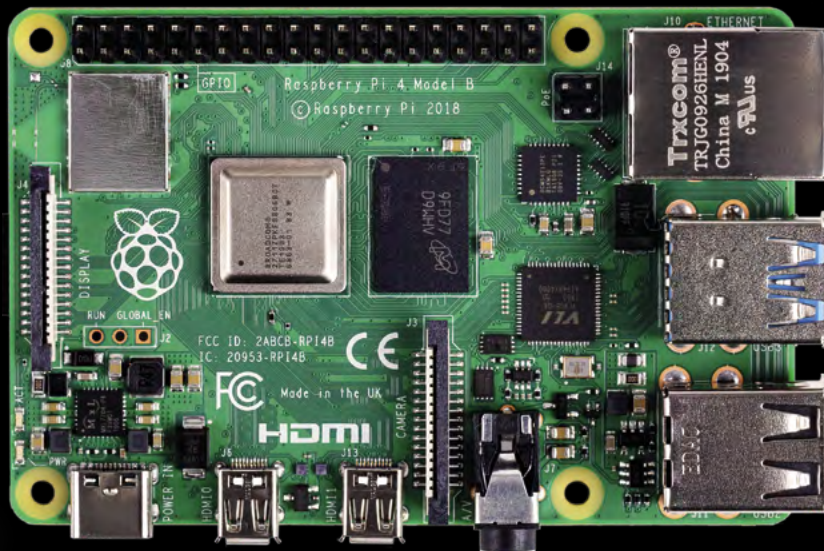# The MagPi

Issue 108 | August 2021 | magpi.cc

The official Raspberry Pi magazine

AWARD WINNING MAKES

Build the ultimate **home server**

Develop games with **Scratch & Python**

Inside the **Smart Campervan**

Pi-top robot **tested!**

WIN! MARTY THE ROBOT V2 UP FOR GRABS!

WELCOME
to *The MagPi* 108

Everybody likes to get a gold star. I remember hosting awards and no matter how cool they try to play it, getting a shiny gong always puts a smile on somebody's face.

And sometimes it feels right to say 'well done!' That's why this month we've gathered together the best Award-Winning Makes (page 30). These are the projects that have won shiny statues, gathered plaudits, and impressed some judges.

These builds are worth your time. They deliver ideas for your next make and stand as glittering examples of Raspberry Pi making the world a better place.

If you want something more down-to-earth and buildable 'right now', then I suggest PJ's new 'Ultimate Home Server' tutorial (page 38). This is the first in a new series of using Raspberry Pi in the home to manage all of the digital files in your life.

And if you've ever fancied yourself making visual entertainment, then check Make Games with Raspberry Pi (page 71). In this article, Mark Vanstone reveals Raspberry Pi software designed to help your digital dreams become reality.

There is a lot to discover in this month's edition of *The MagPi*. Dive in and discover the idea for that project that will one day, perhaps, win you an award.

Lucy Hattersley Editor

EDITOR

Lucy Hattersley

Lucy is editor of *The MagPi* and is back on the train to Cambridge with a smile behind the mask.

@LucyHattersley

GET A
RASPBERRY PI ZERO W KIT
WITH A SUBSCRIPTION!
PAGE 28

Contents

> Issue 108 > August 2021

30

26

08
Callisto II

Kenbak-2/5

The Big Feature

71

Make games with Raspberry Pi

38

Ultimate home server - part 1

66

Isomorphic keyboard

80

pi-top Robotics Kit

86
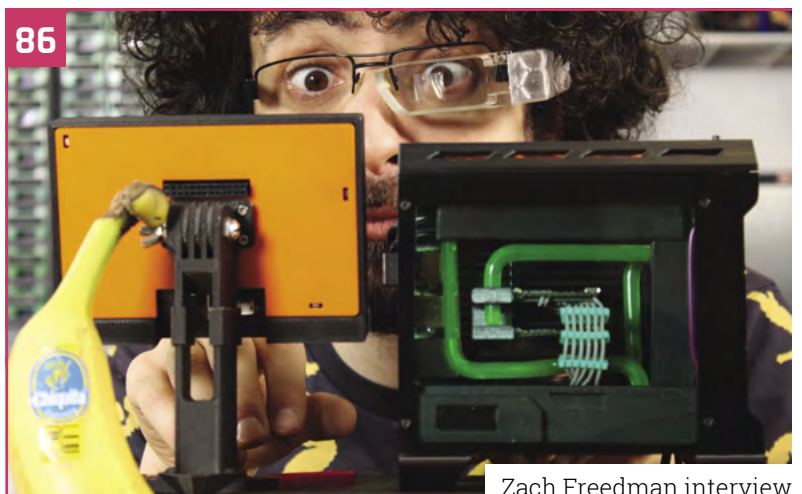
Zach Freedman interview

WIN A MARTY THE **ROBOT V2** **95**

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**

pi-top

Raspberry Pi made simple, robust and modular.

Callisto II

A retro gaming fan gave his Raspberry Pi a 1980s makeover. **Rosie Hattersley** is on the case

MAKER

Kevin Solar

Game developer and software engineer Kevin loves designing physical builds and often uses the powerful capabilities of Raspberry Pi in his projects.

magpi.cc/ solarcomputers

I n his spare time, games developer and maker Kevin loves to tinker with all things retro: "I like to design new NES games using 6502 Assembly" he tells us, deftly setting the scene for how his Callisto II retro computer design came about. A hardware-based project, Callisto marries Kevin's enjoyment of 3D design and printing and his abiding love of retro gaming.

"It started in 2019 when I really wanted to 3D-print a full-sized retro or a terminal style of computer," says Kevin. "When I was looking all over the web, I was surprised that this sort of thing didn't exist. I saw lots of mini 3D printable retro computers and they were really good, but I wanted a full-size one that I could use for everyday tasks. Since this didn't exist, I had to make one."

Raspberry Pi was the obvious choice for the hardware to power his dream of recreating the look and feel of a 1980s computer. "I needed a desktop OS to make my retro computer very functional," he explains, "[and] Raspberry Pi is an inexpensive and very capable computer."

Kevin has previously designed two retro computer cases. The first, Callisto J-29, "was very rough around the edges" while the second one, Europa, "was too heavily inspired by the original Macintosh." For Callisto II he wanted to top these first designs and "make something that was very easy to print and assemble, but still looked great." He wasn't prepared to compromise on computing power either - hence his choice of Raspberry Pi.

Tricky curves

Despite his experience with 3D printing to date, Kevin says the trickiest part of the design was modelling the curves, as he'd not done this before. He persevered, knowing the curves would set his design apart from others. It was also a challenge to find a true 4:3 LCD screen that was inexpensive and readily available, he says, but rarer still to find a retro computer that used the 16:9 aspect ratio of modern displays. He was eventually able to source an 8-inch Pimoroni display to give Callisto that all-important 1980s look.

The curved edges of Callisto II mark an evolution in Kevin's retro computer designs – all available to download and 3D-print from Thingiverse (**magpi.cc/callisto2**)

Few 4:3 screens are available (most are 16:9), but this Pimoroni one provides a suitably retro look

The mechanical keyboard is 60% the size of a standard one and has pleasingly tactile keys

Quick **FACTS**

> Kevin set up a retro website to accompany Callisto's launch

> He also wrote a retro-style user manual

> And an online OS (**magpi.cc/pigeonos**)

> He's particularly proud of Callisto's 3D-printed hatch

> "I love the way it integrates with the case," he beams

▲ Kevin has even designed a retro online operating system for Callisto

▲ Callisto II is the third 3D-printed retro computer Kevin has designed

Callisto's design is inspired by several real retro computers such as the ADM-3A, Zenith Z-89, and Hazeltine 1500, but with no 3D-printable, full-size retro computers available online for comparison, Kevin was on his own when it came to working out the dimensions and 3D design.

Easy ethos

Kevin tried to use readily available parts from online retailers for most of the project. "Not only did I want this to be super-easy to print and put together, I wanted it to be easy to find the parts [and be] something you could put together for an easy weekend project (assuming you spent the previous week printing all the parts)," he says. You don't even need glue as all the parts have been designed to snap together, but you should ensure all the electronics have room to breathe, he cautions.

▲ A trio of Callisto II computers
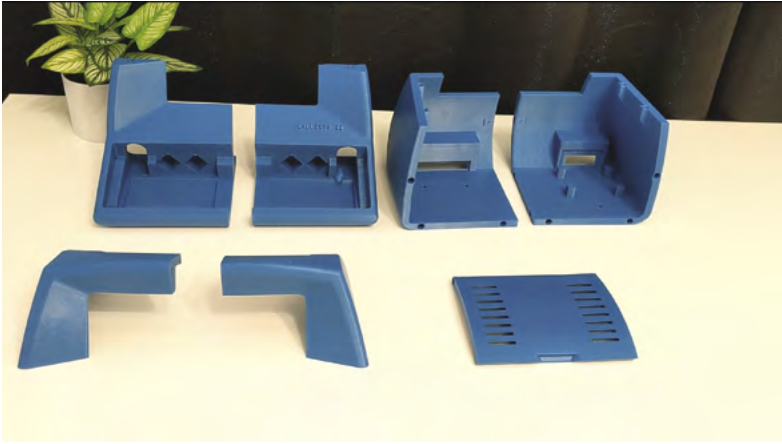
▲ 3D-printed parts ready for assembly

🟦 Something you could put together for an easy weekend project 🟦

The project cost roughly $250 and involved printing six parts on a Prusa Mini 3D printer that each took a day to print. When sourcing a suitably tactile 60% mechanical keyboard, Kevin suggests choosing one that has blue switches. "These give the loudest clicks," he says.

Kevin stuck with Raspberry Pi OS, but part of Raspberry Pi's appeal is that you can load games and emulators to make it look and run however you want, he says. RetroPie is an obvious choice here. For more ideas on mimicking Callisto's retro looks with retro programs, take a look at Retro Computing in *The MagPi* issue 88 (**magpi.cc/88**). 🅜

▼ As Callisto II is 3D-printed, you can have it any colour

Build a retro computer

Download and 3D-print the parts for your Callisto II from **magpi.cc/callisto2**. The case is designed to snap together with pins; glue is optional.

01 Assemble the case and install the hardware. The screen should easily slide into place using the case's built-in slots.

02 Slot the keyboard in place, followed by the USB hub and power supply. Secure them using 3D-printed pins.

03 Insert and plug in Raspberry Pi near the PSU and, optionally, secure it using Velcro, then power up your retro computer.
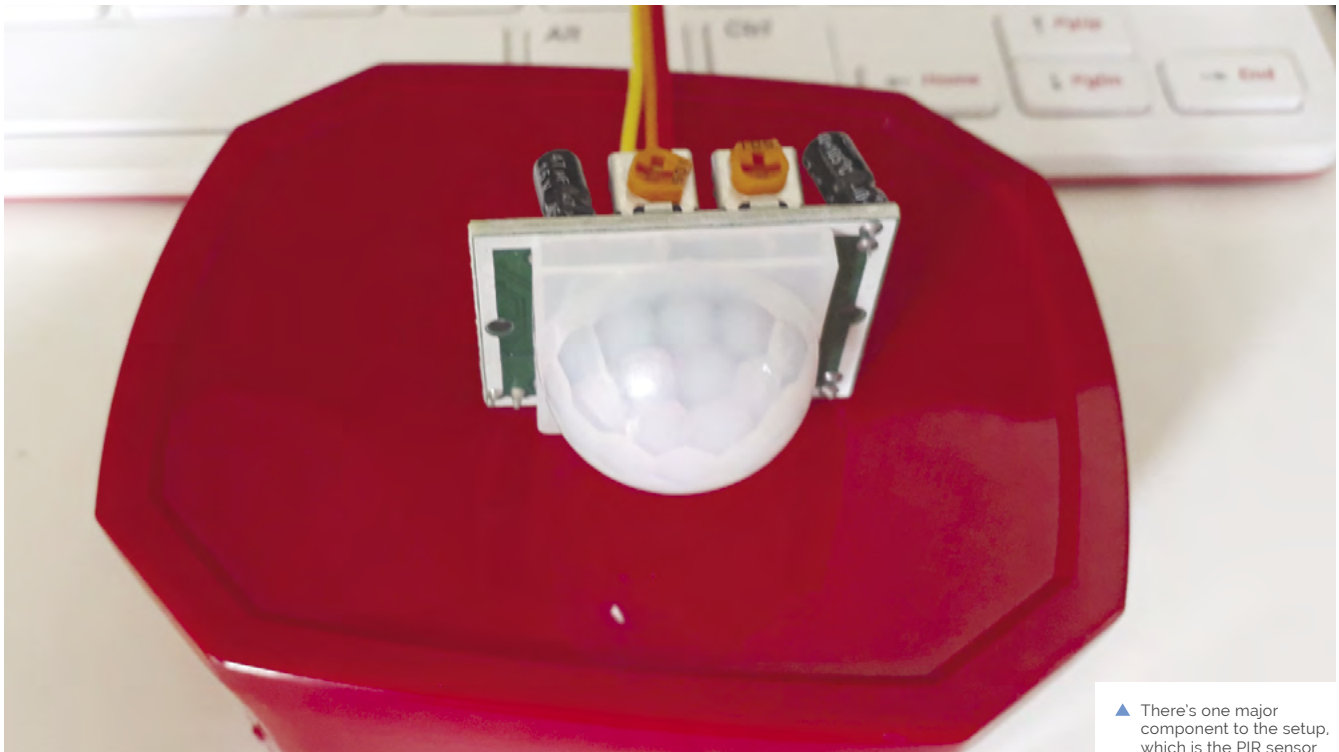
▲ There's one major component to the setup, which is the PIR sensor

Humane mousetrap

Safely catching mice is a better way of fixing a problem, and using Raspberry Pi means it needs less supervision. **Rob Zwetsloot** takes a look

MAKER

Andrew Taylor

A web developer who inherited a Raspberry Pi, allowing him to reconnect with the basics of computing that got him interested in the first place.

ataylor.net

With some IoT projects, it's the little things that help. For example, take Andrew Taylor, who did the good thing of setting up a humane mousetrap. However, checking it to see if any mice had been caught in it, while necessary, was getting a little boring.

"If a mouse had gone in and I did not check it, the mouse would quickly run out of food and water!" Andrew tells us. "Having been interested in Raspberry Pi for a couple of years and having recently begun learning Python using the Enviro+ environment sensors, I figured a Raspberry Pi with a motion sensor would be an ideal way to check."

It's a fairly simple setup, one commonly used in CCTV builds and some fun 'parent detectors' on the Raspberry Pi Foundation's projects site.

Mouse motion

"I came across a couple of automated mousetraps that people had made from scratch, but wanting to keep it simple and cheap," Andrew explains. "I

▲ An old coffee tub is used as a case for the sensor, a good way to recycle

A Raspberry Pi Zero is used to check the motion sensor and send data if it's activated

Installed into an old coffee tub, the standard PIR is something easy to find and connect to GPIO

Quick **FACTS**

> The whole setup costs roughly £28

> At the time of writing, Andrew has safely caught three mice

> Camera modules can detect motion

> The detection rate is calibrated to the environment

> Don't fill your trap with cheese
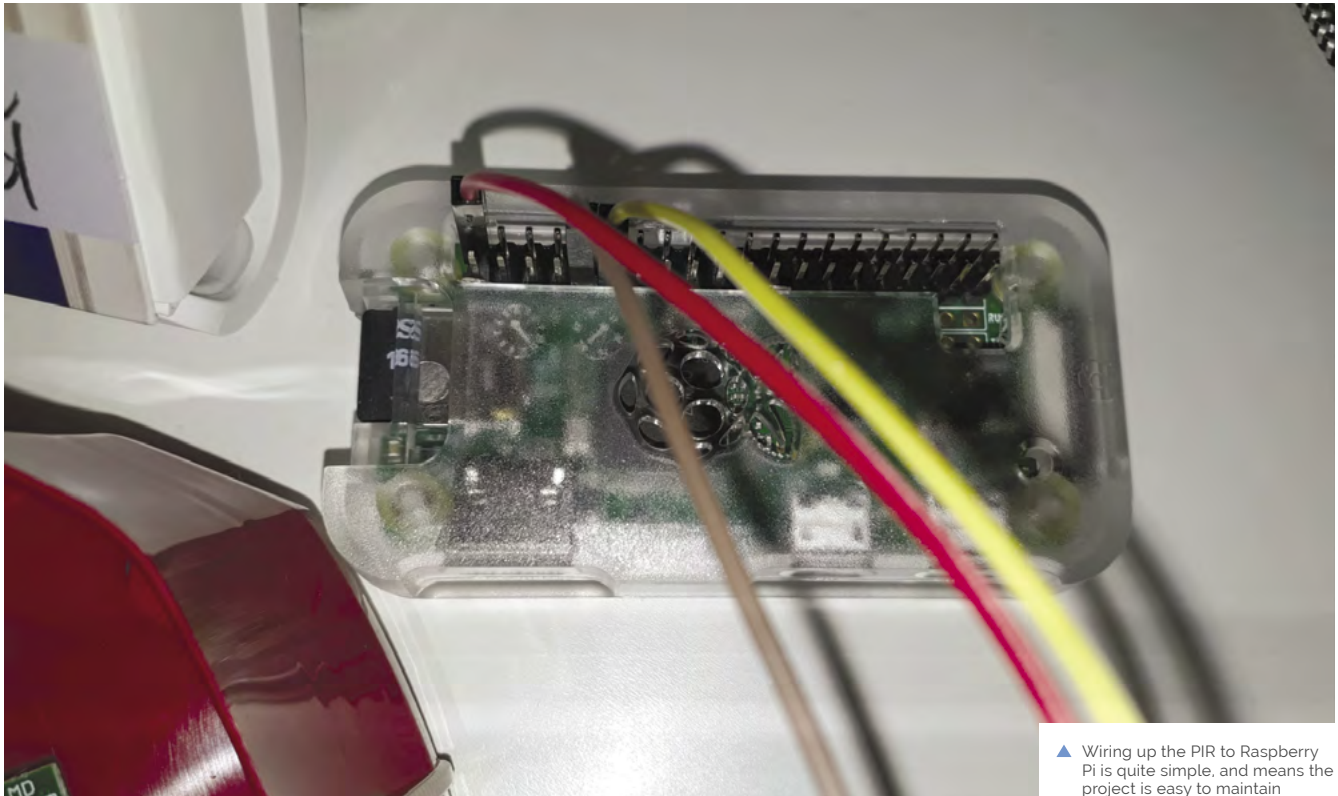
▲ Wiring up the PIR to Raspberry Pi is quite simple, and means the project is easy to maintain

> ❝ I came across a couple of automated mousetraps that people had made from scratch, but wanted to keep it simple and cheap ❞

wanted to use off-the-shelf parts where possible and keep costs down. The Pi Hut had a tutorial for a DIY burglar alarm utilising a PIR sensor, IFTTT, and Pushbullet, which seemed like an ideal starting point (**magpi.cc/pihutifttt**)."

IFTTT – If This Then That – is an online service popular with IoT folks. It's great for small things like cross-posting images on social media services, or sending a push notification when motion is detected in a mousetrap.

"I have only had one mouse since, but it worked!" Andrew says. "I was averaging about 800 detections a day and suddenly got well over a 1000. Sure enough, there was a mouse in the trap which I released shortly afterwards. I do tend to notice that the values fluctuate a bit, so it is always worth checking over the previous day's results to see if it is notably higher."

You might think that 800 push notifications a day is far worse than just occasionally checking your garage, and you'd be right, so Andrew tweaked the code a bit: "The code examples I found sent a
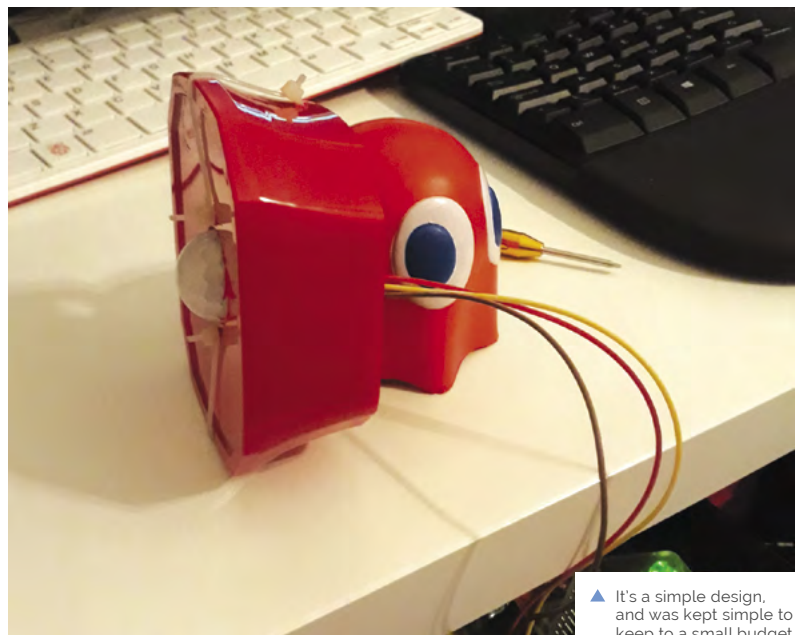
▲ It's a simple design, and was kept simple to keep to a small budget

▲ The first successful capture was released back outside the garage

notification for each movement detection – which I knew would be rather annoying, considering how randomly PIR sensors sometimes seem to trigger. My script instead logs any hits at a max of 1 per 30 seconds and then triggers a notification once every 24 hours, meaning I just get one notification a day."

Beat a path

There's always room for improvement, as Andrew explains: "I intend to improve the code so that it can record running averages and give an indication as to whether it believes there has been a significant spike that might necessitate me checking it out."

Whilst the aim of the project was to keep costs down, Andrew is tempted to experiment by adding a camera, and possibly a light, so he can have a peek remotely when there has been a spike in the readings and to see if it is a false alarm. Which, as he admits, is "a new height in laziness!" M

Catching mice

01 The setup stays on 24/7, and monitors the PIR sensor for movement. The PIR uses infrared so it works just fine even in the dark.

02 Movement is tracked using a Python script which checks, and logs, every 30 seconds. This allows for more accurate long-term readings.

Jun 10, 11:50 pm

Motion Detection Report
June 10, 2021 at 11:50PM - 1117 occurrences in the last 24.02 hours

Jun 12, 12:09 am

Motion Detection Report
June 12, 2021 at 12:09AM - 994 occurrences in the last 24.32 hours

03 "After 24 hours, it triggers a message containing the number of hits over that period by sending a HTTP request to IFTTT which is hooked up to Pushbullet," says Andrew. "This then sends this message as a notification to my phone. The counter is then reset ready for the next day."

Bop It
Minecraft
Controller

Why play Minecraft with a keyboard and mouse when you can use an electronic toy modified with Raspberry Pi Pico? **Nicola King** appreciates the twist

MAKER

Seth Altobelli

Seth is a student studying robotics engineering who enjoys making videos on projects that combine electronics, software, and mechanical elements.

magpi.cc/ sethaltobelli

One of the world's favourite electronic toys, Bop It has been around for 25 years. Normally, you press, pull, flick, spin, and twist its buttons and knobs according to the spoken instructions given. Great fun, if noisy and annoying for parents. Seth Altobelli, however, opted to refit his Bop It with a Raspberry Pi Pico to turn it into a USB controller for Minecraft.

He was inspired to create the project by two prominent YouTubers. "Michael Reeves made a video using a Bop It as an alternative interface to Hawaii's nuclear alert system. This inspired me to use a Bop It, or similar device, as an input for something it was not designed for," he says.

"Another creator, Technoblade, has an old video where he beat Minecraft hardcore, the most challenging difficulty, with a steering wheel. I was watching this a while back and then the idea of playing Minecraft with a Bop It just came to me."

Buttons to Pico to USB

To turn his Bop It XT into a controller, Seth first took the case apart and removed the existing PCB, replacing it with a Raspberry Pi Pico. All of the Bop It's controls link to simple momentary push-buttons, which meant he could simply wire these to digital GPIO inputs on Pico.

When an internal push-button is pressed, it triggers Pico to send the corresponding keyboard command over USB. "Pico can act as a USB device, meaning, as far as the computer running Minecraft knows, it is a normal keyboard," he tells us.

In addition to the button inputs, he installed an accelerometer in the Bop It. This connects to Pico
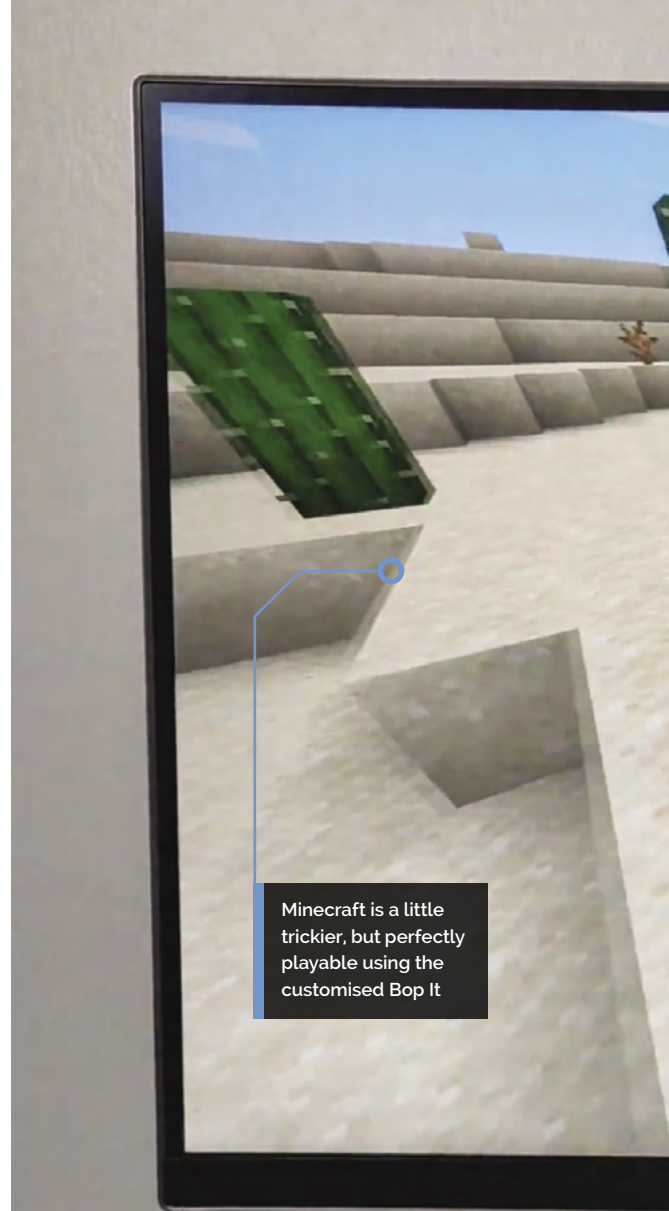
Minecraft is a little trickier, but perfectly playable using the customised Bop It

over I2C. "[Pico] reads the acceleration vector, does some simple vector math, and uses that to send mouse commands over USB," explains Seth. "The fact that Pico can so easily be programmed to act as a keyboard and mouse at the same time made this project possible."

Develop and debounce

The project took Seth two days to develop. "The first day I opened the Bop It, analysed how the inputs interface with the control board, wired the circuit, and wrote a basic program to make sure my wiring was correct," he recalls.

"The next day, I wrote the main control program. I then played Minecraft with it and made minor changes over the next few days when I had a few hours free."

The most challenging element for Seth was the programming as, although an accomplished Python coder, he was using CircuitPython for the first time. "That said, it made the USB functionality surprisingly easy."

The trickiest part of the program was the debounce timers he had to code for each button,
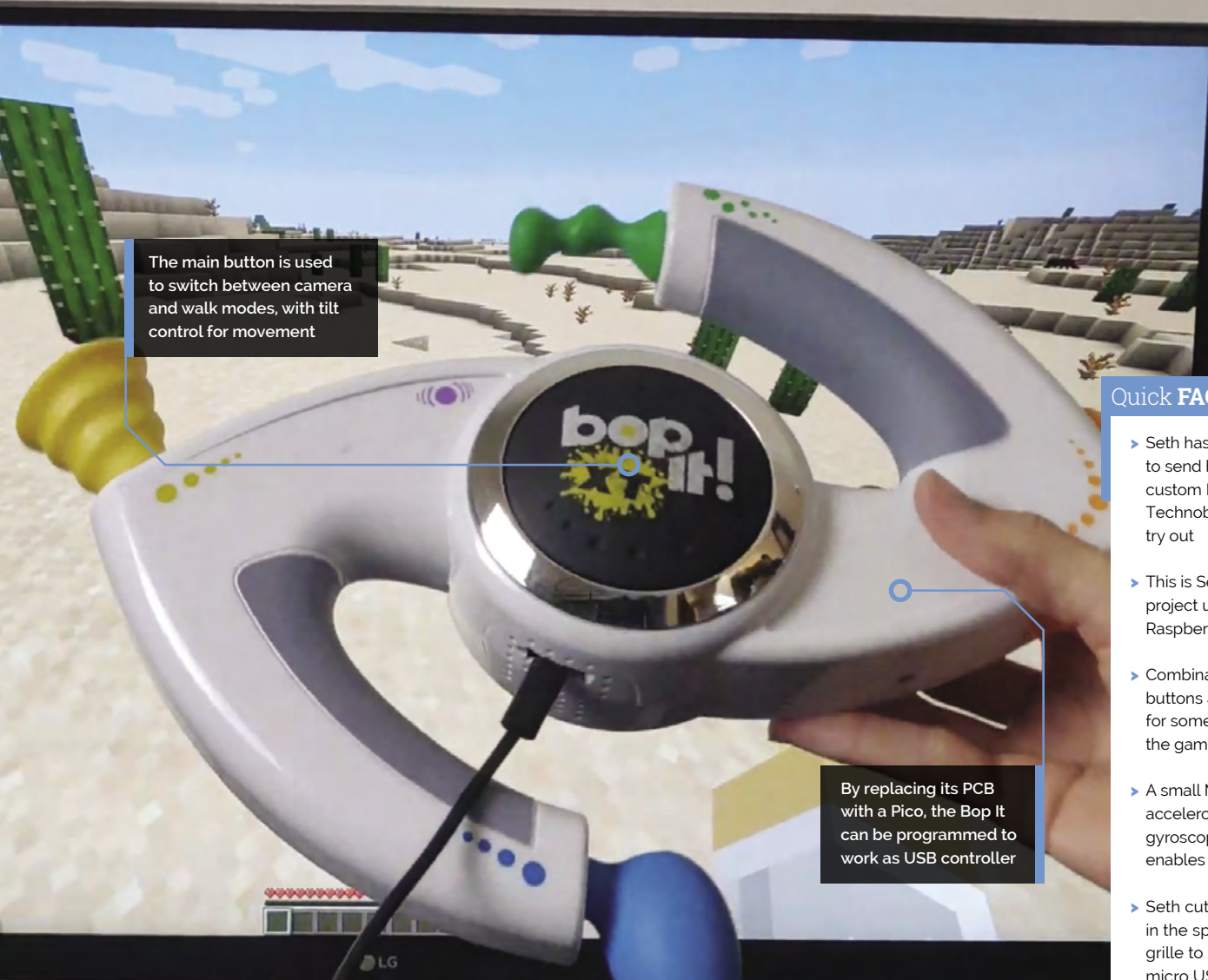
The main button is used to switch between camera and walk modes, with tilt control for movement

By replacing its PCB with a Pico, the Bop It can be programmed to work as USB controller

“ Pico can be programmed to act as a keyboard and mouse at the same time ”

to ensure they only trigger once each time they're pressed. "This was especially important for the buttons that moved in the hotbar and the right-click button that is used for eating and placing blocks," he notes. "The Bop It made this difficult as some of the buttons are challenging to quickly press, meaning the code would toggle them multiple times as though they were being held down. If I increased the timer too much, it would limit the frequency I could press them. I was able to get all the timers to work, but it did take some fiddling."

