align="left")

app.display()
```

Your program should now resemble **worst4.py**. Running it, you will see a single Combo which contains all the letters plus a space and is aligned to the left of the window.

To get a line of letters together, you could continually add Combo widgets to your app, e.g.:

```
a_letter = Combo(app, options=" " + ascii_
letters, align="left")
b_letter = Combo(app, options=" " + ascii_
letters, align="left")
c_letter = Combo(app, options=" " + ascii_
letters, align="left")
```

By aligning each Combo widget to the left, the widgets are displayed next to each other against the left edge.

worst1.py

DOWNLOAD THE FULL CODE:
magpi.cc/guizerocode

Language: **Python 3**

```
# Imports ---------------

from guizero import App, Text


# App -------------------

app = App("it's all gone wrong", bg="dark green")

title = Text(app, text="Hard to read", size="14", font="Comic
Sans", color="green")

app.display()
```

Figure 2 Combos to choose letters

worst2.py

Language: **Python 3**

```
# Imports ---------------

from guizero import App, Text


# Functions -------------

def flash_text():
    if title.visible:
        title.hide()
    else:
        title.show()


# App -------------------

app = App("it's all gone wrong", bg="dark green")

title = Text(app, text="Hard to read", size="14", font="Comic
Sans", color="green")

app.repeat(1000, flash_text)

app.display()
```

CUSTOM PC

THE BEST-SELLING MAG FOR PC HARDWARE, OVERCLOCKING, GAMING & MODDING

THE MAGAZINE FOR

PC HARDWARE ENTHUSIASTS

ISSUE 212 OUT NOW

VISIT **CUSTOMPC.CO.UK** TO LEARN MORE

worst3.py

Language: **Python 3**

```
001.  # Imports ---------------
002.
003.  from guizero import App, Slider, Text
004.  from time import ctime
005.
006.
007.  # Functions -------------
008.
009.  def update_date():
010.      the_date.value = ctime(date_slider.value)
011.
012.
013.  # App -------------------
014.
015.  app = App("Set the date with the slider")
016.  the_date = Text(app)
017.  date_slider = Slider(app, start=0, end=999999999,
018.  command=update_date)
019.
020.  app.display()
```

worst4.py

Language: **Python 3**

```
001.  # Imports ---------------
002.  from guizero import App, Combo
003.  from string import ascii_letters
004.
005.
006.  # App -------------------
007.
008.  app = App("Enter your name")
009.
010.  a_letter = Combo(app, options=" " + ascii_letters, align="left")
011.
012.  app.display()
```

Figure 3
Pointless pop-up

Figure 3

Window widget

Pop-up boxes can be used to ask users questions, but they are really simple.

If you want to do show additional information or ask for supplementary data, you could use the Window widget to create multiple windows.

Window is used in a similar way to App and has many of the same functions.

```
from guizero import App, Window

app = App("Main window")
window = Window(app, "2nd Window")

app.display()
```

You can control whether a Window is on screen using the **show()** and **hide()** methods.

```
window.show()
window.hide()
```

An app can be made to wait for a window to be closed after it has been shown, by passing True to the **wait** parameter of **show**. For example:

```
window.show(wait=True)
```

You can find out more about how to use multiple windows in the guizero documentation: **lawsie.github.io/guizero/multiple_windows**.

Alternatively, you could use a `for` loop, create a list of letters, and append each letter to the list, as shown in **worst5.py**.

Try both these approaches and see which you prefer. The `for` loop is more flexible as it allows you to create as many letters as you like.

Pop-ups

No terrible GUI would be complete without a pop-up box. guizero contains a number of pop-up boxes, which can be used to let users know something important or gather useful information. They can also be used to irritate and annoy users!

First, create an application which pops up a pointless box at the start to let you know the application has started.

```
from guizero import App

app = App(title="pointless pop-ups")

app.info("Application started", "Well done
```

worst5.py

Language: **Python 3**

```
# Imports ---------------

from guizero import App, Combo
from string import ascii_letters


# App -------------------

app = App("Enter your name")

name_letters = []
for count in range(10):
    a_letter = Combo(app, options=" " + ascii_letters,
align="left")
    name_letters.append(a_letter)

app.display()
```

05-worlds-worst-gui.py

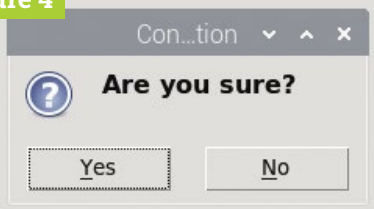
Language: **Python 3**

```
from guizero import App, PushButton

def are_you_sure():
    if app.yesno("Confirmation", "Are you sure?"):
        app.info("Thanks", "Button pressed")
    else:
        app.error("Ok", "Cancelling")

app = App(title="pointless pop-ups")

button = PushButton(app, command=are_you_sure)

app.info("Application started", "Well done you started the
application")

app.display()
```

Create Graphical User Interfaces with Python

For further tutorials on how to make your own GUIs with guizero, take a look at our book, *Create Graphical User Interfaces with Python*. Its 156 pages are packed with essential info and a range of exciting projects.
magpi.cc/pythongui

Figure 4



▲ **Figure 4** Yes, we're sure!

```
you started the application")

 app.display()
```

Running your application, you will see that an 'info' box appears (**Figure 3**). The first parameter passed to info is the title of the window; the second parameter is the message.

You can change the style of this simple pop-up by using `warn` or `error` instead of `info`.

Pop-up boxes can also be used to get information from the user. The simplest is a `yesno` which will ask the user a question and get a True or False response. This is useful if you want a user to confirm before doing something, such as deleting a file. Perhaps not every time that they press a button, though! Import the PushButton widget into your application:

```
from guizero import App, PushButton
```

Create a function which uses the `yesno` pop-up to ask for confirmation.

```
def are_you_sure():
    if app.yesno("Confirmation", "Are you
sure?"):
        app.info("Thanks", "Button
pressed")
    else:
        app.error("Ok", "Cancelling")
```

Add the button to your GUI which calls the function when it is pressed.

```
button = PushButton(app, command=are_you_
sure)
```

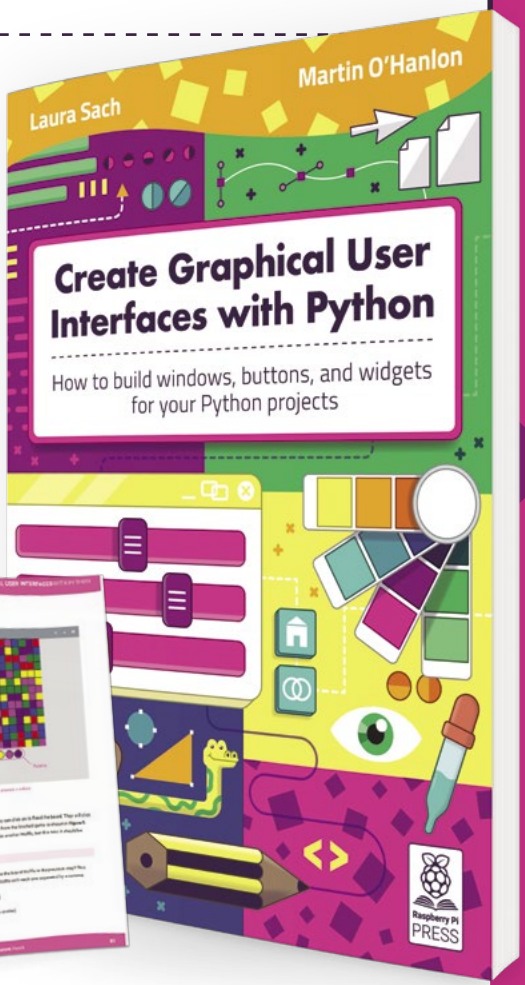
Your code should now resemble **05-worlds-worst-gui.py**. When you run the application and press the button, you will see a pop-up asking to you confirm with a Yes or No (**Figure 4**).

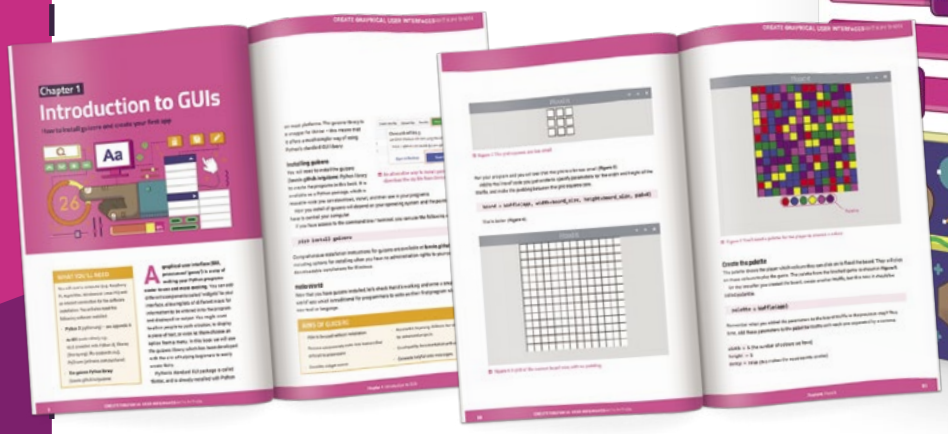
You can find out more about the pop-up boxes in guizero at **lawsie.github.io/guizero/alerts**.

How about combining all of these 'features' into one great GUI?

Create Graphical User Interfaces with Python

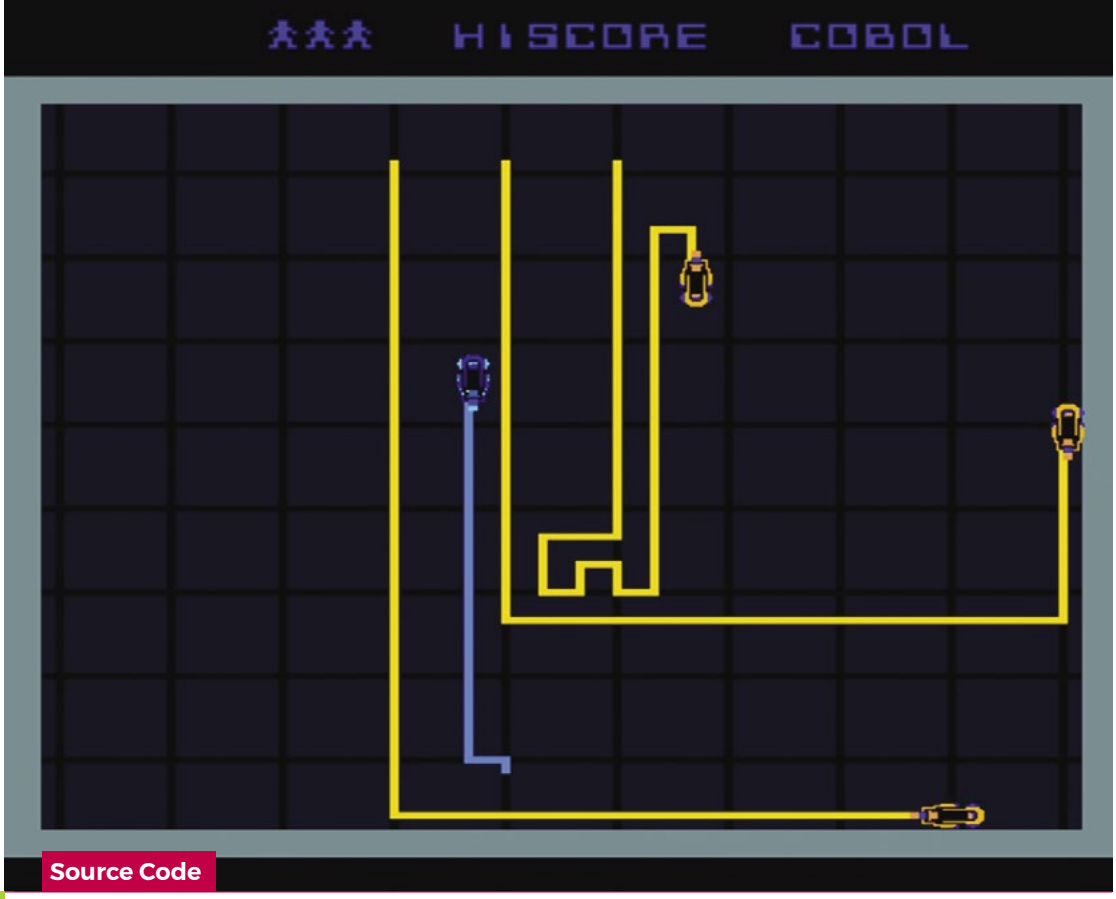
- ✓ **Create games and fun Python programs**
- ✓ Learn how to create your own graphical user interfaces
- ✓ **Use windows, text boxes, buttons, images, and more**
- ✓ Learn about event-based programming
- ✓ **Explore good (and bad) user interface design**





Buy online: magpi.cc/pythongui

The *TRON* cab had two game controllers: a rotary wheel and a joystick.



Battle against AI enemies in the original arcade classic.

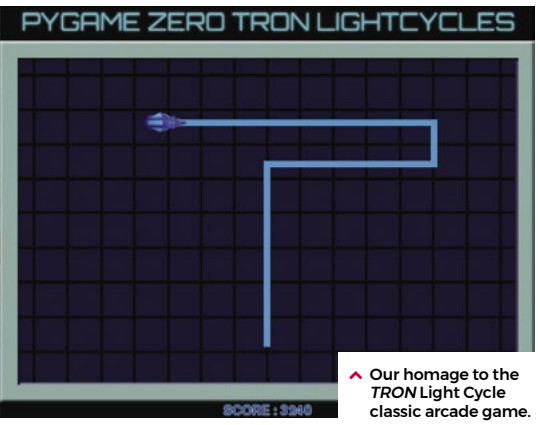
Source Code

Code a Light Cycle arcade minigame

Speed around an arena, avoiding walls and deadly trails

AUTHOR
MARK VANSTONE

At the beginning of the 1980s, Disney made plans for an entirely new kind of animated movie that used cutting-edge computer graphics. The resulting film was 1982's *TRON*, and it inevitably sparked one of the earliest tie-in arcade machines. The game featured several minigames, including one based on the Light Cycle section of the movie, where players speed around an arena on high-tech motorbikes, which leave a deadly trail of light in their wake. If competitors hit any walls or cross the path of any trails, then it's game over. Players progress through the twelve levels which were all named after programming languages. In the Light Cycle game, the players compete against AI players who drive yellow Light Cycles around the arena. As the levels progress, more AI Players are added.



Our homage to the *TRON* Light Cycle classic arcade game.

The *TRON* game, distributed by Bally Midway, was well-received in arcades, and even won Electronic Games Magazine's (presumably) coveted Coin-operated Game of the Year gong. Although the arcade game wasn't ported to home computers at the time, several similar games – and outright clones – emerged, such as the unsubtly named *Light Cycle* for the BBC Micro, Oric, and ZX Spectrum.

The *Light Cycle* minigame is essentially a variation on *Snake*, with the player leaving a trail behind them as they move around the screen. There are various ways to code this with Pygame Zero. In this sample, we'll focus on the movement of the player Light Cycle and creating the trails that are left behind as it moves around the screen. We could use line drawing functions for the trail behind the bike, or go for a system like *Snake*, where blocks are added to the trail as the player moves. In this example, though, we're going to use a two-dimensional list as a matrix of positions on the screen. This means that wherever the player moves on the screen, we can set the position as visited or check to see if it's been visited before and, if so, trigger an end-game event.

For the main `draw()` function, we first blit our background image, which is the cross-hatched arena, then we iterate through our two-dimensional list of screen positions (each 10 pixels square) displaying a square anywhere the Cycle has been. The Cycle is then drawn and we can add a display of the score. The `update()` function contains code to move the Cycle and check for collisions. We use a list of directions in degrees to control

Wireframe

This tutorial first appeared in Wireframe, our sister magazine that lifts the lid on the world of video games. Every issue includes tutorials and in-depth interviews, along with news and reviews of the latest indie and triple-A games.

To find out more, visit their website at **wfmag.cc**.

Check out their subscription offers at **wfmag.cc/subscribe**.

Light Cycles in Python

Here's Mark's code for a Light Cycle minigame straight out of *TRON*. To get it working on your system, you'll need to install Pygame Zero – full instructions are available at **wfmag.cc/pgzero**.

Download the code from GitHub: **wfmag.cc/wfmag47**

```
# TRON

speed = 3
dirs = [0,90,180,270]
moves = [(0,-1),(-1,0),(0,1),(1,0)]

def draw():
    screen.blit("background", (0, 0))
    for x in range(0, 79):
        for y in range(0, 59):
            if matrix[x][y] > 0:
                matrix[x][y] += 1
                screen.blit("dot",((x*10)-5,(y*10)-5))
    bike.draw()
    screen.draw.text("SCORE : "+ str(score), center=(400, 588), owidth=0.5,
ocolor=(0,255,255), color=(0,0,255) , fontsize=28)

def update():
    global matrix,gamestate,score
    if gamestate == 0:
        bike.angle = dirs[bike.direction]
        bike.x += moves[bike.direction][0]*speed
        bike.y += moves[bike.direction][1]*speed
        score += 10
        if matrix[int(bike.x/10)][int(bike.y/10)] < 15 :
            matrix[int(bike.x/10)][int(bike.y/10)] += 1
        else:
            gamestate = 1
        if bike.x < 60 or bike.x > 750 or bike.y < 110 or bike.y > 525:
            gamestate = 1
    else:
        if gamestate < 18:
            bike.image = "bike"+str(int(gamestate/2))
            bike.angle = dirs[bike.direction]
            gamestate += 1

def on_key_down(key):
    if key == keys.LEFT:
        bike.direction += 1
        snapBike()
        if bike.direction == 4 : bike.direction = 0
    if key == keys.RIGHT:
        bike.direction -= 1
        snapBike()
        if bike.direction == -1 : bike.direction = 3
    if key == keys.SPACE and gamestate == 18:
        init()

def snapBike():
    bike.x = int(bike.x/10)*10
    bike.y = int(bike.y/10)*10

def init():
    global bike,matrix,gamestate,score
    bike = Actor('bike1', center=(400, 500))
    bike.direction = 0
    matrix = [[0 for y in range(60)] for x in range(80)]
    gamestate = score = 0

init()
```

the angle the player is pointing, and another list of x and y increments for each direction. Each update, we add x and y co-ordinates to the Cycle actor to move it in the direction that it's pointing, multiplied by our speed variable. We have an **on_key_down()** function defined to handle changing the direction of the Cycle actor with the arrow keys.

We need to wait a while before checking for collisions on the current position, as the Cycle won't have moved away for several updates, so each screen position in the matrix is actually a counter of how many updates it's been there for. We can then test to see if 15 updates have happened before testing the square for collisions, which gives our Cycle enough time to clear the area. If we do detect a collision, then we can start the game-end sequence. We set the **gamestate** variable to 1, which then means the **update()** function uses that variable as a counter to run through the frames of animation for the Cycle's explosion. Once it reaches the end of the sequence, the game stops. We have a key press defined (the **SPACE** bar) in the **on_key_down()** function to call our **init()** function, which will not only set up variables when the game starts, but sets things back to their starting state.

So, that's the fundamentals of the player Light Cycle movement and collision checking. To make it more like the original arcade game, why not try experimenting with the code and adding a few computer-controlled rivals?

MAKER
Gareth Halfacree

With a passion for open-source software and hardware, Gareth was an early adopter of the Raspberry Pi platform and has written several publications on its capabilities and flexibility.

@ghalfacree

Raspberry Pi Pico traffic light controller

Create your own mini pedestrian crossing system using a Raspberry Pi Pico, multiple LEDs, and a push-button

Microcontrollers can be found in almost all the electronic items you use on a daily basis – including traffic lights. A traffic light controller is a specially built system which changes the lights on a timer, watches for pedestrians looking to cross, and can even adjust the timing of the lights depending on how much traffic there is – talking to nearby traffic light systems to ensure the whole traffic network keeps flowing smoothly.

While building a large-scale traffic management system is a pretty advanced project, it's simplicity itself to build a miniature simulator powered by your Raspberry Pi Pico. With this project, you'll see how to control multiple LEDs, set different timings, and how to monitor a push-button input while the rest of the program continues to run using a technique known as threading.

For this project, you'll need your Pico; a breadboard; a red, a yellow or amber, and a green LED; three 330Ω resistors; an active piezoelectric buzzer; and a selection of male-to-male (M2M) jumper wires. You'll also need a micro USB cable, and to connect your Pico to your Raspberry Pi or other computer running the Thonny MicroPython IDE.

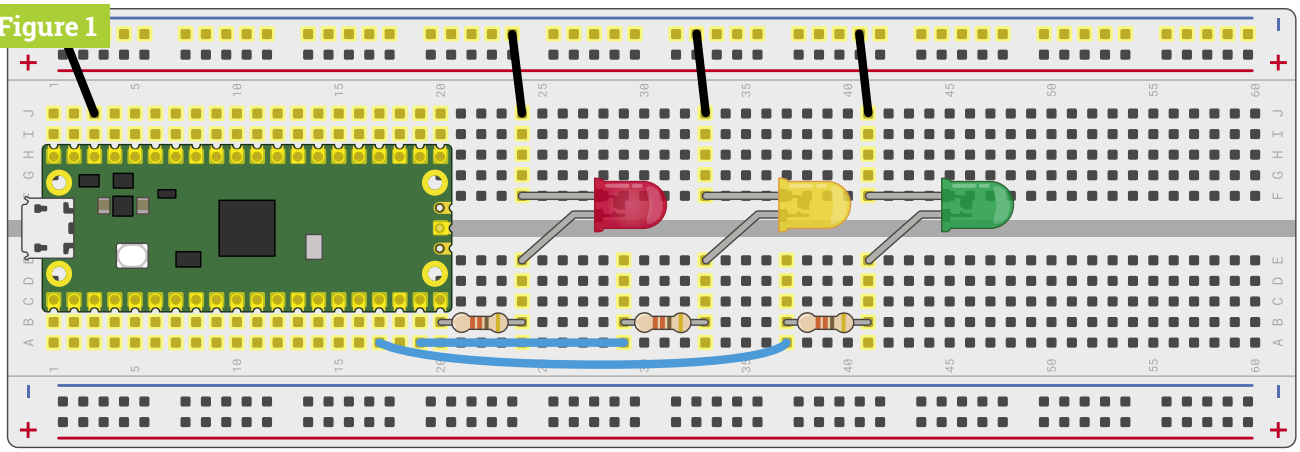
A simple traffic light

Start by building a simple traffic light system, as shown in **Figure 1**. Take your red LED and insert it into the breadboard so it straddles the centre divide. Use one of the 330Ω resistors, and a jumper wire if you need to make a longer connection, to connect the longer leg – the anode – of the LED to the pin at the bottom-left of your Pico as seen from the top with the micro USB cable uppermost, GP15. If you're using a numbered breadboard and have your Pico inserted at the very top, this will be breadboard row 20.

Take a jumper wire and connect the shorter leg – the cathode – of the red LED to your breadboard's

▼ **Figure 1** A basic three-light traffic light system

Figure 1



ground rail. Take another, and connect the ground rail to one of your Pico's ground (GND) pins – in **Figure 1**, we've used the ground pin on row three of the breadboard.

You've now got one LED connected to your Pico, but a real traffic light has at least two more for a total of three: a red light to tell the traffic to stop, an amber or yellow light to tell the traffic the light is about to change, and a green LED to tell the traffic it can go again.

Take your amber or yellow LED and wire it to your Pico in the same way as the red LED, making sure the shorter leg is the one connecting to the ground rail of the breadboard and that you've got the 330 Ω resistor in place to protect it. This time, though, wire the longer leg – via the resistor – to the pin next to the one to which you wired the red LED, GP14.

Finally, take the green LED and wire it up the same way again – remembering the 330 Ω resistor – to pin GP13. This isn't the pin right next to pin GP14, though – that pin is a ground (GND) pin, which you can see if you look closely at your Pico: the ground pins all have a square shape to their pads, while the other pins are round.

When you've finished, your circuit should match **Figure 1**: a red, a yellow or amber, and a green LED, all wired to different GPIO pins on your Pico via individual 330 Ω resistors and connected to a shared ground pin via your breadboard's ground rail.

To program your traffic lights, connect your Pico to your Raspberry Pi (or other computer) and load Thonny. Create a new program, and start by importing the machine library so you can control your Pico's GPIO pins:

```
import machine
```

You'll also need to import the utime library, so you can add delays between the lights going on and off:

```
import utime
```

As with any program using your Pico's GPIO pins, you'll need to set each pin up before you can control it:

```
led_red = machine.Pin(15, machine.Pin.OUT)
led_amber = machine.Pin(14, machine.Pin.OUT)
led_green = machine.Pin(13, machine.Pin.OUT)
```

These lines set pins GP15, GP14, and GP13 up as outputs, and each is given a descriptive name to make it easier to read the code: 'led', so you know the pins control an LED, and then the colour of the LED.

Real traffic lights don't run through once and stop – they keep going, even when there's no traffic there and everyone's asleep. So that your program does the same, you'll need to set up an infinite loop:

```
while True:
```

Each of the lines beneath this need to be indented by four spaces, so MicroPython knows they form part of the loop; when you press the **ENTER** key Thonny will automatically indent the lines for you.

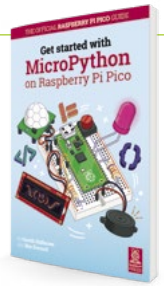
```
    led_red.value(1)
    utime.sleep(5)
    led_amber.value(1)
    utime.sleep(2)
    led_red.value(0)
    led_amber.value(0)
    led_green.value(1)
    utime.sleep(5)
    led_green.value(0)
    led_amber.value(1)
    utime.sleep(5)
    led_amber.value(0)
```

Click the Run icon and save your program to your Pico as **Traffic_Lights.py.** Watch the LEDs: first the red LED will light up, telling the traffic to stop; next, the amber LED will come on to warn drivers the lights are about to change; next both LEDs switch off and the green LED comes on to let traffic know it can pass; then the green LED goes off and the amber one comes on to warn drivers the lights are about to change again; finally, the amber LED goes off – and the loop restarts from the beginning, with the red LED coming on.

> "The pattern will loop until you press Stop, because it forms an infinite loop"

The pattern will loop until you press the Stop button, because it forms an infinite loop. It's based on the traffic light pattern used in real-world traffic control systems in the UK and Ireland, but sped up – giving cars just five seconds to pass through the lights wouldn't let the traffic flow very freely!

Real traffic lights aren't just there for road vehicles, though: they are also there to protect pedestrians, giving them an opportunity to cross a busy road safely. In the UK, the most common type



Get Started with MicroPython on Raspberry Pi Pico

For more physical computing projects to try on your Raspberry Pi Pico, grab a copy of the new book, *Get Started with MicroPython on Raspberry Pi Pico*. As well as learning how to use Raspberry Pi Pico's pins as inputs and outputs, you'll build a simple game, measure temperatures, save and load data to your Pico's file system, and even make a burglar alarm for your room. *Get Started with MicroPython on Raspberry Pi Pico* is available now from **magpi.cc/picobook**.

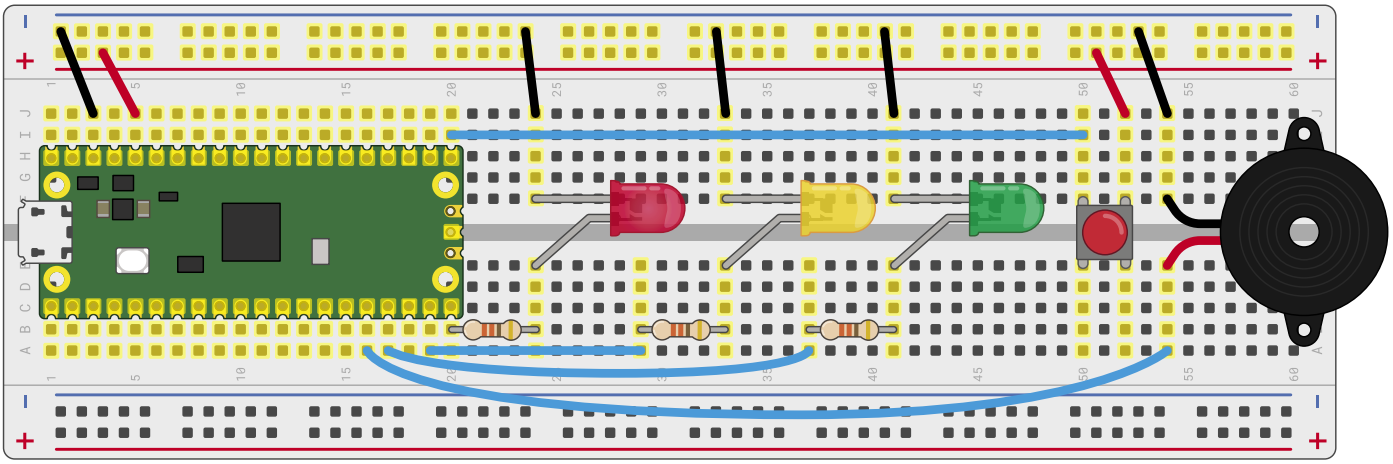


Figure 2

▲ **Figure 2** A puffin crossing traffic light system

of these lights are known as *pedestrian-operated user-friendly intelligent crossings or puffin crossings*.

To turn your traffic lights into a puffin crossing, you'll need two things: a push-button switch, so the pedestrian can ask the lights to let them cross the road; and a buzzer, so the pedestrian knows when it's their turn to cross. Wire those into your breadboard as in **Figure 2**, with the switch wired to pin GP16 and the 3V3 rail of your breadboard, and the buzzer wired to pin GP12 and the ground rail of your breadboard.

If you run your program again, you'll find the button and buzzer do nothing. That's because you haven't yet told your program how to use them. In Thonny, go back to the lines where you set up your LEDs and add the following two new lines below:

```
button = machine.Pin(16, machine.Pin.IN,
machine.Pin.PULL_DOWN)
buzzer = machine.Pin(12, machine.Pin.OUT)
```

This sets the button on pin GP16 up as an input, and the buzzer on pin GP12 as an output. Remember, your Raspberry Pi Pico has built-in programmable resistors for its inputs, which we are setting to pull-down mode for this project. This means that the pin's voltage is pulled down to 0 V (and its logic level is 0), unless it is connected to 3.3 V power (in which case its logic level will be 1 until disconnected).

Next, you need a way for your program to constantly monitor the value of the button. In the last two issues' Pico tutorials, all your programs have worked step-by-step through a list of instructions – only ever doing one thing at a time. Your traffic light program is no different: as it runs, MicroPython walks through your instructions step-by-step, turning the LEDs on and off.

For a basic set of traffic lights, that's enough; for a puffin crossing, though, your program needs to be able to record whether the button has been pressed in a way that doesn't interrupt the traffic lights. To make that work, you'll need a new library: _thread. Go back to the section of your program where you import the machine and utime libraries, and import the _thread library:

```
import _thread
```

A *thread* or *thread of execution* is, effectively, a small and partially independent program. You can think of the loop you wrote earlier, which controls the lights, as the *main thread* of your program – and using the _thread library you can create an additional thread, running at the same time.

An easy way to visualise threads is to think of each one as a separate worker in a kitchen: while the chef is preparing the main dish, someone else is working on a sauce. At the moment, your program has only one thread – the one which controls the traffic lights. The RP2040 microcontroller which powers your Pico, however, has two processing cores – meaning, like the chef and the sous chef in the kitchen, you can run two threads at the same time to get more work done.

Before you can make another thread, you'll need a way for the new thread to pass information back to the main thread – and you can do this using *global variables*. The variables you've been working with prior to this are known as *local variables*, and only work in one section of your program; a global variable works everywhere, meaning one thread can change the value and another can check to see if it has been changed.

To start, you need to create a global variable. Below your `buzzer =` line, add the following:

```
global button_pressed
button_pressed = False
```

This sets up `button_pressed` as a global variable, and gives it a default value of `False` – meaning when the program starts, the button hasn't yet been pushed. The next step is to define your thread, by adding the following lines directly below – adding a blank line, if you want, to make your program more readable:

```
def button_reader_thread():
    global button_pressed
    while True:
        if button.value() == 1:
            button_pressed = True
        utime.sleep(0.01)
```

The first line you've added defines your thread and gives it a descriptive name: it's a thread to read the button input. Like when writing a loop, MicroPython needs everything contained within the thread to be indented by four spaces – so it knows where the thread begins and ends.

The next line lets MicroPython know you will be changing the value of the global `button_pressed` variable. If you only want to check the value, you wouldn't need this line – but without it you can't make any changes to the variable.

> Only when the button is pressed will the final line of your thread run

Next, you've set up a new loop – which means a new four-space indent needs to follow, for eight in total, so MicroPython knows both that the loop is part of the thread and the code below is part of the loop. This nesting of code in multiple levels of indentation is very common in MicroPython, and Thonny will do its best to help you by automatically adding a new level each time it's needed – but it's up to you to remember to delete the spaces it adds when you're finished with a particular section of the program.

The next line is a conditional which checks to see if the value of the button is 1. Because your Pico is using an internal pull-down resistor, when the button isn't being pressed the value read is 0 – meaning the code under the conditional never runs. Only when the button is pressed will the final line of your thread run: a line which sets the `button_pressed` variable to `True`, letting the rest of your program know the button has been pushed. Finally, we add a very short (0.01 second) delay to prevent the `while` loop running too fast.

You might notice there's nothing in the thread to reset the `button_pressed` variable back to `False` when the button is released after being pushed. There's a reason for that: while you can push the button of a puffin crossing at any time during the traffic light cycle, it only takes effect when the light has gone red and it's safe for you to cross. All your new thread needs to do is to change the variable when the button has been pushed; your main thread will handle resetting it back to `False` when the pedestrian has safely crossed the road.

Defining a thread doesn't set it running: it's possible to start a thread at any point in your program, and you'll need to specifically tell the _thread library when you want to launch the thread. Unlike running a normal line of code, running the thread doesn't stop the rest of the program: when the thread starts, MicroPython will carry on and run the next line of your program even as it runs the first line of your new thread.

Create a new line below your thread, deleting all of the indentation Thonny has automatically added for you, which reads:

```
_thread.start_new_thread(button_reader_
thread, ())
```

This tells the _thread library to start the thread you defined earlier. At this point, the thread will start to run and quickly enter its loop – checking the button thousands of times a second to see if it's been pressed yet. The main thread, meanwhile, will carry on with the main part of your program.

Click the Run button now. You'll see the traffic lights carry on their pattern exactly as before, with no delay or pauses. If you press the button, though, nothing will happen – because you haven't added the code to actually react to the button yet.

Go to the start of your main loop, directly underneath the line `while True:`, and add the following code – remembering to pay attention to the nested indentation, and deleting the indentation Thonny has added when it's no longer required:

```
    if button_pressed == True:
        led_red.value(1)
        for i in range(10):
            buzzer.value(1)
            utime.sleep(0.2)
            buzzer.value(0)
```

Warning!

Always remember that an LED needs a current-limiting resistor before it can be connected to your Pico. If you connect an LED without a current-limiting resistor in place, the best outcome is the LED will burn out and no longer work; the worst outcome is it could do the same to your Pico.

traffic_light_controller.py

Language: **MicroPython**

```
import machine
import utime
import _thread

led_red = machine.Pin(15, machine.Pin.OUT)
led_amber = machine.Pin(14, machine.Pin.OUT)
led_green = machine.Pin(13, machine.Pin.OUT)
button = machine.Pin(16, machine.Pin.IN, machine.Pin.PULL_DOWN)
buzzer = machine.Pin(12, machine.Pin.OUT)

global button_pressed
button_pressed = False

def button_reader_thread():
    global button_pressed
    while True:
        if button.value() == 1:
            button_pressed = True
        utime.sleep(0.01)
_thread.start_new_thread(button_reader_thread, ())

while True:
    if button_pressed == True:
        led_red.value(1)
        for i in range(10):
            buzzer.value(1)
            utime.sleep(0.2)
            buzzer.value(0)
            utime.sleep(0.2)
        global button_pressed
        button_pressed = False
    led_red.value(1)
    utime.sleep(5)
    led_amber.value(1)
    utime.sleep(2)
    led_red.value(0)
    led_amber.value(0)
    led_green.value(1)
    utime.sleep(5)
    led_green.value(0)
    led_amber.value(1)
    utime.sleep(5)
    led_amber.value(0)
```

DOWNLOAD THE FULL CODE:
magpi.cc/github