The end result works surprisingly well for playing Minecraft, as demonstrated in his YouTube video (**magpi.cc/bopityt**) – particularly the tilt control for movement, Wii Remote style. Not bad for a project he describes as being "made as a joke from the start."
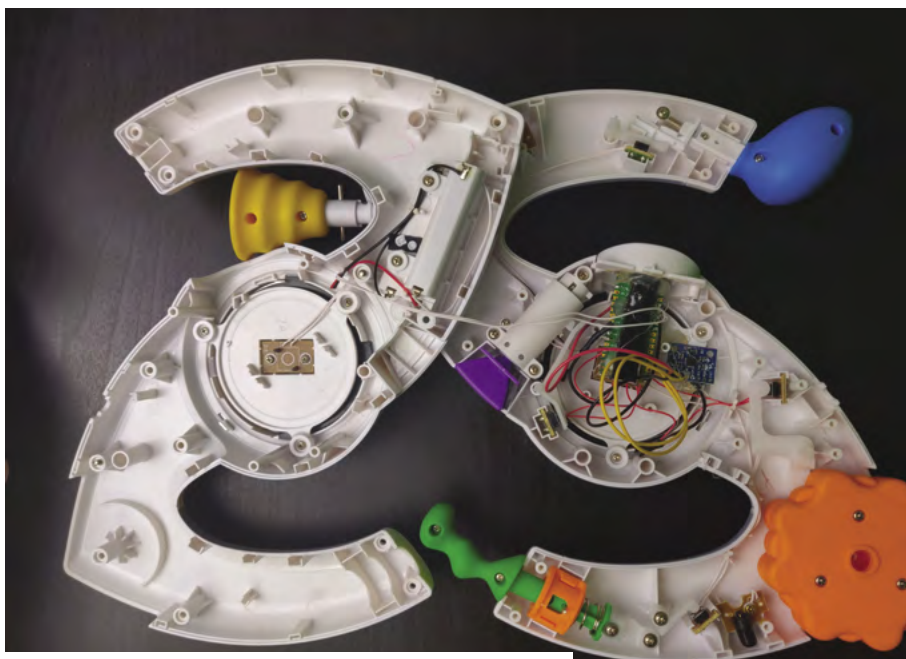
▲ Each of the Bop It's internal push-buttons is connected to a digital input on Pico, which is held in place using a small 3D-printed mount

PrivacyMic

A device that listens to noises in your home but doesn't hear your conversations? **David Crookes** tunes in

MAKER

Yasha Iravantchi

Yasha Iravantchi is a PhD candidate at the University of Michigan. He researches novel sensors that operate in a privacy-preserving way.

magpi.cc/ privacymic

More than 320 million smart speakers have been sold across the world, allowing us to use our voices to play music, ask questions, and discover information. A growing number of Internet of Things (IoT) devices are also being embedded with microphones, but there are concerns they're capable of eavesdropping on our conversations

"It's hard to tell what an IoT device is doing with its microphone beyond an LED signifying its microphone is in a muted or unmuted state," says Yasha Iravantchi, a graduate student at the University of Michigan. "In the future, these devices are going to perform more tasks than just listening for speech commands and there is the possibility that a lot of audio will be captured, recorded, and stored."

For this reason, Yasha has been leading a team developing PrivacyMic, a research project ensuring private conversations cannot be recorded and stored. By only gathering sound at frequencies

above the range of human hearing, the system filters out speech and audible sound, yet can still understand what's happening in our environment.

Sound of silence

PrivacyMic is built around Raspberry Pi and it works with ultrasonic sounds – that is, those with a frequency of 20 kilohertz or higher. Many objects and actions emit ultrasound waves, including compact fluorescent bulbs, dishwashers, computer monitors, flushing toilets, and electric toothbrushes. "It's a frequency that's inoffensive to humans, but can be annoying to dogs," says Yasha.

Most traditional audio equipment won't capture ultrasonic sounds. "Devices are tuned to focus on the range of human speech or hearing and they often actively remove the sounds outside of these ranges as 'noise' from the environment," Yasha explains. By creating a HAT for Raspberry Pi Zero W using an analogue ultrasonic microphone and a filter to remove speech and audible frequencies, however, PrivacyMic can do the opposite.

"The 'noise' that these systems throw away is a valuable signal we can use to recognise and classify daily-use objects," Yasha says. It means PrivacyMic can hear when a light bulb or microwave is turned on, determine when a toothbrush is being used, or when a toilet is flushed, without capturing any conversations. "We've also been using Raspberry Pi 3 to explore ways of performing all tasks – from capture to classification – on the computer itself to ensure no data ever leaves Raspberry Pi."

Listen up

So how does it work? "Analogue filters remove the audible frequencies and only allow ultrasonic frequencies to pass through to the analogue-to-digital converter (ADC)," Yasha explains. "These signals are passed to a machine learning model to recognise these daily-use objects in ultrasound
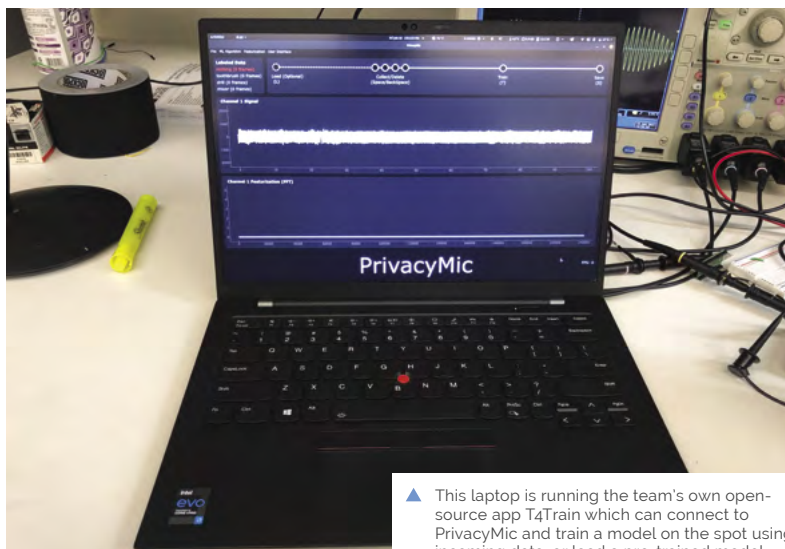
▲ This laptop is running the team's own open-source app T4Train which can connect to PrivacyMic and train a model on the spot using incoming data, or load a pre-trained model
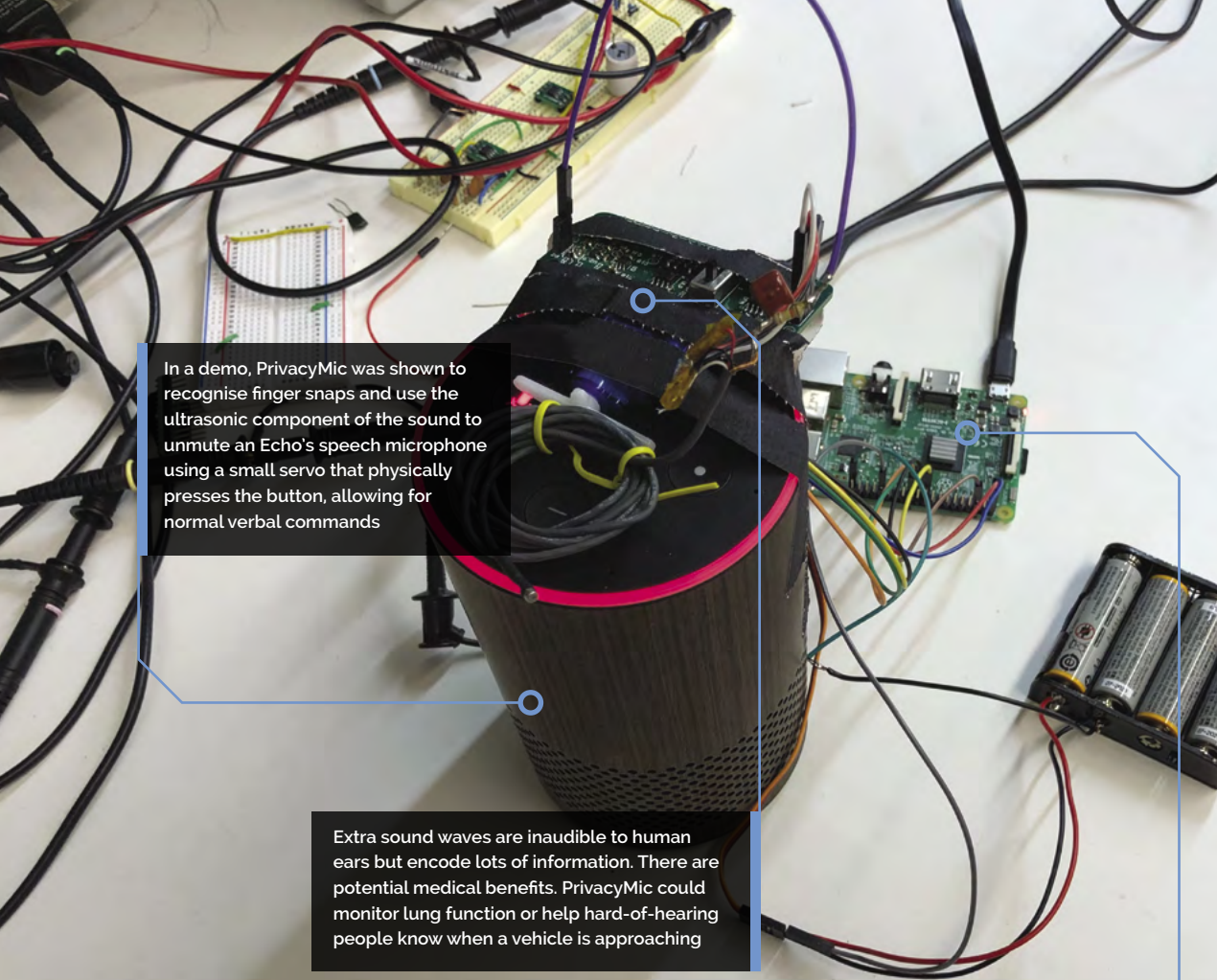
In a demo, PrivacyMic was shown to recognise finger snaps and use the ultrasonic component of the sound to unmute an Echo's speech microphone using a small servo that physically presses the button, allowing for normal verbal commands

Extra sound waves are inaudible to human ears but encode lots of information. There are potential medical benefits. PrivacyMic could monitor lung function or help hard-of-hearing people know when a vehicle is approaching

An audio filter removes all audible speech. Incoming sound is analysed by machine learning software to determine what device is being used

Quick **FACTS**

> PrivacyMic only listens for frequencies of 16kHz+

> Sound has been collected from 127 objects

> The device can identify activities with 95% accuracy

> The HAT can move between Raspberry Pi computers

> The team has applied for patent protection

> ## It's a frequency that's inoffensive to humans, but can be annoying to dogs >>

only, and the model then records an entry, including a time stamp and the name of the thing that turned on."

In doing so, the filtered ultrasound is not kept. "The idea is that there are multiple layers of safeguards in place that allow us to minimise the privacy exposure to users while keeping the artefacts that matter. In the case of activities of daily living tracking, it's the activity log, not the sounds themselves that are of value," Yasha says. Accomplishing this has entailed a lot of coding. C is used to efficiently send samples from the ADC to a laptop via TCP, and a new framework for interactive machine learning has been developed in Python. "It's called T4Train (**magpi.cc/t4train**) and it's a work-in-progress, but it's already compatible with Raspberry Pi," Yasha says.

Even so, work is far from complete. "One of the hardest challenges is trying to train PrivacyMic to learn to recognise sounds that you yourself cannot hear," Yasha laments. It means the device won't likely be out of the proof-of-concept stage for a few years yet, but we'll keep our ears to the grou for more news as the project progresses. M
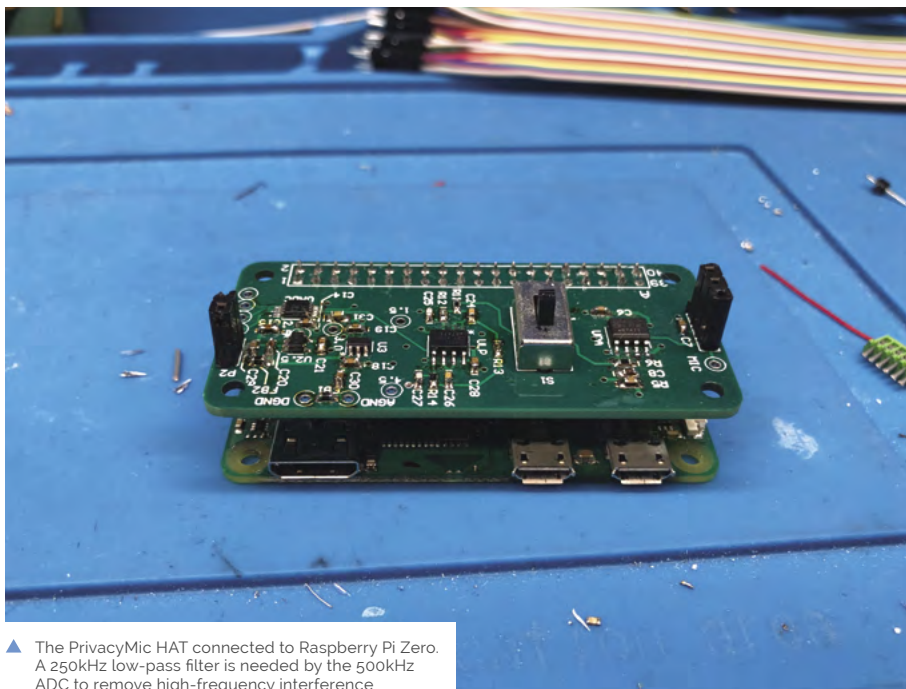
▲ The PrivacyMic HAT connected to Raspberry Pi Zero. A 250kHz low-pass filter is needed by the 500kHz ADC to remove high-frequency interference

Campervan LAN

Needing to travel to several countries for work, one maker chose to kit out his campervan as a mobile office. **Rosie Hattersley** was intrigued

MAKER

Enrico Miglino

Enrico designs and develops multi-platform software projects for clients across Europe (and occasionally India). He's been using and helping road-test Raspberry Pi since its first Linux prototype.

we-are-borg. com

Campervans are all the rage right now, since they offer a chance to escape the home in favour of fresh air and a change of scene. Enrico Miglino's job often needs him to pitch up in Germany, Spain, or Belgium where his software development architect consultancy is in demand, so it made sense for him to take the digital nomad concept literally, and adapt a campervan to create a mobile office.

Although most campsites and caravan parks offer WiFi to guests, such connections are often unsecured, limited to only a couple of hours' free use, and require each device to log in separately. Enrico needed a setup that was far more robust. His vehicle, Jan The Van, sports a secure mobile LAN, is powered by three Raspberry Pi computers, and allows Enrico to use a single login to provide internet access to any devices on his network. The whole setup cost less than 500 euros, meaning he could also afford to add a solar panel.

Hack The Van

Enrico bought his van specifically for use as a mobile office. "The project aims to create a modular set of technological improvements in a standard vehicle to convert it to a secured and efficient mobile unit for living, working, and travelling," he explains.

It's named after Enrico's friend Jan Cumps who helped him work out the electronics that would be needed to successfully hack the van. The pair spent many evenings working together studying how to develop projects, debug hardware circuits, and teaching workshops at the Ingegno Makerspace in Ghent in Belgium (**ingegno.be**).

From the driving seat, Enrico can get live updates about on-board essentials such as water and fuel, as well as alerts about intruders

Jan The Van doubles as a mobile office and panel van conversion

Two Raspberry Pi 4 boards control the on-board LAN and provide wireless logins for passenger's smartphones, laptops, and tablets

Quick **FACTS**

> Enrico has an original Raspberry Pi prototype

> He participated in the design of its first Linux distro

> He loves Raspberry Pi for its flexibility and adaptability

> A modular approach means project components are readily reusable

> The biggest Jan The Van challenge: space, of course

▲ Jan The Van – a mobile office with a constantly changing landscape

▼ A quick glance inside and Jan seems like a relatively normal luxurious campervan

The duo took a modular approach to the build, starting with an 8GB Raspberry Pi 4B mounted in a case with a 7-inch touchscreen. Developed with Node-RED, the display shows the state of the network, firewall status, and the local weather. It also has screens monitoring any physical intrusions around Jan The Van.

DHCP features and the wireless LAN to Ethernet router make it safe for Enrico to connect his shielded network to insecure public WiFi networks. He's currently adding sensors to monitor gas and smoke levels from the van's kitchen, and to show how much water he's storing on board, providing reassurance there are plenty of provisions on lengthier, more remote journeys.

Enrico found lots of ideas online for how to create his vision, building and testing each element before installing it. "I tried to design it to be as modular as possible so it's easy to replicate it and adapt the modules to similar but not identical environments," he says.

More Raspberry Pi

Internal devices on Jan The Van's network could only be connected to the Ethernet port, which wouldn't work for iPads, iPhones, and other mobile devices that lack the necessary port. To overcome this, Enrico added an Ethernet switch and a second Raspberry Pi 4B configured as a bridge to which

▲ Jan The Van's third passenger reportedly enjoys the ride

> ❝ I tried to design it to be as modular as possible so it's easy to replicate it and adapt the modules ❞

these mobile devices can connect wirelessly. A third Raspberry Pi connected to a full HD webcam provides live visuals of the rear of the vehicle while he's driving, and acts as a motion sensor security camera when the campervan is parked. It "detects motion around the camera field of view and records video just in case," Enrico says.

Jan The Van's first big trip was a ten-day journey from Belgium to a "nice campsite" in Spain – some 2800 km – where Enrico easily hooked up his campervan LAN and was able to work as a consultant each morning and tinker with his mobile office setup each evening. Planned improvements include adding a third Raspberry Pi 4 with a display to stream images from the van's rear camera and other information to the dashboard or cockpit. Most important, says Enrico, "is travelling to nice places to test the prototype in the real world." **M**

⚠ Warning!
Electric hookup

Be careful when adding a power inverter to a van for electric hookup and seek professional advice.

magpi.cc/ powerinverter

Hack your van

01 To create your own campervan network, you'll need two Raspberry Pi 4 computers for the LAN, one of which acts as a wireless bridge to provide ad hoc access, and a 300W or less power inverter. Laptops can also connect via Ethernet.

02 The main interface uses Node-RED software to manage the hardware, with C/C++ to program the ESP3266 modules attached to the sensors. The screen orientation is corrected in configuration.

03 Enrico used Python bash scripting to manage the network, execute timed tasks, and automate image saving from the on-board camera. This can be either a webcam or a Raspberry Pi Camera Module. Full details of Enrico's hack the van project are at **magpi.cc/janthevan**.

Air Quality
Traffic Light

Worried about the effects of poor air in his home city, Dmytro Panin has created a unique alert system, as **David Crookes** explains

MAKER

Dmytro Panin

Dmytro Panin is a programmer based in Kyiv, Ukraine, and he wrote his first line of code aged eight. He works for a large provider of nearshore software engineering services.

magpi.cc/ airquality

Kyiv in Ukraine is one of Europe's most polluted cities. It has high smog levels, which can cause health problems. "It also occurred to me that, in summer, the air quality in our area can deteriorate due to peat deposits igniting and smouldering somewhere in the province," says Dmytro Panin.

For that reason, he decided to take a break from a project he was working on – a device which monitors currency prices using Raspberry Pi Zero W – to create an air quality reporting system. Having found a battery-operated toy traffic light, he came up with a plan that was both practical and fun.

"If the wind blows in the direction of the city, you can smell the smoke, which some say resembles whiskey, and it means you don't want to have the windows open for too long," Dmytro says. "So, I came up with the idea of having the red light of a traffic light illuminate if this was to happen."

Capturing air quality

Dmytro chose a traffic light made by German toy manufacturer Dickie. "It's compact, has a button on top, and two additional LEDs accompanying the standard three," he explains. But before he could connect it to a Raspberry Pi 3B+ computer, he needed to open the toy.

"I had to get rid of the traffic light's internal components without damaging the LEDs or the internal structure of the toy," he continues. "I also had to change the resistors so that it would work with Raspberry Pi's GPIO pins, which produce 3.3 V."

From there, it was a matter of working out the best way of capturing air quality, starting by experimenting with the MQ-135 gas sensor which monitors benzene, alcohol, and smoke. "I got it working after a few hours by connecting an additional analogue-to-digital chip to the serial port of Raspberry Pi," he says. "But the measurements were not that precise and I had to write quite a bit

of code to account for inaccuracies which varied drastically based on the environment temperature."

Good to go

Instead, he turned to an MH-Z19B carbon dioxide sensor. "It was way more accurate and it worked, but only temporarily, as the elements in my area can be on the extreme side," Dmytro says.

The solution, therefore, was to write a short program using the open-source programming language Go (**golang.org**). It fetches Air Quality Index data from the website IQAir (**iqair.com**), which uses information from consumer devices

▶ The traffic light is only used during the day. It's been coded to turn itself off at night, but can be activated for a minute by pressing the button on top
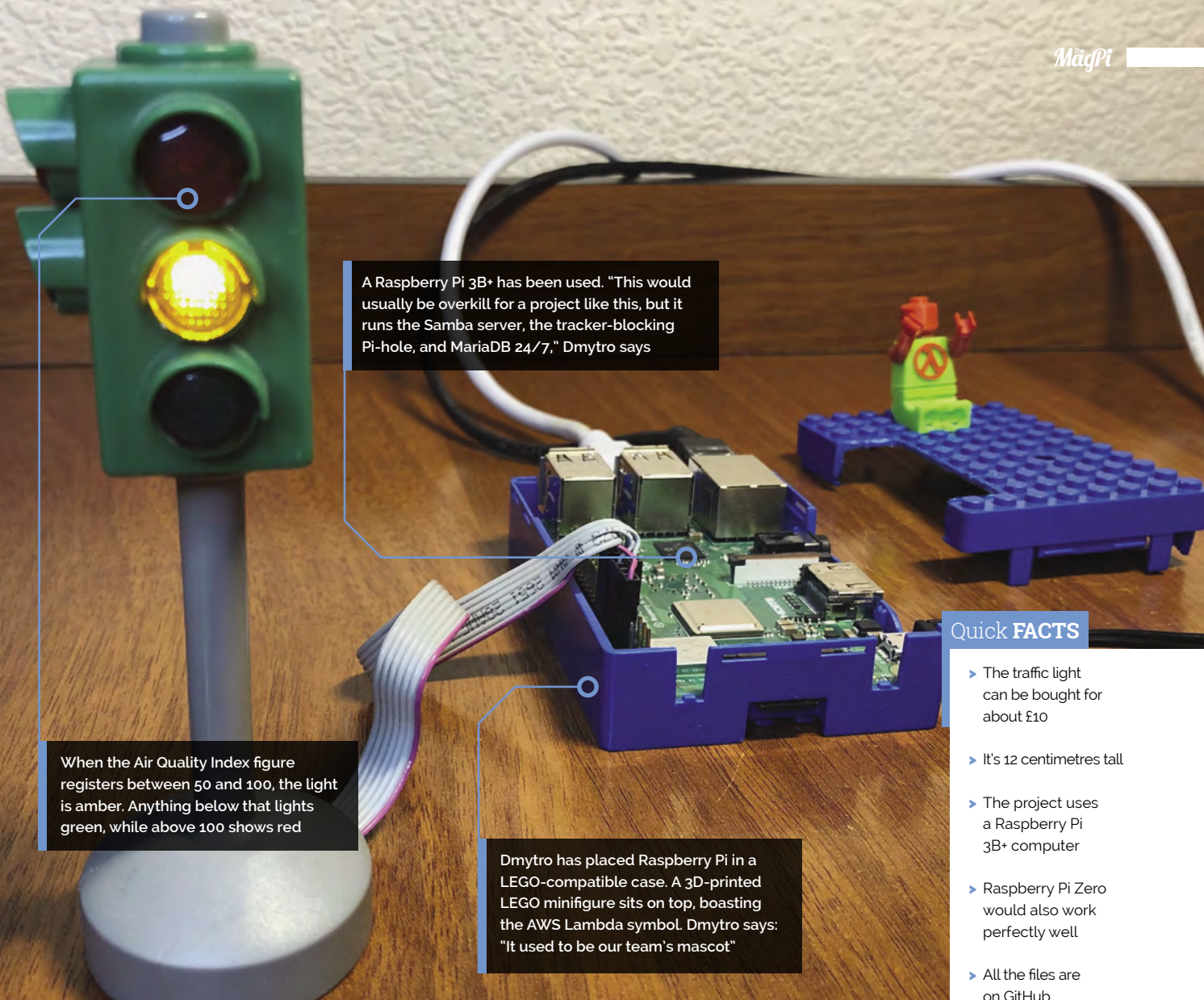
A Raspberry Pi 3B+ has been used. "This would usually be overkill for a project like this, but it runs the Samba server, the tracker-blocking Pi-hole, and MariaDB 24/7," Dmytro says

When the Air Quality Index figure registers between 50 and 100, the light is amber. Anything below that lights green, while above 100 shows red

Dmytro has placed Raspberry Pi in a LEGO-compatible case. A 3D-printed LEGO minifigure sits on top, boasting the AWS Lambda symbol. Dmytro says: "It used to be our team's mascot"

Quick **FACTS**

> The traffic light can be bought for about £10

> It's 12 centimetres tall

> The project uses a Raspberry Pi 3B+ computer

> Raspberry Pi Zero would also work perfectly well

> All the files are on GitHub

❝ The aggregated data is quite accurate, but it covers the entire city ❞

located across Kyiv. It also grabs current data from OpenWeather (**openweathermap.org**) in parallel.

By storing the data on Raspberry Pi, Dmytro's script – written in Python – can determine which light to illuminate, controlling the LEDs using the Python library GPIO Zero. Red obviously shows that the air quality is poor, prompting Dmytro to close the windows and take precautions when venturing outside.

Even so, there is room for improvement. "The aggregated data is quite accurate but it covers the entire city," he laments. "Sometimes it would make more sense for measurements to be more local. Collecting data myself and combining it with the information I get from the API should be the next step." ◾
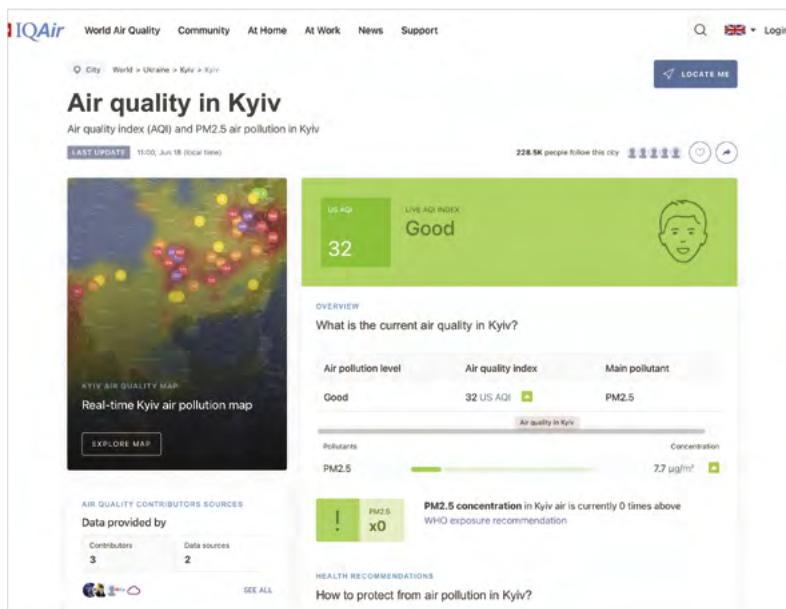
▲ Data from IQAir is used to help determine which LED should illuminate on the traffic light

Kenbak-2/5

Reproducing a personal computer that was first created way back in 1971 has been, for Michael Gardi, a labour of love. **Nicola King** turns back the years
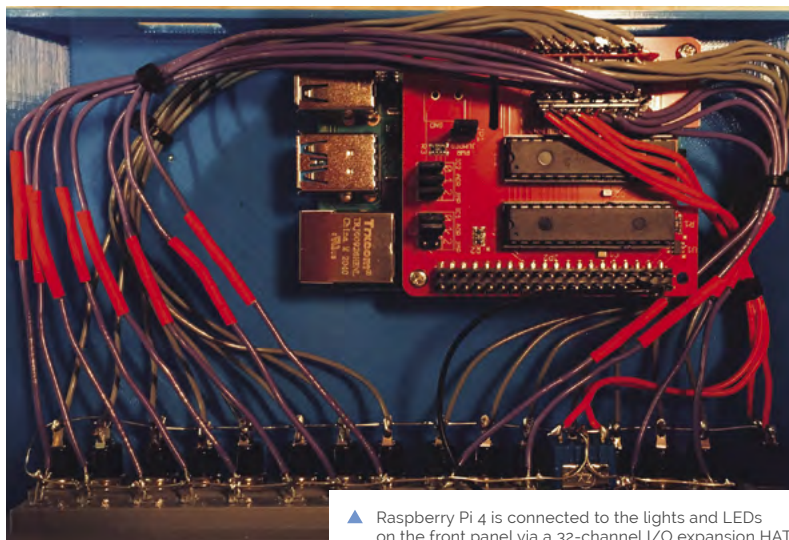
▲ Raspberry Pi 4 is connected to the lights and LEDs on the front panel via a 32-channel I/O expansion HAT
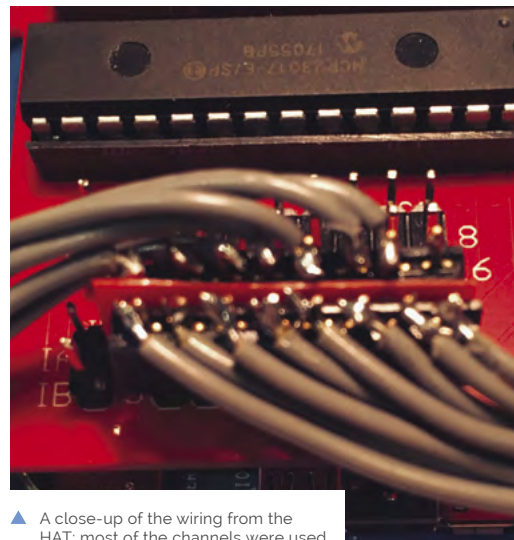
▲ A close-up of the wiring from the HAT; most of the channels were used

MAKER

Michael Gardi

A retired software developer, living in Waterloo, Ontario, Canada with his wife, who appreciates having the time to make whatever the heck he damn well feels like!

magpi.cc/mgardi

Regular readers of *The MagPi* may recall that in issue 102 (magpi.cc/102), we covered a fantastic Turing machine reproduction by Canadian retro enthusiast Michael Gardi. Well, Michael is back with another blast from the past – this time with his take on the Kenbak-1 which, as he shares, was "considered by many to be the first commercial personal computer, despite its failure in the marketplace."

The Kenbak-1 was created by John Blankenbaker, a computer engineer and inventor, and Michael was drawn to the idea of reproducing it in the form of the Kenbak-2/5 (a 2:5 scale model of the original) for a key reason. "For me, the thing that attracted me to this project, aside from the historical significance, was the sophisticated instruction set that the Kenbak-1 supported," he explains. "This was surprising because the computer was implemented with discrete logic chips and did not contain a microprocessor. The instructions were on a par with those used

in microprocessors that were to follow years later, like the Motorola 6800 (1974) and the MOS Technology 6502 (1975)."

A contemporary take

Michael has a YouTube video (**magpi.cc/kenbakyt**) that explains in detail how the original computer worked, as he enters a program through the model's front panel buttons. He then details how the Kenbak-2/5 supplements the original by allowing the user to enter a program by typing in assembly language instructions. In Michael's words, his version of this computer "allows one to have a 'classic' Kenbak-1 experience, then enhances that experience by allowing the user to use more modern tools like an assembler and a debugger." He also has a detailed Hackaday page (**magpi.cc/kenbak**), if you feel like having a go yourself.

Using photographs of the original machine as a guide, because finding an original Kenbak-1 would have been difficult and hugely expensive,
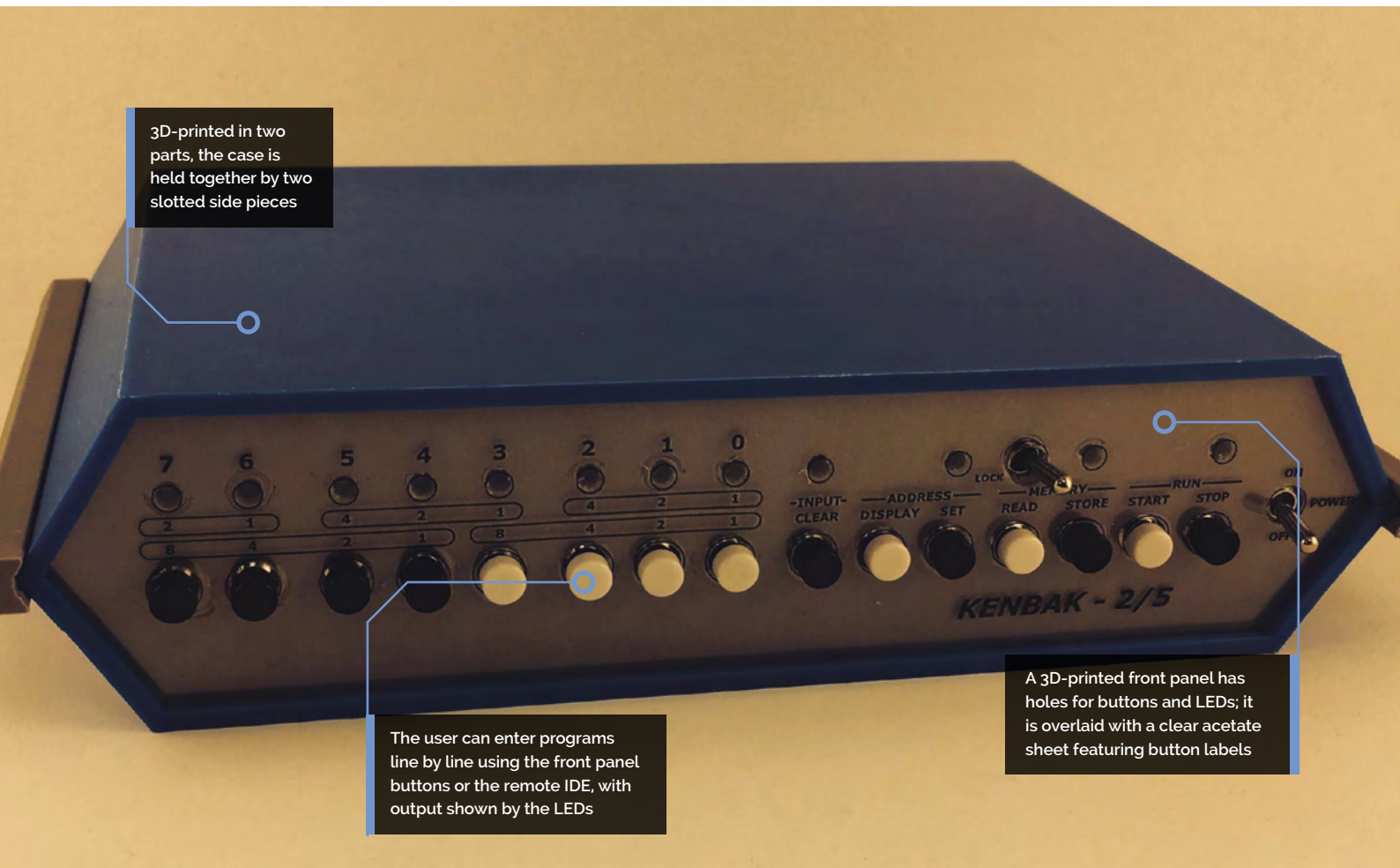
3D-printed in two parts, the case is held together by two slotted side pieces

The user can enter programs line by line using the front panel buttons or the remote IDE, with output shown by the LEDs

A 3D-printed front panel has holes for buttons and LEDs; it is overlaid with a clear acetate sheet featuring button labels

" I think that the devices themselves, and their designers, deserve to be remembered and honoured "

Michael 3D-printed the frame of the project. "One of the reasons that I chose to make this reproduction at 2/5 (40%) scale was so that all the 3D-printed parts (of which there were only five) would fit onto my printer. So, this simplified the assembly," he says. "Aside from the screws used to mount the Raspberry Pi to the frame, everything just snaps together."

A built-in Raspberry Pi 4 is connected to the front panel via a 32-channel expansion HAT. It also runs an IDE that can be accessed on a remote machine via SSH or VNC; the source code is on GitHub (along with the STL files for 3D printing the case): **magpi.cc/kenbakide**.

Michael created the Kenbak-1 Assembler and Emulator using Python and it's all based on John Blankenbaker's Programming Reference Manual. Any qualms he had about his understanding of the documentation were laid to rest by fellow enthusiasts who helped him out with potential glitches. "Fortunately, there is a pretty active Kenbak-1 community out there and some people actually tried out my software and found a few bugs and misunderstandings."

Architect's approval

The icing on the cake is that Michael's make has received praise from the original inventor. "I sent John Blankenbaker (91 years young) an email with pictures of my reproduction and a few questions, and he was gracious enough to answer. The first thing he said in his reply was 'Your email was the most interesting thing I have received today. I was very impressed!'. That sure made my day too!"

Praise indeed, and this underlines one of the reasons why Michael undertook the challenge: "I think that the devices themselves, and their designers, deserve to be remembered and honoured." Many enthusiasts would certainly agree with that sentiment. ◾

Quick **FACTS**

> Only 50 of the original Kenbak-1 computers were ever sold

> As they are so rare, the last one sold at auction went for $40,000

> Michael's version took one month to create

> Most of his projects are based on very retro reproductions…

> …like this impressive replica of a 1968 computer: **magpi.cc/vt100**

JOIN FOR 12 MONTHS AND GET A

FREE Raspberry Pi Zero W Starter Kit

WITH YOUR FIRST 12-MONTH SUBSCRIPTION

Subscribe in print today and you'll receive all this:

WORTH **£20**

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

This is a limited offer. Not included with renewals. Offer subject to change or withdrawal at any time.
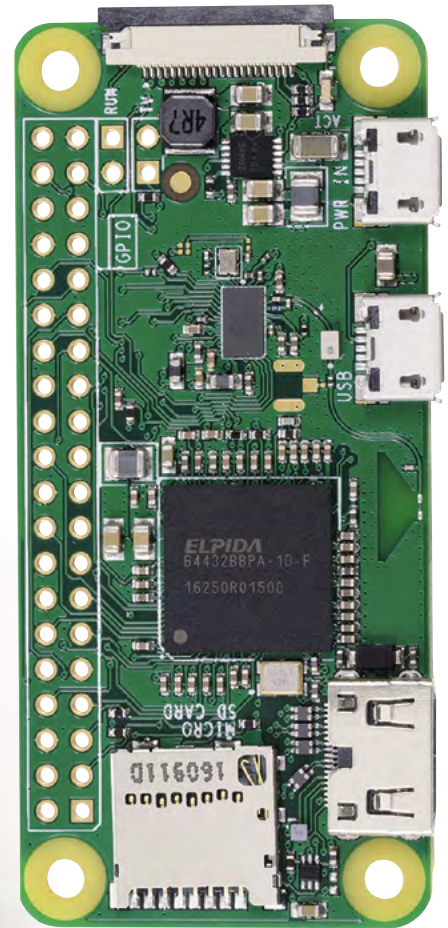
◉ Buy now: magpi.cc/subscribe

YOUR **OFFICIAL** RASPBERRY PI MAGAZINE

The MagPi
Issue 107 | July 2021 | magpi.cc | The official Raspberry Pi magazine

BECOME A **RASPBERRY PI** GENIUS
Troubleshooting tips for everyone

Make a Weather Watcher
Starter problems Solved!
Brilliant Handheld projects
Recreating The Mars Helicopter

50 PAGES OF PROJECTS & TUTORIALS

SUBSCRIBE on app stores
From **£2.29**

Available on the App Store

GET IT ON Google Play

AWARD WINNING MAKES

Incredible builds that have won competitions and prestigious honours

We like to hope that at least some of the projects we publish in *The MagPi* inspire people to make more. When it comes to inspiration, though, nothing beats projects that are changing the world for the better, and have won awards for doing so.

There are many kinds of awards, and even more kinds of winners, but here is just a selection of these prestigious projects that will make you go 'ooh' and 'aah' in appreciation, and hopefully be the spark of inspiration for your next build.

Nemo-Pi
Save Nemo
magpi.cc/nemopi

This underwater 'weather station' from the Save Nemo Foundation is being used to protect coral reefs from climate change and divers.

They work by attaching to a concrete block that has been placed on the sea floor – these blocks are attached to a buoy that boats can moor up to, rather than randomly dropping anchors that can damage the reefs. The Nemo-Pi can detect temperature, visibility, pH levels, and concentration of certain gasses in the water. As well as monitoring the ocean, it will allow holidaymakers to know if the conditions for diving are good.

Nemo-Pi won the Google.org Impact Challenge 2018 competition.

Space Vegetables
feiticeir0
magpi.cc/spacevegetables

Our friends over at element14 regularly have community design challenges that involve Raspberry Pi. A recent one, 1 Meter of Pi, was to create a $1\,m^3$ area that could grow food on a hypothetical trip to Mars. The winner was feiticeir0 with their Space Vegetables, a hydroponic garden that had a time-lapse camera and the ability to post to Twitter.

Like a lot of the element14 competitions, you can read a detailed, regular analysis of how the build and growing went. To summarise, the water, nutrient, and air pumps, along with grow lights, were all Raspberry Pi controlled using a mixture of sensors and code. The results of this cosmic produce has even been sampled.

Ecosystem monitoring
Sarab Sethi
magpi.cc/ecosystem

"The health of a forest ecosystem can often be attributed to how much noise it creates," says Professor Rob Ewers of Imperial College London, the university where this project that listens to the rainforest was created.

These solar-powered audio recorders using Raspberry Pi were the creation of a small team at ICL, including Dr Sarab Sethi and Dr Lorenzo Picinali, and are placed around the rainforest in Borneo. They record and transmit the noise 24/7 for scientific study or for you to just chill out to via the SAFE Acoustics live stream: **magpi.cc/SAFE**.

This system won a prestigious NETEXPLO Innovation Forum Award 2018, an award scheme from UNESCO that looks at university projects.

> ❝ **They record and transmit the noise 24/7 for scientific study** ❞

Coolest Projects Online 2021

I f you've read much of *The MagPi*, you'll know we love Coolest Projects. It's an event run by CoderDojo that showcases incredible projects from younger makers. For the last couple of years, Coolest Projects has had to go online, and this has meant not only more entrants but also more winners. For 2021, several judges chose their favourite projects out of over a thousand entries – here are the ones that used Raspberry Pi.

Cap For The Blind
Archit Garg and Omkar Prasad Rout from India
magpi.cc/capforblind

This cap has an ultrasonic distance sensor attached to the brim, making a warning noise if any objects are within 60 cm off the person wearing it. It's supposed to complement a cane for blind people which is used to detect objects lower down the user's body.

BROADCOM CODING WITH COMMITMENT

MediBuddy
Jayanth Ramganesh from the UAE
magpi.cc/medibuddy

A system to remotely dispense medicine to a patient, as decided by a doctor. A custom interface is given to the caregiver, with patient info and a button to dispense the medication, which is then pushed out of a hole in the MediBuddy case with a sound alert.

JAMES WHELTON'S FAVOURITE HARDWARE PROJECT
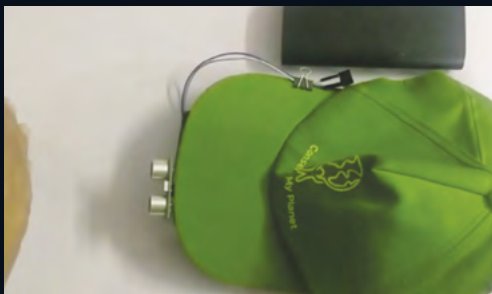
Friction experiments made easy
Artur Panek from Poland
magpi.cc/frictionexp

School physics lessons like to do practical experiments to determine friction on an object. Artur created a Raspberry Pi 3B+ box that you can place on the object, which not only aids in calculating the coefficient of friction, it provides a much more accurate measurement.

JAMES WHELTON'S FAVOURITE ADVANCED PROGRAMMING PROJECT

Backyard Squirrel Detection System
Arvin Zhang from the USA
magpi.cc/backyardsquirrel

This project's full name is 'A Deep Learning Based Backyard Squirrel Detection System Utilizing Raspberry Pi', and it does exactly that. Using Google's Coral Edge TPU, and a squirrel-trained deep learning model, Arvin was able to warn his grandma of invading squirrels so she could protect their peach tree.

MELISSA PICKERING'S FAVOURITE ADVANCED PROGRAMMING PROJECT

> ❝ **This consumer grade (under $100/£72) box can accurately detect earthquakes** ❞

Playable LEGO Steinway Piano
Lavie from the USA
magpi.cc/legopiano

This clever hack takes a LEGO piano that is fully mechanically functional, and uses a mixture of handmade paper PCBs with conductive ink and a Raspberry Pi to translate the mechanical functions to the notes it should play. Lavie had to debug her code to account for the way multiple keys are pressed in a smart way.

MELISSA PICKERING'S FAVOURITE HARDWARE PROJECT

Earthquake Early Warning
Vivien He from the USA
magpi.cc/earthquakewarn

This consumer grade (under $100/£72) box can accurately detect earthquakes over a magnitude of 3.0, and filter out footsteps, jumping, and other nearby human activity. It then sends a warning to the user's phone. Vivien hopes this kind of tech can be used to save lives and reduce damage.

COLIN FURZE'S FAVOURITE HARDWARE PROJECT

Past winner highlights

AZ-Tech Teddy
Andrei, Ioana, David Daniel and Bianca Laura from Romania

This smart teddy bear helps track eating habits in younger people, and helps them pay closer attention to what they're eating. It connects to various apps via Raspberry Pi.

BEST INTERNATIONAL PRIZE 2018

Door Pi Plus
Freddie
magpi.cc/doorpiplus

A door security system for elderly users that makes use of facial recognition software. This project has won more awards as well from other UK and global competitions.

COOLEST PROJECTS UK 2019 HARDWARE WINNER

Vital Signs Monitor
Adarsh Ambati
magpi.cc/vitalsigns

Adarsh's project is a system for monitoring vital signs to diagnose illness without touching the patient. It also provides limitless possibilities, from preventing sudden infant death syndrome to diagnosing viruses.

OVERALL WINNER OF COOLEST PROJECTS USA 2020

Astro Pi

You can find Raspberry Pi everywhere, even in space! There are yearly competitions to upload and run science experiments aboard two special Raspberry Pi computers, called Astro Pi, aboard the International Space Station orbiting 408 km above the Earth.

Here we present some of the past winners, along with highlights from the entrants for Mission Space Lab 2021.

2018/2019 Mission Space Lab

TheHappy.Pi
I Liceum Ogólnokształc ce im. Bolesława Krzywoustego w Słupsku, Poland

One of the winners from this Mission Space Lab were able to process images to measure concentrations of chlorophyll concentrations in Earth vegetation. You can read about the full list of winners here: **magpi.cc/msl2019**.

2019/2020 Mission Space Lab

Pardubice Pi
SPŠE a VOŠ Pardubice, Czech Republic

Using NDVI (Normalised Difference Vegetation Index) analysis on photos of vegetation, the team was able to analyse loss of vegetation compared to historic images of the same locations. You can read about all the winners here: **magpi.cc/msl2020**.
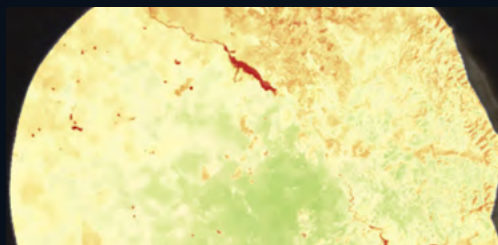
Astro Pi in numbers

426 total team submissions

232 experiments run on the ISS

1695 participants

From **23** countries

489 **(28.8%)** female participants

149 teams eligible to win

Astro Pi Mission Space Lab
2020/2021 highlights

The winning teams of the latest Astro Pi mission have not been announced at the time of writing, but here are some amazing highlights. Look forward to next issue to read about the winning teams!

> " This team is looking to study the evolution of vegetation around the world "
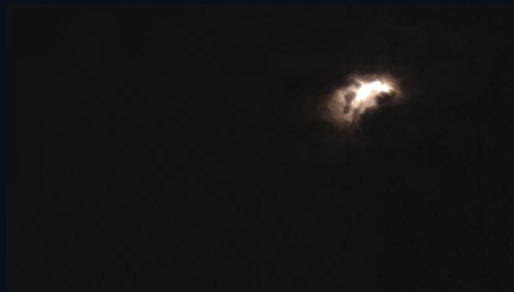
Team HamCloud

This English team captured lightning in the clouds during a night-time storm over South America – the first time this has ever happened during Astro Pi. Their experiment is to test cloud density to see if it's been affected by climate change.

Team Blue-3A

Italian team Blue-3A captured Turks and Caicos. The blue filter really accentuates the blueness of the barrier reef. Images are being taken to analyse how vegetation and inland waters (rivers, lakes, etc.) influence local climates.

Want to learn more about Astro Pi? Head to astropi.org

Team Imposter

A team from Ireland, they captured a great photo of the Maldives. Their experiment hopes to be able to compare historial data for regions that have had wildfires, to help predict what areas may be in danger of future fires.

Team BetVeg

Team BetVeg from Spain got this amazing photo of the southern part of Great Britain. This team is looking to study the evolution of vegetation around the world, comparing images to photos from previous Astro Pi missions.

Tech4Good
Young Pioneers

The Tech4Good awards are an annual event that celebrate many aspects of technology. One accolade is the BT Young Pioneers Award, given to young folks that are using tech to support climate action and other social good projects. Several winners have all done so with Raspberry Pi projects, showing how versatile Raspberry Pi is.

Kiera McKillop and Sinead McKeown – 2017
Dyslexic Aid
magpi.cc/dyslexicaid

Dyslexia effects a significant portion of the population, and results in people having some difficulties with reading and writing. The Dyslexic Aid creates a multi-sensory learning environment that stimulates different senses, aimed specifically at helping dyslexic pupils. With a very limited budget, they were able to do research and build the device with just a Raspberry Pi and a Sense HAT, and make use of its full suite of functions.

Arnav Sharma – 2016
AsthmaPi
magpi.cc/asthmapi

Arnav is one of the youngest award winners in this article at only nine years old. After studying the intricacies of the respiratory condition asthma, he hooked up a suite of sensors to a Raspberry Pi to monitor the quality of the air in an asthma sufferer's home and send messages with reminders, alerts, and even warnings to their phone. It can also help figure out any unknown triggers that can result in an asthma attack.

Mihika Sharma – 2019
Smart Stick
magpi.cc/smartstick

Mihika is also nine years old, and made the Smart Stick for blind people after witnessing her mother helping a blind person across the road. The Smart Stick is surprisingly, well, smart. As well as the visible ultrasonic distance sensors to detect obstacles, it can detect puddles, it has motors that guide the user left or right depending on smartphone satnav directions, LED lights for night-time, and Braille writing to let users know where to place a left or right hand. It's won other awards as well, and for good reason.

> 66 **The Smart Stick is surprisingly, well, smart** 99

Awards from The MagPi

As well as favourites, there have been winners…

Issue 50 community award

To celebrate our 50th issue, we decided to count down 50 incredible projects, with the top 20 voted for by our community. The top five included incredible creations, such as the SeeMore cluster computer art project and the BrewPi beer brewing system that a lot of adult makers love.

Magic Mirror
magicmirror.builders

A classic project. The concept, popularised by Michael Teeuw, has exploded in popularity, with people creating straight smart mirrors, smart vanity mirrors, and even recycling a broken iPad to act as a mirror. It's deceptively easy to make one, and installing useful code for it is a doddle. It's very extendible as well, and we've seen folk add seasonal extras and voice control.

Issue 75 community award

In the two years between issues 50 and 75, we were able to collate 75 brand new and amazing projects to once again show off in *The MagPi*. Instead of just voting for 20 of them, this time we had a top 50, as voted for by you.

PiSwitch
magpi.cc/piswitch

The design of the Nintendo Switch is an all-timer, and many folks have been inspired by its removable controllers. PiSwitch went a step further and had them connect to a 3D-printed case that also had a big display for a Raspberry Pi running RetroPie, with the Joy-Cons connected via Bluetooth and completely detachable like its inspiration machine.

Raspberry Pi awards timeline

A lot of people love Raspberry Pi, and over the last few years it's won plenty of awards itself. Here are just some of them…

T3 Award for Innovation of the Year	Software Satisfaction Awards – BASDA Theo von Dort award	Inquirer Tech Hero Awards – Best Product (Raspberry Pi 2)	MacRobert Award for innovation and engineering	Time's The 10 Best Gadgets of the 2010s
2012	2012	2015	2017	2019

PJ Evans

PJ is a writer, software engineer and tinkerer. His server has just told him it's time for another coffee.

twitter.com/ mrpjevans

MAKER

Part 01

Build the ultimate
home server with Raspberry Pi

Revolutionise your home network with shared storage, media serving, network management, home automation and more. All in one tiny box!

The rear of the case organises all the ports neatly and the bridge connects the M.2 drive to your Raspberry Pi 4

Raspberry Pi also makes a great home server. Why do you need a home server? Well, there's all kinds of uses: file sharing, protecting your network from dodgy advertisers, controlling your smart devices; the list goes on. In this new series, we'll build a home server with all the bells and whistles to inspire you to create your own ultimate build, with plenty of ideas and tips along the way.

01 Choose the right model

It's probably no surprise that we're using a Raspberry Pi 4 with 8GB RAM model for our ultimate server. It's the most powerful Raspberry Pi. However, you can go for lower memory sizes and get similar performance. If you're interested in using advanced file systems like ZFS for protecting your data, you'll need at least 4GB for it to work effectively. Then consider what the server will be doing. Serving files doesn't require much effort and is well within the capabilities of Raspberry Pi Zero, and we strongly recommend a wired network connection.

02 Choose a case

Our Raspberry Pi 4 needs a good home if it's going to be a server. There's no shortage of cases available and if you're lucky enough to have a 3D printer, many more designs are available for download. Have a think about storage requirements such as additional disks or access to USB or GPIO connectors. We've chosen the Argon One case. Not only is it beautiful, it also features

You'll Need

> Raspberry Pi 4
> **magpi.cc/ raspberrypi4**

> Argon One M.2 case
> **magpi.cc/ argononem2**

> 1TB M.2 SATA SSD drive
> **magpi.cc/ wdbluem2**

The Argon One is an attractive and practical case for a server

Wired Ethernet reduces load on your wireless network helping speed and reliability

passive cooling (as well as a fan) and support for M.2 storage devices. As a server is intended to be running 24/7, make sure your choice of case is well ventilated and will be placed where it can 'breathe'.

03 Choose storage

The standard application for a server is file sharing: the ability for anyone in your household to access a library of files. If you're thinking of a media server, consider carefully what type of storage to use and how much you need. Movies, especially in high-definition formats, eat up a lot of space but don't require fast storage to play back reliably. We're making use of the Argon One's M.2 interface to add 1TB of fast SSD-based storage, but you may prefer one or more USB SSD drives. Using a microSD card is still an option, but slower and less reliable.

04 Update your bootloader

By default, a Raspberry Pi 4 and Raspberry Pi 400 will try to boot from the microSD card and then from a USB attached drive. So you will not need to do any additional tweaks to get the M.2 drive working. USB boot is also available on most other Raspberry Pi models, including Raspberry Pi 2, 3A,

3B, and Compute Modules 3+ and 4. However, you will need to update the bootloader on these devices. See **magpi.cc/bootloader** for info and instructions.

05 Argon assemble!

Now we have the computer, case and storage, it's time to put them together. The Argon One comes with a comprehensive installation guide. Add the expansion board to bring all the connections to the rear of the case, then mount in the case as instructed, using the supplied thermal pads to get a good connection between the board and the heatsink. Install your M.2 SSD drive in the lower part of the case, then screw the two halves together. The M.2 SSD device is connected to Raspberry Pi 4 using an external USB-to-USB bridge.

06 Prepare the operating systems

Preparing an M.2 SSD device is identical to preparing an SD card, but connecting the M.2 device may be tricky. Either purchase an inexpensive M.2 to USB device or start with an existing Raspberry Pi build and use Argon One M.2's USB bridge to connect it. Then use Raspberry Pi Imager to 'burn' a Raspberry Pi OS disk image onto the SSD device so

Top Tip 👍

USB drives

If you're going for mass storage and fancy adding an array of drives, don't forget a powered USB hub to avoid undervolting – this can damage your data and hardware.

▲ The Argon One's clever expansion board brings all the connectors to the back and converts the HDMI ports to full-size

it's ready to boot. As this is a server, we don't need a graphical user interface, so we can keep things lightweight. Using Raspberry Pi Imager, select Raspberry Pi OS Lite and check out the advanced menu (**CTRL/CMD+SHIFT+X**) to set things like hostname and networking before burning the image to the card. We are naming our hostname 'ultimate' so it is easy to find on the network.

Top Tip

Which M.2?

Be careful when selecting your M.2 SSD. The Argon One supports SATA devices, not newer NVMe types.

07 Networking choices

Full-time servers are not normally connected to your network over wireless LAN. Whilst wireless is a great technology for end-users, servers crave reliability and the best way of delivering that is a wired Ethernet connection. We recommend placing your server close to your home broadband router (or whichever device handles your network traffic). That way, the network is not trying to handle two wireless conversations: one between you and the router and another between the router and the server. Wired will be much faster and reliable. If this isn't an option, don't worry: it'll still work, but larger files may be slow to transfer.

08 Boot time

Let your server boot for the first time, making sure no microSD card is inserted. Even if you configured it as headless, it's worth plugging a monitor in so you can check everything is OK. It will quickly reboot to resize the file system and then start up. We used the Advanced menu in Raspberry Pi Imager to set the hostname to 'ultimate', so with a wired connection we can ping 'ultimate.local' from another computer to confirm it's on the

▲ The aluminium case connects to the CPU and memory using thermal paste to act as a giant heatsink

network. If you didn't do this, set up your hostname by running `sudo raspi-config` from a Terminal. You can also set up networking here if you need to. Update everything with:

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

09 Boot from microSD

There is an argument to be made for still using a microSD card as the primary boot device. This dedicates your M.2 SSD to data only. In the event of an operating system failure, your data can still be accessed by removing the SSD drive. Likewise, if you need to increase your storage capability, you avoid having to rebuild your operating system from scratch. If you want to take this approach, make sure your bootloader prioritises the SD card (you can check in raspi-config > Advanced Options > Bootloader Version). You will you'll need to partition and format the second USB drive with an ext4 file system.

10 Fan and power button support

The Argon One case incorporates not only passive cooling but also has a fan for keeping our Raspberry Pi 4B nice and cool. It's not the quietest but does support active control. To install this, enter the following command in a Terminal:

```
curl https://download.argon40.com/argon1.
sh | bash
```

This will download the drivers and install them automatically. The fan will now adjust its speed

> " We recommend placing your server close to your home broadband router "

based on temperature. It will also allow the power button to cleanly shut down after being pressed for three seconds. To change these settings, run `argonone-config` from the command line.

11 Access from other devices

Let's go back to networking. Our server's hostname is 'ultimate', so using the ZeroConf DNS system (aka mDNS), it should be on the network as 'ultimate.local'. To test this from another machine, run `ping ultimate.local` from the command line and see if you get a response from the server. If you get no response, check your network settings on Raspberry Pi using `sudo raspi-config`. If using wireless LAN, try a wired connection instead. You can also use utilities like Fing (**fing.com**) to see if your server has acquired an IP address. (Note that versions of Windows prior to 10 do not natively support .local addresses.)

12 Create a user

All Raspberry Pi operating systems start with a pre-installed user called 'pi'. So we can share files and manage permissions, it makes sense to create our own user accounts. To do this, open a Terminal as the 'pi' user and run this command:

```
sudo adduser <name>
```

…where <name> is the username you would like (avoid spaces and special characters). It will ask you a few optional questions about your full name and location. Finally, you can set a password (be sensible!).

To give the user 'super user' or 'sudo' rights:

```
sudo adduser <name> sudo
```

To give the new user all the same permissions as the 'pi' user:

```
sudo usermod -a -G
```



```
sudo usermod -a -G adm, dialout, cdrom,
sudo, audio, video, plugdev, games, users,
input, netdev, gpio, i2c, spi <name>
```

▲ The lower part of the case accepts an M.2 SSD for large storage needs

You should now be able to log in as your new user. You can create as many of these as you need.

13 Create and exchange SSH keys

A fast and secure way to log into your new server is SSH key exchange. If you anticipate lots of logging into your new server with SSH, this is an essential step. If you're using Windows, start by installing the free OpenSSH client from Microsoft's instructions (**magpi.cc/msopenssh**).

From a command line, first generate your keypair (if you don't already have one):

```
ssh-keygen
```

This stores a public and private key in your home directory under **~/.ssh**.

Copy your public key over to your server:

```
ssh-copy-id <name>@ultimate.local
```

…where <name> is the username you created previously. Now you can log into your server without a password using SSH:

```
ssh <name>@ultimate.local
```

If your local username matches the server's, you don't even need to include it or the '@'.