```
        utime.sleep(0.2)
    global button_pressed
    button_pressed = False
```

This chunk of code checks the `button_pressed` global variable to see if the push-button switch has been pressed at any time since the loop last ran. If it has, as reported by the button reading thread you made earlier, it begins running a section of code which starts by turning the red LED on to stop traffic and then beeps the buzzer ten times – letting the pedestrian know it's time to cross.

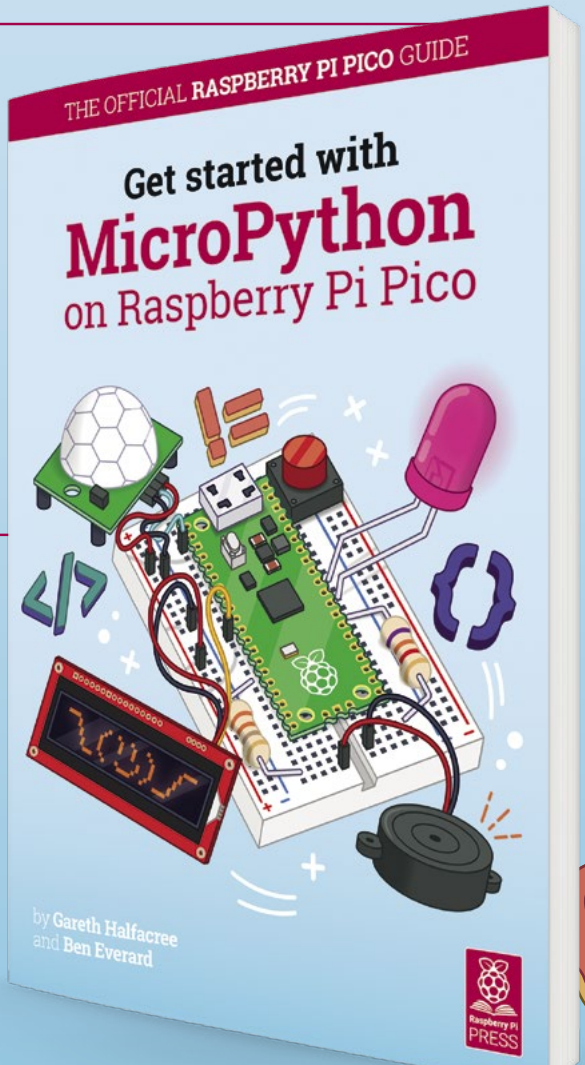
Finally, the last two lines reset the button pressed variable back to `False` – so the next time the loop runs it won't trigger the pedestrian crossing code unless the button has been pushed again. You'll see you didn't need the line `global button_pressed` to check the status of the variable in the conditional; it's only needed when you want to change the variable and have that change affect other parts of your program.

Your program should look like the code in **traffic_light_controller.py**. Click Run. At first, the program will run as normal: the traffic lights will go on and off in the usual pattern. Press the push-button switch: if the program is currently in the middle of its loop, nothing will happen until it reaches the end and loops back around again – at which point the light will go red and the buzzer will beep to let you know it's safe to cross the road. The conditional section of code for crossing the road runs before the code you wrote earlier for turning the lights on and off in a cyclic pattern: after it's finished, the pattern will begin as usual with the red LED staying lit for a further five seconds on top of the time it was lit while the buzzer was going. This mimics how a real puffin crossing works: the red light remains lit even after the buzzer has stopped sounding, so anyone who started to cross the road while the buzzer was going has time to reach the other side before the traffic is allowed to go.

Let the traffic lights loop through their cycle a few more times, then press the button again to trigger another crossing. Congratulations: you've built your own puffin crossing!

Get started with MicroPython on Raspberry Pi Pico

Learn how to use your new Raspberry Pi Pico microcontroller board and program it using MicroPython. Connect hardware to make your Pico interact with the world around it. Create your own electro-mechanical projects, whether for fun or to make your life easier.



- **Set up your Raspberry Pi Pico and start using it**
- **Start writing programs using MicroPython**
- **Control and sense electronic components**
- **Discover how to use Pico's unique Programmable IO**

Available now: magpi.cc/picobook

Use Visual Studio Code with Raspberry Pi

Install and use Microsoft's development environment

MAKER
Lucy Hattersley

Lucy is editor of *The MagPi* and loves it when her code works. Although not the very first time. That's just a bit weird.

magpi.cc

We love Microsoft's Visual Studio Code. It's an excellent programming environment that Microsoft has made available for free. Installing Visual Studio Code is just an apt command away, and once installed it's a joy to use.

It's powerful too. While Raspberry Pi OS's default IDEs (integrated development environments) Thonny and Geany are both beginner-friendly, Visual Studio Code is the recommended environment if you want to debug your code with a graphical interface.

Visual Studio Code has deep integration with all the programming languages you are likely to use. Having support for everything from HTML, through to Python, and up to C++ ensures that you only have to learn one tool.

Git support is baked in, so you can push and pull remotely without having to switch to the Terminal. There's a vast extensions marketplace offering a huge range of features, including code syntax extensions and code completions.

All this makes it our go-to IDE for coding on Raspberry Pi. In this tutorial, we're going to show you how to install Visual Studio Code and set it up for coding with Raspberry Pi and Pico.

01 Install Visual Studio Code

Visual Studio Code is now available in the default apt packages in Raspberry Pi OS. This means you can install it with just a simple Terminal command.

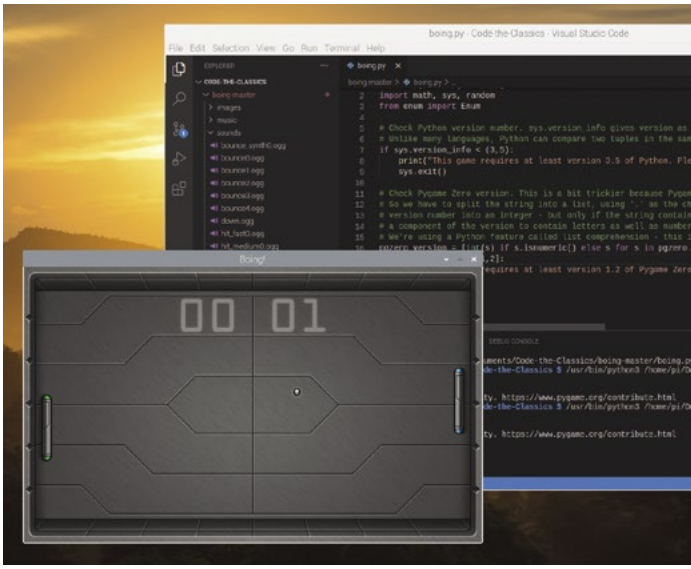
```
sudo apt update
sudo apt install code -y
```

Click the Raspberry Pi applications menu and choose Programming > Visual Studio Code to open the program.

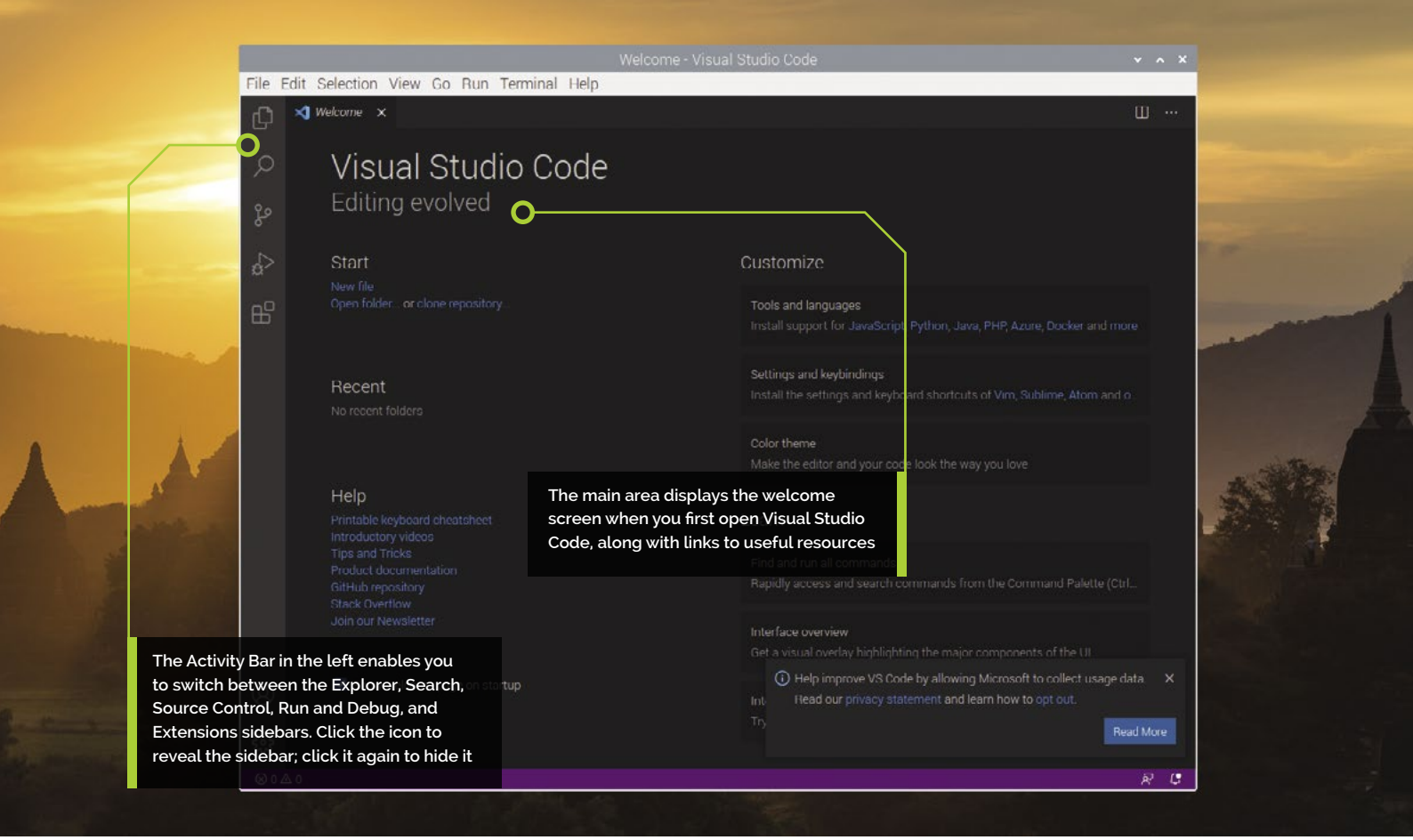
If you're looking to use Visual Studio Code along with the C/C++ SDK for Raspberry Pi Pico then it is best to use the Pico installation script (see 'Install with Pico'). This installs Visual Studio Code along with the extensions needed to debug Pico.

02 Explore Visual Studio Code

When Visual Studio Code first opens, you will see the Welcome screen. This has quick links for creating a new file, opening a file, and links to various Help documents (including a cheat sheet). In the bottom-left you will see a 'Show welcome



The Explorer in the Activity Bar makes it easy to navigate code and files, while the code preview makes it easy to scan long files. Here we are playing Boing from *Code The Classics*



page on startup' option. Keep it ticked for now. On the left of the screen is the Activity Bar. Click the icons and a sidebar will appear; click the same icon again to get rid of it. The Activity Bar is used to navigate most of the features of Visual Studio Code while you code in the main window.

03 Install the Python extension

We have no code to play with at the moment. So let's set up Visual Studio Code for use with Python and create the classic Hello World program.

Click on the Extensions icon at the bottom of the Activity Bar. The Extensions sidebar will appear. From here you can install a wide range of features that expand Visual Studio Code.

> "There's a vast extensions marketplace offering a huge range of features"

Click the 'Search extensions' box, enter 'python', and press **ENTER**. Because Python is such a popular language, hundreds of extensions will appear. Below each extension will be a short description and the name of the maker. We're looking for the one made by 'Microsoft'. It will almost certainly be the first result. Click on it and the extension details will appear in the Editor area. You will see the details and installation instructions. At the top are the number of times the extension has been installed (over 32 million at the time of writing) and a star rating from other Visual Studio Code users (four-and-a-half stars). These can give you an idea of how useful an extension will be.

Note that we are not installing Python itself into Raspberry Pi OS. You will already have Python installed there. We are installing support for it in Visual Studio Code. Click the blue Install button.

04 Create a Python file

Now let's create a Python file. Close the Extensions sidebar by clicking the Extensions icon and choose File > New File from the menu. A second tab will appear with Untitled-1. Choose File > Save As and name the file **hello.py**.

Visual Studio Code will automatically detect you are working on a Python program and will load the Python extension from Step 3.

On the first run, the Python Extension window will appear in the Editor, along with a notification in the bottom-right corner that 'Linter pylint is not installed'. Pylint is a code analysis tool that checks your code against the PEP-8 style guide (**magpi.cc/pylint**). It's handy to have around, but we're not going to use it here. Close the notification. Click on the **hello.py** file to switch back to it.

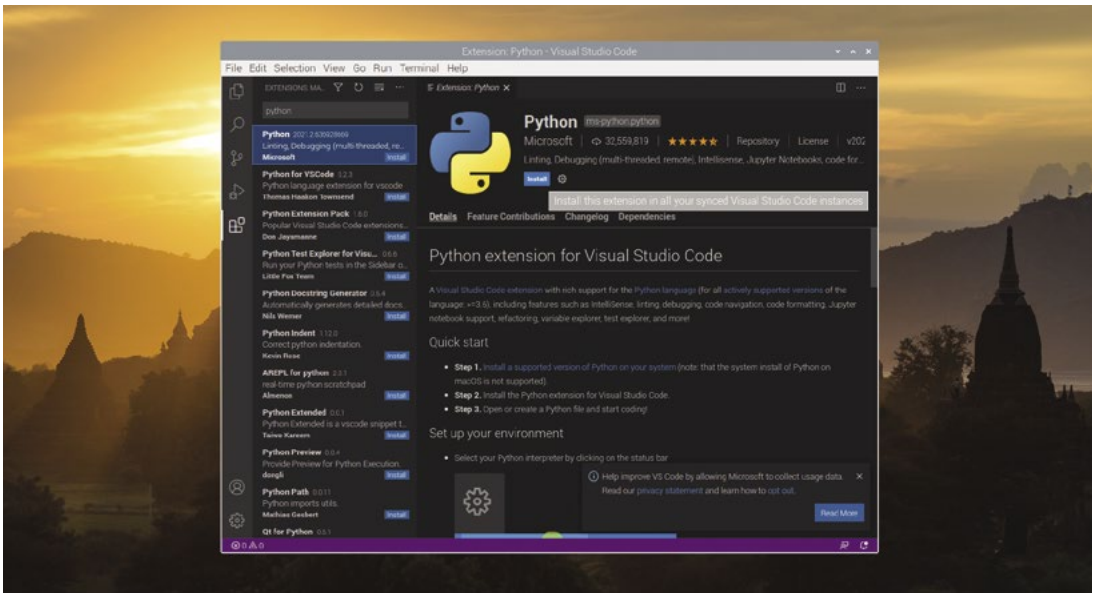
Top Tip

Install with Pico

If you have set up Raspberry Pi Pico with C/C++ development, you may already have Visual Studio Code installed (**magpi.cc/picoc**).

You'll Need

- Raspberry Pi
- Raspberry Pi OS
- Internet connection



Visual Studio Code has extensions for all major programming languages. The Python extension provides features such as colour code syntax and IntelliSense code completion

Top Tip

IntelliSense

Microsoft's code-completion is known as 'IntelliSense' (**magpi.cc/intellisense**) and you can expand code by pressing the **TAB** key. It's a great way to get parameter information, and to list members of objects (when you use dot notation). It's really helpful when coding.

Blog

Microsoft's Jim Bennett has written two blogs on Visual Studio Code for Raspberry Pi. Both are well worth taking a look at.

Visual Studio Code on Raspberry Pi
magpi.cc/vscodeblog

Code Remotely
magpi.cc/vscoderemotely

05 Run Hello World

Enter the classic Hello World program in the editor window:

```
print("Hello World")
```

As you start to type the 'pr' in 'print', IntelliSense will display options for Code Completion. You can carry on typing the code outt, or use the arrows to select a command from the list. Select 'print' and press **ENTER**.

Before running the program, we should pick the right interpreter. By default Visual Studio Code is set to use Python 2.7, whereas we want to code in Python 3.7

Click on 'Python 2.7.16 32-bit' in the bottom-left corner (the version number may have updated) and a list of interpreters will appear at the top. Choose 'Python 3.7.3 32-bit' from the **/usr/bin/python3** folder.

To run the program, click the small green play icon in the top-right corner (hover the mouse on it to see 'Run Python file in Terminal'. There are numerous other ways to run programs in Visual Studio code, including the Run and Debug option in the Activity Bar and the Run menu.

A Terminal window will appear at the bottom and you will see 'Hello World' as the output.

06 Debug

Now you've got the basics of Visual Studio Code, let's split Hello World onto two lines:

```
message = "Hello World"
print(message)
```

Hover the mouse to the left of the 'p' on line 2 and a small red dot will appear. Click it to set a breakpoint on line 2. Click Run and Debug in the Activity Bar and click Run and Debug. A list of configurations will appear at the top of the screen. Choose 'Python File' and press **ENTER**.

The program will run line 1 and stop on line 2 (with our breakpoint). The sidebar will display the Variables in your program (in this instance just 'message' with a value of 'Hello World').

At the top of the screen, a debug toolbar will appear with Continue, Step Over, Step Into, Step Out, Restart, and Stop options. Choose Continue to run line 2 of the program.

07 REPL and Debug Console

You can use REPL (Read-Eval-Print-Loop) in Visual Studio Code by choosing it from the Command Palette. Choose View > Command Palette (press **CTRL+SHIFT+P**) and enter 'repl'. Choose 'Python: Start REPL' from the command list.

A REPL instance will appear in the Panel. You can now enter Python programs a line at a time.

You can't, however, interact with your Hello World program from REPL. A better approach is to use the Debug Console. Click Run and Debug again. The program should halt on line 2 with the breakpoint you set in step 6.

Now click on Debug Console in the console window. At the bottom of the screen is an interactive Debug Console REPL. Enter 'message' in it to see the contents of your `message` variable. The Debug console will say 'Hello World'.

You can adjust variables from the Debug Console REPL. Enter:

```
message = "Hello from The MagPi"
```

Notice that the `message` variable in the Run and Debug window changes to display the new variable contents.

You can interact with the contents of the code using breakpoints, the Debug Console, and the debug toolbar. Click the Continue button to carry on running the program, and click on Terminal in the console to see the output message.

08 GitHub integration

Now that you have got the basics of coding and debugging, let's get some code. We're going to download some games from the *Code The Classics* book (**magpi.cc/codetheclassics**).

Click on Source Control in the Activity Bar and click Clone Repository. Enter the following URL into the text field that appears.

https://github.com/Wireframe-Magazine/Code-the-Classics.git

A window will appear so you can choose a location for the files. Select the **Documents** folder and click Select Repository Location.

The files will be downloaded and a notification will appear saying 'Would you like to open the cloned repository?' Click Open.

09 Looking at detailed code

The Explorer sidebar will display all the folders for the five games used in Code The Classics. Open Boing Master and click on **boing.py**.

Notice that the right of the screen now displays a preview of all the code. You can use this to quickly zoom around the code. Click the Split Editor Right button in the top-right corner. This splits the window in half and enables you to view different parts of the code at the same time. You can continue to split the code into three, or more, views. Any changes you make will appear in all windows. Click the close icon on any view.

The Explorer sidebar also presents folders for images, music, sounds, and other components. You can view the images in the Editor window.

Press Run to play a game of Boing. Being able to view all the code and files in Visual Studio Code will make it much easier to understand the code construction.

Part 03

Cheap Trills for all

Make a Trill Guitar that is so easy to play you won't believe it

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 Pi Projects for Dummies*.

magpi.cc/mikecook

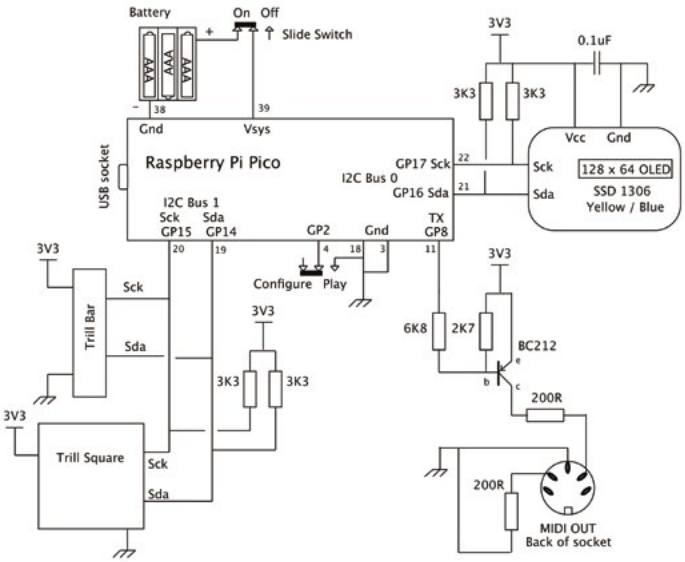
Having seen how to read the Trill sensors in Python, we are now going to show you how to make a standalone Trill MIDI guitar using a Pico Pi Controller. Here, we show you how to make it, and next month how the software makes it play.

01 How to play the flute

In a satire of the long-running *Blue Peter* children's program, *Monty Python* once famously told viewers how to play a flute, by blowing down one end and moving your fingers over the keys. That is a bit complex compared to how you can play our Trill Guitar. A Trill bar with a square below it looks like a guitar already. We had planned to make one when we started our look at the Trill sensors, but the arrival of Raspberry Pi Pico changed the direction of our design completely.

Figure 1

▼ **Figure 1** Schematic of the Trill Guitar



02 The concept

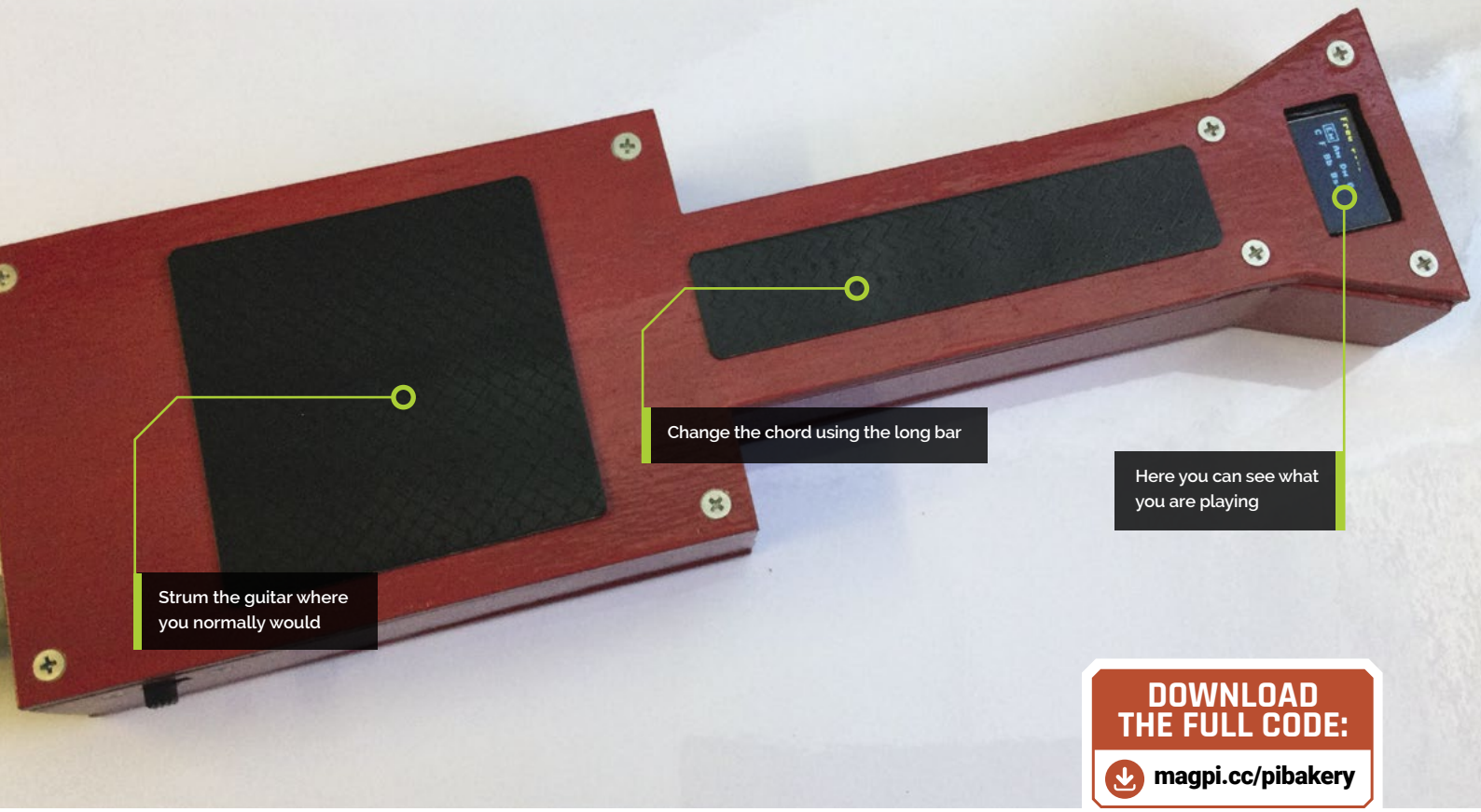
The idea is that we use the square Trill sensor as the guitar strings to strum or pluck, and the Trill bar sensor to change the chords, so you don't have to learn the finger patterns for each chord. Then these touches are converted into MIDI note on and note off messages, and sent out to a MIDI sound generator, like the one we made in our MIDI sound box project in *The MagPi* #63 (**magpi.cc/63**). You could even combine this project with an amplifier and speaker for an all-in-one instrument if you like.

03 The design

The big advantage in using a Pico over something like Raspberry Pi Zero is that it consumes way less power, and you don't have to be careful about turning it off, just disconnect the power. It also has zero boot time. A Pico design can run off three AA batteries; this project draws only 30 mA, so you should get around 65 hours of continuous play from a single set. Finally, adding an OLED display allows you to customise how the guitar works when you are away from your computer, and show you what chord you are playing.

04 I2C buses

As Pico has two I2C buses, we used Bus 1 for the Trill sensors, and Bus 0 for the OLED display. Changing the I2C speed on the Pico is simple, so we could run the Trill sensors at 400kHz and run the OLED display at 1MHz for a quick screen refresh rate. We chose one of those two-colour OLED displays, the sort that has a built-in filter over sections of the screen. We thought the yellow top line, with the rest in blue, looked good; however, any colour will work without modifying any code or hardware.



DOWNLOAD THE FULL CODE:
magpi.cc/pibakery

05 Schematic

The schematic for our project is shown in **Figure 1**, and is quite simple. Perhaps the most involved part is the MIDI output circuit in the lower right corner. We used a BC212 PNP transistor, but any general-purpose PNP transistor will do. The two I2C buses each need their own set of pull-up resistors; when using a Raspberry Pi computer, you do not need these because they are built into the hardware of the board for Bus 1. We put a decoupling capacitor on the power and ground feeds out to the three I2C devices.

06 Switches

There are two change-over switches; we used slider switches, but any sort should work. One controls the power from the batteries, and the other controls the guitar's mode: either playing or configuring the guitar. If the USB is connected to Pico for programming, then that will power the system, and the battery power should not be switched on, otherwise the USB's voltage will be applied to the batteries. As connecting a USB lead is not the normal way to use the project, we thought it not worth adding protection components if this were to happen.

07 Construction

The circuit is built on a piece of stripboard 25 rows by 20 holes. For mounting, four 3 mm holes are drilled in each corner. **Figure 2** shows the back of the stripboard, and you should cut the tracks at the places shown with the grey squares over the holes, or grey bars between the holes. The front of the board is shown in **Figure 3**. This Pico has had

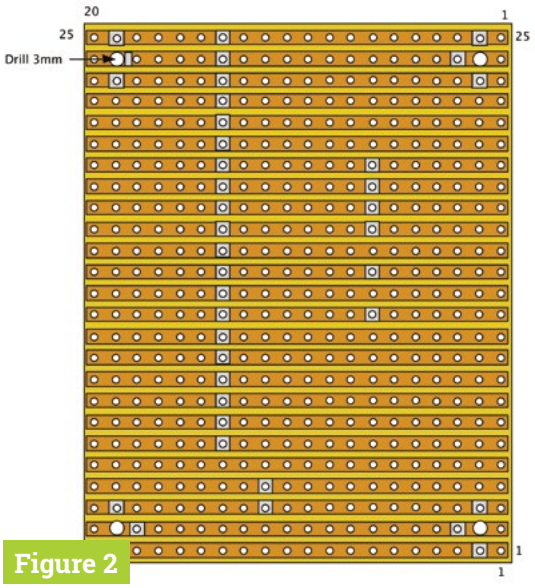


Figure 2

You'll Need

- Trill Bar and Square sensor **magpi.cc/trill**
- OLED 128×64 I2C display SSD 1306 yellow/blue **magpi.cc/trilloled**
- Raspberry Pi Pico **magpi.cc/pico**

Warning!
Sharp tools

Many cutting tools are used in this tutorial. Take necessary precautions when using them.

Figure 2 The stripboard tracks to cut

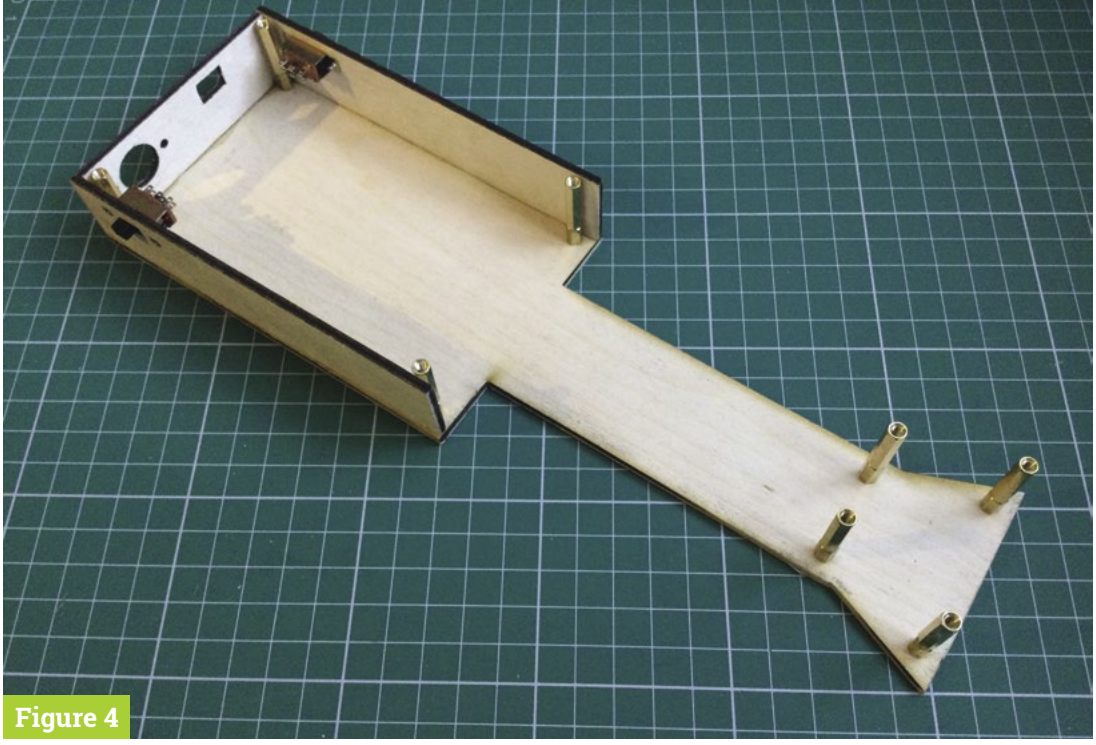


Figure 4

▶ **Figure 4** Three sides added to the base

downward-pointing pins soldered to it and plugs into two rows of sockets, making it stand tall from the board. You can miss out the sockets if you want to solder your Pico directly onto the board.

Top Tip

Mounting the Trill sensors

Pass the cables through the top of the case, and fix them in to the top with some small strips of double-sided sticky tape.

08 The guitar body

The guitar body is constricted from 3 mm plywood; the drawings can be found on our GitHub page (**magpi.cc/pibakery**). We printed these out on a piece of paper, and used spray mount glue to stick this temporarily to the wood sheets. Then, we used a combination of a tabletop saw and fretsaw (scroll-saw) to cut them out. Of the three large holes in the top, only the display's needs to be neat, as the other two are covered by the Trill sensors. Be sure to measure your OLED display because there are two slightly different sizes; the drawings are for the larger one.

▼ **Figure 3** The front of the board assembly

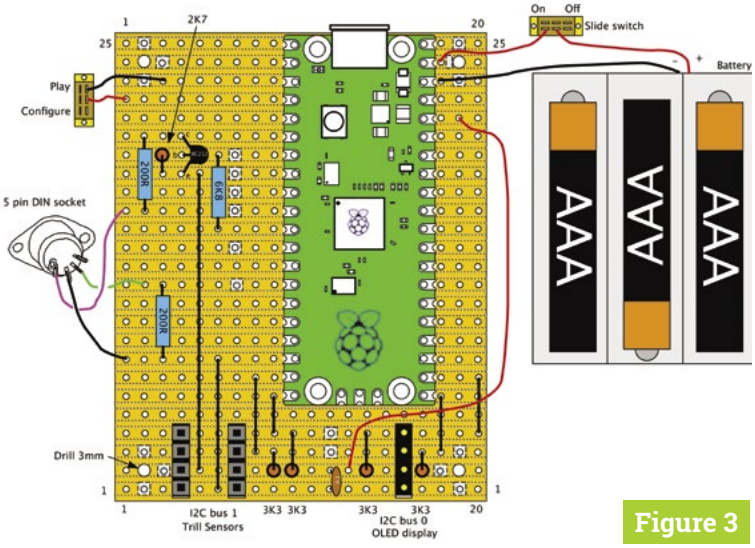


Figure 3

09 The sides

The top and the base were taped together, and the holes drilled through them both to ensure they aligned exactly. These holes are used for the pillars that hold the top and bottom apart, and should be countersunk on the outside. The sides of the guitar should be glued on one at a time to the bottom. We used superglue, and a small engineer's set square to make sure they were vertical. **Figure 4** shows when three sides had been added. The sides around the display need to be bevelled to fit, but all the other sides are butt joints.

10 Finishing the case

As more sides are glued in place, the structure becomes more solid. To finish off, we put a fillet of wood filler on the inside, as well as in the gaps where the bevelled edges did not meet exactly. Then the circuit board should be placed

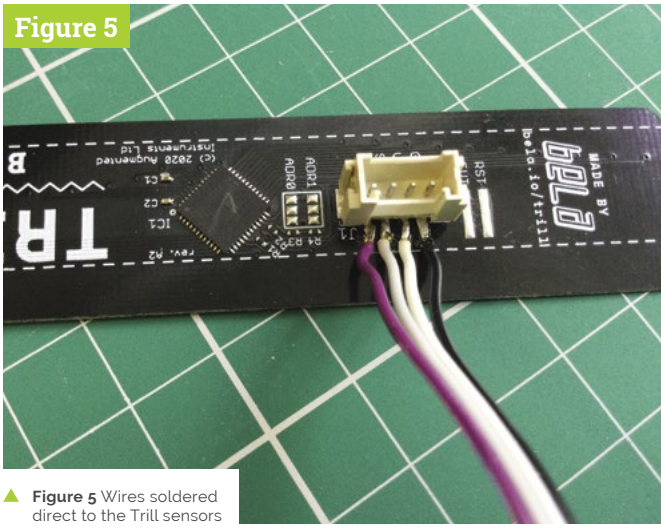


Figure 5

▲ **Figure 5** Wires soldered direct to the Trill sensors

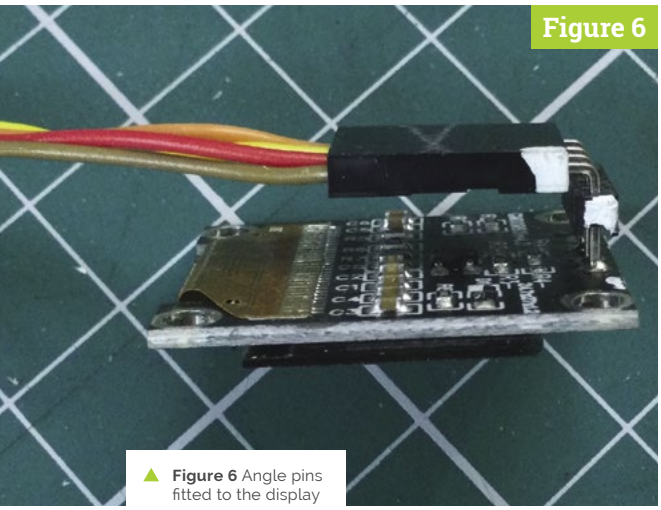


Figure 6

Figure 6 Angle pins fitted to the display

in line with the USB hole and 3 mm holes drilled through into the base of the guitar. Following this, the board should be mounted on 3 mm nylon standoff spacers, and the USB socket should be in the middle of the hole. Remove the board, sand off, and paint the whole case; we used red.

> The sides around the display need to be bevelled to fit

11 I2C peripherals

We soldered wires direct to the Trill sensors (**Figure 5**) rather than using the supplied connector, to reduce the height needed for a socket. Instead, we used pin headers on the board for all I2C connectors, and made up leads accordingly. With the OLED display, we removed the straight connector by inserting the blunt end of a Stanley knife between the PCB and plastic, and left the plastic pin holders off the board. This allowed the pins to be unsoldered one at a time, and a right-angled pin header fitted instead, as shown in **Figure 6**.

12 Final assembly

Note that the clock and data of the display were swapped over compared to the Trill sensors, so we kept the same board layout and just compensated for that in the lead from the display to the board. **Figure 7** shows the whole assembly prior to fitting the top. The top and bottom of the box are held together using pillars; the trick is to have the pillars just a millimetre or two shorter than the sides in order to provide room for a bit of tension to be applied. The battery pack is held in place with adhesive foam pads.

13 Software

Next month we will look at the software to turn this into a fully-fledged instrument. To get good results is not as simple as you might think, but the complexity is all hidden from the user. We will use the two-dimensional reporting of a touch on the square sensor to not only pick which string is being plucked, but also the velocity (volume) of the MIDI note settings, so that you can play loud or soft depending on where you strum. The bar sensor can also be used in two modes, either defining the chord that is played or as feedback while modifying how the guitar plays.

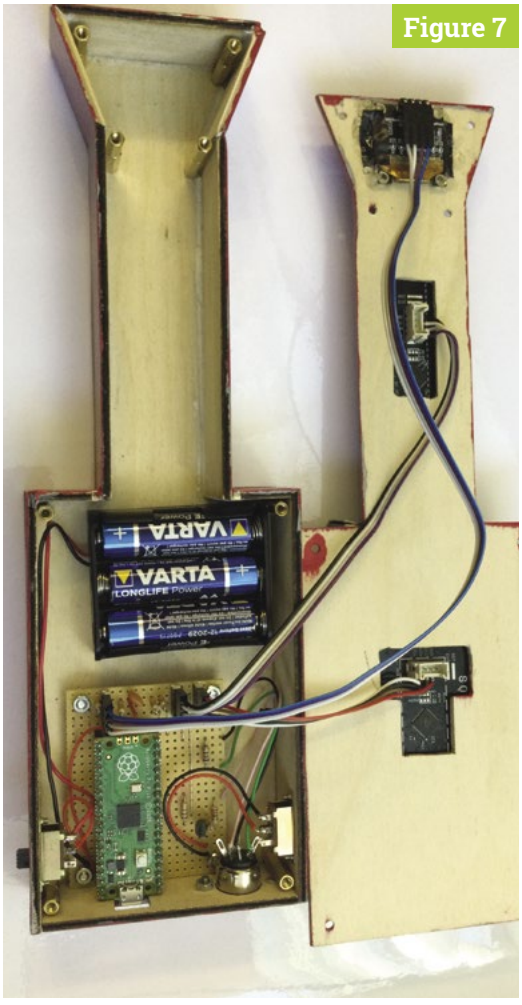


Figure 7

Figure 7 All parts assembled, now put the top on

Top Tip

BC212

Note there are variants of the BC212 transistor with different pin configurations. The BC212L is not the same as the BC212.

Pico-fied Raspberry Pi projects

Raspberry Pi Pico is already making waves as a cheaper, smaller alternative for traditional Raspberry Pi projects. By **Rob Zwetsloot**

When we first heard about Raspberry Pi Zero, our minds raced with how the tiny version of Raspberry Pi could be used in similar ways to its credit-card-sized siblings, yet fit into even tinier spaces. Raspberry Pi Pico was no different; however, as it was now a microcontroller, different ways of using it sparked our imagination.