Next time

You've now got the basis for building a powerful home server. Next month we'll move on to the most useful aspect of any home server: file sharing. M

Build an arcade machine:
RetroPie and stream from Steam

Use RetroPie as your arcade operating system and add extra emulators with support for Steam Link. Stream games from a powerful PC to Rasbperry Pi

MAKER

K.G.
Orphanides

K.G. is a writer, maker of odd games, and software preservation enthusiast. Their household can now hold very retro Street Fighter II tournaments, and that's beautiful

@KGOrphanides

▼ More button assignments are available on RetroPie than you have arcade controls. You can skip the ones that don't match up

Last month, we used Recalbox for our main arcade cabinet operating system, but it's not your only choice. In this final instalment of the 'Build an arcade machine' series, we'll use the RetroPie distribution, currently at version 4.7, to provide extra features such as Steam Link support, as well as taking a longer look at where to buy arcade games and how to get them onto your system. This tutorial assumes that you already have a fully assembled and wired arcade cabinet.

01 Install and prepare RetroPie
Fire up Raspberry Pi Imager, connect your microSD card writer, and install RetroPie from its Choose OS menu. Re-mount the microSD card once you've finished flashing it, because we've got some changes to make.

CONFIGURING

GAMEPAD 36

HOLD ANY BUTTON TO SKIP

⊕ BUTTON A / EAST	BUTTON 7
⊕ BUTTON B / SOUTH	BUTTON 0
⊕ BUTTON X / NORTH	BUTTON 3
⊕ BUTTON Y / WEST	BUTTON 2
◧ LEFT SHOULDER	BUTTON 4

OK

As with our DB9 joystick project in issue 101 (**magpi.cc/101**), we have to tell the GPIO to treat the controls as pull-up switches. Recalbox, by comparison, implements this by default.

Create the **pullup.sh** file we've supplied (**magpi.cc/pullupfix**). You can put it anywhere you like – we stuck ours in **/home/pi/**. Now open **/etc/rc.local** on the SD card and, above the `exit` line, add:

```
/home/pi/pullup.sh
```

This will load your pull-up settings on boot. If you're setting up your disk on a Linux system, you can set **pullup.sh** as executable now. Otherwise, we'll do that on first boot.

02 First boot
Make sure you have a keyboard plugged into your cabinet for this bit. We left ours propped up against the marquee acrylic during setup for easy access. A Bluetooth keyboard is a viable alternative, but it's easier to start with a wired connection.

Plug in Raspberry Pi's power. It should boot to the EmulationStation interface, but we can't configure the controls until we've set our pull-up script executable. Press **F4** to exit to the command line and type:

```
chmod /home/pi/pullup.sh +x
```

While we're here, let's enable SSH:

```
sudo raspi-config
1 system option
s3 password
```
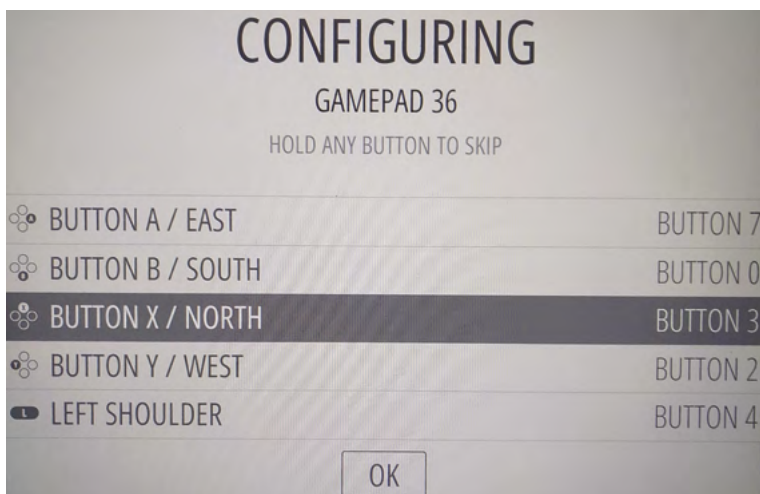
Raspberry Pi is powerful enough to run most games, and you can stream classic and modern arcade titles from a separate PC on your network via Steam Link

Steam Link lets you create a dedicated control map. Remember to map your hotkey button to its guide button

```
enter a new password
3 interface options
enable ssh
yes
```

You can now SSH into Raspberry Pi from another PC using a client such as Remmina or PuTTY.

03 Add hotkey button support

If, like ours, your arcade cabinet's GPIO controller setup has either one or two extra hotkey buttons for easy access to save, load, and exit shortcuts while playing, then the standard version of the mk_arcade_joystick_rpi driver available from RetroPie's package manager won't support them. We'll have to manually add an updated version from maintainer Recalbox's GitLab repo.

At the command line, type:

```
git clone --branch v0.1.9 https://gitlab.
com/recalbox/mk_arcade_joystick_rpi.git
sudo mkdir /usr/src/mk_arcade_joystick_rpi-
0.1.9/
cd mk_arcade_joystick_rpi/
sudo cp -a * /usr/src/mk_arcade_joystick_
rpi-0.1.9/
nano /usr/src/mk_arcade_joystick_rpi-0.1.9/
dkms.conf
```
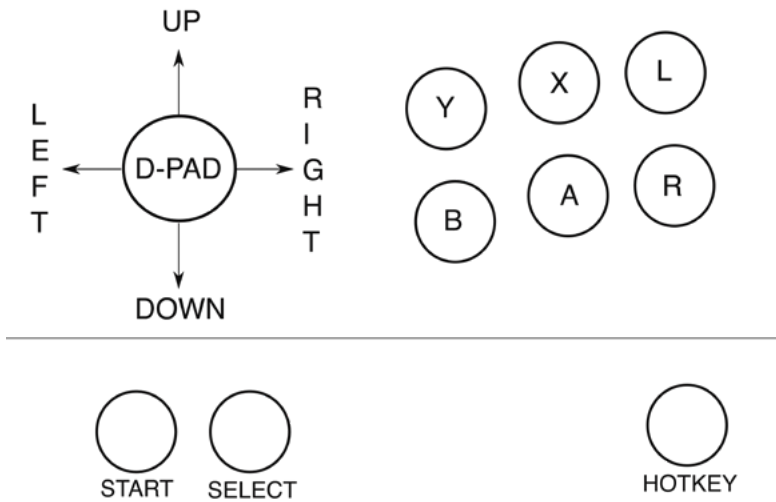
In this file, change PACKAGE_VERSION="$MKVERSION" to PACKAGE_VERSION="0.1.9". Press **CTRL+X** to exit, then **Y** to save.

Back at the command line, enter:

```
sudo dkms build -m mk_arcade_joystick_rpi
-v 0.1.9
sudo dkms install -m mk_arcade_joystick_rpi
-v 0.1.9

reboot
```

❝ We'll use the RetroPie distribution to provide extra features such as Steam Link support ❞

▲ Button and joystick correspondences for player controls; we recommend this configuration for use with RetroPie. L and R map to the right and left shoulder buttons

04 Optional: Load your hotkey driver

When Raspberry Pi has rebooted, SSH back in and type:

```
sudo modprobe mk_arcade_joystick_rpi map=1,2
```

Go over to the arcade machine and press **F4** to get to the command line and test your controllers:

```
jstest /dev/input/js0
jstest /dev/input/js1
```

If that works, it's time to load that module on boot. At the command line:

```
sudo nano /etc/modules
```

In this file, add the following on a new line, then save and exit.

```
mk_arcade_joystick_rpi
```

Next, at the command line:

```
sudo nano /etc/modprobe.d/mk_arcade_
joystick.conf
```

In this file, add the following:

```
options mk_arcade_joystick_rpi map=1,2
```

Now save, exit and reboot.

05 Configure RetroPie

There's a bit more configuration to do before RetroPie is ready to go. SSH in and type:

```
sudo ~/RetroPie-Setup/retropie_setup.sh
```

…to open the ncurses configuration menu.

If you did not manually install a hotkey version of the mk_arcade_joystick driver in the previous steps and do not need one, go to:

```
P manage packages
driver
819 mkarcadejoystick
```

…and install it.

If you need to connect any Bluetooth keyboards or controllers, go to:

```
C Configuration / tools
804 bluetooth
```

Press **R** to register a device and follow the pairing instructions.

832 samba in the configuration menu sets up Samba shares so you can easily transfer ROMs and BIOS images over your local network

You can add extra emulators here, but we'll come to that later. For now, select the R Perform reboot option from the main menu.

06 Input configuration

When RetroPie reboots, it should inform you that it can detect two GPIO controllers. Press and hold any button on the left-hand button bank to configure controls for player 1. Because arcade controls don't map perfectly to a gamepad, you'll have to skip some buttons by pressing and holding any key.

> 🟩 When RetroPie reboots, it should inform you that it can detect two GPIO controllers 🟩

Map up, down, left, and right on the arcade stick to the D-pad. Follow our button assignment diagram to map the top row to button Y, X, and L(eft shoulder), and the button below to buttons B, A, R(ight shoulder).

Map Start to player 1's left-hand front function button and Select to their right-hand front function button – this will be their 'insert coin'

button. In our wiring configuration, our single hotkey button – the last we set – is associated with player 1.

Approve your configuration, then set up player 2's controls in the same way.

07 Getting to know RetroPie

With your controllers configured, RetroPie's main interface will open. Press A to select menus and items and B to go back. Press Start to open the main menu and Select to open the options menu. Press the same button again to close each of these.

As you have yet to put any games on the system, only the RetroPie menu will be available. Here, you'll find easy access links to configuration tools, including some we used earlier. Install new emulators and drivers from the RetroPie Setup menu.

You'll probably need to disable overscan to get rid of a black border around the screen. In the ReotroPie menu, select Raspi-config > Display options > Underscan > No and then reboot to solve the problem. Note that button B is mapped to the **ENTER** key in this set of menus.

When you add any new games, ROMs or emulators, you'll have to restart EmulationStation by pressing Select, going to Quit, and then Restart EmulationStation.

08 Install more emulators

Although this is an arcade machine, you can play what you like on it. The core lr-mame2003 and lr-fbneo emulators are included, along with those for popular consoles

such as the Sega Mega Drive, used in some arcade systems and for which original games are legally available.

Some emulators require system BIOS images. Sadly, very few of these have been made legally available to emulation enthusiasts. SNK distributes a UniBIOS compatible BIOS set in its 40th Anniversary Collection. We recommend adding the following:

opt > 327 opentyrian – arcade-like DOS shoot-'em-up Tyrian 2.1 is now freeware.

exp > 241 lr-mame – a more up-to-date version of MAME that supports a wider range of ROMs. Install from source for bleeding edge.

exp > 307 digger – a sanctioned remaster of Windmill Software's Dig-Dug.

exp > 334 steamlink – this allows you to stream less emulation-friendly titles directly from a Steam installation on a Windows or Linux PC.

09 Configure your emulator

Once you've installed a new emulator, such as lr-mame, you'll have to configure the libretro back end to use it by default for either all games or selected titles. The easiest way to do this is to browse to the game you want to play in the EmulationStation front end.

Go to the Arcade menu, press B to start any game – it doesn't matter if it currently works or not – and then press B again when you're briefly prompted to 'press a button to configure'. Select option 1 to set the default emulator for arcade

---

Top Tip 👍

Steam Link smoothly

Use a wired Ethernet connection for optimal Steam Link game streaming.

► RetroPie is significantly more configurable than Recalbox, although its interface doesn't look quite as slick
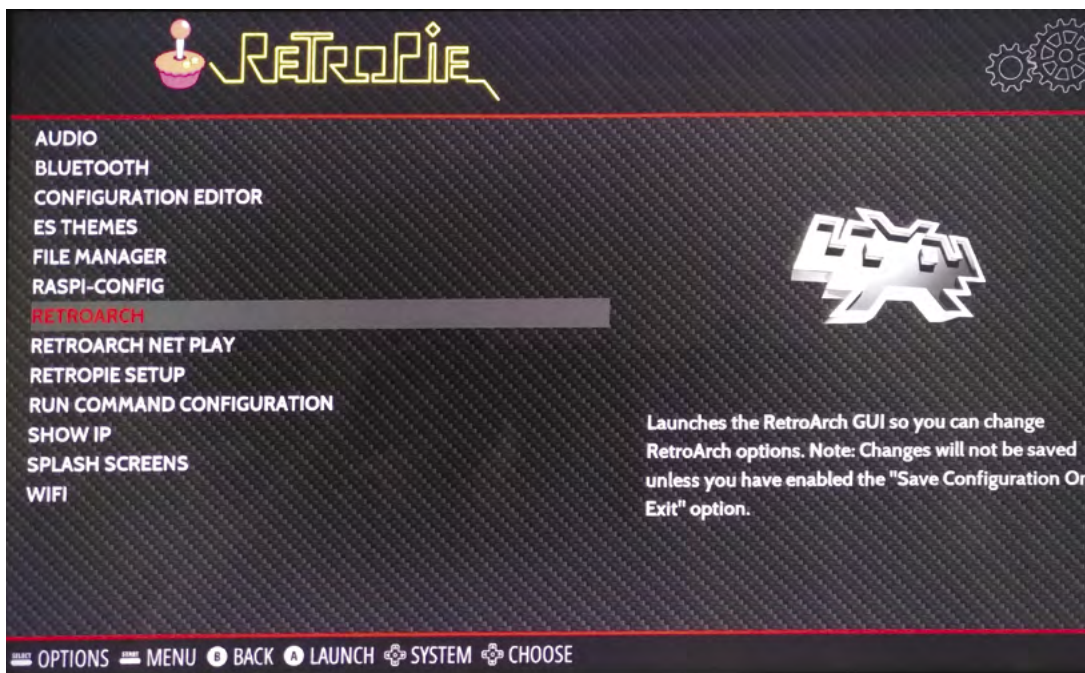
games and choose lr-mame. Option 2 allows you to select a different emulator for anything that doesn't work well with this.

10 Connect Steam Link

Linking Steam to your arcade cabinet lets you stream a wealth of modern and classic arcade games to Raspberry Pi from a more powerful PC, like Melty Blood, Guilty Gear, Horizon Chase Turbo, and Street Fighter V. After you've installed it and restarted EmulationStation, go to the Ports menu and select Steam Link.

It'll download updates – you will need a keyboard plugged in to approve these – and then run. Make sure Steam is running on a PC on your local network and that Enable Remote Play is ticked under Settings > Remote Play.

On the arcade machine, select the computer you want to link to. Steam Link will show a code. Enter this in your PC's Steam client when prompted. To avoid a resolution mismatch, run Steam with a monitor that matches the resolution of your arcade machine set as your primary display.

11 Configure Steam Link

You may want to reconfigure your controls, as Steam Link doesn't inherit the control layout from RetroPie's EmulationStation, and some games do better with alternative button assignments – for example, to more closely match

an Xbox or Steam Controller, which swaps the position of the B and A buttons.

To set these, launch Steam Link, press up to highlight the gear icon, press A (per our button assignment diagram), go right to highlight Controller and press A. Select the controller you wish to configure, then press down and right twice, and select Setup Controller.

Hit the button you want to associate with each Steam Controller-style button as it's displayed on screen. Use a keyboard or your second set of controls to use the skip button at the bottom to bypass extraneous buttons.

12 Why use Steam Link?

Steam Link is an invaluable tool for arcade emulation enthusiasts, not only because you can play more CPU-intensive games, but also because it's the best way of ensuring copyright compliance for a number of re-released arcade games.

We've been playing Ghouls 'N Ghosts from the Capcom Arcade Stadium on our cabinet via Steam Link. Unlike some SNK and Sega re-releases, Capcom doesn't supply emulator-ready ROM files and the EULA for that compilation doesn't allow you to extract its PAK files.

Neil Brown of decoded.legal opines (**magpi.cc/ romextractionlegal**) that "when even a legitimate Steam purchaser extracts the ROMs and runs them on their own Raspberry Pi, they infringe Capcom's copyright", making streaming these titles your best option for fully legal home arcade action. M

Code

Design
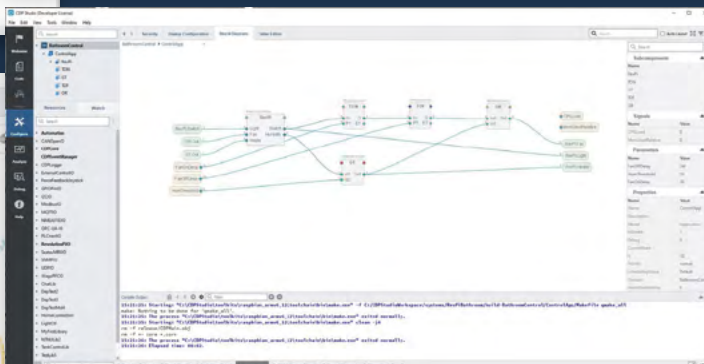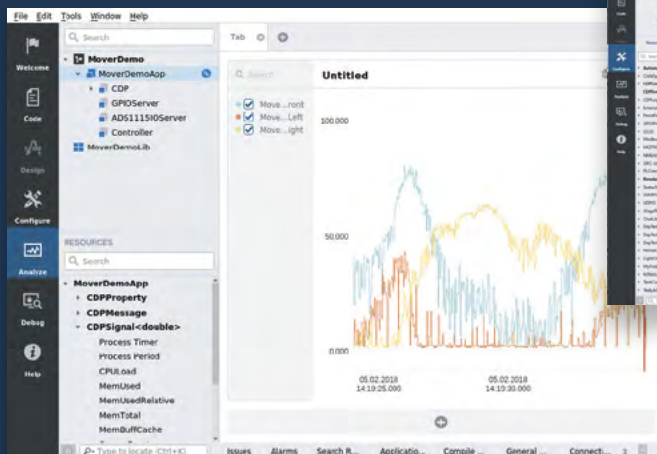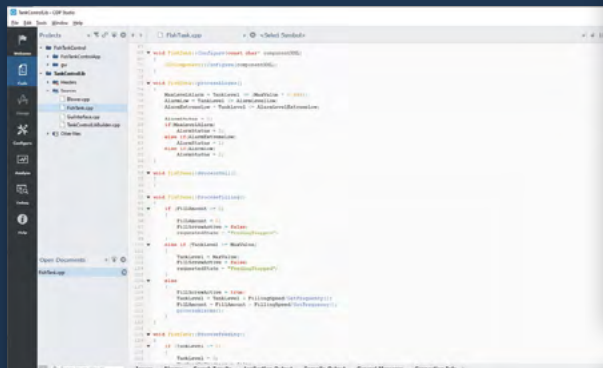
Configure

Analyze

PROFESSIONAL CONTROL SYSTEM DEVELOPMENT TOOL

Home projects made easy.

CDP Studio, a great software development tool for your home projects. Build systems for Raspberry Pi, use C++ or NoCode programming, open source libraries, out of the box support for GPIO, I2C, MQTT, OPC UA and more. Create beautiful user interfaces. Built for industrial control system development, **FREE for home projects**.

cdpstudio.com
Tel: +47 990 80 900 • info@cdptech.com
CDP Technologies AS // Hundsværgata 8, 6008 Ålesund, Norway

CDPStudio

Part 08

Create GUIs with Python:
Emoji Match

Create a fun picture-matching game

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

You are going to build an emoji picture-matching game (Figure 1). The object of the game is to spot the one emoji that appears in two different sets. You get a point for each correct match and lose a point for an incorrect match.

Loading emojis
To create the game, you will need some emojis. You can use the emojis created for Twitter (**twemoji.twitter.com**). Download the **emojis.zip** file from **magpi.cc/guizeroemojis**, open the zip file, and copy the **emojis** folder to the folder where you save your code.

The game will need to choose nine emojis at random and arrange them into a grid. A simple way to do this is to put all of the emojis into a list and randomly shuffle them.

Create a new program with the usual commented lines for different sections (Imports, Variables, Functions, App), as we've done in previous instalments. Under imports, add:

▲ **Figure 1** The finished game

```
import os
from random import shuffle
```

Then, under variables, enter this code which creates a shuffled list of emojis, each in the form **path/emoji_file_name**.

```
# set the path to the emoji folder on your
computer
emojis_dir = "emojis"
emojis = [os.path.join(emojis_dir, f) for f
in os.listdir(emojis_dir)]
shuffle(emojis)
```

The **emojis_dir** variable is the path of the emojis on your computer; it will tell the code that loads the emojis where to find them.

Test your program. Try printing the `emojis` list to the screen with `print(emojis)`. You should see a long list of file names. The list should be in a different order each time you run it.

Displaying the emojis
Next, the code needs to create two 3×3 grids of Picture and PushButton widgets which will show the emojis.

Modify your program to create a guizero app and a Box to hold the picture widgets using a `"grid"` layout. In the imports section, add this line to import the required widgets:

```
from guizero import App, Box
```

In the app section, add the following code:

```
app = App("emoji match")

pictures_box = Box(app, layout="grid")
```

The Box widget is really useful for laying out your GUI. It's an invisible area of your GUI where you can group widgets together. A Box can have its own

layout, size, and bg (background). They can also be hidden or shown, meaning you can easily make a collection of widgets invisible.

If you wish to see the Box, you can add a border by setting the parameter to True.

```
pictures_box = Box(app, layout="grid",
border=True)
```

Now, add the Picture widget to your imports:

```
from guizero import App, Box, Picture
```

In the app section, add in the code to create the Picture widgets and add them to a list.

```
pictures = []

for x in range(0,3):
    for y in range(0,3):
        picture = Picture(pictures_box,
grid=[x,y])
        pictures.append(picture)
```

To assign co-ordinates to each Picture widget, two for loops are used. They both run through the range 0–2; one assigns its value to the variable x and the other to the variable y. The grid position of each widget is set using the x and y values. The widgets are appended to a list so they can be referenced later in the game.

Do the same for PushButton widgets to create the second 3×3 grid. First, add the widget to your imports:

```
from guizero import App, Box, Picture,
PushButton
```

In the app section, add lines so it looks like this:

```
app = App("emoji match")

pictures_box = Box(app, layout="grid")
buttons_box = Box(app, layout="grid")

pictures = []
buttons = []

for x in range(0,3):
    for y in range(0,3):
        picture = Picture(pictures_box,
grid=[x,y])
        pictures.append(picture)
```

> ❝ The widgets are appended to a list so they can be referenced later in the game ❞

emoji1.py

DOWNLOAD THE FULL CODE:

magpi.cc/guizerocode

> Language: **Python 3**

```
001.   # ----------------------------
002.   # Imports
003.   # ----------------------------
004.
005.   import os
006.   from random import shuffle
007.   from guizero import App, Box, Picture, PushButton
008.
009.   # ----------------------------
010.   # Variables
011.   # ----------------------------
012.
013.   # set the path to the emoji folder on your computer
014.   emojis_dir = "emojis"
015.   emojis = [os.path.join(emojis_dir, f) for f in os.listdir(
       emojis_dir)]
016.   shuffle(emojis)
017.
018.   # ----------------------------
019.   # Functions
020.   # ----------------------------
021.
022.   def setup_round():
023.       for picture in pictures:
024.           picture.image = emojis.pop()
025.
026.       for button in buttons:
027.           button.image = emojis.pop()
028.
029.   # ----------------------------
030.   # App
031.   # ----------------------------
032.
033.   app = App("emoji match")
034.
035.   pictures_box = Box(app, layout="grid")
036.   buttons_box = Box(app, layout="grid")
037.
038.   pictures = []
039.   buttons = []
040.
041.   for x in range(0,3):
042.       for y in range(0,3):
043.           picture = Picture(pictures_box, grid=[x,y])
044.           pictures.append(picture)
045.
046.           button = PushButton(buttons_box, grid=[x,y])
047.           buttons.append(button)
048.
049.   setup_round()
050.
051.   app.display()
```

emoji2.py

> Language: **Python 3**

```
001.    # -----------------------------
002.    # Imports
003.    # -----------------------------
004.
005.    import os
006.    from random import shuffle, randint
007.    from guizero import App, Box, Picture, PushButton
008.
009.    # -----------------------------
010.    # Variables
011.    # -----------------------------
012.
013.    # set the path to the emoji folder on your computer
014.    emojis_dir = "emojis"
015.    emojis = [os.path.join(emojis_dir, f) for f in os.listdir(emojis_dir)]
016.    shuffle(emojis)
017.
018.    # -----------------------------
019.    # Functions
020.    # -----------------------------
021.
022.    def setup_round():
023.        for picture in pictures:
024.            picture.image = emojis.pop()
025.
026.        for button in buttons:
027.            button.image = emojis.pop()
028.
029.        matched_emoji = emojis.pop()
030.
031.        random_picture = randint(0,8)
032.        pictures[random_picture].image = matched_emoji
033.
034.        random_button = randint(0,8)
035.        buttons[random_button].image = matched_emoji
036.
037.    # -----------------------------
038.    # App
039.    # -----------------------------
040.
041.    app = App("emoji match")
042.
043.    pictures_box = Box(app, layout="grid")
044.    buttons_box = Box(app, layout="grid")
045.
046.    pictures = []
047.    buttons = []
048.
049.    for x in range(0,3):
050.        for y in range(0,3):
051.            picture = Picture(pictures_box, grid=[x,y])
052.            pictures.append(picture)
053.
054.            button = PushButton(buttons_box, grid=[x,y])
055.            buttons.append(button)
056.
057.    setup_round()
058.
059.    app.display()
```
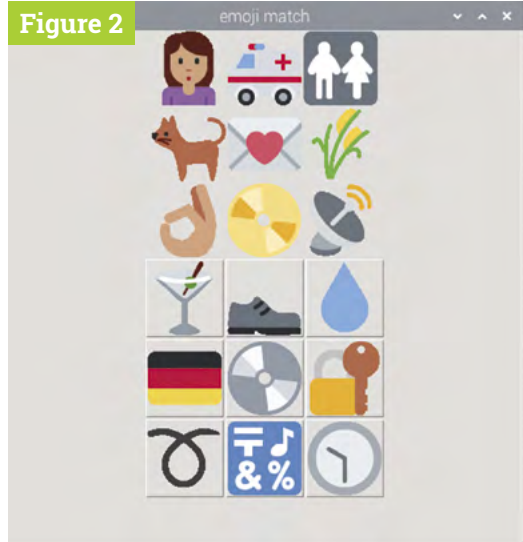

Figure 2

▲ **Figure 2** No matching emojis

```
        button = PushButton(buttons_box,
grid=[x,y])
        buttons.append(button)
```

In the functions section, create a function to set up each round of the game.

```
def setup_round():
    for picture in pictures:
        picture.image = emojis.pop()

    for button in buttons:
        button.image = emojis.pop()
```

To assign each **picture** and **button** widget an emoji, the **image** property is set to an item from the **emojis** list. Emojis are selected using **pop()**, which chooses the last item in a list and then removes it from the list. We've used this function because it will prevent any emoji appearing in the game more than once.

At the bottom of your program, call the **setup_round** function and display the app.

```
setup_round()

app.display()
```

Your program should now resemble **emoji1.py** (see previous page). Test it and you should see two grids of nine emojis.

Matching emojis

At the moment, all of the emojis in your app will be different (**Figure 2**). In the next step, you will pick another emoji to match, and update one picture and one button so they have the same matching emoji.

emoji3.py

> Language: **Python 3**

```
001.    # ----------------------------
002.    # Imports
003.    # ----------------------------
004.
005.    import os
006.    from random import shuffle, randint
007.    from guizero import App, Box, Picture, PushButton,
008.    Text
009.
010.    # ----------------------------
011.    # Variables
012.    # ----------------------------
013.
014.    # set the path to the emoji folder on your computer
015.    emojis_dir = "emojis"
016.    emojis = [os.path.join(emojis_dir, f) for f in
017.    os.listdir(emojis_dir)]
018.    shuffle(emojis)
019.
020.    # ----------------------------
021.    # Functions
022.    # ----------------------------
023.
024.    def setup_round():
025.        for picture in pictures:
026.            picture.image = emojis.pop()
027.
028.        for button in buttons:
029.            button.image = emojis.pop()
030.            button.update_command(
        match_emoji, args=[False])
031.
032.        matched_emoji = emojis.pop()
033.
034.        random_picture = randint(0,8)
035.        pictures[random_picture].image = matched_emoji
036.
037.        random_button = randint(0,8)
038.        buttons[random_button].image = matched_emoji
039.
040.        buttons[random_button].update_command(
        match_emoji, [True])
041.
042.    def match_emoji(matched):
043.        if matched:
044.            result.value = "correct"
045.        else:
046.            result.value = "incorrect"
047.
048.        setup_round()
049.
050.    # ----------------------------
051.    # App
052.    # ----------------------------
053.
054.    app = App("emoji match")
055.
056.    pictures_box = Box(app, layout="grid")
057.    buttons_box = Box(app, layout="grid")
058.
059.    pictures = []
060.    buttons = []
061.
062.    for x in range(0,3):
063.        for y in range(0,3):
064.            picture = Picture(pictures_box, grid=[x,y])
065.            pictures.append(picture)
066.
067.            button = PushButton(buttons_box, grid=[x,y])
068.            buttons.append(button)
069.
070.    result = Text(app)
071.
072.    setup_round()
073.
074.    app.display()
```

Add `randint` to your `random` import line. This is used to obtain a number from 0 to 8 for each picture and button.

```
from random import shuffle, randint
```

Then add this code (indented) to the bottom of the `setup_round` function to pop another emoji from the list and set it to be the image of a random picture and button.

```
matched_emoji = emojis.pop()

random_picture = randint(0,8)
pictures[random_picture].image = matched_
emoji

random_button = randint(0,8)
buttons[random_button].image = matched_
emoji
```

Your code should now look like **emoji2.py**. Run your program now; one of the emojis should match. Look carefully – the matching emoji can be hard to spot.

Check the guess

Each time one of the PushButtons is pressed, it will need to check if this is the matching emoji and put the result 'correct' or 'incorrect' on the screen. After the player's guess, a new round will be set up and different set of emojis displayed.

Your app will need a Text widget display the result. Add it to your imports:

```
from guizero import App, Box, Picture,
PushButton, Text
```

Add this line in your app section:

```
result = Text(app)
```

emoji4.py

> Language: **Python 3**

```python
001.    # ----------------------------
002.    # Imports
003.    # ----------------------------
004.
005.    import os
006.    from random import shuffle, randint
007.    from guizero import App, Box, Picture, PushButton, Text
008.
009.    # ----------------------------
010.    # Variables
011.    # ----------------------------
012.
013.    # set the path to the emoji folder on your computer
014.    emojis_dir = "emojis"
015.    emojis = [os.path.join(emojis_dir, f) for f in
016.    os.listdir(emojis_dir)]
017.    shuffle(emojis)
018.
019.    # ----------------------------
020.    # Functions
021.    # ----------------------------
022.
023.    def setup_round():
024.        for picture in pictures:
025.            picture.image = emojis.pop()
026.
027.        for button in buttons:
028.            button.image = emojis.pop()
029.            button.update_command(match_emoji, args=[False])
030.
031.        matched_emoji = emojis.pop()
032.
033.        random_picture = randint(0,8)
034.        pictures[random_picture].image = matched_emoji
035.
036.        random_button = randint(0,8)
037.        buttons[random_button].image = matched_emoji
038.
039.        buttons[random_button].update_command(
        match_emoji, [True])
040.
041.    def match_emoji(matched):
042.        if matched:
043.            result.value = "correct"
044.            score.value = int(score.value) + 1
045.        else:

046.            result.value = "incorrect"
047.            score.value = int(score.value) - 1
048.
049.        setup_round()
050.
051.    def reduce_time():
052.        timer.value = int(timer.value) - 1
053.        # is it game over?
054.        if int(timer.value) < 0:
055.            result.value = "Game over! Score = " + score.value
056.            # hide the game
057.            pictures_box.hide()
058.            buttons_box.hide()
059.            timer.hide()
060.            score.hide()
061.
062.    # ----------------------------
063.    # App
064.    # ----------------------------
065.
066.    app = App("emoji match")
067.
068.    score = Text(app, text="0")
069.    timer = Text(app, text="30")
070.
071.    pictures_box = Box(app, layout="grid")
072.    buttons_box = Box(app, layout="grid")
073.
074.    pictures = []
075.    buttons = []
076.
077.    for x in range(0,3):
078.        for y in range(0,3):
079.            picture = Picture(pictures_box, grid=[x,y])
080.            pictures.append(picture)
081.
082.            button = PushButton(buttons_box, grid=[x,y])
083.            buttons.append(button)
084.
085.    result = Text(app)
086.
087.    setup_round()
088.
089.    app.repeat(1000, reduce_time)
090.
091.    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
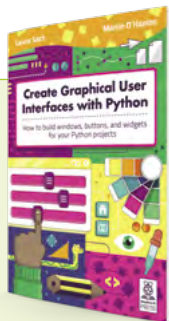
Create a new function which will be called when one of the emoji buttons is pressed. It will display 'correct' or 'incorrect' and call **setup_round** to create the next set of emojis.