While we knew it may be better suited to some classic Raspberry Pi projects than an actual Raspberry Pi, we did not consider the number of ways people would Pico-fy their projects. Here's what people have made, and here are some ideas for you to pursue...

Motor controller

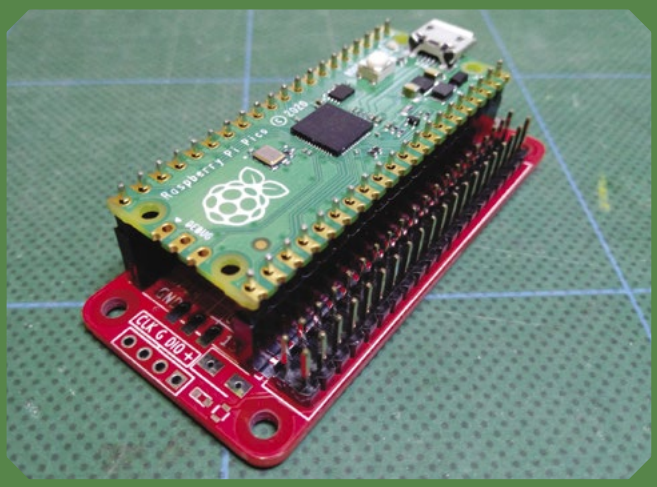
A way to control robot motor control boards designed for Raspberry Pi with Raspberry Pi Pico – it's different enough that you need a solution!

How it works:

We don't want to say that this is cheating – it's not – but it is an adapter for a board designed with Raspberry Pi Zero in mind. GPIO functions are programmable to be similar enough to work. This will possibly also be for sale at some point.

Maker:

Neil Lambeth (**@NeilRedRobotics**)



Radio receiver

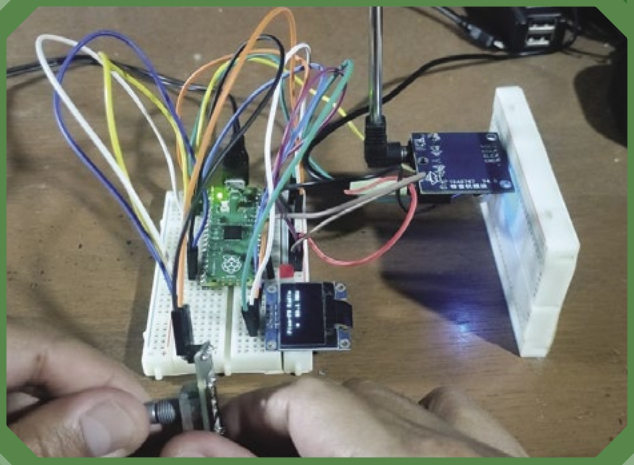
Listen to the radio, or at least tune into the right frequency, using this Raspberry Pi Pico project. Case to come.

How it works:

A TEA5767 module is used as an FM receiver and hooked up to a Pico. A tiny little LCD screen shows the frequency that Pico is currently tuned in to, using a little variable resistor to change the frequency just like on an older radio. Raspberry Pi Pico also has the ability to output sound, so with some extra tweaks you could probably upcycle an old radio or create your own.

Maker:

tkraspilabs (**@tkraspilabs**)



Ideas to try

Physical games and board games that have a little bit of electronics can be easily made or upgraded with a Pico. While someone has already made a Simon, we've not seen a dice roller or a Monopoly money system that allows for less cheating. With some motion sensors, they could easily be used to control open drones and such as well.

Simon says game

Like the Simon toy of old, this Pico will flash specific LEDs in a sequence and you have to remember it, with each sequence getting longer and, in theory, harder.

How it works:

A simple script which lights an LED, stores what that was, and then waits for player input. Like much game code, it involves a lot of lists and checking of lists. You can find out more here on the website for it: **magpi.cc/picosimon**.

Maker:

GeekDude (**@geektechstuff**)

Music and MIDI generation

Whether it's simple beeps or harmonic piano, it seems like Pico has extraordinary audio capabilities, as long as you can find them.

How it works:

The MagPi regular Mike Cook has made use of Pico's PIO (Programmable Input/Output) to generate simple tones by toggling a pin, based on one of the Pico code examples: **magpi.cc/pio1hz**. Kevin attached some audio gear to Pico and programmed it in MicroPython to produce MIDI notes, in this case some Bach: **magpi.cc/picomidi**.

Maker:

Mike Cooke (**@Wee_Grumphie**) and Kevin (**magpi.cc/diyelectro**)

Amazing uses – OS installation

If you want to run an operating system on Pico, look no further than FUZIX. It's a very small version of UNIX. It very likely won't replace, well, any of your main computers now, but it's a remarkable feat to port it to RP2040. You can find out more about how it works, and how to experiment it with yourself, on this Raspberry Pi blog post: **magpi.cc/fuzix**.

Temperature logger

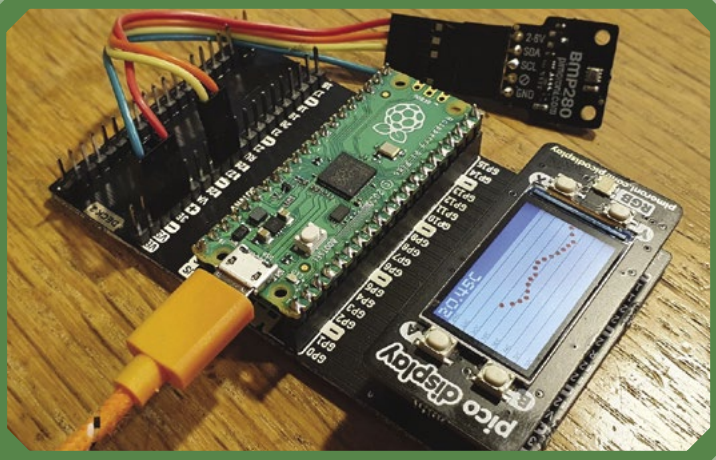
Keep track of the temperature in a given location, and display the results on a graph. This is great for smart thermostats or manually setting times.

How it works:

Taking temperature data from a thermistor is fairly simple with Python – all you then need to do is store that data, preferably in a text file. The data from that file is then plotted on a graph and displayed on a Pico Display, which you can find out how to use from Pimoroni's GitHub: **magpi.cc/picodisplay**.

Maker:

David Booth (**@Worlds6440**)



Auxiliary keyboard

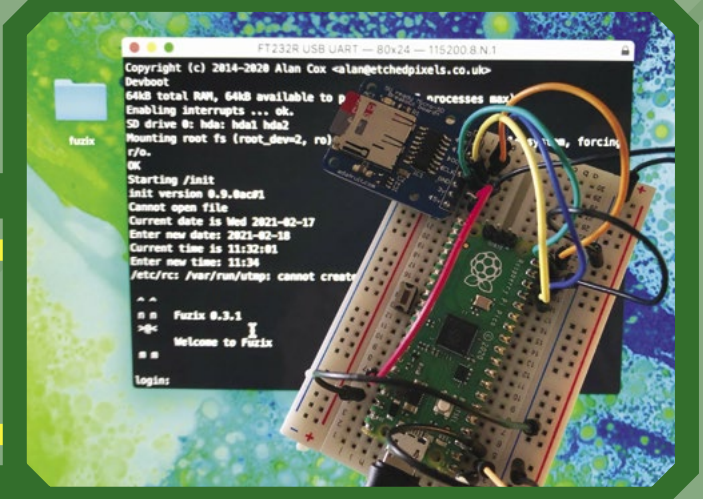
Perfectly described by Airton himself: "A mini keyboard with Raspberry Pi Pico, it will be my hotkey to mute and unmute Zoom, even when I'm not on Zoom window."

How it works:

Pico can function as a USB keyboard using C and TinyUSB (**magpi.cc/tinyusb**), or by using MicroPython (**magpi.cc/mpkeyb**). It works great with custom keys like Airton used, or with something like a Pimoroni RGB Keypad.

Maker:

Airton Zanon (**@airtonzanon**)



Ideas to try

Home automation is something Raspberry Pi is great at, and we expect to see amazing homes powered by Raspberry Pi Pico in the near future. Even if it's working with a Raspberry Pi that powers the whole system, Picos can be strategically placed to control individual lights and curtains and garage doors.

Sprite rendering

The Sega Mega Drive did not have Mode-7 style graphical rotation and scaling like the Super Nintendo, but that hasn't stopped James from implementing it on a Pico with Pico Display. With the transparency, it also goes into Game Boy Advance territory.

How it works:

Manipulating sprites like this is far easier than it was in 1991, especially with Python. You can have it create a scene that is updated at a specified fraction of a second, with different sprites' position, rotation, and transparency updated each time, using some simple maths to make it animated.

Maker:

James Sutherland (**@jamessutherland**)

Hydrometer

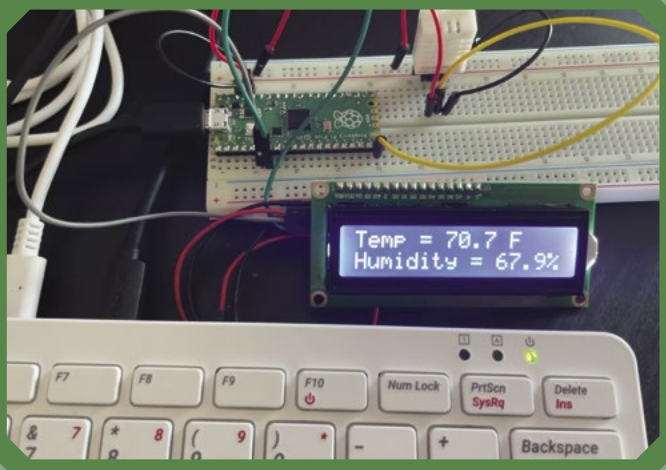
Check the temperature and humidity with Raspberry Pi Pico. Perfect for green houses, planters, or rooms you need to environmentally control.

How it works:

This is one is very simple, and in some ways slightly easier than with Raspberry Pi, as Pico allows for analogue inputs. With a bit of standard code and a nice LCD screen, you can easily convert the data coming from a sensor into something you can read, like in the photo.

Maker:

Caroline Dunn (**@thecarolinedunn**)



Pico robotics

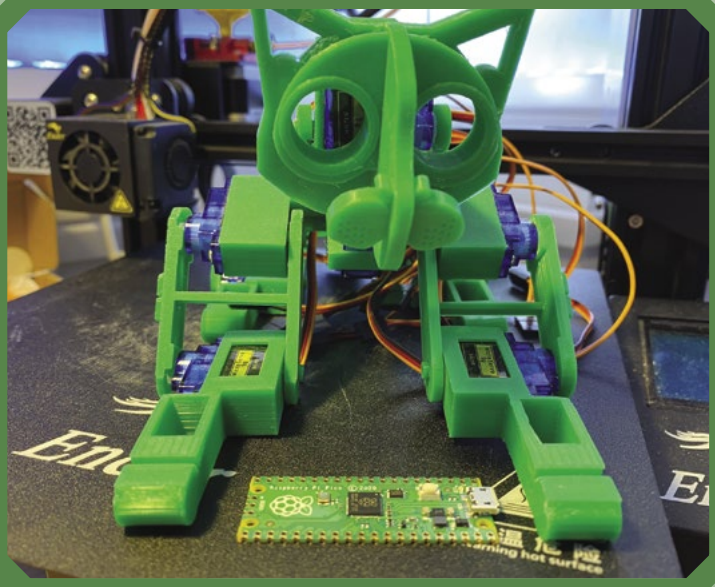
Using a Pico as the controller for motors in a robot means you can skip the motor controller step, and reduce down on parts and power.

How it works:

Pico supports analogue on some of its pins, so with the right code and tweaking you can get it to supply the right power to the motors for specific tasks. This one is a bit of trial and error, though, and we would recommend making use of a motor controller until we can get some good code and tutorials for this kind of usage.

Maker:

Kevin McAleer (**@kevsmac**)



Disco lights

Using a MOSFET switch to power lights that require much more power than something like a Pico can provide, all in the name of funk.

How it works:

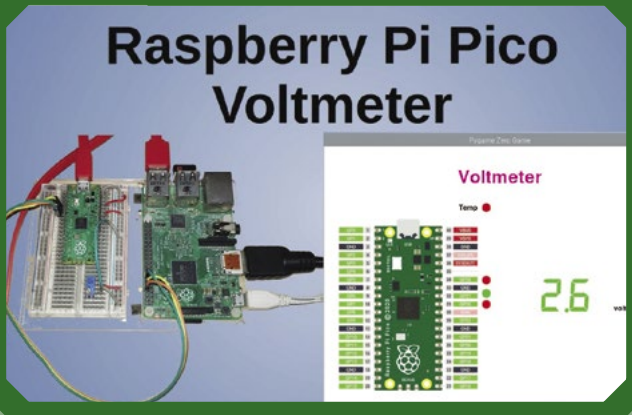
A MOSFET is able to control larger currents using a small input signal, like how a car starter motor works. In this case, for quite powerful lights that a Pico, Raspberry Pi, or other microcontroller would not be able to power themselves. This requires a bit of electronics engineering to get right, along with code to activate the right parts of the circuit, and Stewart goes into it with his video: **magpi.cc/picodisco**.

Maker:

Stewart Watkiss (**@stewartwatkiss**)

Amazing uses – Pico collaboration

Friend of the magazine Stewart Watkiss has been doing a lot of work with getting Raspberry Pi and Pico to work together – similar to how we've written about using Arduino with Raspberry Pi in the past! In fact, he's managed to get Raspberry Pi, Pico, and an Arduino working together, as you can see in this video: **magpi.cc/picoi2c**.



IoT Cricket

SPECS

INPUTS:
1 × digital, 1 × digital or analogue (8-bit resolution), wake-up trigger, on-board temperature sensor

DIMENSIONS:
37.2 × 16.4 × 4 mm

WIRELESS RANGE:
Up to 100 metres

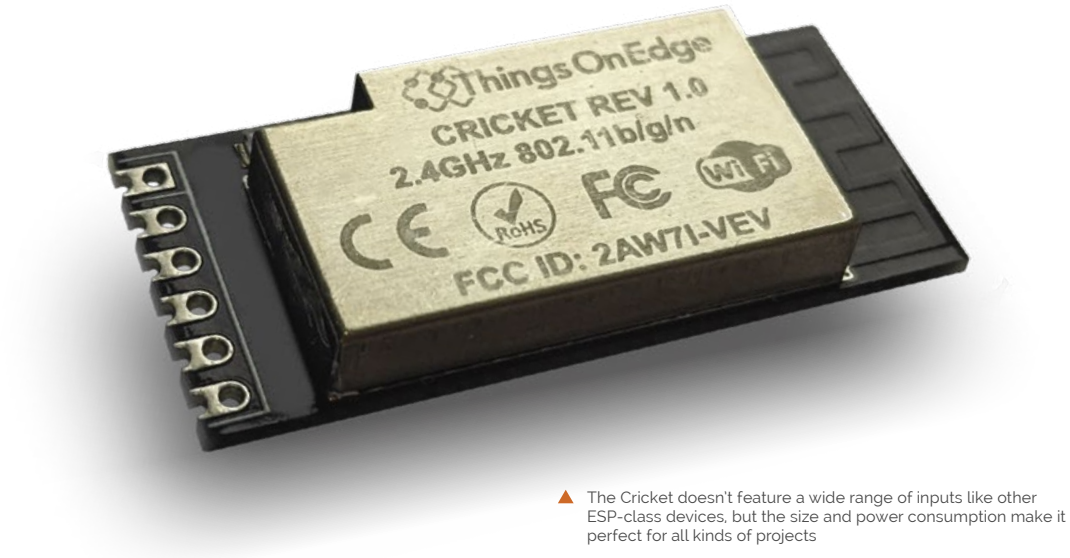
WIFI:
2.4GHz 802.11 b/g/n WPA / WPA2

OPERATING VOLTAGE:
1V–3.5V

CHIPSET:
ESP8266EX 32-bit @160MHz

▶ Things On Edge ▶ **magpi.cc/wificricket** ▶ £16 / $22

This new British-designed device aims to make building the Internet of Things as easy as a few clicks. **PJ Evans** checks out the no-code approach



▲ The Cricket doesn't feature a wide range of inputs like other ESP-class devices, but the size and power consumption make it perfect for all kinds of projects

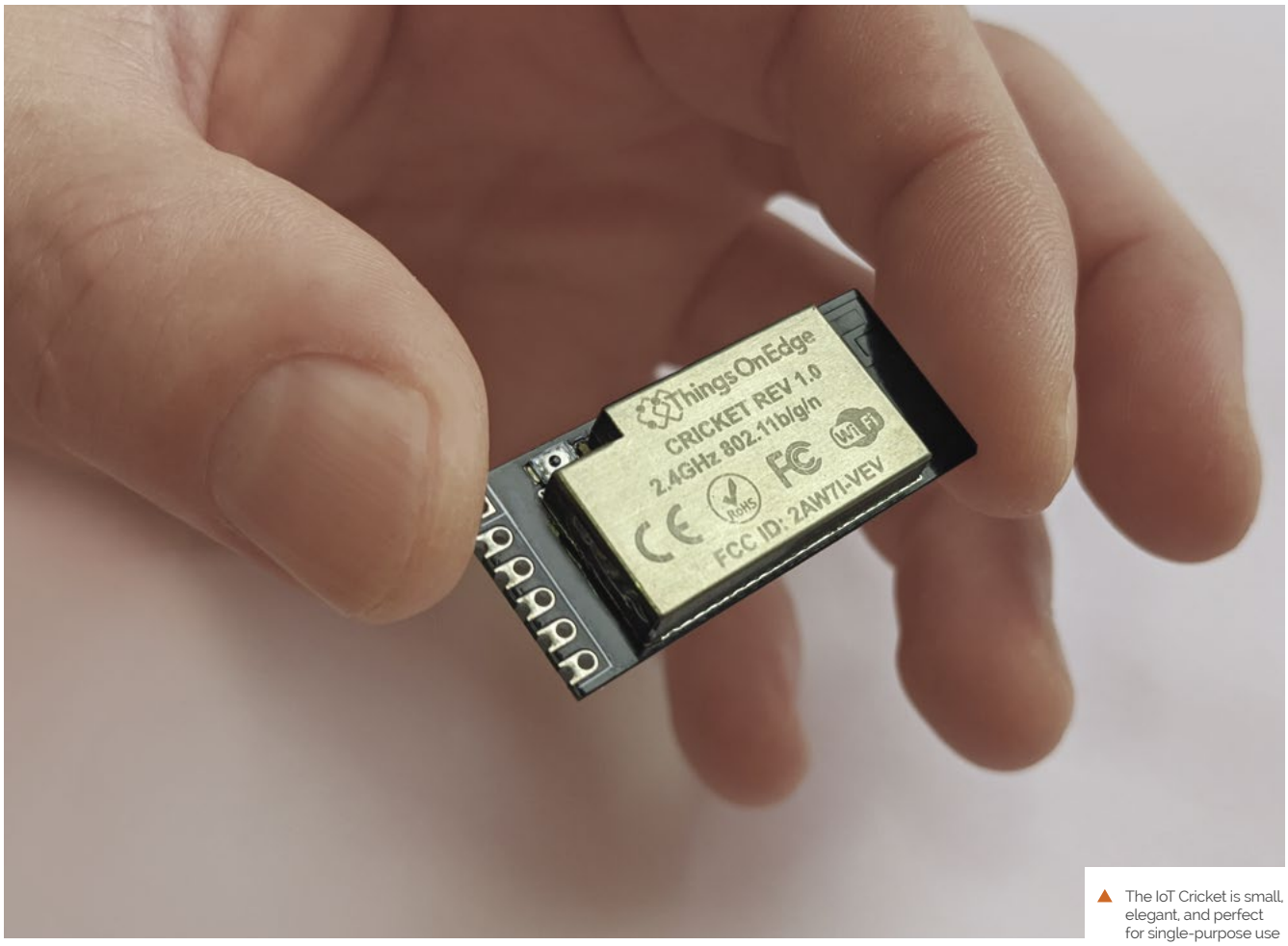
There's been a lot of talk recently about microcontrollers, and Raspberry Pi has recently entered the field with Raspberry Pi Pico (we're *sure* we've mentioned it).

Microcontrollers have many different use cases, and this new gizmo from Cambridge-based Things On Edge is squarely targeting the Internet of Things (IoT) market. The IoT Cricket is a small package based on the ubiquitous ESP8266 chipset, popular for its solid wireless LAN support. ESP microcontrollers are widely available and some can be had for just a couple of pounds, so it may come as a surprise to see a British-designed and made product introduced to an already saturated market. Turns out there are a few things that make this device different from the rest.

The IoT Cricket appears to have been designed around the philosophy of 'do one thing and do it well'. At first glance it may not seem like a good deal. It's more expensive than many similar controllers and lacks a full range of inputs as seen on Arduino-compatible ESP devices. In fact, the IoT Cricket boasts a total of one digital and one analogue input and a 3.3V output. This simplistic approach and some clever design solves one of the great headaches of ESP-based IoT devices: power. By combining a real-time clock into the design to control wake-ups as well as a 'wake up' line, it draws true 0A when idle. It can also operate on voltages from 1V up to 3.3V. This means you can power it from a single AAA battery for potentially months, even years, depending on activity.

Zero-code configuration

You can't write code for the IoT Cricket. Everything is configured using an on-board web interface (there is also an over-the-air solution). Placing it into configuration mode (using the smallest button we've ever seen) starts a wireless hot spot that allows you to set the behaviour of the device. Options include using the RTC to wake the device at given intervals, how to read inputs,



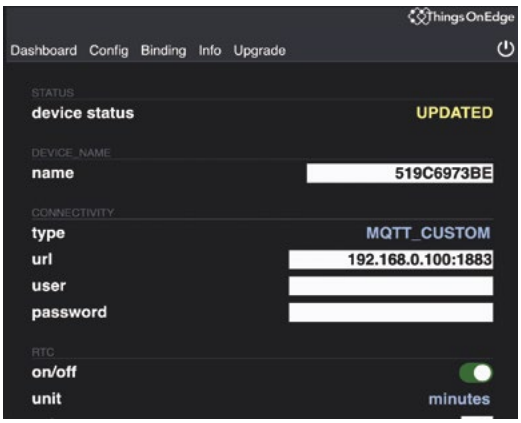
▲ The IoT Cricket is small, elegant, and perfect for single-purpose use

> The IoT Cricket appears to have been designed around the philosophy of 'do one thing and do it well'

and also what to do with the data. IoT Crickets support MQTT and HTTP GET or POST actions, making them instantly compatible with a huge range of monitoring and alerting services. Things On Edge even provides a free MQTT broker if you don't want to set up your own.

Programming microcontrollers is not for everyone (especially if you're using C), so this novel approach of configuring the IoT Cricket places the Internet of Things within the reach of a much wider audience. Even if you are accustomed to coding ESP devices, getting a simple project up and running with this approach is much quicker.

Things On Edge has provided several examples of projects using the IoT Cricket, including door sensors, wireless light controllers, motion detection, and more. There is also considerable documentation on integrating with services such as IFTTT and Home Assistant. They've even included an on-board temperature sensor (TMP1075DSG) so you can get started with no soldering. The online documentation can be a little hard to wade through, but seems to be constantly improving. All in all, it's a very impressive device if you want ultra-low power consumption and very easy setup.