```python
def match_emoji(matched):
    if matched:
        result.value = "correct"
    else:
        result.value = "incorrect"

    setup_round()
```

Using other images

The emoji match game uses picture buttons to allow the user to pick which emoji matches. You can make any PushButton widget into a picture button by setting the image parameter; for example:

```
button = PushButton(app, image="my_picture.gif")
```

The button will scale to fit the size of your image. The type of image you can use is determined by your operating system and how you installed guizero, although any setup will support GIF images. To find the image file types supported by your setup, you can run:

```
from guizero import system_config
print(system_config.supported_image_types)
```

You can find out more about image support in guizero at **lawsie.github.io/guizero/images**.

The incorrect emoji buttons will pass False to the `match_emoji` function; the matching emoji will pass True.

Update the `setup_round` function so that all the 'incorrect' buttons call the `match_emoji` function.

```
    for button in buttons:
        button.image = emojis.pop()
        button.update_command(match_emoji,
args=[False])
```

The `update_command` method sets the function which will be called when the button is pressed. The `args` list `[False]` will be used as the parameters to the `match_emoji` function.

Finally, update the command for the matching button so it calls `match_emoji`, but this time passes True as the argument.

```
    buttons[random_button].update_
command(match_emoji, [True])
```

Your code should now resemble **emoji3.py** (previous page). Play the game. In each round there will be a matching emoji – press the matching picture button. Did you get it right?

Adding a score and timer

At the moment, the game continues forever (or until you run out of emojis in the list). Add a score and a timer which counts down to the end of the game to give a challenge.

In the app section, create two Text widgets to show the score and the timer.

Figure 3 With box for score and timer

```
score = Text(app, text="0")
timer = Text(app, text="30")
```

The timer is set to `"30"`, which will be the number of seconds in each round.

Modify the `match_emoji` function to either add or subtract 1 to/from the player's score.

```
def match_emoji(matched):
    if matched:
        result.value = "correct"
        score.value = int(score.value) + 1
    else:
        result.value = "incorrect"
        score.value = int(score.value) - 1
```

To create the timer, you will use a feature of guizero which allows you to ask the application to continuously call a function every 1 second.

Create a function which will reduce the value of the timer by 1.

```
def reduce_time():
    timer.value = int(timer.value) - 1
```

Before the app is displayed, use the `app.repeat()` function to call the `reduce_time` function every second (1000 milliseconds).

```
app.repeat(1000, reduce_time)

app.display()
```

Running your game now, you will notice that the timer counts down from 30. Unfortunately, it will continue counting down past 0 and never stop.

Update the `reduce_time` function to check if the timer is less than zero and then stop the game.

09-emoji-match.py

> Language: **Python 3**

```python
001.   # ----------------------------
002.   # Imports
003.   # ----------------------------
004.
005.   import os
006.   from random import shuffle, randint
007.   from guizero import App, Box, Picture,  PushButton, Text
008.
009.   # ----------------------------
010.   # Variables
011.   # ----------------------------
012.
013.   # set the path to the emoji folder on your computer
014.   emojis_dir = "emojis"
015.   emojis = [os.path.join(emojis_dir, f) for f in
016.   os.listdir(emojis_dir)]
017.   shuffle(emojis)
018.
019.   # ----------------------------
020.   # Functions
021.   # ----------------------------
022.
023.   def setup_round():
024.       for picture in pictures:
025.           picture.image = emojis.pop()
026.
027.       for button in buttons:
028.           button.image = emojis.pop()
029.           button.update_command(match_emoji, args=[False])
030.
031.       matched_emoji = emojis.pop()
032.
033.       random_picture = randint(0,8)
034.       pictures[random_picture].image = matched_emoji
035.
036.       random_button = randint(0,8)
037.       buttons[random_button].image = matched_emoji
038.
039.       buttons[random_button].update_command(
       match_emoji, [True])
040.
041.   def match_emoji(matched):
042.       if matched:
043.           result.value = "correct"
044.           score.value = int(score.value) + 1
045.       else:
046.           result.value = "incorrect"
047.           score.value = int(score.value) - 1
048.
049.       setup_round()
050.
051.   def reduce_time():
052.       timer.value = int(timer.value) - 1
053.       # is it game over?
054.       if int(timer.value) < 0:
055.           result.value = "Game over! Score = " + score.value
056.           # hide the game
057.           game_box.hide()
058.
059.   # ----------------------------
060.   # App
061.   # ----------------------------
062.
063.   app = App("emoji match")
064.
065.   game_box = Box(app, align="top")
066.
067.   top_box = Box(game_box, align="top", width="fill")
068.   Text(top_box, align="left", text="Score ")
069.   score = Text(top_box, text="4", align="left")
070.   timer = Text(top_box, text="30", align="right")
071.   Text(top_box, text="Time", align="right")
072.
073.   pictures_box = Box(game_box, layout="grid")
074.   buttons_box = Box(game_box, layout="grid")
075.
076.   pictures = []
077.   buttons = []
078.
079.   for x in range(0,3):
080.       for y in range(0,3):
081.           picture = Picture(pictures_box, grid=[x,y])
082.           pictures.append(picture)
083.
084.           button = PushButton(buttons_box, grid=[x,y])
085.           buttons.append(button)
086.
087.   result = Text(app)
088.
089.   setup_round()
090.
091.   app.repeat(1000, reduce_time)
092.
093.   app.display()
```

```python
def reduce_time():
    timer.value = int(timer.value) - 1
    # is it game over?
    if int(timer.value) < 0:
        result.value = "Game over! Score = 
" + score.value
        # hide the game
        pictures_box.hide()
        buttons_box.hide()
        timer.hide()
        score.hide()
```

When the timer is less than 0, the message 'game over' is displayed and the game's widgets are hidden so the user can no longer play.

See **emoji4.py** to get an idea of how your code should now look. Run it and play the emoji match game. Challenge a friend or family member to a game.

You may want to put the score and timer widgets into a Box so they can be laid out better (**Figure 3**, previous page) – see the complete **09-emoji-match.py** listing for how to do this. **M**

CREATE FANTASTIC ELECTRONIC PROJECTS WITH
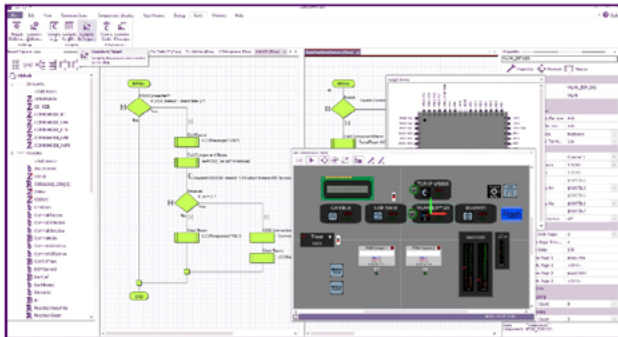
FLOWCODE

There are two parts to Flowcode:

Flowcode Embedded - Allows you to create advanced microcontroller projects using graphical programming.

Flowcode App Developer - Allows you to create fantastic looking HMI's with low cost hardware for your Windows PC or tablet.

EMBEDDED

APP DEVELOPER

- Create highly functional microcontroller projects using RPi, Arduino, PIC, ESP, ARM and AVR processors

- Graphical programming: use Flowcharts, state diagrams and data flow techniques

- Full simulation - electrical and mechanical

- Huge library of parts

- Create great Windows Human Machine interfaces for control and data gathering using RPi, Arduino, PIC, ESP

- Graphical programming: use Flowcharts, state diagrams and data flow techniques

- Comprehensive library of dials, switches, indicators, graphs and other components

Compatible — SparkFun Pro Micro RP2040 — SparkFun Thing Plus RP2040 — SparkFun MicroMod RP2040 Processor

Custom USB games controllers with Pico

Play games the way you want to

Ben Everard

🐦 @ben_everard

Ben's house is slowly being taken over by 3D printers. He plans to solve this by printing an extension, once he gets enough printers.

G ames controllers – like keyboards – are very personal things. What works for one person may not work for another. Why, then, should we all use almost identical off-the-shelf controllers? Let's take a look at how to use Raspberry Pi Pico to create a controller that's just right for you.

We'll use CircuitPython for this as it has excellent support for USB interfaces. The sort of USB devices that we interact with are called human interface devices (HIDs), and there are standard protocols for common HIDs, including keyboards and mice. This is why, for example, you can plug almost any USB keyboard into almost any computer and it will just work, with no need to install drivers.

We'll be using the Keyboard type, as that works best with the sorts of games that this author likes to play, but you can use exactly the same technique to simulate a mouse or a gamepad.

Before we get onto this, though, let's take a look at the buttons and how to wire them up.

We're going to use eight buttons: four for direction, and four as additional 'action' buttons. We'll connect these between an I/O pin and ground. You can use any I/O pin you like. We're going to use slightly different ones in two different setups, just because they made sense with the physical layout of the hardware. Let's take a look at the hardware we're using. Remember, this is just the hardware we want to use. The whole idea of this is to create a setup that's right for you, so there's no need to use the same. Think about how you want to interact with your games and take a look at the available input devices and build what you want.

The first setup we're creating is an Arcade box. This author would really like an arcade machine in his house. However, space limitations mean that this isn't going to be possible in the near future. The first setup, then, is an attempt to recreate the control setup of an arcade machine, but use it to play games on a laptop rather than a full-sized cabinet.

Arcade controls are quite standard, and you can get them from a range of sources. We used one of Pimoroni's Arcade Parts sets, which includes a joystick and ten buttons (we only used four of these). The important thing about the joystick you pick is that it's a button-based joystick and not an analogue one (sometimes called a dual-axis joystick), as the latter won't work with a keyboard interface. If you want to use an analogue joystick, you'll need to switch the code around to use a mouse or gamepad as an input device.

As well as the electronics, you'll need some way of mounting them. We used a wooden craft box. These are available for about £10 from a range of online or bricks and mortar stores. You can use anything that is strong enough to hold the components.

The second setup we're using is a much simpler button-based system on breadboard-compatible tactile buttons and protoboard. It's smaller, cheaper, and quicker to put together. The protoboard holds everything together, so there's nothing extra to add unless you want to. You can personalise it by selecting different-sized buttons, changing the layout, or building a larger chassis around this.

INSERT COIN TO CONTINUE

Let's take a look at the arcade setup first. The joystick has five pins. One is a common ground and the others are up, down, left, and right. When you push the joystick up, a switch closes, linking ground to the up pin. On our joystick the outermost pin is ground, but it's worth checking on your joystick which pin is which by using a multimeter. Select continuity mode and, if you push the joystick up, you should find a continuous

> **The joystick has five pins. One is a common ground** and the others are up, down, left, and right

connection between the up pin and ground. A bit of experimentation should confirm which pin is which.

In order to read the pins, we just need to connect the directional output from the joystick to an I/O pin on Pico. We can use one of Pico's internal pull-up resistors to pull the pin high when the button isn't pressed. Then, when the button is pressed, it will connect to ground and read low. The joystick should come with a cable that slots onto the joystick. This should have five outputs, and this conveniently slots into the I/O outputs of Pico with a ground on one end.

The buttons, similarly, just need to be connected between ground and an I/O pin. These came with cables that pushed onto the button and plugged into

adjacent pins. Since Pico has eight grounds available, there are enough that each button can have its own ground, and you don't have to mess around joining cables together.

Once all the cables are soldered together, it's just a case of building the chassis. For this, you need five large holes (one for the joystick and four for the buttons). We didn't have an appropriately sized drill bit and, given how soft the wood on these boxes is, a large drill bit may have split the wood anyway. Instead, we drilled a 20 mm hole and then used a rotary tool with sanding attachment to enlarge the hole until it was the right size. You have to go quite easy with both the drill and the sanding tool to avoid →
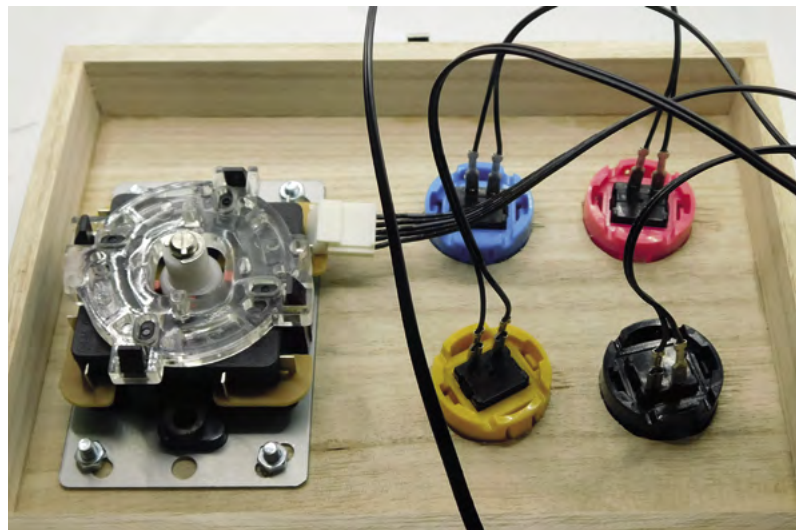
Above
Gaming like it's 1989

Below
The connectors should just push onto the buttons and joysticks
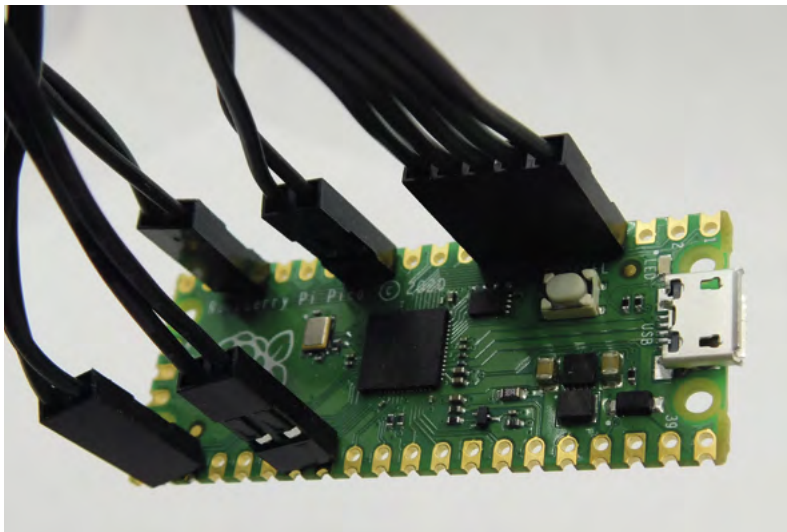
⚠️

Warning!
Soldering iron

Be careful when using a hot soldering iron and read Raspberry Pi's Getting started with soldering guide.

magpi.cc/soldering

turning everything into shards of broken wood. Four small holes then allow bolts to keep the joystick in place (we used M5 bolts). The buttons just push into place.

The only remaining thing was a 12 mm hole for a micro USB cable to pass through to Pico. If you don't have a 12 mm drill bit, two overlapping smaller holes may work if you're careful.

The buttons just push-fit into place, and that's everything ready to go.

A SMALLER APPROACH

Our smaller option used protoboard over the back of Pico. Since we didn't want to block the BOOTSEL button, we only soldered it over part of Pico. However, before soldering it on at all, we soldered the buttons in place.

Tactile switches typically have four connections. Well, really they have two connections, but each connection has two tabs that fit into the protoboard. This means that you have to orientate them correctly. Again, your multimeter's continuity function will confirm which pins are connected and which are switched.

Protoboard is a PCB that contains lots and lots of holes and nothing else. You solder your components into the holes and then you have to create connections between them.

We placed the buttons in the protoboard in positions we liked before worrying about the wiring. First, we looked to connect one side of each switch to ground. To minimise the wiring, we did this in two groups. We connected one side of each of the direction buttons together and then linked them to ground. Then we did the same to all the action buttons.

There are two ways of connecting things on protoboard. One is to use jumper wire. This works well if the points are more than a couple of holes apart. For holes that are next to each other, or very close, you can bridge them. On some protoboard (which doesn't have a solder mask), you might simply be able to drag a blob of solder across with your soldering iron so that it joins both holes. On protoboard with solder mask, this doesn't work quite so well, so you need to add a little strand of wire in a surface-mount position between the two points and solder it in. If you've got a pair of tweezers to hold the wire in place while you solder it, it will be much easier.

For longer connections, you'll need to use jumper wire. Sometimes you'll be able to poke it through

> **We placed the buttons in the protoboard in positions** we liked before worrying about the wiring

the protoboard and use the leg to join. Other times you'll have to surface-mount it. This all sounds a bit complicated, but while it can be a bit fiddly, it's all fairly straightforward once you put solder to iron.

PROGRAM IT UP

Now that we've got the hardware ready, let's code it up. You'll first need to load CircuitPython onto your Pico. You can download the latest release from **circuitpython.org**. Press the BOOTSEL button as you plug Pico into your USB port, and then drag and drop the downloaded UF2 file onto the RP2 USB drive that should appear.
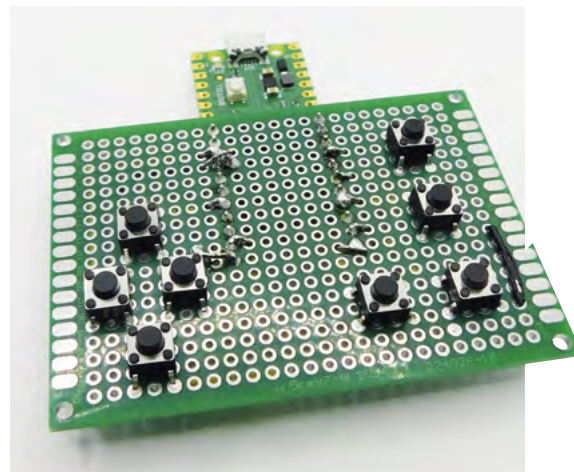
We'll use Mu to program Pico. If you've not used CircuitPython before, it's probably worth having a quick look through the 'getting started' guide here: **hsmag.cc/CircuitPythonGuide**.

The code to run our games controller is:

DOWNLOAD THE FULL CODE:
magpi.cc/picocontroller

```python
import board
import digitalio
import gamepad
import time
import usb_hid
from adafruit_hid.keyboard import Keyboard
from adafruit_hid.keycode import Keycode

kbd = Keyboard(usb_hid.devices)

keycodes = [Keycode.UP_ARROW, Keycode.DOWN_ARROW,
Keycode.LEFT_ARROW, Keycode.RIGHT_ARROW,
    Keycode.X, Keycode.Z, Keycode.SPACE, Keycode.
ENTER]

pad = gamepad.GamePad(
    digitalio.DigitalInOut(board.GP12),
    digitalio.DigitalInOut(board.GP14),
    digitalio.DigitalInOut(board.GP9),
    digitalio.DigitalInOut(board.GP15),
    digitalio.DigitalInOut(board.GP16),
    digitalio.DigitalInOut(board.GP17),
    digitalio.DigitalInOut(board.GP18),
    digitalio.DigitalInOut(board.GP20),
)
last_pressed = 0
while True:
    this_pressed = pad.get_pressed()
    if (this_pressed != last_pressed):
        for i in range(8):
            if (this_pressed & 1<<i) and not
(last_pressed & 1<<i):
                kbd.press(keycodes[i])
            if (last_pressed & 1<<i) and not
(this_pressed & 1<<i):
                kbd.release(keycodes[i])
        last_pressed = this_pressed
    time.sleep(0.01)
```

This uses the HID keyboard object (called **kbd**) to send key press and release events for different key codes depending on what buttons are pressed or released. We've used the gamepad module that is for keeping track of up to eight buttons. When you initialise it, it will automatically add pull-up resistors and set the I/O pins to input. Then, it will keep track of what buttons are pressed. When you call **get_pressed()**, it will return a byte of data where each digit corresponds to an I/O pin. So, the following number (in binary) means that the first and third buttons have been pressed: 00000101. This is a little confusing, because this is the opposite order to how the I/Os are passed when you initialise the GamePad object.

The **while** loop may look a little unusual as it's not particularly common to use this sort of binary comparison in Python code, but in essence, it's just looking at one bit at a time and seeing either: it's now pressed but wasn't last time the loop ran (in which case, it's a new button press and we should send it to the computer), or it isn't pressed this loop but was the previous loop (in which case, it's newly released so we can call the release method).

The << operator shifts a value by a number of bits to the left. So, 1<<2 is 100, and 1<<3 is 1000. The & operator is bitwise and so it looks at a binary number and does a logical AND on each bit in turn. Since the right-hand side of the & is all zeros apart from one bit (at a different position depending on the value of **i**), the result will be dependent on whether the value of **this_pressed** or **last_pressed** is 1 or 0 at the position **i**. When you have an **if** condition that's a number, it's true if the number is anything other than 0. So, **(this_pressed & 1<<2)** will evaluate to true if there's a 1 at position 2 in the binary form of **this_pressed**. In our case, that means if the joystick is pushed left.

You can grab this code from the following link – **hsmag.cc/USBKeyboard**. Obviously, you will need to update the GPIO values to the correct ones for your setup when you initialise GamePad.

We've taken a look at two ways to build a gamepad, but it's up to you how you want to design yours. ◻

Above ◈
With a combination of small sections of wire and jumpers, you can create whatever pattern of wiring you like on protoboard

Raspberry Pi Pico data logger

Turn Raspberry Pi Pico into a temperature data-logging device and untether it from the computer to make it fully portable
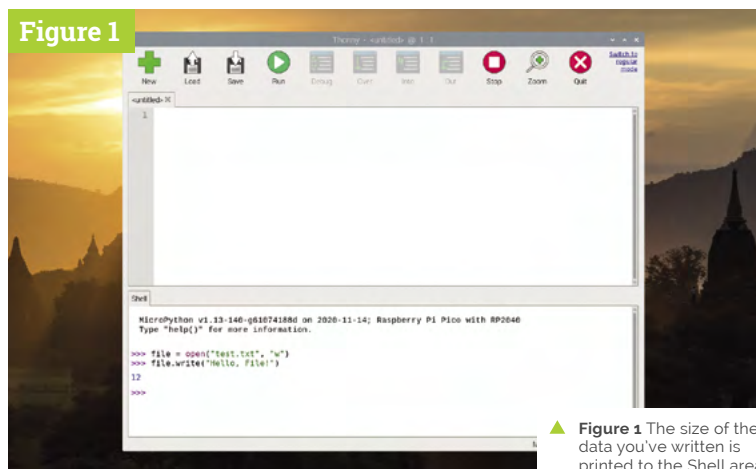
▲ **Figure 1** The size of the data you've written is printed to the Shell area

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

Throughout our Raspberry Pi Pico tutorials, you've been using your Pico connected to a Raspberry Pi or other computer via its micro USB port. Like all microcontrollers, though, there's no reason your Pico has to be tethered in this way: it's a fully functional self-contained system, with processing capabilities, memory, and everything it needs to work on its own.

In this guide you'll learn how to use the file system to create, write to, and read from files, allowing you to put your Pico anywhere you like and have it record data for later access – turning it into what is known as a data logger. For this you'll only need your Pico and, if you want to use it away from your Raspberry Pi, a micro USB charger or battery pack; once you have finished the tutorial, you can connect additional analogue sensors if you want to expand your project.

The file system

The file system is where your Pico stores all the programs you've been writing. It's equivalent in function to the microSD card in your Raspberry Pi, or the hard drive or solid-state drive in your laptop or desktop computer: it's a form of non-volatile storage, which means that whatever you save there stays in place even when you unplug your Pico's micro USB cable.

Connect your Pico to your Raspberry Pi and load Thonny, if you don't already have it open. Click the Open icon, and click 'Raspberry Pi Pico' in the pop-up which appears. You'll see a list of all the programs you've been writing so far, stored on your Pico's file system. You're not going to open one right now, so click Cancel.

Click at the bottom of the Shell area to start working with your Pico in interactive mode. Type:

```
file = open("test.txt", "w")
```

This tells MicroPython to open a file called **test.txt** for writing – the **"w"** part of the instruction. You won't see anything print to the Shell area when you press **ENTER** at the end of the line, because although you've opened the file, you haven't done anything with it yet. Type:

```
file.write("Hello, File!")
```

When you press **ENTER** at the end of this line, you'll see the number 12 appear (**Figure 1**). That's MicroPython confirming to you that it has written 12 bytes to the file you opened. Count the number of characters in the message you wrote: including the letters, comma, space, and exclamation mark, there are twelve – each of which takes up a single byte.

When you've written to a file, you need to close it – this ensures that the data you've told MicroPython to write is actually written to the file system. If you don't close the file, the data might not have been written yet – a bit like writing a letter in LibreOffice Writer or another word processor and forgetting to save it. Type:

```
file.close()
```

Your file is now safely stored on your Pico's file system. Click the Open icon on Thonny's toolbar, click 'Raspberry Pi Pico', and scroll through the list of files until you find **test.txt**. Click on it, then click OK to open it: you'll see your message pop up in a new Thonny tab.

You don't have to use the Open icon to read files, though: you can do it directly in MicroPython itself. Click back into the bottom of the Shell area and type:

```
file = open("test.txt")
```

You'll notice that this time around there's no `"w"`: that's because instead of writing to the file, you're going to be reading it. You could replace the `"w"` with an `"r"`, but MicroPython defaults to opening a file in read mode – so it's fine to simply leave that part of the instruction off.

Next, type:

```
file.read()
```

You'll see the message you wrote to the file print to the Shell area (**Figure 2**). Congratulations: you can read and write files on your Pico's file system!

Before you finish, remember to close the file – it's not as important to properly close a file after reading it as it is when writing to it, but it's a good habit to get into anyway:

```
file.close()
```

Logging temperatures

Now you know how to open, write to, and read from files, you have everything you need to build a data logger on your Pico. Click the New icon to start a new program in Thonny, and start your program by typing:

```
import machine
import utime

sensor_temp = machine.ADC(machine.ADC.
CORE_TEMP)

conversion_factor = 3.3 / (65535)
reading = sensor_temp.read_u16() *
conversion_factor
temperature = 27 - (reading -
0.706)/0.001721
```

You might recognise this code: it's the same as you used in *The MagPi* #107 (**magpi.cc/107**) to read

from your Pico's on-board temperature sensor. The readings from the sensor are the data you're going to be logging to the file system, so you don't want to simply print them out as you did before.

Start by opening a file for writing by adding the following line at the bottom:

```
file = open("temps.txt", "w")
```

If the file doesn't already exist on the file system, this creates it; if it does, it overwrites it – emptying its contents ready for you to write new data.

Now you need to write something to the file – the reading from the temperature sensor:

```
file.write(str(temperature))
```

Rather than writing a fixed string in quotes, as you did before, this time you're converting the variable temperature – which is a floating-point number, in other words one with a decimal point in it – to a string, then writing that to the file.

As before, to make sure the data is written you need to close the file:

❝ Now write something to the file – the reading from the temperature sensor ❞

Warning!

Opening a file for writing in MicroPython will delete anything you've already stored in it. Always make sure you've opened the file for reading and saved the contents somewhere if you want to keep it!

▼ **Figure 2** Printing the stored message to the Shell area
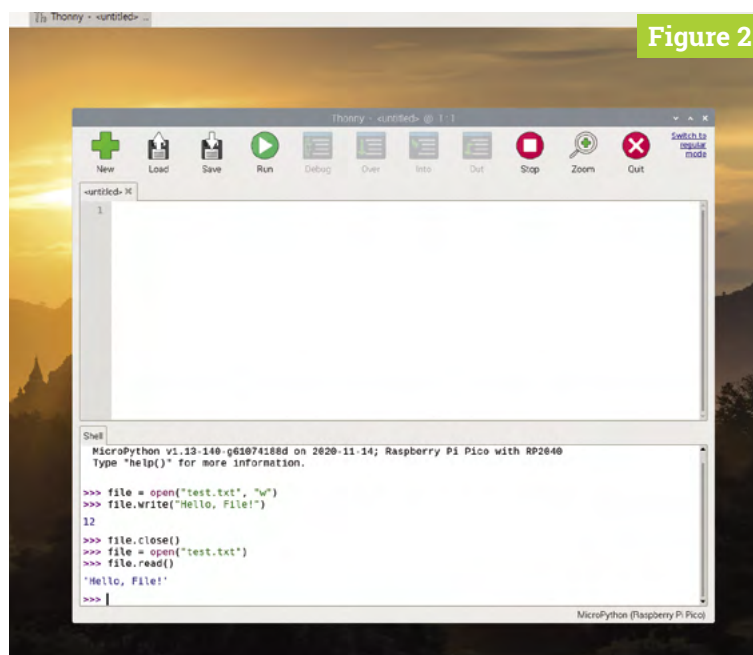
Figure 2

Figure 3
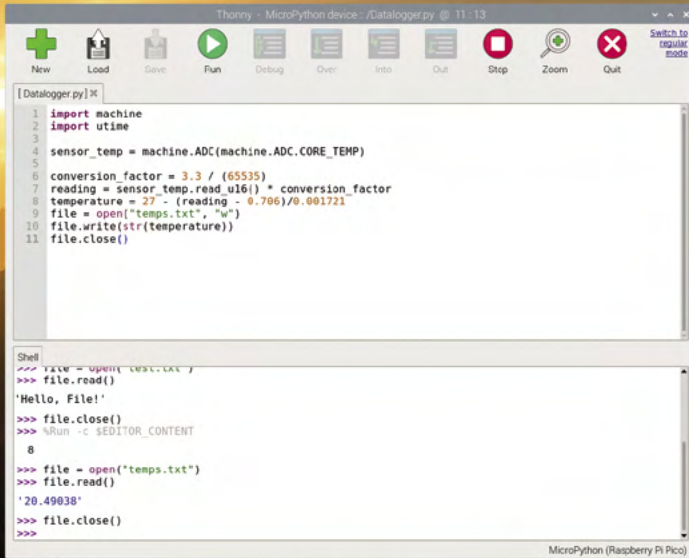
▲ **Figure 3** Your file records the temperature at the time the measurement was taken

▼ **Figure 4** All the measurements are there, but the formatting makes it difficult to read

Figure 4
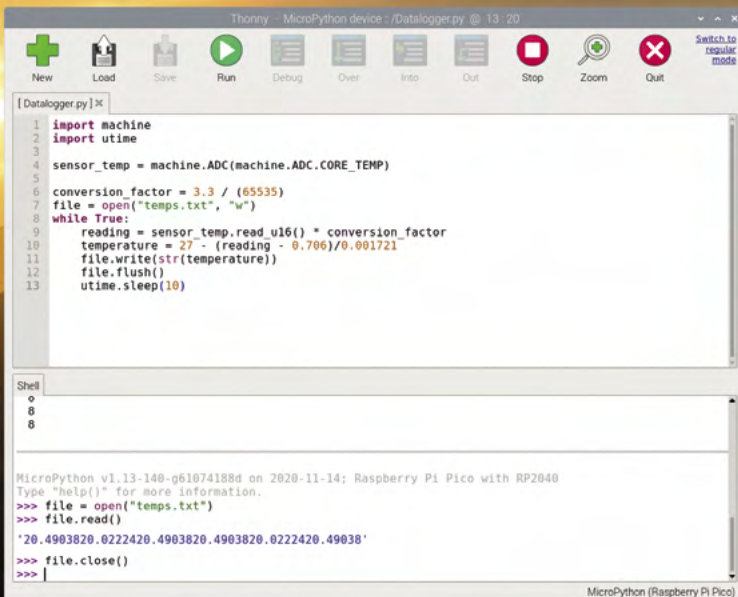
```
file.close()
```

Click the Run icon and save your program to the Raspberry Pi Pico as **Datalogger.py**. The program will only take a few seconds to run; when the >>> prompt reappears at the bottom of the Shell area, click into it and type the following to open and read your new file:

```
file = open("temps.txt")
file.read()
file.close()
```

You'll see the temperature reading your program took appear in the Shell (**Figure 3**). Congratulations: your data logger works!

A data logger that only logs a single reading – a datum – isn't that useful, though. To make your data logger more powerful, you need to modify it so it takes lots of readings. Click Run again, and read the file again:.

```
file = open("temps.txt")
file.read()
file.close()
```

Notice how there's still only one reading in the file. When your program opened the file for writing again, it automatically wiped its previous contents – meaning that each time your program runs, it will wipe the file and store a single reading.

To fix that, you need to modify your program. Start by clicking and dragging your mouse cursor to highlight the lines:

```
reading = sensor_temp.read_u16() *
conversion_factor
temperature = 27 - (reading -
0.706)/0.001721
```

When you've highlighted both lines – making sure not to miss any parts – let go of the mouse button and press **CTRL+X** on your keyboard to cut the lines; you'll see them disappear from the program.

Next, go to the bottom of your program and delete everything after:

```
file = open("temps.txt", "w")
```

❝ To make your data logger more powerful, you need to modify it ❞

Now type:

```
while True:
```

After pressing **ENTER** at the end of that line, hold down the **CTRL** key and press the **V** key to paste the two lines you cut earlier. You'll see them appear, which saves you having to type them in – but only the first line will be properly indented so it's nested as part of the infinite loop you just opened. Put your cursor at the start of the line below by clicking and press the **SPACE** bar four times to indent the line properly, then move your cursor to the end of the line and press **ENTER**.

Type the following line, making sure it's properly indented:

```
file.write(str(temperature)
```

Now, though, you're going to need to do something new. If you close the file as you did before, you won't be able to write to it again without reopening it and wiping its contents. If you don't close the file, the data will never actually get written to the file system.

The solution: flush the file, rather than close it. Type:

```
file.flush()
```

When you're writing to a file but the data isn't actually being written to the file system, it's stored in what's known as a buffer – a temporary storage area. When you close the file, the buffer is written to the file in a process known as flushing. Using `file.flush()` is equivalent to `file.close()`, in that it flushes the contents of the buffer into the file – but unlike `file.close()`, the file remains open for you to write more data to it later.

Now you just need to pause your program between readings:

```
utime.sleep(10)
```

Your finished program will look like this:

```
import machine
import utime

sensor_temp = machine.ADC(machine.ADC.
CORE_TEMP)

conversion_factor = 3.3 / (65535)
file = open("temps.txt", "w")
```

```
while True:
    reading = sensor_temp.read_u16() *
conversion_factor
    temperature = 27 - (reading -
0.706)/0.001721
    file.write(str(temperature))
    file.flush()
    utime.sleep(10)
```

Click the Run icon and count to 60, then click the Stop icon. Now open and read your file in the script area:

```
file = open("temps.txt")
file.read()
file.close()
```

The good news is that your program has worked, and you've recorded multiple readings in a single file – around six, give or take a few depending on how quickly you counted. The bad news is that they're all mushed together on a single line (**Figure 4**) – making it difficult to read. To fix that problem, you need to format the data as it's written to the file. Go back to the `file.write()` line in your program, and modify it so it looks like: `file.write(str(temperature) + "\n")`.

The plus symbol (**+**) tells MicroPython that you want to append what follows, concatenating the two strings together; `"\n"` is a special string known

▼ **Figure 5** You're getting closer to an easy-to-read printout here

Figure 5
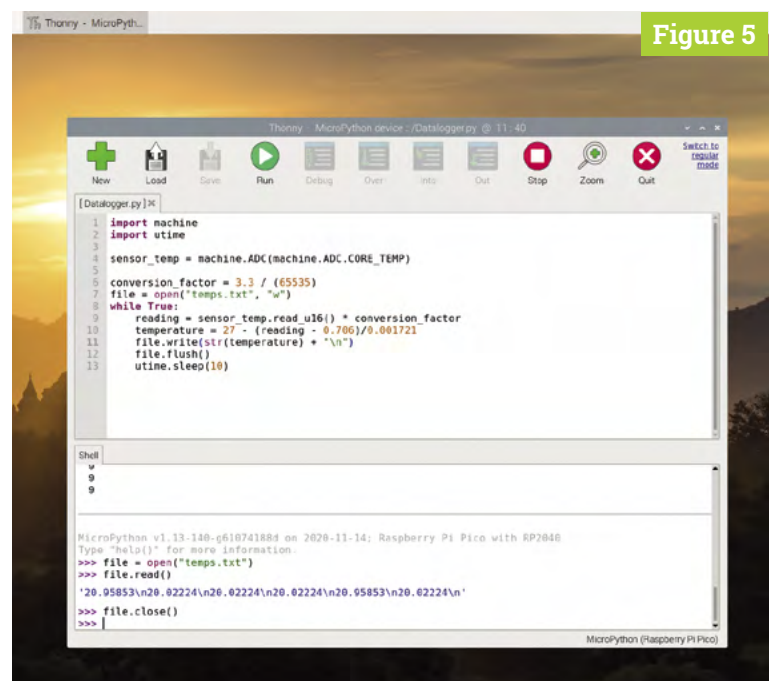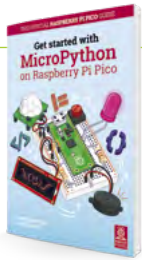
Get Started with MicroPython on Raspberry Pi Pico

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**.

as a control character – it acts as the equivalent of pressing the **ENTER** key, meaning that each line in your data log should be on its own separate line.

Click the Run icon, count to 60 again, and click Stop. Open and read your file:

```
file = open("temps.txt")
file.read()
file.close()
```

You've made progress, but it's still not right: the `\n` control character isn't acting like a press of **ENTER**, but printing as two visible characters (**Figure 5**, previous page). That's because `file.read()` is bringing in the raw contents of the file, and making no attempt at formatting it for the screen.

To fix the formatting problem, you need to wrap the file read in a `print()` function:

```
file = open("temps.txt")
print(file.read())
file.close()
```

This time you'll see each reading print out on its own line, neatly formatted and easy to read (**Figure 6**).

Congratulations: you've built a data logger which can take multiple readings and store them on your Pico's file system!

Your Pico's file system works regardless of whether or not it's connected to your Raspberry Pi or another computer. If you have a micro USB

▼ **Figure 6** Now you can read all the temperatures your data logger has captured

Figure 6

> **You can take your data logger to any room and have it run by itself**

mains charger or a USB battery pack with a micro USB cable, you can take your data logger to any room in your house and have it run by itself – but you'll need a way to get your program running without having to click the Run icon in Thonny.

For use without a connected computer – known as headless operation – you can save your program under a special file name: **main.py**. When MicroPython finds a file called **main.py** in its file system, it runs that automatically every time it's powered on or reset – without you having to click Run.

In Thonny, after stopping the program if running, click the File menu then Save As. Click 'Raspberry Pi Pico' in the pop-up that appears, then type 'main.py' as the file name before clicking Save. At first, nothing will seem to happen: that's because Thonny puts your Pico into interactive mode, which stops it from automatically running the program you just saved.

To force the program to run, click into the bottom of the Shell area and hold down the **CTRL** key, press the **D** key, then let go of the **CTRL** key. This sends your Pico a soft reset command, which will break it out of interactive mode and start the program running. Find something else to do for five minutes or so, then press the Stop icon and open your data log:

```
file = open("temps.txt")
print(file.read())
file.close()
```

You'll see a list of temperature readings, even though you didn't click the Run icon – because your program ran automatically when your Pico reset. If you have a micro USB charger or USB battery pack, disconnect your Pico from your Raspberry Pi, take it to another room, and connect it to the charger or battery pack. Leave it there for ten minutes, then come back and unplug it. Take it back to your Raspberry Pi, plug it back in, and read your file again: you'll see the readings from the other room, proving that your Pico can run perfectly well without your Raspberry Pi helping it along.

Congratulations: your data logger is now fully functional and wholly portable, ready to go with you wherever you need to record data! M
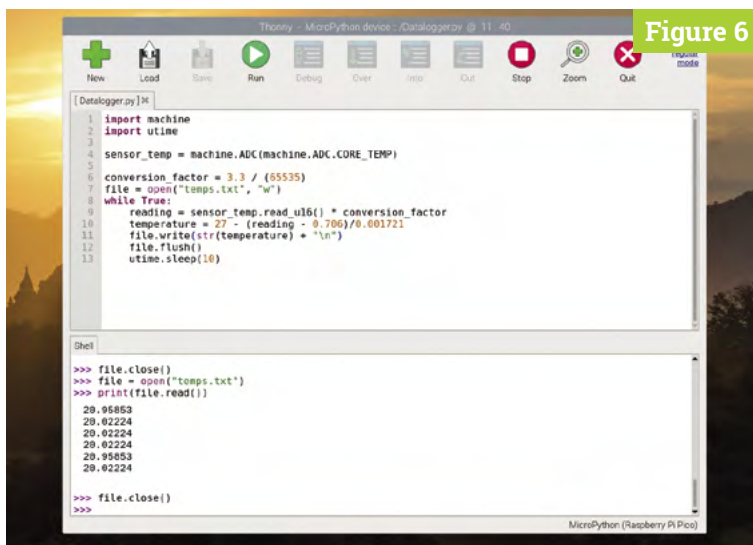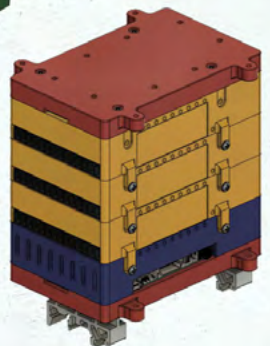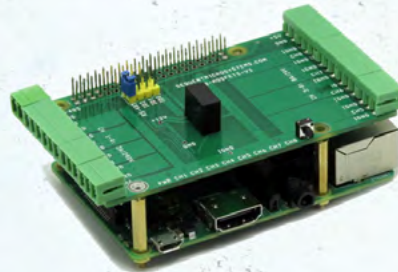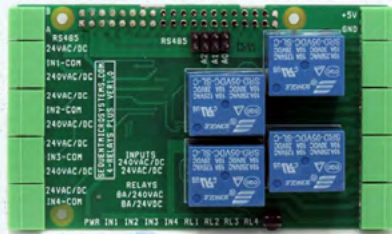
MAKER

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

Part 01

Isomorphic keyboard: hex keys

Use Raspberry Pi Pico to make an isomorphic music keyboard with hexagons

One alternative layout to a conventional piano keyboard is an isomorphic keyboard. Here, the notes are laid out so that adjacent keys differ by only a semitone. It is said that such a keyboard is easier to learn to play in any key. This month, we show you how to make a USB interfaced MIDI keyboard with an isomorphic layout.

01 Isomorphic keyboards

While the principle of adjacent keys differing only by a semitone is simple enough, there is considerable variety in exactly how they are laid out. The arrangement of keys can be on a square grid, or an offset grid; the keys themselves can be square, round, or, our favourite, hexagonal. There are a number of different layouts that detail exactly what keys are where; these have names like Gerhard, Park, Maupin, Wicki-Hayden, Harmonic, Janko, C-system, and B-system. We are going to use the Harmonic type that was used by the now defunct C-Thru Music company.

02 The enclosure

We came up with the concept of a two-layer box – this is shown in **Figure 1**. The top surface is where the keys are. They are cut out and press down on the tack switches underneath. To stop them falling out when the box is turned upside down, each key has an oversize version of itself glued to it. The tack switches are mounted on a large piece of stripboard, supported by 3 mm spacers screwed to the middle level. Underneath this is a Raspberry Pi Pico board, supported on five 30 mm pillars between the base and middle level.

03 Scan the Matrix

A scanning switch matrix is a way of reading a lot of switches from a few inputs/outputs. **Figure 2** shows the schematic of a small 3×4 matrix. There are many variations on this, but we have chosen to use a column drive with a row input. A walking zero signal is applied to the columns, and pressing a key connects a column to an input. As you know what column was low when an input was low, you

You'll Need

> Stripboard
> **magpi.cc/stripboard**

> 102 tack switches
> 4-pin 6×6×5 mm
> **magpi.cc/tackswitch**

> Hex Keys CAD diagrams
> **magpi.cc/pibakery**

> Access to laser cutter

> **Figure 1** Our double box concept

Figure 1

The switch under a key, which is hooked up to a Pico

A hexagonal key with its corresponding note written on
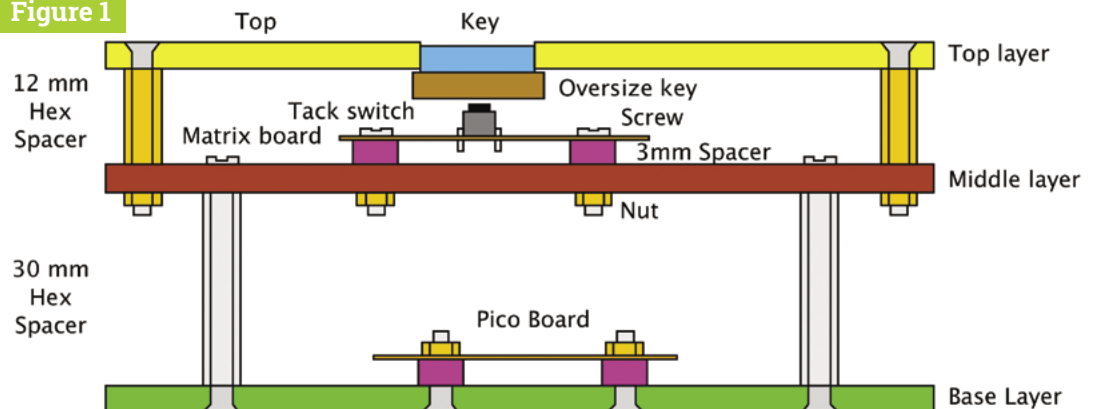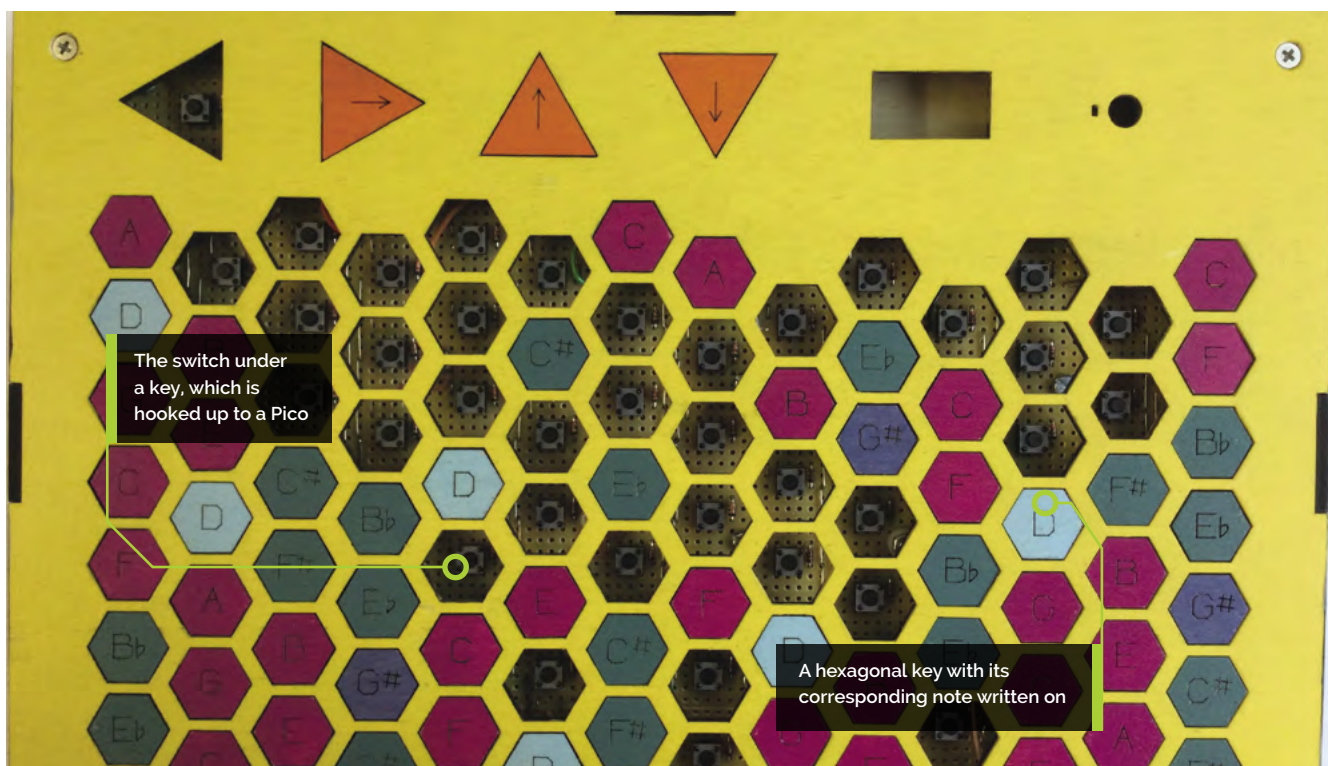
know what key was pressed. We will be making a 15 column by 7 row matrix for this project.

04 The laser files

We have started using the software LightBurn to produce our laser cutter artwork, and all the files can be found on our GitHub page. If you use another system, then you can get a full version of this software on a 30-day free trial to use or convert the files. Alternatively, we have also supplied SVG files for each sheet you have to cut. The top panel also includes vector engraving on the keys – this has to be done first at a lower setting so as not to burn all the way through.

05 Enclosure construction

Figure 3 shows the construction of the two-layer box. Note how the side with only a 1.5 mm tab sticking out goes to the middle layer of the box, and the 3 mm tabs on the other side go to the top and the base. We assembled and glued the sides into two rings, and then we reinforced each corner with a small length of half round dowel fixed with superglue. Then we plugged up the bottom end with Blu Tack (tack putty) and dribbled some Gorilla clear glue down the joint for added rigidity. See **Figure 4**.

06 Painting

We have come up with a new way to paint plywood boxes that doesn't show brush marks, by using a half-inch foam brush and normal emulsion paint. This allows us to use paint from the wide range of match-pot colours, and gives a nice flat finish. To keep things clean from finger marks, we add three coats of a water-based clear matt varnish. We used yellow for the top and base, and matt black for the sides and visible edges of the box.

07 Painting the keys

Figure 5 shows the colours we used to paint the keys. They keys can be painted any colours you like, but keep the groups consistent for the

Top Tip 👍

Box assembly

We marked the sides and middle layer of the inside of the box so that we always assembled it consistently.
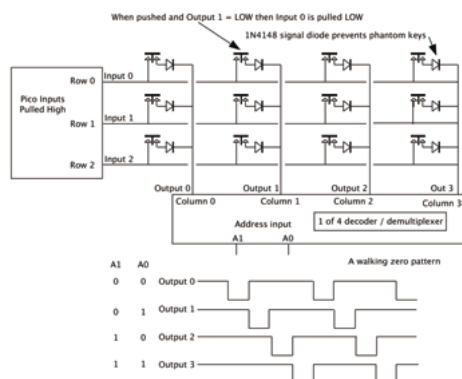
Figure 2

◀ **Figure 2** A 3×4 switch matrix

Figure 3
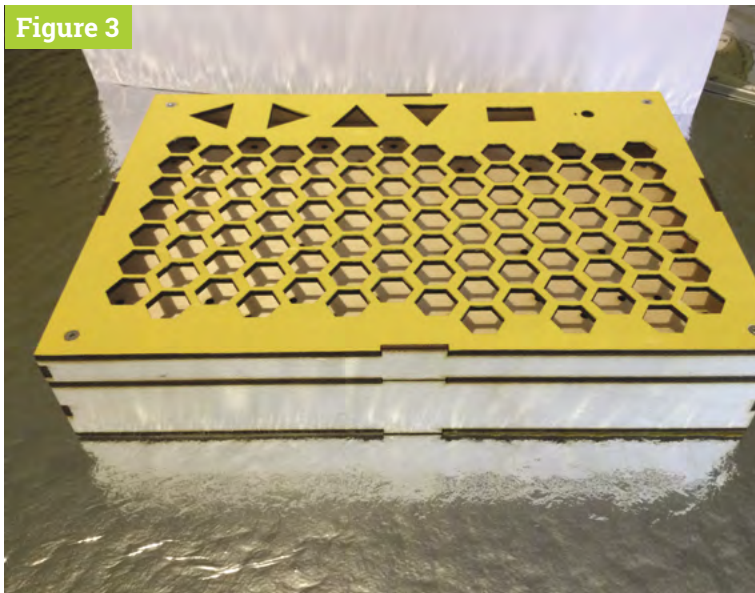
▲ **Figure 3** The assembled double box

Figure 4

▶ **Figure 4** Reinforcing the corners of the two rings formed from the sides

▼ **Figure 5** The colours we used for the keys – magenta and sky blue are the white keys on a piano
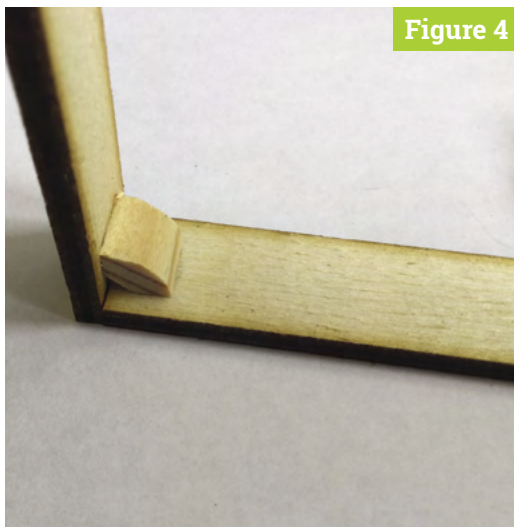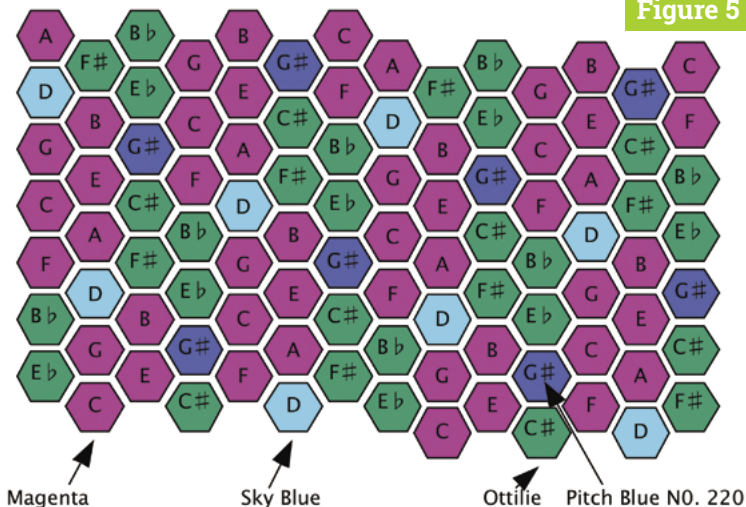

Figure 5

> ❝ Once painted, you can glue the keys to the oversized key blanks ❞

best playing experience. Again, three coats of matt varnish are applied to each key. Once painted, you can glue the keys to the oversized key blanks. We used a small drop of superglue in the middle of each and pressed down for ten seconds. Try to get them centralised; if not, this can be corrected with a disc sander. The top and bottom flat of each key blank should be about 1mm from the edge of the actual key.

08 The stripboard

We need a piece of stripboard 101 holes long by 58 strips high. Unfortunately, we couldn't find one, so we had to make one out of three pieces of board. Fortunately, this could be done by buying just one 119 by 455mm sheet of stripboard. The maximum height of this board is 46 strips, so we cut out a 101 hole by 46 strip high main piece, a 71 hole by 12 strip, and a 30 hole by 12 strip piece. We also got a 58 hole by 8 strip piece for the triangular control button board from the same piece of stripboard. See our GitHub page for more notes about joining these together.

09 The Matrix

Figure 6 shows the first five columns. There are two additional diagrams showing the rest of the wiring on our GitHub page, but the basic pattern is the same. The even and odd numbered columns both have identical patterns of key placement and track breaks round them. We fitted the switch first and then, before soldering it, cut the tracks between the switch pins and surrounding the tracks. To make sure you have picked up the correct place to make the break, insert a wire through the place indicated, then turn the board over and make the break where you see the wire.

10 The Matrix 2

Make sure the tack switches are flat to the board when you solder them. They have a habit of pushing themselves out once the board is turned over for soldering. To make sure they are OK, solder

one of the four pins and check. If it is not flat then hold down the corner of the key with your finger nail and reapply the soldering iron and push it into place. Do the same for the opposite corner. Once two corners are in place, simply solder up the other two. Then fit the 1N4148 diode next to each switch; get it the right way round.

11 The Matrix 3

The long tracks of tinned copper wire with a black dot on them denote a joint where one wire enters the hole and another leaves it. Here, push all the wires in place before soldering them. We found this easier if these holes were enlarged slightly with a 1.2 mm drill. When fitting the keys, turn the top over and insert from the back; we numbered the columns and rows on the board and wrote the column and row number, along with its name, on the back of each key. We then placed a blank board over the top and inverted it. Finally, we attached each key to the top with masking tape so they would not drop out when we fitted the top – see **Figure 7**.

12 Warp one, Mr Sulu

One problem we encountered was that the top board had an inwards warp. This meant that when the top retaining screws were tightened, the top was pressing some switches. To counter this,
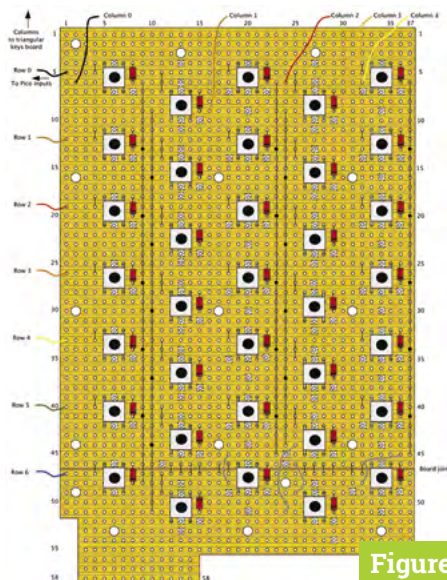
Figure 6
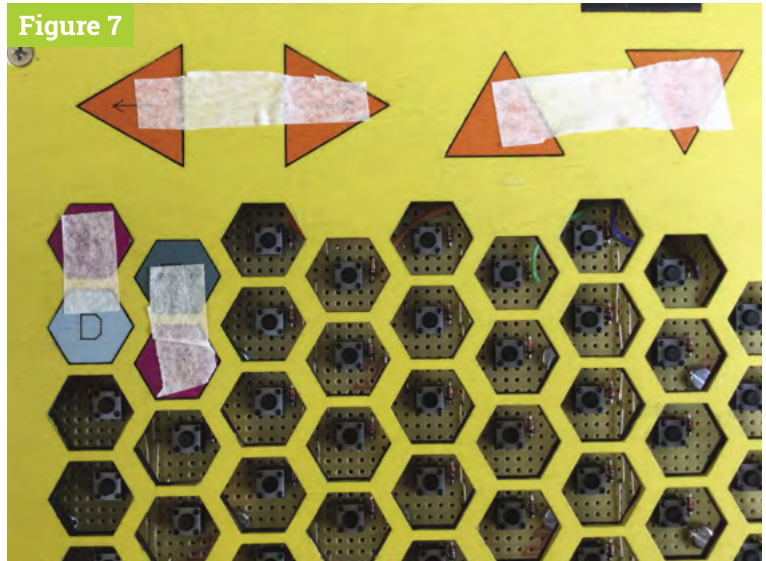
Figure 7

one or two securing holes were turned round so that a screw pushed up on the board, correcting the warping. We used a 20 cm pan-head M3 screw with a 3 mm spacing and three 0.5 mm solder tags on the underside of the board – see **Figure 8**. The solder tags allowed for fine adjustment, making the switch tops flush with the board. We had to cut away at the oversize key mount on surrounding keys to ensure the push-up screw was not fouled.

13 Part 2

Next issue, we'll look at the construction of Raspberry Pi Pico, control switches, OLED display, and rotary encoder, as well as how to wire it all together. We will also look at the software we need to test what we have built, and the final software bring it all to life. ᴍ
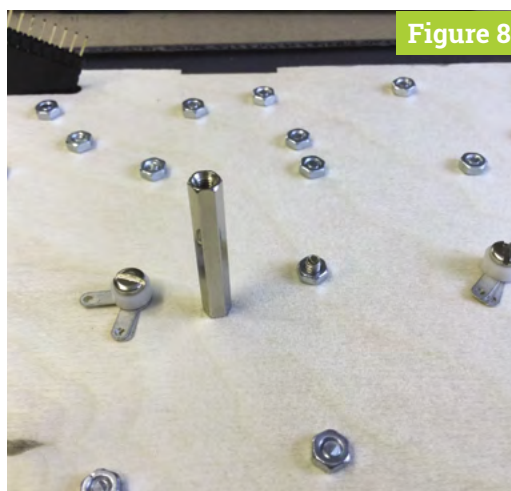
▲ **Figure 7** Use masking tape to stop the keys falling out when fitting the top to the switches

◄ **Figure 6** Physical wiring of a third of the switch matrix

Top Tip 👍

Deburr the stripboard holes

When drilling through the stripboard, you'll notice that there will be bits of copper that could make contact with the screw. Deburr the holes by lightly applying a larger drill bit by hand to remove the copper at the very edge of the hole.

Figure 8

◄ **Figure 8** Two push-up support screws and the central base support

MAKE GAMES WITH
RASPBERRY PI

Did you know that Raspberry Pi is a game-creation machine? There are many ways to write games and **Mark Vanstone** will show you a few to get you started

Mark Vanstone

MAKER

Educational games author from the 1990s, author of the ArcVenture series, disappeared into the corporate software wasteland. Rescued by Raspberry Pi!
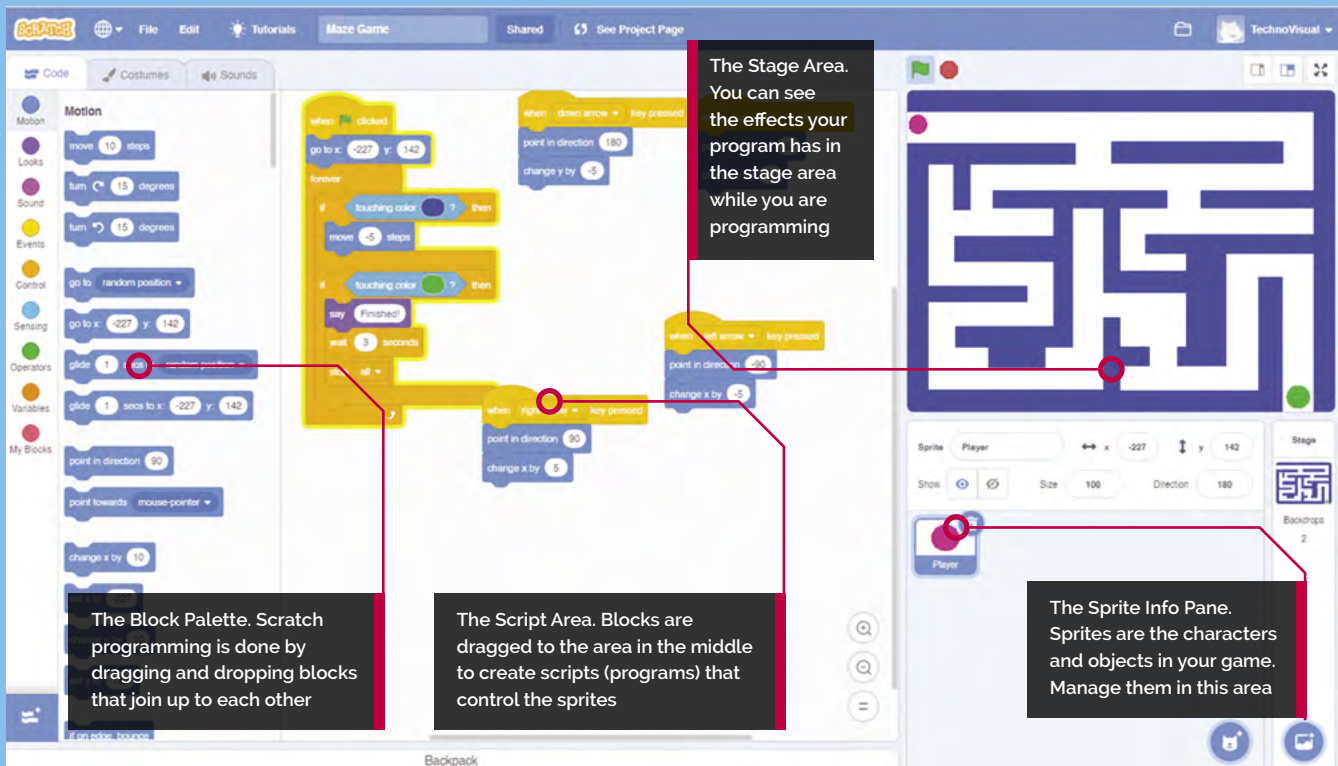
magpi.cc/ technovisual

The UK computer games industry has grown and grown since its origin in the eighties; grown so much that it's on a par with the UK film industry. This trend is true in many other areas of the world. If you are learning how to write games, then Raspberry Pi is a great way to get your teeth into the subject. There are many ways to write games and get started quickly.

If you are just starting out and not ready for text-based coding, don't worry. There are block-based systems like Scratch where you can lay out your game graphics on the screen. You can code games using blocks that you drag and drop into place to create your program. Once you have mastered that, you may want to move on to text-based coding like Python and Pygame. If you aim to start a career in the games industry, you will find that these days game designers use both methods: visual block editing and text-based programming. One of the most popular game engines, Unreal Engine uses a block editor called Blueprint that is underpinned by libraries of C++ code.

Game programming is a great way to learn a wide range of techniques that are useful for other areas of programming. You don't need to start with advanced scripting but can easily get quick results with the tools in this article. Let's make some games!

MAKE GAMES WITH SCRATCH

Scratch is an ideal place to start making games, with a great online community of creators who share their games so that you can see how they were made

The Stage Area. You can see the effects your program has in the stage area while you are programming

The Block Palette. Scratch programming is done by dragging and dropping blocks that join up to each other

The Script Area. Blocks are dragged to the area in the middle to create scripts (programs) that control the sprites

The Sprite Info Pane. Sprites are the characters and objects in your game. Manage them in this area

---

Top Tip 👍

Using sounds

If you want to use sound in your project, you can go to the Sounds tab. You will find a cat meow sound to start you off.

Scratch is a block-based visual editing programming language. Instead of writing commands in text, you click and drag objects (known as 'sprites') and control them with block commands. It's designed to make object-oriented programming easy to understand, and is a great way to get to grips with coding concepts. Due to its visual nature, it's ideal for creating basic games and interactive stories.

There are several versions of Scratch that are compatible with all versions of Raspberry Pi although the latest version, Scratch 3 is recommended for Raspberry Pi 3 and Raspberry Pi 4.

01 Get Raspberry Pi ready

It's always a good idea to keep your system files up-to-date. You can either download a fresh install of Raspberry Pi OS to your system card using the instructions at **magpi.cc/imager** or from your Terminal window use the commands `sudo apt update` and then `sudo apt upgrade`. It is wise to always go through this procedure before installing anything new on your Raspberry Pi to make sure you have the latest version of all the system files. Of course, for any installs or updates, you will need a connection to the internet.

02 Install Scratch

There are three versions of Scratch and an online editor. You can install Scratch 3 by clicking on Menu > Preferences > Recommended Software. In the Programming section, you will see Scratch 3. Place a tick in the Install checkbox to the right and click Apply.

03 Your first Scratch

If you have not used Scratch before, you probably want to jump straight in and make something happen. With Scratch, you can do just that. You'll find Scratch in Menu > Programming > Scratch 3. You will see a cartoon cat on the right-hand side and a set of blue boxes on the left. Drag the **turn 15 degrees** block into the Script area in the middle (this is where you assemble your program). Click the **turn 15 degrees** block and the cat will rotate.

04 The green flag

Our rotation block is good, but what if you want something more? We can build a program of blocks by joining them together. Click Control in the sidebar and then drag and drop the **repeat 10 block** into the Script area. Then move your **turn 15 degrees** block so that it's inside the **repeat** block. Then click Events in the Blocks palette and drag the **when (green flag) clicked** block to sit on top of the **repeat** block. Now if you click the green flag at the top of the window, the program will run.

05 Customising your sprites

In Scratch, a sprite is a graphic on the screen that you are controlling. You can change

RASPBERRY PI SCRATCH PROJECTS

Go to the projects section of Raspberry Pi's website, **magpi.cc/projects**. In the 'Find a Project' section, select Games from the Topic drop-down, and Scratch from the Software drop-down. You will be shown a selection of game projects for Scratch. Each of these projects is laid out as a step-by-step tutorial to help you build the game. There are lots of different game projects available, so you shouldn't run out!

Or you can try PICO-8

PICO-8 is a fantasy console for making, sharing, and playing tiny games and other computer programs. It feels like a regular console and runs on a variety of platforms. It has a suite of cartridge creation tools and an online cartridge browser called SPLORE. The programs are distributed in the form of a PNG file and each program has a memory limit of 32kB, so it is like programming a retro-style, 8-bit computer. The PICO-8 development system costs £11 / $15 and can be downloaded from **magpi.cc/pico8**.

what it looks like by loading in your graphics or editing the image. Select the Costumes tab at the top of the screen. From there you can use the painting tools to make a new image or alter the one that is already there. Try drawing some lines or filling in some shapes to see how it works. You can also write text. If you select items with the arrow tool, you can also change their colour.
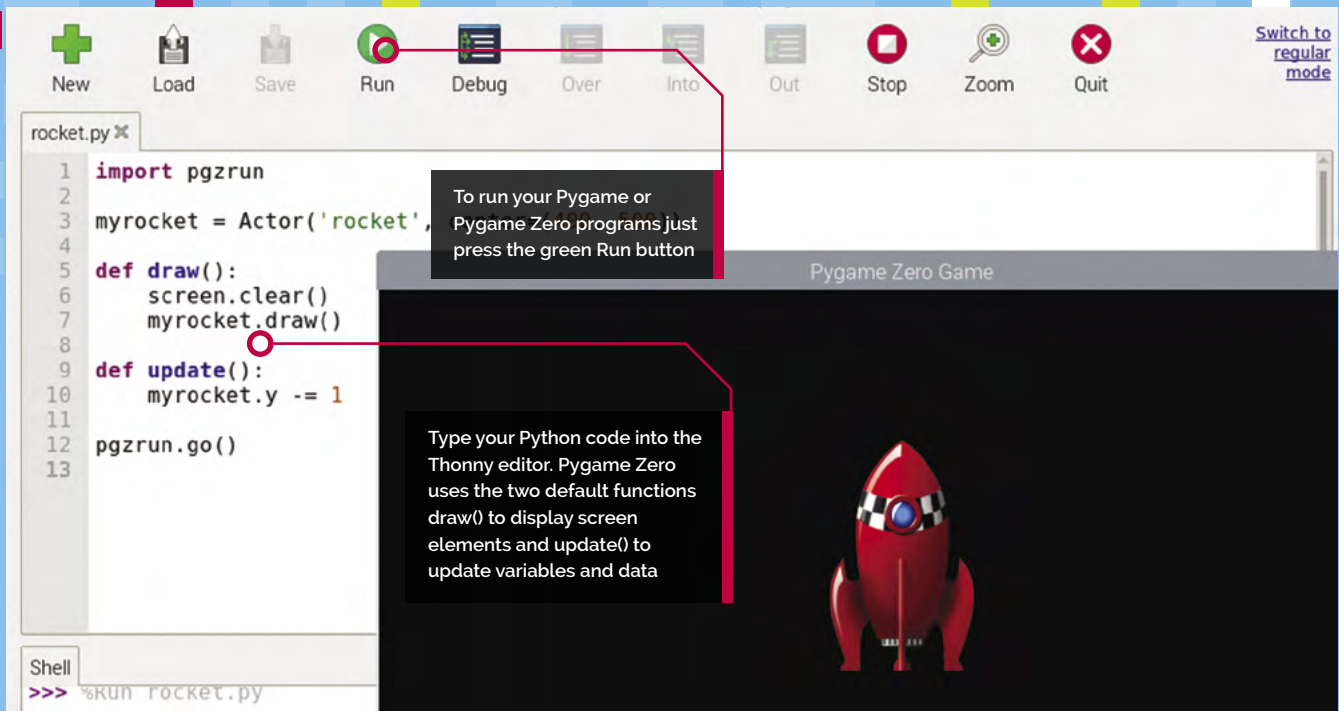
06 Exploring the Scratch community

You will probably want to find out lots more about how to use Scratch, and there are lots of tutorials if you select the Tutorials section in the top menu bar of the desktop or online version. You can also get lots more help from the main Scratch web page at **scratch.mit.edu/ideas**. If you look at the Explore section on the website, you will be able to find lots of projects that other people have made and if you find one you like, you can see how they did it by selecting the 'See inside' button at the top-right corner.

Top Tip 👍

Loading and saving

You can load/save Scratch projects to your Raspberry Pi. The online version can save projects to the Scratch server if you log in.

◀ The costume editor allows you to edit and design graphics for your games

To run your Pygame or Pygame Zero programs just press the green Run button

Type your Python code into the Thonny editor. Pygame Zero uses the two default functions draw() to display screen elements and update() to update variables and data

CODE PYTHON GAMES WITH PYGAME

Raspberry Pi can be used to make some super games and Pygame gives you a great head start

Top Tip 👍

Mix and Match

Even if you start your program with Pygame Zero, if you need a function from Pygame you can include parts of the Pygame module too!

One of the best ways to get started with text-based programming on your Raspberry Pi is to jump straight into Pygame or Pygame Zero. These are both available with the Python programming language and all three are already installed by default with Raspberry Pi OS. If you are not familiar with Python, you can get it running from the Programming menu by selecting the Thonny Python IDE. This will open up an editor to use Python 3. Python is easy -to learn and read, and this article will show you how to use it with Pygame and Pygame Zero.

01 First Pygame Zero

Pygame Zero was designed to require as little code as possible to get a game running. If you launch the Thonny Editor (IDE) and type `import pgzrun` to load the Pygame Zero module and then after that, write `pgzrun.go()` to start the game, you can then save the file and run it (with the green play button). If you have typed the code correctly, you will see a black window appear titled 'Pygame Zero Game'. You have written your first Pygame Zero game! It's not a great game yet but that's all you need to get the game engine running.

02 More than zero

Let's get a graphic moving on the screen. You will need to find a suitable image to use, perhaps a spaceship or little green man. Have a look at the 'Graphics Resources' section near the end of this feature about where to find graphics. A PNG is best; you can find our rocket at **magpi.cc/rocketart**. Now make a subdirectory in the same place as you saved your Python file and call it **images** and put your graphic file inside that directory. Now load that graphic into an Actor object in your code. Name your graphic file **rocket.png** (you must keep to lower-case letters) and load it by typing `myrocket = Actor('rocket', center=(400, 500))`.

03 Seeing the rocket

Now to get our rocket to display on the screen, we need to add some code to draw it. We do this with a `draw()` function, so type `def draw():` and press **RETURN**, then type `myrocket.draw()`. Then, to make the rocket move up the screen, we need to add an `update()` function by typing `def update():` and underneath type `myrocket.y -=1`. If we save and run this program, we should see the rocket moving up

the screen. If you don't, check the **rocket.py** code to see what you have done differently. It may have some drawing left behind, so add `screen.clear()` at the beginning of the `draw()` function. If all is well, you have the start of your Pygame Zero game.

ROCKET.PY

- Language: Python