◀ You can configure the IoT Cricket through its web interface, even remotely using its OTA feature

Verdict

If you're not into coding, or are looking for devices that could truly last years on batteries, this might be the smoothest route to your own IoT utopia. Cheaper options are available, but none is this easy to use.

9/10

Pico Explorer Base

SPECS

DISPLAY::
1.54-inch IPS LCD screen, 240×240 pixels

FEATURES:
170-point breadboard; breakout header including I2C, SPI, ADC; dual H-bridge DW8833 motor driver; 2 × Breakout Garden I2C sockets; 4 × tactile buttons

DIMENSIONS:
117×63×20 mm

▶ Pimoroni ▶ **magpi.cc/picoexplorer** ▶ £22 / $26

Experiment with electronics and a mini display with this add-on board for Raspberry Pi Pico. By **Phil King**

The launch of Raspberry Pi Pico saw a whole raft of add-ons created for the tiny, but powerful, microcontroller board. The Pico Explorer Base is one of the most interesting offerings, enabling you to plug and play with standard electronic components to explore physical computing more easily. It also has the bonus of a mini LCD display, dual H-bridge motor driver, and a couple of breakout slots.

To use the Explorer Base, your Pico will need to have male pin headers soldered, facing downwards – if you don't fancy doing this yourself, it's possible to buy Pico boards with pre-soldered pins. It's then just a case of mounting your Pico in the dual female headers; a helpful 'landing area' graphic on the Explorer Base indicates which way round to place it.

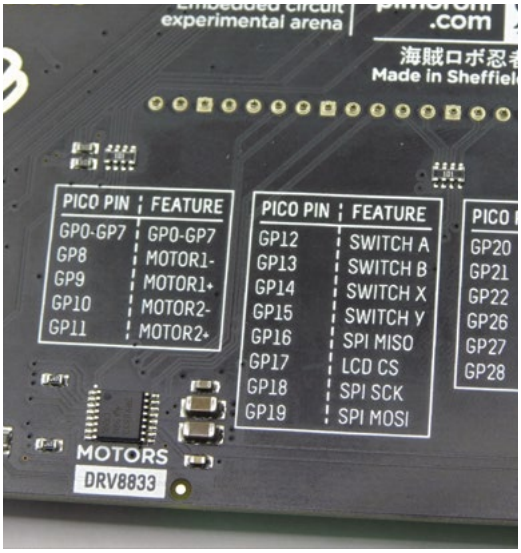
Making connections

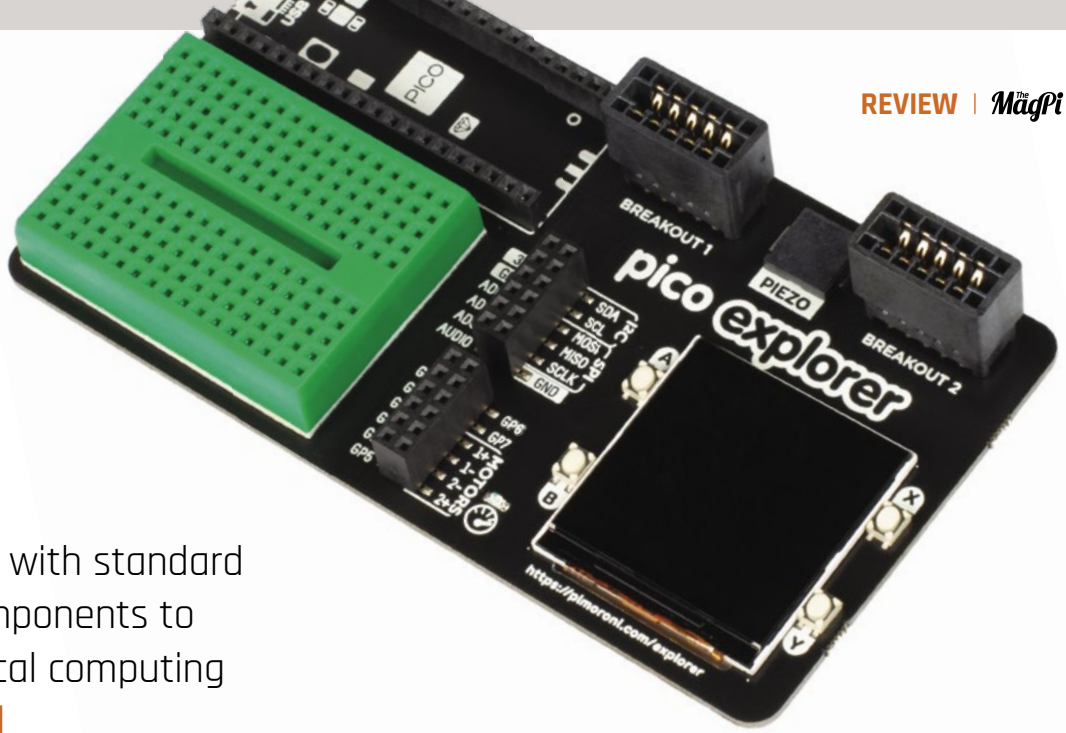
On the left side of the Explorer Base is a mini green breadboard with 170 points. While this may prove a little cramped for some projects, you could always just use a separate breadboard to house extra components. Note that no electronic components are supplied with the board, so it's up to you to source your own LEDs, buttons, sensors, etc., along with the male-to-male jumper wires to connect them.

Rather than wiring components to Pico's pins directly, a selection of its pins are broken out via two 12-pin female headers. These are clearly labelled and include I2C, SPI, ADC, and seven standard GP pins. While there are two GND connections, the only power option is 3V3, so this rules out any components requiring 5 V power, such as NeoPixels. An Audio pin is connected to the on-board piezo speaker.

The remaining four breakout pins are allocated to motor connections. Making use of a DRV8833 dual H-bridge motor driver chip, these can deliver 1.5 A RMS current output to control two

▶ Connect a jumper wire from a GP female pin to the Audio pin to send sound to the on-board piezo speaker





> Plug and play with standard electronic components to explore physical computing more easily

DC motors (or other power-hungry devices such as bright LEDs) – there's even a handy overcurrent warning LED next to them.

Mini display

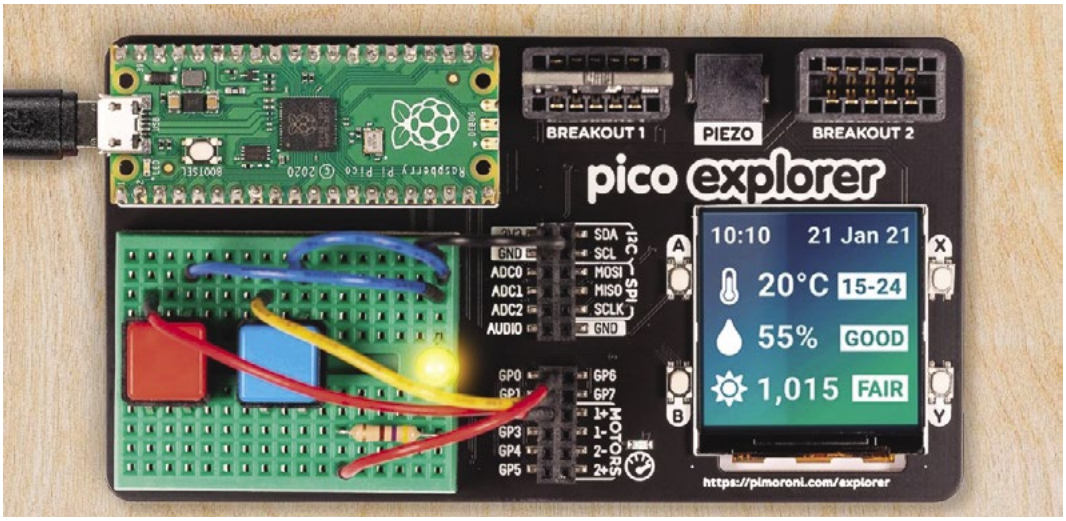
One of the highlights of the Explorer Base is the mini LCD screen on its right-hand side. This 1.54-inch, 240×240 IPS display is vibrant and useful for showing data such as sensor readings, as well graphs, text, and colourful graphics. Like most of Pimoroni's mini displays, it features four tiny tactile buttons around the outside for user input. You could even use them to play simple games.

Above the screen are a couple of five-pin I2C-based breakout slots that are compatible with Pimoroni's large range of Breakout Garden boards. Making use of them isn't quite so simple, however, since the Explorer Base's supporting software libraries – for C and MicroPython – are still a work in progress at the time of writing, and only include drivers for a few breakouts.

Pimoroni tells us more will be made available soon. With a great deal of jiggery-pokery, and the help of a CircuitPython bus conversion library (**magpi.cc/cpbusdevice**) created by Ben Everard from our sister magazine HackSpace – we managed to get a BME680 breakout sensor working in MicroPython. Note that you'll also need to flash Pimoroni's custom UF2 firmware to Pico to use the Explorer Base with MicroPython.

Driving the LCD display and reading its buttons is made fairly simple by a MicroPython module. This enables you to set pixels, create filled rectangles and circles, change pen colours, and display text strings and characters (using a preset upper-case font). With a bit of effort, it's possible to create some more advanced effects, such as lines, hollow shapes, and even scrolling text on a path – as demonstrated by Tony Goodhew in his excellent Instructable (**magpi.cc/explorerworkout**), which shows the power of the display and Pico itself.

▲ Packed with features, the Pico Explorer Base is billed as an 'electronic adventure playground'



◀ Connect components on the breadboard to the clearly labelled female breakout headers. Pico's USB connection also powers the Explorer Base

Verdict

While the software library support and documentation is currently lacking, and may prove befuddling for beginners, the Pico Explorer Base packs in a lot of functionality to explore physical computing with Pico.

8/10

10 Amazing:

Maker tools

Help make your dreams a reality with the right set of tools

Practice and experience is generally more important than the tools you have for the job. However, the right tools can make it easier, whatever your level of experience. A well-stocked workshop can really help bring your projects to life – here are ten tools we think you should have.

▲ Helping hands

Handy holders

Due to various reasons, humans have two hands. Sometimes less. Balancing wires and solder and a (very hot) soldering iron, or holding a small piece steady painting, can be tricky even for the most dextrous. With movable clamps and a magnifying glass, helping hands make it easier for everyone.

▲ 3D printer

Print your dreams

A quite expensive item for sure, but one that has endless possibilities. Thanks to online modelling communities and easy access to basic CAD software, you can quickly prototype and test designs for projects, or create that part you can't get elsewhere.

◀ Wire stripper

No knife required

Snipping wires to length is easy; removing the plastic insulation to reveal the actual conductive wire underneath can be a little tricky. We recommend a good wire stripper, especially when you get past the breadboard prototyping phase.

▶ Glue gun

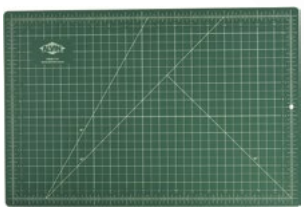
Hot fusion

Easy to apply, quick to set, strong, and insulating? Glue guns are a maker's best friend. Always make sure to get one with at least two settings, though. And don't rely on just one for all your gluing needs. You can even get sealing wax sticks for them, as well.

▶ Cutting mat

Precision slicing

While you can cut on the surface of a wooden workbench in a pinch (and with permission), a cutting mat is definitely the preferred method. It won't dull the blade as much, they self-heal, and they usually come with angles and measuring tools built onto the surface so you can have a better visual guide.



▲ Rotary multi-tool

Spinning trimmer

Otherwise known as a Dremel, which is a brand of tool maker that makes popular rotary tools, they're great for small projects. You can sand, buff, cut, drill, shave, and more by switching out the different tool heads. It's great for makers as they're small and cover a lot of bases.

▸ Directional light

See your project

It's easy to find little USB LED lights on a goose neck these days. They're bright, low energy, can be clamped to a table, and moved around however you wish. For finer work, good illumination is key, and they're a bit more flexible than a head torch.

▼ Needle-nose pliers

Grab small things

You never realise just how much you need a slim pair of pliers until you really need them and all you have are tweezers or heavy-duty pliers bigger than a Raspberry Pi. Of course, a good set of tweezers and regular-sized pliers are great to have as well.

▲ Soldering iron

Create circuits

An essential tool for electronics, a soldering iron is basically a very hot metal pen that allows you to melt specially treated, soft metal that allows for wires and circuits to be connected and have electricity run through them. Make sure to tin the tip before using, though: **magpi.cc/tin**.

▲ Clamps/vice

Steady on

Vital for a lot of wood- and metalwork – even some painting! A clamp can keep your projects sturdy so that drilling or cutting won't go awry, or make sure glue will dry firmly, or even make sure metal is aligned for welding. Got a miniature with a base? You can lightly clamp it in one to keep it steady too.

Safety gear!

Every good maker knows they need safety equipment. Dust masks, goggles, welding mask, sturdy gloves, and even a good set of coveralls are a good idea as well. Before you do any making, make sure you're doing it as safely as possible.

Learn ARM assembly with Raspberry Pi

Get down to the metal by learning ARM assembly language. By **Lucy Hattersley**

Introduction to ARM assembly basics

AUTHOR
Maria Markstedter, Azeria Labs

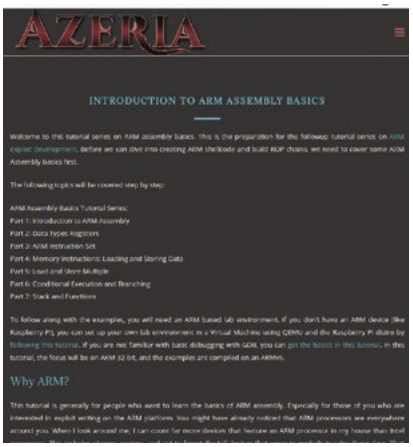
Price: Free

magpi.cc/azeriaassembly

Assembly is a low-level language sitting just next to machine code. It's also known as 'symbolic machine code' because assembly statements are typically translated directly into their machine code counterparts.

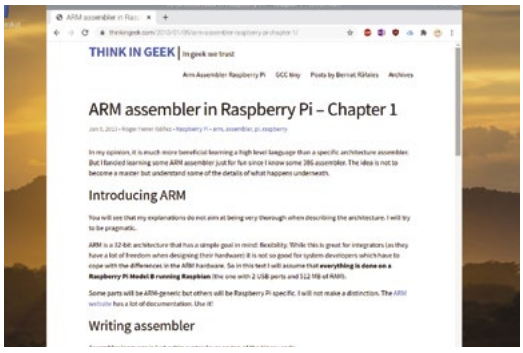
Assembly is more readable than entering pure hexadecimal or binary code, and you get the benefit of comments and symbolic jump locations. Even so, it's not for the faint of heart. Assembly is as close to bare-metal programming as you are likely to get. It's cryptic in nature and you're unlikely to ever create an app, web service, or even useful program in assembly. Instead, this is all about the purity of learning.

Azeria Labs' website has a starter course in ARM assembly as a preparation for a much more detailed course on exploits. Created by Maria Markstedter (**@Fox0x01**), the course is split into seven parts, and covers the ARM-based assembly environment, data types, registers, the instruction set, load and store, conditional executions, stack, and functions.



Hello Assembler

These tutorials will help you get started



ARM ASSEMBLY HELLO WORLD

The classic doing introduction is covered in this online tutorial. Discover how to write and compile 'Hello World' in ARM assembly.
magpi.cc/helloarm

ARM ASSEMBLER IN RASPBERRY PI

Think in Geek's tutorial is worth bookmarking, but Chapter 1 takes you through writing and compiling binary code on Raspberry Pi and creating a 'makefile'.
magpi.cc/thinkingeek

PROGRAM IN ARM6 ASSEMBLY LANGUAGE ON A RASPBERRY PI

This great tutorial by Marty Kalin features a walkthrough of the hailstone function, first in C and then in ARM assembly.
magpi.cc/programarm6

ARM Assembly Language Fundamentals

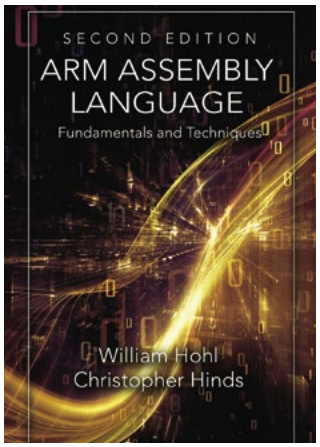
AUTHOR

William Hohl, Christopher Hinds

Price:
£85 / $110

magpi.cc/assemblyfundamentals

Hohl and Hinds's book is a widely regarded classic on the subject of ARM assembly language. It's also a good introduction to computing in a broader context, starting with an overview of computer architecture and history.