```python
import pgzrun

myrocket = Actor('rocket', center=(400, 500))

def draw():
    screen.clear()
    myrocket.draw()

def update():
    myrocket.y -= 1

pgzrun.go()
```


◀ Lots of Pygame game developers share their creations online like this game called Dynamite

PGTEST.PY

- Language: Python

```python
import pygame
pygame.init()

screen = pygame.display.set_mode([400, 400])
running = True

while running:
    # Get events from the user
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
    # Fill the screen
    screen.fill((0, 0, 0))
    # Draw a red circle at 200,200 with
radius of 50
    pygame.draw.circle(screen, (255, 0, 0),
(200, 200), 50)
    # Switch from buffered screen to visible
    pygame.display.flip()

# Quit the program
pygame.quit()
```

04 Moving on to Pygame

Pygame Zero makes it very quick and easy to get games working on your Raspberry Pi, but if you want more flexibility you may find that Pygame is what you require. You will need to write a bit more code, but you will be able to access some functions like using game controllers. To start a Pygame program, you will need to import the pygame module using `import pygame` and then after that, make a call to `pygame.init()`. This starts the game engine off, but we won't see anything happen if we run it.

05 Making a screen

We make a screen for our game by calling a function called `pygame.display.set_mode()` and give it the width and height that we want the screen to be. Once that is set up, we will need to start a loop (in this case a `while` loop) to check that the program is still running – and in the loop we blank the screen, draw our graphics on an invisible buffered screen, and then flip the screen from the buffer to the visible screen. All this keeps happening until the user exits the program by using the window close icon. Have a look at **pgtest.py** to see how all this is done with Pygame.

06 Taking it further

Both of these examples are very simple, just to get you started, but there are lots of amazing games that can be made with Pygame and Pygame Zero. Over the years, *The MagPi* magazine has featured many tutorials about making games in Python and has even produced three books dedicated to teaching Python Games by example: *Retro Gaming with Raspberry Pi* (**magpi.cc/retrogaming**) and *Code the Classics – Volume 1* (**magpi.cc/ctc1**) which have many Pygame Zero example games, and the other is called *Essentials – Make Games with Python* (**magpi.cc/essentialgames**) which takes you through game creation with Pygame.

Top Tip 👍

Watch your naming

When you save your code, don't call it 'pygame' or Python will think that you are referring to the pygame module.

WHERE TO GET IDEAS

Did you know that *The MagPi's* sister magazine, Wireframe, features a section called Source Code every month with Pygame Zero game examples? **wfmag.cc**

MAKE GREAT GAMES WITH APPGAMEKIT

With AppGameKit you can develop professional-looking games, not just for Raspberry Pi but also for desktop and even mobile devices
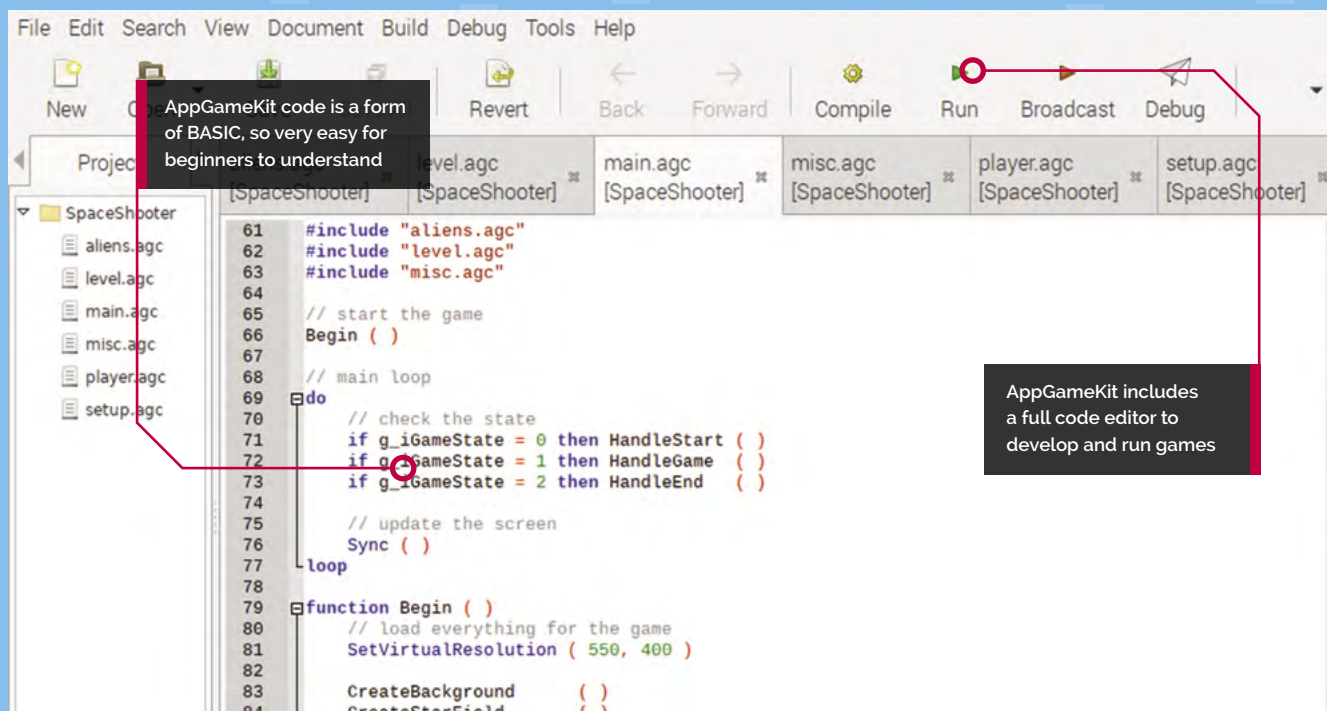
AppGameKit code is a form of BASIC, so very easy for beginners to understand

AppGameKit includes a full code editor to develop and run games

AppGameKit provides a cross-platform development system that was originally for PC desktops, but recently it has become available to download free for Raspberry Pi. You can use the same system on other platforms too, and develop on one system to run on a different one. You can even publish your games and earn money without paying any royalties. The engine has many tools to help you build your game, like 2D sprites, 3D, physics, sound, and even virtual reality. This guide will get you started with AppGameKit so that you can explore all the features.

Compatibility alert

Until recently, AppGameKit was compatible with all Raspberry Pi computers, but at the time of writing, it is difficult to get running on Raspberry Pi 4. Some system updates are needed for other Raspberry Pi computers, even with the latest version of Raspberry Pi OS. Make sure you have backed up any data from your Raspberry Pi microSD card before you start.

01 Get the download

First, we need to get the AppGameKit files. You'll need to go to the website **appgamekit.com** and sign up for an account. When that's done, go to the 'AppGameKit For Raspberry Pi' section in the 'Classic' menu item and download the editor files (they are free). Double-click the gzip file to open it and extract the files to somewhere suitable like your home directory. When it's unpacked, you will see a directory called **AGKPi**. Inside that, you will find the AGK launcher. Double-click to open the editor. If you want to see an error log when it's running, select 'Execute in Terminal' when prompted.

02 Doing the update

If you try to run any of the samples provided with AGK, you may find that you get some errors. This may only be an issue at the time of writing as there are regular updates available.

From a Terminal window, enter `sudo apt update`, followed by `sudo apt upgrade`, just to make sure we have everything up to date. Then, if you are getting errors about libgles2 (graphics library), type `sudo apt install libgles2-mesa libsdl2-dev`, which will install the necessary libraries. Then enter `sudo rpi-update` – this is a firmware update, so a bit more extreme than the usual updates, and the reason why you should make a backup of your memory card before issuing this command. Now reboot your Raspberry Pi.

◀ With the AppGameKit samples, you can quickly see how to build many types of games

03 Load a sample

A good sample to start with, to make sure everything is working, is the Space Shooter game. Select the Open icon on the toolbar of the editor and then navigate to the **SpaceShooter** directory, which is found in the **Games** folder inside **Projects**. Open the .agk file and you will see several files open in the editor. AGK uses a language very much like BASIC, so if you have used BASIC before you should be right at home. If you haven't learnt BASIC, then it's quite easy as it was designed for beginners.

04 Run the game

If everything has gone well with the install and updates, when you press the green Run arrow you should see a window open up titled AGK and a Start Game screen with spaceships floating about. If you don't see that, then check the Terminal window that launched the editor to see if there are any errors. You may see some warnings there anyway as some of the shader modes are not supported on Raspberry Pi, but the game should work fine. Start the game by clicking the screen and move the player ship up and down with cursor keys.

05 Make your program

Now you have the editor building a game, why not start your own? Start a new project by clicking the New icon on the toolbar. You will be asked for a name for the new project and a base path. Select the folder icon to the right of the base path input box, and navigate to somewhere suitable inside your home directory. Select Create and you will see a new file called **main.agc** open in the editor. In that file, there will already be

a listing of the base code you need to start your game. Run it and you will see a black window open with the title 'test'.

06 The sky's the limit

Have a look through the samples to see the range of what AGK can do. You will find a huge range of tutorials at **magpi.cc/agkyt** and there is a full user guide at **magpi.cc/agkguide**. There is also an active and helpful community forum at **magpi.cc/agkforum** where you will find more hints and tips to help you on your way. If you are having a problem with something, you'll find someone who has solved it and will tell you how. Don't be afraid to get stuck in and just start coding: the compiler will give you feedback on anything you get wrong.

Top Tip 👍

Raspberry Pi 4

Feeling brave and want to run AGK on a Raspberry Pi 4? Check out the forum post at: **magpi.cc/agkpi4** for instructions.

You'll Need

- AppGameKit: **appgamekit.com**
- Raspberry Pi SDK: **appgamekit.com/agk-pi**

GRAPHICS RESOURCES

There are many graphics resources that are free to download. You can get images, animations and programs. Here are a few places to visit:

- **opengameart.org** has a wide range of artwork for backgrounds and character images to include in your games free of charge.

- **spriters-resource.com** specialises in sprites, which are the characters to include in games. They are often available in sprite sheets which have all the frames in one image file.

- **free3d.com** has many free (and paid-for) 3D models for you to download and use. There are models for just about any situation, some of them specifically designed for games.

- **gimp.org** is a great image manipulation program and should be all you need for creating 2D graphics for your games. It can be installed using `sudo apt install gimp` in a Terminal window.

- **blender.org** is best for creating a game with 3D graphics. Install Blender free from your Terminal with `sudo apt install blender`. Discover a range of Blender projects on Raspberry Pi's website (**magpi.cc/blenderprojects**).

MAKE ADVENTURE GAMES WITH REN'PY

This game engine is for storytelling. Use Raspberry Pi to combine words, images, and sounds to create interactive visual novels and life simulation games
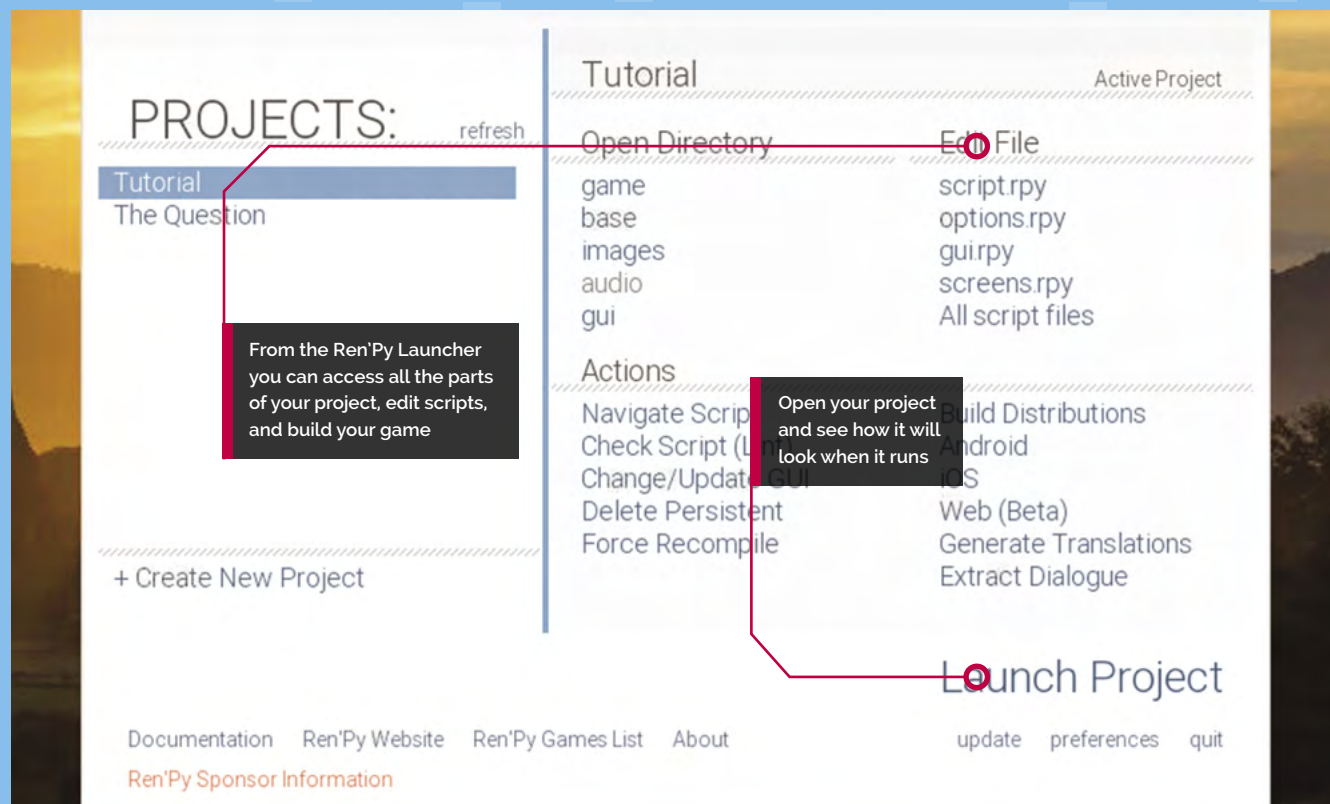
From the Ren'Py Launcher you can access all the parts of your project, edit scripts, and build your game

Open your project and see how it will look when it runs

Ren'Py is open-source and free to download and use. You can even share your creations without paying a penny in royalties or licences. Ren'Py includes a simple scripting language to control the flow of your story and add interactivity to the pages. The engine also includes a wide selection of animation and transition effects to bring your games and graphic novels to life without needing to learn complicated animation software and supports the most common graphics and sound formats like JPG, PNG, MP3, and a whole lot more.

01 Get the files

First, download the install files from the Ren'Py website at **renpy.org/latest.html**. You will need the .bz2 version for Raspberry Pi. When it has downloaded, double-click to open the archive and extract it to a suitable place such as your home directory. You will also need to download and extract Raspberry Pi support files from the Additional Downloads section. Once this is all in place, you will find a file in the directory you have extracted called **renpy.sh**. Double-click this file and select 'Execute'. After a few seconds, you will see the Ren'Py Launcher open.

02 Tutorial time

Ren'Py includes a getting started tutorial, which is probably the best place to begin. By selecting the Tutorial project from the launcher, you will be introduced to Ren'Py's features by Eileen. She will show you how to start a new project and the ways to set colours and screen sizes. There are also sections in the tutorial to

Eileen

In this tutorial, we'll teach you the basics of Ren'Py, so you can make games of your own. We'll also demonstrate many features, so you can see what Ren'Py is capable of.

cover adding your images, text, and sound to your pages. It then goes on to creating interactions and transitions to make your game engaging for your audience. Have a look at the Choices and Python section to see how scripting is used to ask questions and branch to different options.

the left of the window that opens. If you make changes to your script, you can then press **SHIFT+R** to reload your script and start the game again. If you need further help, select the Documentation link at the bottom left of the Launcher window, or check out the forums at **magpi.cc/renpyforum**. M

▲ Ren'Py includes a tutorial where Eileen talks you through all the features of the system

03 Let's make a game

Going back to the Ren'Py Launcher, start a new project with the 'Create New Project' link on the left-hand side. You will then be asked where you want to save your project and what it should be named. Next, choose what screen resolution you want your production to use and the colour scheme that you would like. After a short pause for processing, your project will be created and listed with the tutorial in the Projects section in the Launcher.

04 Let's get scripting

Start scripting the game by selecting the 'script.rpy' option under Edit File in the Launcher. It will ask you to select the editor you want to use and then open the script. From there you can make changes to the default script. When you want to test your changes, select your project and Launch Project, then select 'Start' from the list on

SOUND RESOURCES

If you need to find sounds for your games, you can get a whole range of sound effects and background soundtracks from **freesound.org**, **zapsplat.com,** or **musopen.org/music**, all of which provide free downloads.

You may need to edit your sounds, in which case use Audacity – available for you to install using `sudo apt install audacity` from your Terminal window.

▼ Audacity enables you to edit sound files in a variety of formats such as WAV and MP3
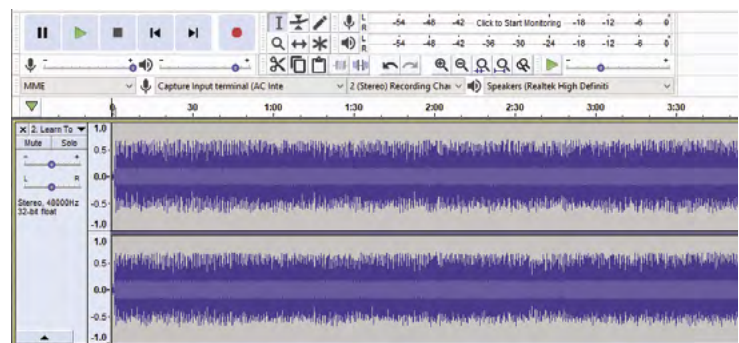
pi-top [4] Robotics Kit
and Expansion Plate

Build a range of robots with aluminium plates and reusable rivets, then control them with Raspberry Pi and pi-top [4] DIY Edition case. This kit clicks with **Lucy Hattersley**

Pi-top [4] Robotics Kit is a long-awaited robot that complements the pi-top [4] DIY Edition case. It promises to be a sturdy and versatile building platform, with competent software and good educational chops.

And it delivers. Inside the box is a series of aluminium plates and plastic rivets that act a little like LEGO Bricks meet Meccano. Two rivet-compatible servo motors and two encoder DC motors, plus a webcam and ultrasonic distance sensor. Everything you need to build a wheeled robot that can see and sense the world around it.

A plan comes together

We reviewed the pi-top [4] DIY Edition (**magpi.cc/pitop4review**) back in *The MagPi* issue 99 (**magpi.cc/99**). At the time, we were impressed with the build quality, but noted the oddity of the built-in battery and 128×64 OLED display.

Once clipped into the Expansion Plate on the robot, it all makes sense. The pi-top [4] case powers both Raspberry Pi and the motors in the robot; the OLED display provides feedback on the IP address and remaining battery level (we got around two hours of use). The kit tested comes with an Expansion Plate that connects to the bottom of the pi-top [4] DIY Edition case and breaks it out into several control ports.

Making robots

Thanks to the rivet system, you can get creative with your robots. Three designs are included: Alex, a regular wheeled robot with a pan-tilt mechanism for the camera and ultrasonic sensor; Bobbie uses the servos to control two ping-pong ball grabbing pincers; Prax is angled in a vertical position and the servos create a moving face for a desk-based interactive assistant. Instructions for all three builds are available as a PDF download from the pi-top Start website (**pi-top.com/start**) and it's a good place to get an overview of how the builds work.

Going further

Each build took us around an hour. When the build is complete, you attach a pi-top [4] DIY Edition to

▼ The innovative rivet system enables you to connect aluminum plates and build the robot

The "Prax" design uses the components to build an interactive desktop assistant

▲ The pi-top Robotics Kit designs are sturdy. The "Bobbie" design uses servo motors as two pincer arms

> ❝ Superbly designed with a clever rivet construction system and seamless integration with pi-topOS and Further courses ❞

the Expansion Plate and insert the cables to connect the electronic parts to the Expansion Plate.

Here is where pi-top [4]'s on-board battery and OLED display spring into useful action, providing network information that you can use to quickly SSH into the robot.

Pi-top has clearly put a lot of elbow-grease into its pi-topOS and its Further 2.0 system (**magpi.cc/further**). The pi-topOS ensures elements like SSH are enabled by

default; along with baked-in support for the hardware components.

The Further website has demonstration code to follow, and you can SSH directly to your robot from the Further website and run the interactive code examples directly from the web. Students can log in separately and collaborate on code and practise programming techniques together. We enjoyed the coding courses and challenges, and integration with OpenCV for object and face recognition.

Moving beyond the Further educational port, you can code pi-top [4] Robotics Kit directly with the pi-top Python SDK (**magpi.cc/pitopsdk**) and there is support for ROS (**magpi.cc/pitopros**) and Microsoft's .NET (**magpi.cc/pitopnet**).

There's a lot here. First of all, the whole kit is not particularly cheap if you include the price of the pi-top [4] DIY Edition and a Raspberry Pi 4. It is, however, superbly designed with a clever rivet construction system and seamless integration with pi-topOS and Further courses. All in all, pi-top [4] Robotics Kit is plain nice to use, and will sit neatly into an education environment. Good job. M

◀ The pi-top [4] DIY Edition sits on top of the chassis as the brains and battery of the robot

Verdict

Very high-quality robotic education kit, with an innovative rivet construction mechanism and well-thought-out software. It's been worth the wait.

10/10

10 Amazing:

Wearable projects

Ways to wear your Raspberry Pi with pride

As Raspberry Pi is so small and draws so little power, it's a perfect device for powering more complex wearable projects. We've done cosplay stuff with it in the past, and here are some other incredible projects that let you wear your Raspberry Pi. M
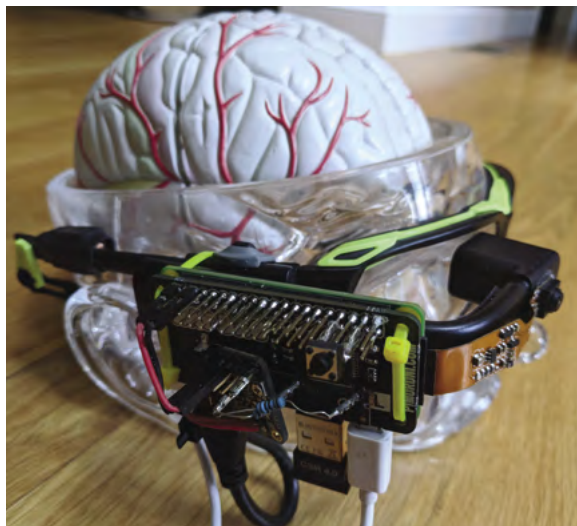
▲ PiGlass

Smart vision

We maintain that Google Glass was cool, so seeing excellent DIY versions using a Raspberry Pi is always a delight. This one packs in a lot of extra features as well.

magpi.cc/piglass

▲ Raspberry Pi Smart Watch

Wrist computer

With screens getting smaller and smaller, discreet and wrist-bound Raspberry Pi builds are easier than ever to do. This Smart Watch build is a very fun example.

magpi.cc/smartwatch

▶ Social media without the internet

Offline socialising

This art piece made full use of wearable tech by having interactive objects all over these coveralls. This allowed for social media-style interactions in real life.