This is a complete introduction to assembly language over 18 chapters. There are detailed introductions to the instruction sets, assembler rules, loads, constants, logic and arithmetic. It works into more detailed areas, such as floating point instructions, exception handling, and memory mapped peripherals. The final chapter shows you how to mix C and assembly together. The sticking point is the retail price of the book: quality costs. However, you can pick up a digital edition for less, and we have found it on sale at various outlets. Or, you can pick up a second-hand copy for roughly half the price.

Introduction to Computer Organization

AUTHOR

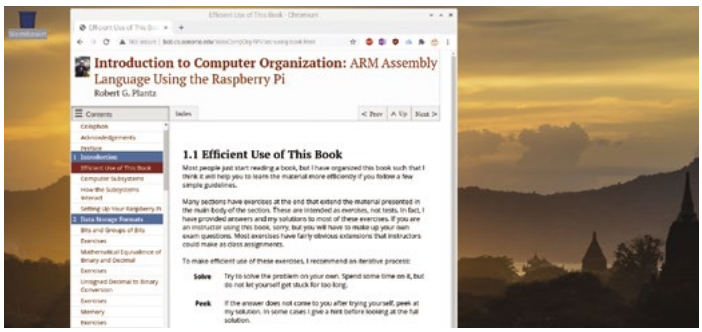
Robert G Plantz

Price:
Free

magpi.cc/introcomporg

Robert G Plantz's book, available for free on the Sonoma State University website, is a solid overview of ARM assembly language, interspersed with exercises to flex your mental muscles. Each exercise has hints and full solutions to help you keep moving along.

The course covers data, arithmetic, algebra, logic gates, circuits, and a crash course in CPU architecture before you even start on ARM assembly. At which point you get a thorough grounding in compiling assembly language and in an in-depth analysis of every feature. This is one of the most detailed courses around, and the bonus that it is written directly for Raspberry Pi ensures you are likely to stay on point. And getting it all for free is remarkable.



Also of interest...

Bookmark these resources

ARM DEVELOPER

ARM has a wealth of documentation available, including programmers' guides and reference documentation. It can be a little detailed for the beginner, but bookmark it for when you start asking questions.
developer.arm.com

RASPBERRY PI SOC DATASHEETS

Raspberry Pi's system-on-chip datasheets are a good way to understand how Raspberry Pi hardware implements with ARM SoC.
magpi.cc/socdatasheets

ARM REFERENCE CARD

This ARMv7 cheat sheet from the ARM documentation website is a great PDF to download and print out..
magpi.cc/armcheatsheet

Kevin Johnson

A man of many talents, Kevin helps to organise official Raspberry Pi events and produce free resources

- Name Kevin Johnson | Occupation Coding club liaison
- Community role Event runner | URL @KJberrypi

If you're part of the community for Raspberry Pi in North America, and have been to one of the many events the Raspberry Pi Foundation North America (RPFNA) has organised, you've likely got Kevin Johnson to thank. If you've ever met him, you'll know he loves doing it, and we love him for it.

"So my official title is Club Programs Coordinator," Kevin tells us. "What that means is I help develop engagement strategies and also implement them to keep our youth programs community (Code Club, CoderDojo, Coolest Projects) in the USA fully immersed in all of the free resources we have. It involves a lot of writing, which I love because my background is in creative writing, so I write monthly newsletters and blogs, I design seasonal competitions for young people to participate in, I try to maintain our presence on social media, and a bunch of other things. I'm truly a wearer of many hats!"

▼ Kevin's job is to make sure this mission statement is made reality

What did you know about Raspberry Pi before joining?
Full transparency, I did not know much about Raspberry Pi prior to joining the Foundation because my introduction to tech was mainly focused in Adobe Creative Suite programs like Photoshop, Illustrator, Premier Pro, etc. I was in a multimedia academy (CMMA) in high school, so I dabbled a little bit with coding, but not enough to know about the different hardware out there. My knowledge of Raspberry Pi came when I discovered the job listing and after reading through the Code Club and CoderDojo blogs, I was immediately hooked. To see such a small device have such a large impact on people all over the world, especially people from different socioeconomic backgrounds, I had to have the job. Fast-forward two years and here I am, still loving every minute!

Kevin helps Code Clubs and CoderDojos around the country to make sure they're up-to-date with all the latest projects

Coolest Projects USA is a huge event which Kevin not only co-hosted, but he was also one of the organisers

What are some of your favourite moments with the Raspberry Pi community?
I had a blast co-hosting last year's Coolest Projects USA event that we held in Santa Ana, CA at the Discovery Cube Orange County! I love the energy that young people bring to any space and in that space the energy was at an all-time high. Also,

> I had a blast co-hosting last year's Coolest Projects USA

who hasn't dreamed of being a talk show host?! If you can get on a stage and keep a group of young people entertained and laughing, it's a great sign that you're doing alright in life. I have to also mention the few trips our team here in the USA have taken to Cambridge to visit the Foundation headquarters; those have been some of the greatest moments of my life! Being part of a global community, learning about other cultures, connecting with colleagues on a more personal level is just amazing. It means the world to me to work for a global organisation that finds importance in bringing its employees together not just to work, but to also play and have fun.

What are some of your favourite moments with the Raspberry Pi community?
We're really excited about this year's Coolest Projects event, especially because this will be a global event where we will hold one event for all of our participating regions. Here in the USA, we were lucky to be able to have the event in-person right before the lockdowns started last year, so while we did get to experience that, we did not get to experience the event online like everyone else. Collaborating with the global team on the upcoming Coolest Projects Online has birthed some really awesome ideas that we can't wait to share with everyone come early June when this year's showcase happens!

You'll see Kevin helping out at just about every event in the US, and some elsewhere in the world!

Share your stories

"As a storyteller, I love telling stories of my own, but I also love sharing the stories of others, so please connect with me on Twitter at **@KJberryPi** or through email at **kevin.johnson@raspberrypi.org** if you have a story that you'd like to share! Community is everything to me, so it brings me so much joy to share the stories of our global community!"

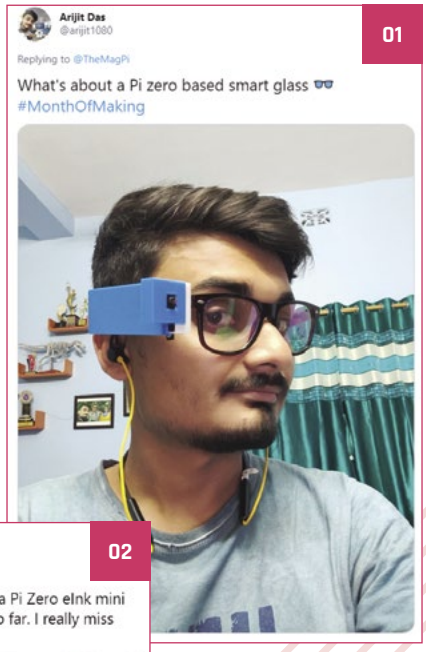
This Month in Raspberry Pi

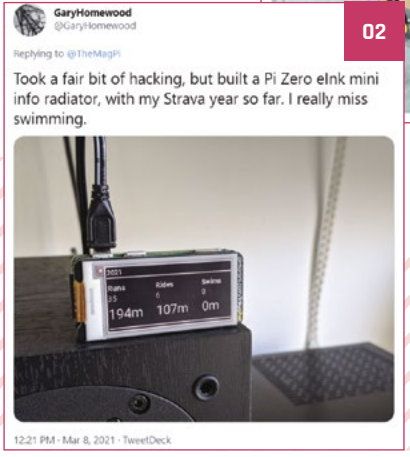
#MonthOfMaking on Mondays

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, especially during #MonthOfMaking.

Here's a selection of some of the awesome things we got sent this month – and remember to follow along at the hashtag #MagPiMonday!





01. Smart glasses are very cool, and we're looking forward to seeing what Arjit does
02. A nice little fitness tracker
03. A skill that many of us on the mag would like to learn
04. Load up *Xanadu* on the fake vinyl
05. #MonthOfMaking is for more than just Raspberry Pi stuff, as Alex shows here!
06. We're not 100% sure this watering can would work well with water
07. Ambient info/art is a cool idea. Ours would frown depending how close to deadline we are!
08. We like seeing Jitesh's robot develop further!
09. Easy, yet a cool thing to learn thanks to a bit of clever Python code
10. A very fast robot indeed, and custom as well!

03

Jonny
@mcjonnyw

Replying to @TheMagPi

I made this, now I just need to learn how to play piano 😅
#MonthOfMaking

> **Jonny** @mcjonnyw · Mar 10
> Welcome to my musical piano stool. Korg Monotron using #RaspberryPi and MCP4725 DAC as MIDI to CV converter.
> Followed this guide by @yakczar schollz.com/raspberrypi/mo...

04

Mark Cantrill
@AstroDesignsLtd

Replying to @TheMagPi

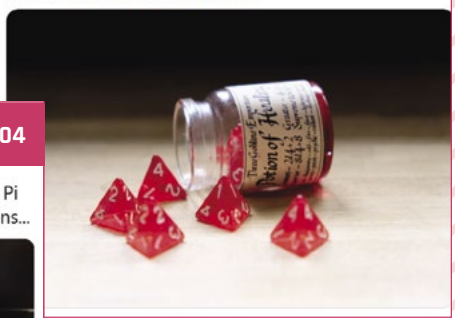
Another step closer to completion of the Raspberry Pi Jukebox, this time with rattles, clicks, whirrs and toons...

5:43 PM · Mar 1, 2021 · Twitter Web App

05

alex j'rassic
@alexjrassic

For #MonthOfMaking I decided to start a whole side business making and selling #DungeonsandDragons gifts on @Etsy. I've had some great sales and feedback for these Potions of Healing and can't wait to design more! etsy.com/uk/listing/974...



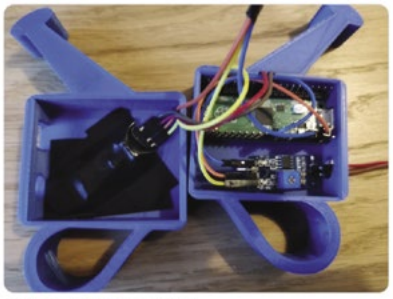
06

S Organ
@makercupboard

Replying to @TheMagPi

I haven't planned my March make yet but .....

At the weekend I completed my @Raspberry_Pi Pico watering can.
themakercupboard.space/index.php/2021...



1:23 PM · Mar 1, 2021 · Twitter for Android

07

Mike Fell
@mikefsway

Replying to @TheMagPi

Inspired by @openbookuk's eink electricity display from a week ago

> **Mike Fell** @mikefsway · Mar 1
> Experimenting with ambient info/art this weekend. This portrait stands in my kitchen and smiles/frowns depending on current level of renewable generation (via @ng_eso) and @octopus_energy Agile price. Uses a @Raspberry_Pi Zero and low-powered @pimoroni eink display.
> Show this thread

08

Jitesh Saini
@jiteshsaini10

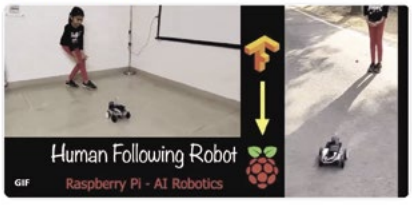
Replying to @TheMagPi

Made a human following robot using Raspberry Pi, PiCamera, a Pre-trained Object Detection #MachineLearning model, #Tensorflow Lite and Coral USB Accelerator

Read the article:-
helloworld.co.in/article/ai-rob...

Video:-
youtu.be/GSYtn8BlGBI

#monthofmaking



09

Robert Kelso
@cumbiebob

Probably the easiest make ever. PiZero + old webcam Motioneyeos software, Pushover App and a python script from MagPpi magazine and hey presto a cat detection system up and running #monthofmaking

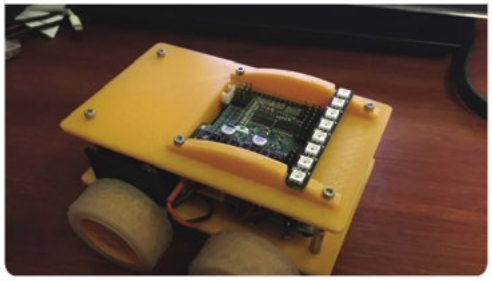
10

Dr Footleg - Roboteer
@drfootleg

Replying to @HackSpaceMag

I made a new Raspberry Pi based robot!
#MonthOfMaking

> **Dr Footleg - Roboteer** @drfootleg · Mar 1
> It is finished! Showcased on the @PrestonRJam tonight. My Turbo 4WD. It's really fast, and powered by my own designed electronics.
> Show this thread



Coolest Projects online!

Submit your projects for the next Coolest Projects online showcase before Monday 3 May!

Last year, Coolest Projects went online like a lot of other events. It went so well it's happening again this year, and you can take part! No matter where you live, as long as you're 18 or under, you can apply! Here's what CoderDojo has to say about it…

"We welcome all projects, all experience levels, and all kinds of projects, from the very first Scratch animation to a robot with machine learning capacity! The beauty of Coolest Projects is in the diversity of what the young tech creators make.

"Young people can register projects in six categories: Hardware, Scratch, Mobile Apps, Websites, Games, and Advanced Programming. Projects need to be fully registered by Monday 3 May 2021, but they don't need to be finished then — at Coolest Projects we celebrate works in progress just as much as finished creations!"

Thinking about entering but not sure how? There's a fantastic video (**magpi.cc/cpregistervid**) and even a worksheet (**magpi.cc/cpworksheet**) that will get you ready for it.

Once you've figured out a project, head over to **magpi.cc/coolestprojects21** to register. Good luck!

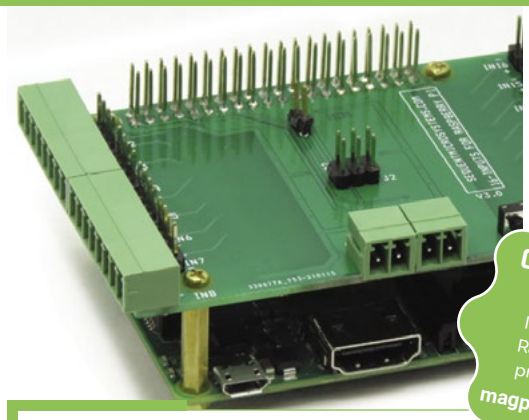
Crowdfund **this!**

Raspberry Pi projects you can crowdfund this month

CarPlay and Android Auto on Raspberry Pi

If you have a car that hooks up to your phone for better navigation and a car-friendly UI, you'll know how good it can be. This Kickstarter is to help develop further some software called OpenAuto Pro so that it can plug into the Apple CarPlay and Android Auto API so you use it on Raspberry Pi. A very interesting prospect.

▶ **kck.st/3k6KDVa**



128 Opto-isolated Inputs plus RS485

If you need to read a *lot* of digital inputs on a Raspberry Pi, and the GPIO pins just aren't enough, this big HAT-style add-on might be the solution. It has already hit its funding goal, so is definitely worth a look.

▶ **kck.st/3aA2T6j**

CROWDFUNDING A PROJECT?

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

magpi@raspberrypi.com

Best of the rest!

Other amazing things from the community

AUTOMATIC WATER DISPENSER

We've seen computer vision used before to grade cucumbers, but to make sure a water bottle is safely filled with the right amount of water is excellent. We like the little light display that speeds up as the bottle is nearly full.

▶ **magpi.cc/awdispenser**

RASPBERRY PI 3D SCAN

A 3D model of a Raspberry Pi that was created by a 3D scanner powered by a Raspberry Pi. This is an amazing result, and we're glad to see 3D scanners have improved so much over the last few years.

▶ **magpi.cc/diy3dscan**

ANALOGUE BANDWIDTH GAUGE

An old DC voltmeter has been upcycled with a Raspberry Pi Zero W to show the bandwidth in user jllauser's home network. It's very fancy.

▶ **magpi.cc/bwgauge**

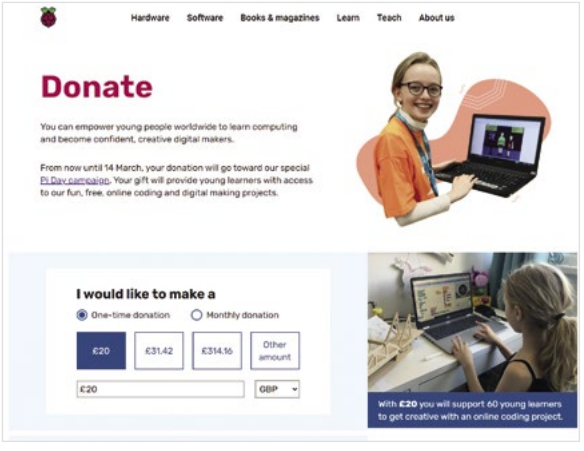
Your Letters

Pi Day

I donated to the Raspberry Pi Foundation for Pi Day and I learned later that my name might end up in the magazine. Which issue should I be looking out for? I've read your magazine for years and would love seeing my name in it!

Sylvia via Facebook

Due to the timing of how *The MagPi* is produced, unfortunately we couldn't make it happen in this issue. However, we'll be making sure it gets printed in issue 105 – check out page 97 to see when it comes out.



▲ The Raspberry Pi Foundation is a non-profit charity, so every little helps

▼ Rafa used an old notebook as a screen for a desktop Raspberry Pi setup

#MonthOfMaking email

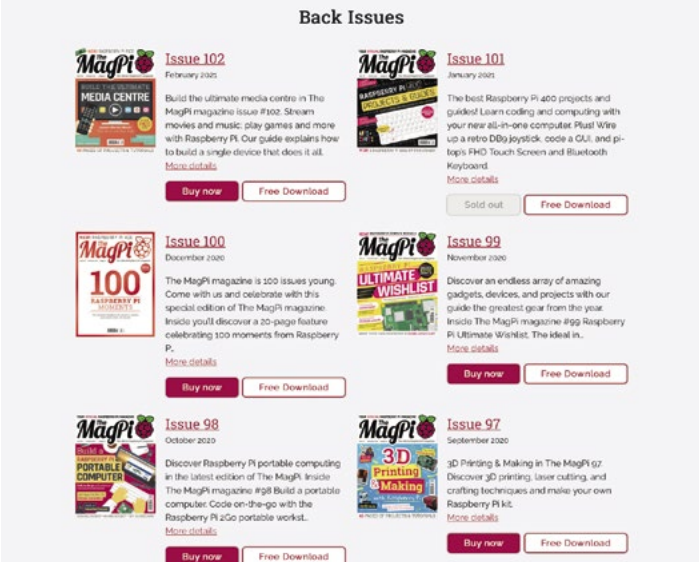
I emailed you details and photos of my #MonthOfMaking project, but just read in the magazine that they should be posted on Instagram and Twitter. I don't have either of those. Will my project not be counted?

Rafa via email

It still counts! You still made something, and we will put it in the magazine when we do a full #MonthOfMaking roundup next issue. We think it's great to post it on social media so other folks might be inspired to make something.

If you have a project you'd like to share with us, you can always email us any time of year.

▼ You can get PDFs of all our issues from **magpi.cc/issues**



Out of stock books

I was looking over the list of books you've released and noticed that some that I wanted were out of stock. When will they return to the store?

Ibs via email

Some books take a little longer to restock than others, although some may not see a print run again. However, all the books we've released are available as a free PDF which you can print (or get printed) yourself if you need a physical copy.

There are probably some knocking around on eBay as well.

GitHub repos

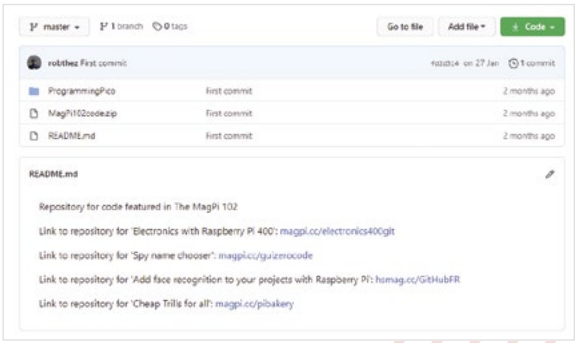
I was looking at your GitHub and noticed not all the code is available in repositories, some of it is in a zip file. Why not just have the code in the issue repo?

Daniel via GitHub

There are a couple of different reasons for why this is: when we started putting code on GitHub, it only included the code that would only live on that repository. All the repos are linked from *The MagPi* article they appear in, so Mike Cook's work appears on his Pi Bakery GitHub.

Someone asked us if we could provide all of this code for the mag as a zip as well, so we started doing that and uploading it with any other code that needed a repo.

We have now started to provide a link to the other repositories in each issue's repo now. As we figured people would be going from the article to the repo, it didn't occur to us to put a link there. Now they are!



▲ Need a hub for all the code in an issue? Our GitHub will have it

Contact us!

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

HackSpace

TECHNOLOGY IN YOUR HANDS

THE MAGAZINE FOR THE MODERN MAKER

SUBSCRIBE AND SAVE UP TO 35% on the cover price

ISSUE **#41**

OUT NOW

hsmag.cc

Available on the App Store

GET IT ON Google Play

WIN ONE OF SIX SMARTIPI TOUCH PRO CASES!

IN ASSOCIATION WITH SmartiPi

We reviewed the SmartiPi Touch Pro in the last issue of *The MagPi* and gave it a phenomenal 9/10. If you're still not sure if you want one, maybe we can tempt you with a competition to win your very own?

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

Terms & Conditions

Competition opens on **24 March 2021** and closes on **29 April 2021**. 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.

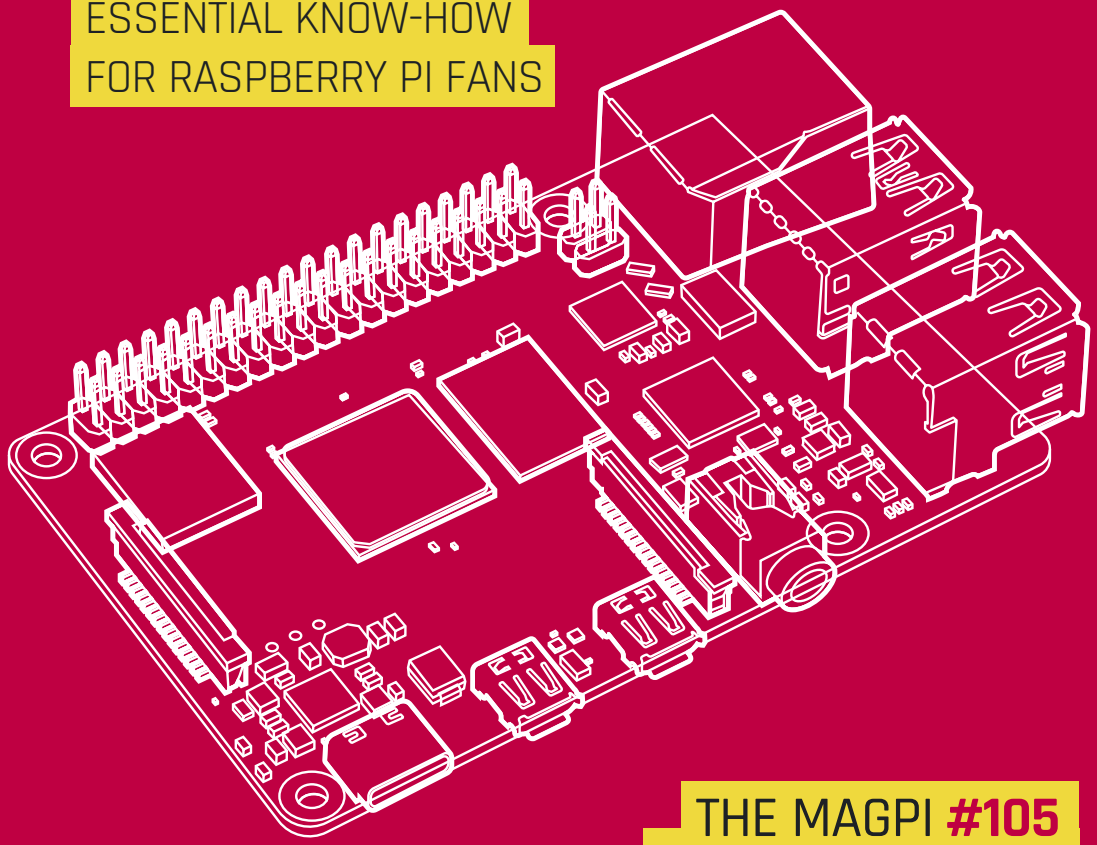
Wireframe

Join us as we lift the lid on video games

Visit **wfmag.cc** to learn more

RASPBERRY PI TIPS & TRICKS

ESSENTIAL KNOW-HOW FOR RASPBERRY PI FANS



THE MAGPI #105 ON SALE 29 APRIL

Plus!

Star Wars arcade cabinet

Portable Raspberry Pi projects

Grove Starter Kit for Pico

DON'T MISS OUT!

magpi.cc/subscribe

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

EDITORIAL

Editor
Lucy Hattersley
lucy@raspberrypi.com

Features Editor
Rob Zwetsloot
rob@raspberrypi.com

Sub Editors
Phil King and Nicola King

ADVERTISING

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

DESIGN

criticalmedia.co.uk

Head of Design
Lee Allen

Designers
Lucy Cowan, Sam Ribbits

Illustrator
Sam Alder

CONTRIBUTORS

Mike Cook, David Crookes, PJ Evans, Gareth Halfacree, Martin O'Hanlon, Rosemary Hattersley, Nicola King, KG Orphanides, Nik Rawlinson, Laura Sach, Mark Vanstone

PUBLISHING

Publishing Director
Russell Barnes
russell@raspberrypi.com

Director of Communications
Liz Upton

CEO
Eben Upton

DISTRIBUTION

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

SUBSCRIPTIONS

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

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

The MagPi magazine is published by Raspberry Pi (Trading) Ltd., Maurice Wilkes Building, St. John's Innovation Park, Cowley Road, Cambridge, CB4 0DS. The publisher, editor, and contributors accept no responsibility in respect of any omissions or errors relating to goods, products, or services referred to or advertised in the magazine. Except where otherwise noted, content in this magazine is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported (CC BY-NC-SA 3.0).
ISSN: 2051-9982.

Running up that hill

How patience can be its own reward. By **Lucy Hattersley**

Computers are truly wonderful inventions. The miracle of the modern age, as some would say. But it's fair to say they can also be opaque. While it's easy to look at a clock or a steam train and see the moving parts, the 'micro' part of a microcomputer means you have to use a very powerful microscope – or, more likely, your imagination – to see the moving parts.

There are billions of bits shifting thousands of times per second. The act of teaching a rock to think is rather overwhelming for our small simian brains.

Learning computer architecture is a task worthy of Sisyphus, only you are carrying a rock that frequently crashes and stalls as you carry it up to the mountain of understanding. Advances in artificial intelligence ensure that it'll be soon giving you lip on the way up.

Carry on coding

Let's look on the bright side. One way in which the last few months have been positive for me is regarding concentration. With everything closed, I've been able to sit down and focus on a few things.

Learning about low-level computing, right down at the mechanical level is one of them. And it has been tough going. It requires a clear head and a steady concentration. I've also learned a new level of meaning to the word 'perseverance'.

This isn't the first time I've tried to understand low-level computing. I first got the coding bug when I was very young: introduced to a ZX-81 in pre-school and then working my way through the BBC Micro at school and a ZX Spectrum (then Commodore 64) at home.

> "Learning computer architecture is a task worthy of Sisyphus, only you are carrying a rock that frequently crashes"

For all the coding tricks I've learned over the years, it's always saddened me that I didn't quite get the hang of Z80 machine code when I was young. For some reason, I just couldn't get any of my Z80 code to work and eventually I moved on.

Lately, I've been working my way through the Nand2Tetris course (**nand2tetris.org**), partly as an attempt to get back to those binary and machine code roots so that I can at least get my head around Z80 coding as a present to my younger self. Also, to get a better understanding of ARM architecture so I know what's going on when I switch on my Raspberry Pi. (Always handy when you do that for a living.)

I've been using a Raspberry Pi 400 to work through the course materials, and the experience is surprisingly similar to the ZX Spectrum of my youth – although the keyboard is infinitely better, as is the operating system. I've also got a new Raspberry Pi Pico hooked up to the back, just like an expansion card.

Hopefully, I'll carry this Nand2Tetris rock to the top of the hill. And I suspect I will roll it down at the end and reveal something new to learn. There's always a shiny new technological concept waiting to be discovered. Onwards and upwards!

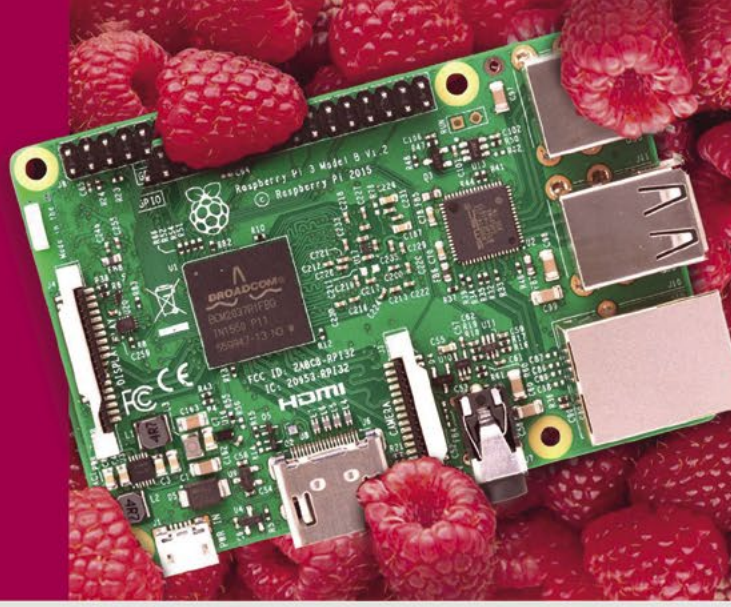
AUTHOR

Lucy Hattersley

Lucy is editor of *The MagPi* and recently learned how to burst a water pipe with a drill. Look carefully, kids. Look, then drill. In that order. Every day is a school day!

magpi.cc

American **Raspberry Pi** Shop



- Displays
- HATs
- Sensors
- Cases
- Arcade
- Swag
- Project Kits
- Cameras
- Power Options
- Add-on Boards
- Cables and Connectors
- GPIO and Prototyping

Partner and official reseller for top Pi brands:

adafruit · sparkfun ELECTRONICS · PIMORONI · HiFiBerry · and many others!

Price, service, design, and logistics support for **VOLUME PROJECTS**

USA PiShop.us

Canada BuyaPi.ca

Raspberry Pi APPROVED RESELLER

pi-top [4]

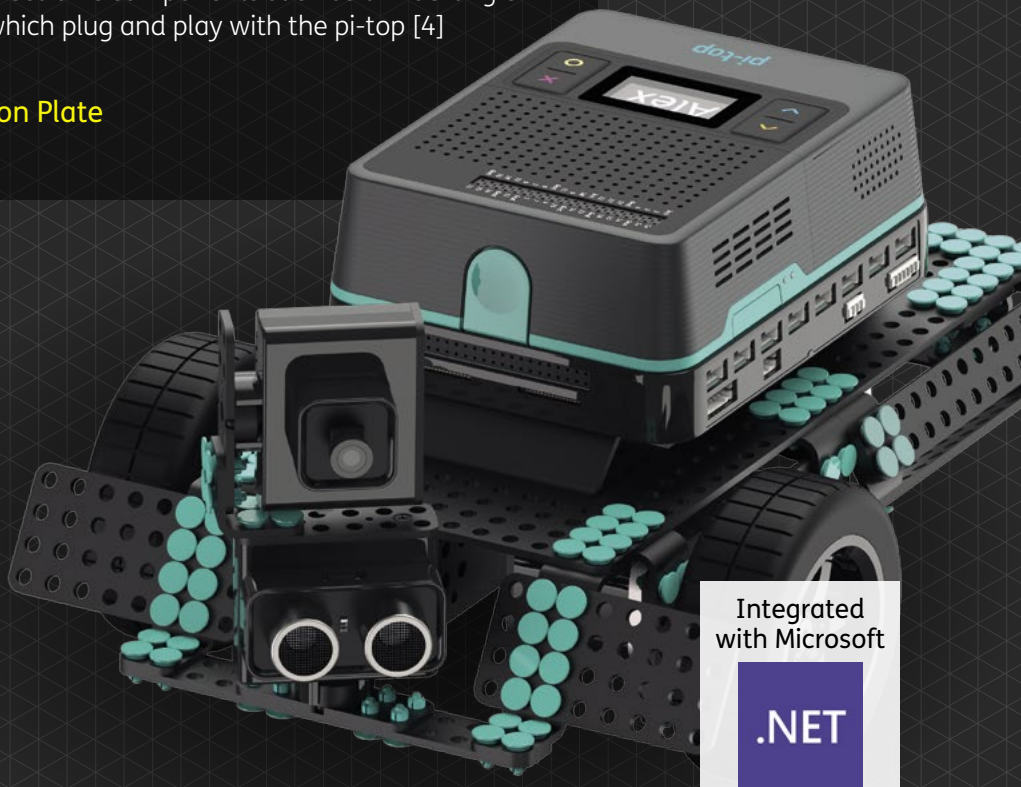
ROBOTICS KIT

Robotics & rapid prototyping with your Raspberry Pi

Power your projects with computer vision and applied AI

pi-top [4] Robotics Kit comes with electronic components such as a wide-angle camera, servos and motors, all of which plug and play with the pi-top [4] Complete or pi-top [4] DIY Edition†.

pi-top Robotics Kit with Expansion Plate
187.90 / $199.90



- Gesture Control
- Obstacle Avoidance
- Autonomous Driving
- Object Recognition
- Emotion Mapping
- Line Recognition
- Interaction
- Face Tracking

Integrated with Microsoft .NET

pi-top.com/MagPi

Raspberry Pi is a trademark of the Raspberry Pi Foundation. †pi-top [4] and Robotics Kit with Expansion Plate sold separately.

pi-top

Raspberry Pi made simple, robust and modular.