magpi.cc/socialwear

◀ Pip-Boy built from scrap

Apocalyptic wrist computer

You can buy official Pip-Boy cases, or 3D-print carefully crafted models. However, the recycling nature of this project makes it a firm favourite of ours.

magpi.cc/pipboyscrap

Wearable Tech Projects

Want to see more wearables, and maybe make some yourself? Check out *Wearable Tech Projects* by Sophy Wong and our sibling mag, HackSpace Magazine:

magpi.cc/wearableprojects

▲ PiE-Ink Name Badge

Who are you?

This is sure to impress folks when you're at an event (whenever they come back) or when you're the newbie in the (geeky) office.

magpi.cc/pieink
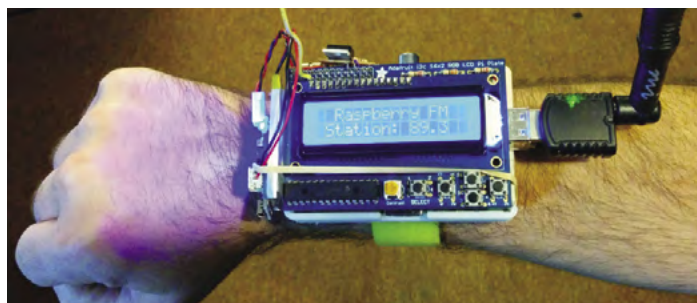
▲ RaspWristRadio

Portable FM

We like the very 1980s movie kid inventor look of this DIY radio that fits on your wrist. Don't be rude, though: bring headphones to listen!

magpi.cc/raspwristradio
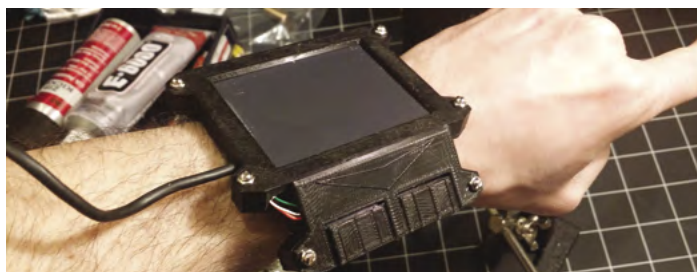
▲ Wearable Cyberpunk Gesture Pad

Hacker wear

Need a wrist-mounted touchpad with multi-touch gestures? Then look no further than Zack's cyberpunk-inspired gesture pad.

magpi.cc/gesturepad

▲ Wearable Time-Lapse Camera

Taking a walk

This Raspberry Pi Zero project is simple yet cool, and we think it would make for some great music video footage.

magpi.cc/timelapsecam

▶ Smart Cap

Cyber eye

This more conspicuous take on a pair of smart glasses can be used on multiple types of headwear. It's also completely open.

magpi.cc/smartcap
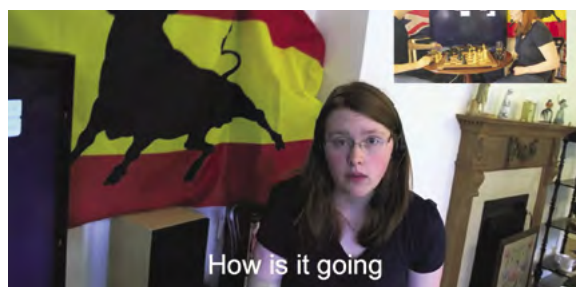
▲ Project Glass

Real-life subtitles

Real-time translation, while not 100% accurate, is quite impressive. Using this AR-like Raspberry Pi glasses system, you can use it to try to talk to real people.

magpi.cc/projectglass

Learn Visual Studio
Code with Raspberry Pi

Discover the best tutorials and resources for Microsoft's development environment.

By **Lucy Hattersley**

Getting started with Visual Studio Code

AUTHOR

Microsoft

Price:
Free

magpi.cc/vsintro

Microsoft's Visual Studio Code is a modern text editor and IDE (integrated development environment). Back in 2019, StackOverflow reported a massive leap in popularity for Visual Studio Code. It immediately took the IDE number one spot with over 50% of respondents to their annual survey using it (**magpi.cc/so2019**).

Visual Studio Code is available for Raspberry Pi and can be easily installed in Raspberry Pi OS. Click on Menu > Preferences > Recommended Software and choose Programming from the sidebar. Scroll down and place a tick next to Visual Studio Code and click Apply.

Open Visual Studio Code (under Programming) and click Help > Introductory Videos.
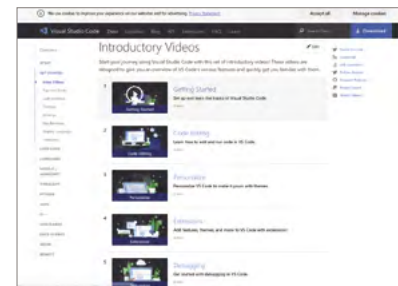
In these videos, Ornella Altunyan walks you through the user interface and settings, with tips and tricks. Once you've done that, click User Guide in the sidebar to start on the documentation. 🅜

Visual Studio Code resources
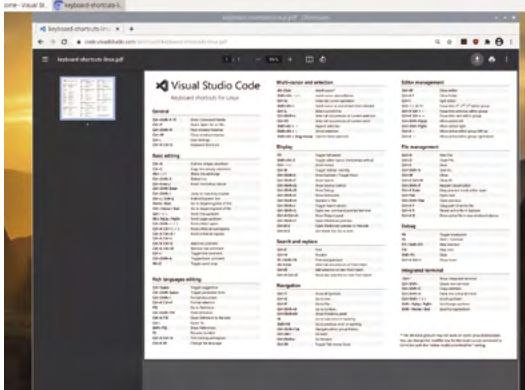
Bookmark these pages and return to them while you code

KEYBOARD SHORT CUTS FOR LINUX
Worth printing out if possible, but certainly keep these keyboard short cuts bookmarked while you run and debug code.
▶ **magpi.cc/vscodeshortcuts**

AWESOME VSCODE
This GitHub page by Viatsko has a curated list of excellent packages and resources, all placed into handy lists and sections.
▶ **magpi.cc/awesomevscode**

VS CODE CAN DO THAT?!
This website by Burke Holland and Sarah Drasner is a list of interesting things that VS Code can do. Genuinely fascinating.
▶ **vscodecandothat.com**

Set up your extensions

AUTHOR

Microsoft

Price:
Free

**magpi.cc/
vscextensions**

Once you've installed Visual Studio Code, you will want to investigate extensions. A huge array of additions to the base program enable you to add support for languages, debuggers, editing assistants, and general tools.

Choose View > Extensions from the menu bar inside Visual Studio Code (or press **CTRL+SHIFT+X**) to see the Extensions interface. By default, it will show some of the most popular extensions. The Extension Marketplace Documentation has a great overview of how to search for, install, update, and remove the extensions. M

VSCode Power

AUTHOR

Ahmad Awais

Price:
£35/$50

vscode.pro

In our experience, most of what you need to know can be found in the extensive Visual Studio Code documentation and introductory video tutorials.

If you want to deep-dive into Visual Studio Code with an online course, then there is one that everybody recommends: Ahmad Awais's video tutorials called 'VSCode Power'.

Ahmad is an entertaining course leader and the video tutorials are engaging and fun. The course covers everything from the basics, and setting up Visual Studio Code, through version control with GitHub and a professional extension setup. M

VISUAL STUDIO CODE ON RASPBERRY PI

Microsoft has created its own guide to installing and running Visual Studio Code on Raspberry Pi. A great place to start.
▶ **magpi.cc/
vscoderaspberrypi**

VISUAL STUDIO CODE AND HTML

Codecademy's article on using Visual Studio Code to build a basic HTML document walks you through setting up development folders and adding files.
▶ **magpi.cc/vscodecademy**

HOW TO USE VISUAL STUDIO CODE

Flavio Copes has written a great introduction to Visual Studio Code that works through installation, setting up fonts and workspaces.
▶ **magpi.cc/flaviovscode**

Zack Freedman

Zack loves wearables, and now makes them
full-time on YouTube, among other things

> Name **Zack Freedman** | > Occupation **YouTuber**

> Community role **Maker** | > URL **magpi.cc/zackfreedman**

Many people dream of being a YouTuber or online content creator of some kind. Zack Freedman basically stumbled into it.

"I used to be a freelance prototype developer in New York City," Zack tells us. "But when… all the meetups and conferences [shutdown last year], the clients dried up. I knew that social interaction wouldn't just go away – people would spend more time online, substituting streams for meetups and videos for talks. So, I started making YouTube videos to promote myself, and people watched – not many at first, but out of nowhere, lots of people, way more than I expected. The videos ended up doing better than the prototyping contracts! Now, I'm a full-time YouTuber, making videos about 3D printing and DIY electronics, producing videos and streams to show different parts of the stuff-making process."

What is your history with making?
I tried and failed to build tech projects a number of times: websites as a kid, RC cars as a teen, a Nerf ammo counter in college, and many more dead-end ideas that my very generous parents supported. I even got into engineering school, only to find that it was all math, and I hate math. I've never been a great student, even when I want to learn the topic – I get

▶ Some of the amazing projects that Zack has put together over the years

> " I've built a crazy number of projects for fun and work, at least 200 in total "

obsessed with something like Nerf blasters and plunge hundreds of hours into the rabbit hole, but I just can't force myself to systematically learn anything soup to nuts. I was spinning my wheels and felt like a moron.

Everything changed during an internship in the summer of 2011 – I was working for an app store startup ('twas a different world) and got assigned to write for the company blog. At the time, 'appcessories' – gizmos designed to pair with cell phones – were the Silicon Valley hotness, and my research brought me to Hackaday, where someone had turned a $25 toy into a wearable computer. I thought that was the coolest thing ever, so I instantly got obsessed with turning myself into a cyborg.

Since then, I've built a crazy number of projects for fun and work, at least 200 in total. It was all going pretty well until people started coughing a lot and pretending it wasn't dangerous, and here we are!

When did you first learn of Raspberry Pi?

I'm by no means an early adopter, but a MakerBar member was. He pre-ordered a huge pile of original Raspberry Pi before the first release, so I got in on the ground floor. [As] I was obsessed with wearable computers, Raspberry Pi was the only pocket-sized device below $500 that output 480p composite video, so it instantly became a critical part in my projects. One of our members learned that Eben Upton himself was travelling through the East Coast, so we invited him to speak at our hackerspace and somehow got a meet-and-greet with the team. I showed off one of my earliest wearable computers and Eben signed a Raspberry Pi for me. I put it in a frame. It's one of my most prized possessions.

What was your first Raspberry Pi project?

My first Raspberry Pi project was a wearable computer built out of a Vuzix AV230, an original Raspberry Pi, and a USB battery pack. It wasn't the most useful device… still, it was an authentic fully functional wearable computer, the kind that took MIT geniuses years of work and thousands of dollars only a couple of decades earlier. It was the springboard that helped me broaden my knowledge and build more projects; as I printed better enclosures, designed better electronics, and experimented with different control devices, I filled out my skills. 𝕄

This Month in
Raspberry Pi

#**MagPi**Monday

Amazing projects direct from our Twitter!

Every Monday we ask the question: have you made something with a Raspberry Pi over the weekend? Every Monday, our followers send us amazing photos and videos of the things they've made.

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

01. This robot can be controlled over the internet! Hopefully it doesn't grab the router's power cable.

02. Astronomy and Raspberry Pi are two great things that go great together.

03. Ravi has been doing some amazing things with flying Pico projects!

04. This is proper science fiction stuff – hopefully we can get portable versions soon.

05. Pi Wars is still going, even if it is currently a remote event.

06. This is a very cool project from one of our regular writers, PJ.

07. CutiePi is a cool tablet project that we love seeing.

08. The number of Pico applications is continually rising.

09. This microscope is very cool and powerful. Apparently, that is the stinger on a stinging nettle!

Stan Dmitriev
@DmitrievStan

Replying to @TheMagPi

Added a claw and a y-axis camera servo for my internet controlled @Sphero RVR.

Runs on Raspberry Pi 4 + Surrogate.tv SDK.

0:18 962 views

01

Ward Hills
@wardhills

Replying to @TheMagPi

Telescope testing last night.
It has a Pico running an IMU and display, with a RPi3 sending data to Stellarium Planatarium.
Designed during lockdowns and bulid in between @cammakespace. Captured Alkaid(Ursa Major), Antaries. Mars, Venus and Crescent moon.
#madeinmakespace

02

electronics_monk
@ravi_a_geek

Replying to @TheMagPi

Raspberrypi pico based #piwings flight controller.. as @STEM education platform

03

Michael Pick
@TheCasualEngr

Replying to @TheMagPi

I made an ultra compact computer that folds up into a tube! Powered by a raspberry pi 4. Full build here:

04

The Nomad Computer
World's Most Compact/rugged off the grid Computer. The goal of this project

Brian Corteil
@CannonFodder

Replying to @TheMagPi

The first heats of #PiWars #RemotePiNoon were run.
The event was hosted on a Pi 4 & the robot have Pi 3
As as their brains #MagPiMonday
Jasper was our arena kitten.

05

Penk Chen
@penk

Replying to @TheMagPi

Verified injection parts for the soon-to-be-shipped CutiePi tablet (cutiepi.io), also completed the battery standby time test in 8 hours 50 minutes from a single charge 😌

#raspberrypi #computemodule4 #opensourcehardware

07

PJ Evans
@MrPJEvans

Replying to @TheMagPi

My piMac project edges towards completion. Power button, sound and camera all working.

06

Stewart Watkiss
@stewartwatkiss

Replying to @TheMagPi

I've been working at getting a RC controller working with a Raspberry Pi Pico.
Fairly successful, I get the occasional invalid reading, but managed to get it working reliable enough.

youtu.be/kCZb6CD6Vgg

#RaspberryPiPico #MagPiMonday

08

RC Remote Control Raspberry Pi Pico
The Raspberry Pi Pico does not come with any kind of wireless build-in, but that doesn't mean you can't control it...
youtube.com

DomObvious
@DomObvious

Replying to @TheMagPi

I am still working on by Raspberry pi based microscope. Uses a Pi 4 and waveshare stepper hat for focus. Wrote my own Python based GUI. I need to sort out the illumination and working on a moving stage. Can you guess what the image is? @TheMagPi

09
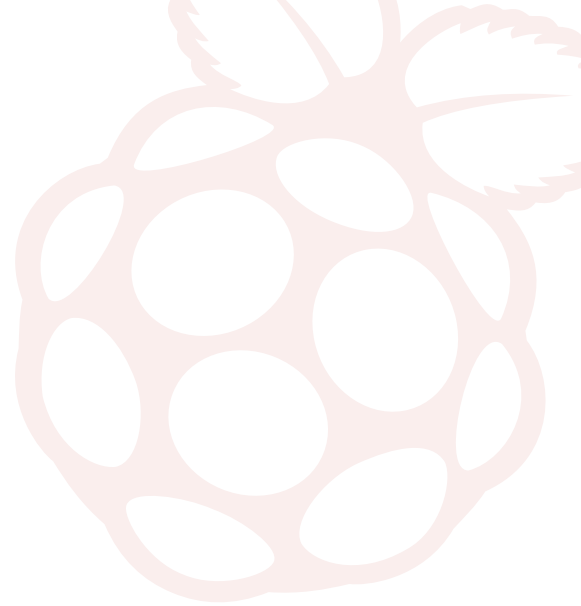
Maple Syrup Pi

A Google Coral smart camera

Smart camera tech is something that has been around for a little while, although it is very often under-utilised for many reasons. Ricardo de Azambuja is making a fresh attempt to use it in a practical and exciting manner.

"Maple Syrup Pi Camera is part of my research project," Ricardo tells us. "I'm one of the 25 TRAIN@Ed Research Fellows at University of Edinburgh. I am working on a project focused on helping local tourist attractions to better manage tourist flow, still GDPR compliant by design. My solution is to process information in the camera without ever saving or transmitting personal data. In addition to that, I wanted something open-source that could be customised, and as low-power as possible, allowing it to run a full day on an off-the-shelf power bank. Raspberry Pi Zero W is my go-to IoT device and, together with the Google Coral USB Accelerator, it has become my powerhouse for machine learning inference."

The Coral USB Accelerator attached to Raspberry Pi contains an Edge TPU processor which allows for offline TensorFlow models to run. It has some similar tech to the AIY Vision Kit that Google released a little while back.

If you're interested in reading about how it works in more detail, you can head to Ricardo's project page at **magpi.cc/maplesyrup**. M

◀ The finished device is fairly small

▼ The case holds a Raspberry Pi Zero W and a Camera Module

▼ The Coral TPU is mounted simply on the back
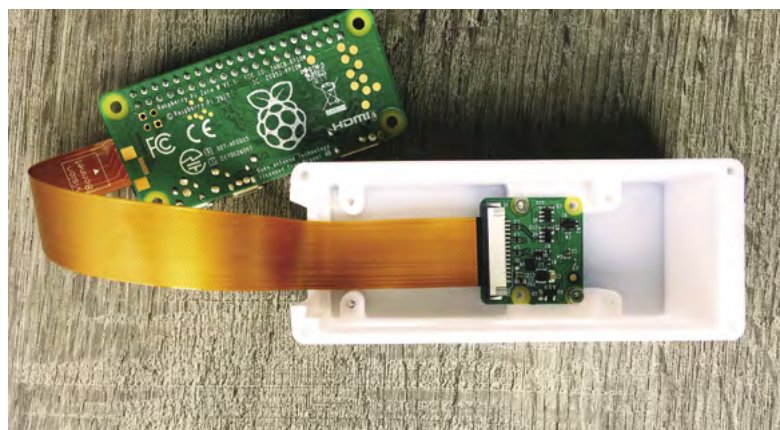
Crowdfund **this!** Raspberry Pi projects you can crowdfund this month

Lite Berry

The design of the Nintendo Switch is very appealing for many folks who play games, so this kit that helps you turn a Raspberry Pi CM4 into a handheld gaming powerhouse using the same form factor looks quite cool.

▶ **kck.st/3ivYCVY**

PiRelay 8

This board, which is being made in association with SB Components, really expands the IoT uses of a Raspberry Pi. As its name suggests, it adds eight relays and even has a built-in touchscreen that controls each one.

▶ **kck.st/3qAzLCg**

Your Letters
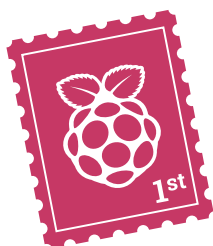
Backwards compatibility, part two

I am the Chair of a small charity, The Thika Alumni Trust, that helps build e-Learning Centres in schools in Kenya. Typically a 50 desk environment consists of four Raspberry Pi Internet-In-A-Box (IIAB) Servers and 50 Raspberry Pi Desktops. We also encourage schools, via Kenyan Raspberry Pi Code Club champions, to hold local code clubs. We provide help and support with experimental kits and any old IT equipment in our possession.

Old Raspberry Pi computers are wonderful for Kenyan school children to learn coding skills and also to build robots etc. Following on from the article by Rob Zwetsloot in *The MagPi* 107, in a similar vein I am appealing to any Raspberry Pi enthusiasts out there, who may have an old Raspberry Pi in their possession which they no longer have a need for, to send them to us and we will transport them to Kenya for use by school children in Kenya. You will be helping enormously in enhancing ICT skills of students in the developing world.

Harper via email

If you missed Rob's Final Word in the previous issue, he talked about how older Raspberry Pi still work just fine, and can be used for a variety of project types. Organisations like The Thika Alumni Trust are able to do amazing things with an older Raspberry Pi, so if you're not sure what to do with one, you can always check out their website here: **thethikaalumnitrust.org**. You can also email them at: **ttatenquiries@gmail.com**.
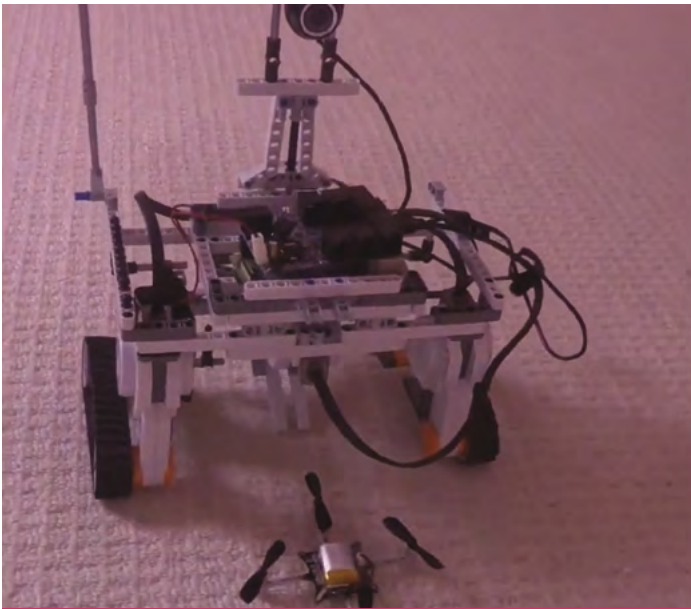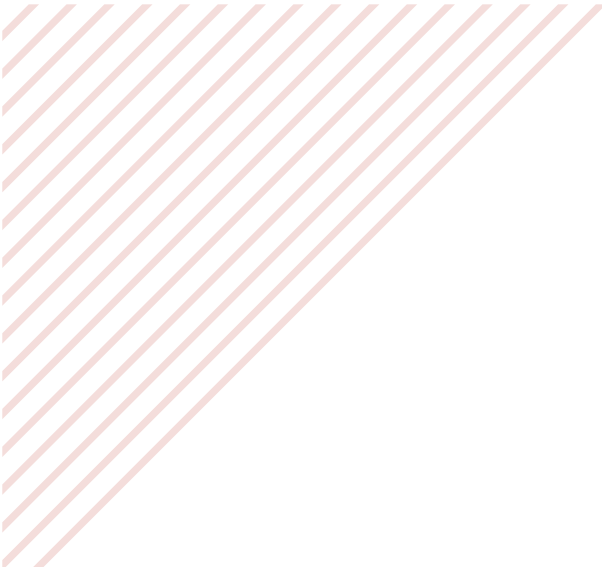
The Martians

The Martians

I read the article about Avra Saslow and her emulation of the Mars 2020 helicopter with interest.

I've made a short video of my 'homage' to Perseverance and Ingenuity. It is far, far less sophisticated, but tries to emulate rover and drone: magpi.cc/legomars.

Architecture is a Raspberry Pi 3B, LEGO Mindstorms, BrickPi, CrazyFlie, OpenCV, and Python.

Raspberry Pi controls the rover and provides the video feed – the OpenCV component currently runs on a laptop and communicates with Raspberry Pi over 0MQ. Porting the OpenCV code to Raspberry Pi is a future step.

Nigel via email

This is an incredible build – Raspberry Pi is perfect for OpenCV as well, so this should be an easy port. Keep us updated with what you're making as we'd love to cover it more in the magazine.

More Raspberry Pi 400

I've been wanting to get a Raspberry Pi 400 for a while now. However, with all the different keyboard types that were being released, I was waiting for a Swedish version. Is that on the cards, do you know? I'd be fine with a QWERTY keyboard. How long should I hold out for one?

Tuisku via Twitter

Good news: there are now some new variants of Raspberry Pi 400 including a Swedish one. It's joined by keyboard layouts for Portugal, Norway, and Denmark. A Japanese version is coming soon for people looking forward to that as well.

Check out more at the following blog post: **magpi.cc/new400layouts**.

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 #45

OUT NOW

hsmag.CC

WIN

A MARTY THE ROBOT V2!
IN ASSOCIATION WITH ROBOTICAL

"Marty the Robot is the best-value humanoid robot that can offer the breadth of learning progression from screen-free coding through to Scratch and Python" – Robotical

We reviewed Marty last issue and thought it was great, describing it as a deceptively powerful robot with a cute design. Here's your chance to win one…
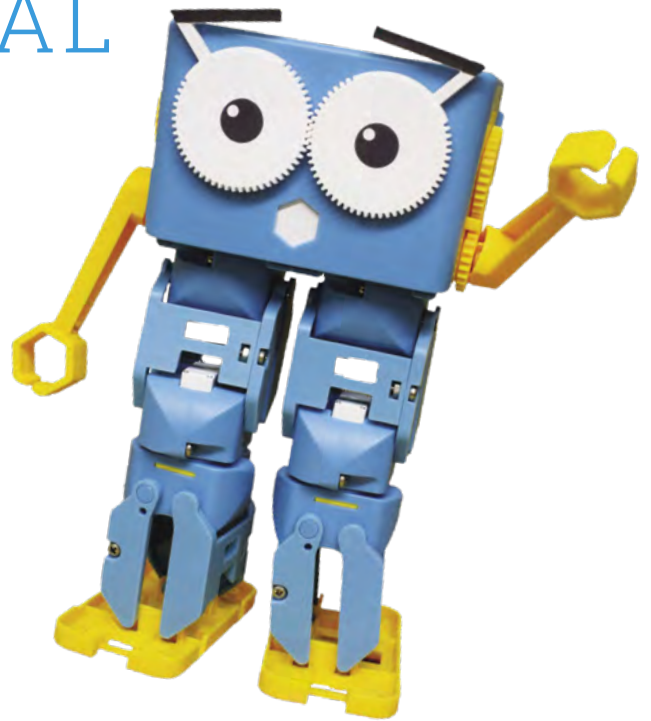
Head here to enter: magpi.cc/win | **Learn more:** robotical.io

Terms & Conditions

Competition opens on **28 July 2021** and closes on **26 August 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.

20
PICO
PROJECTS
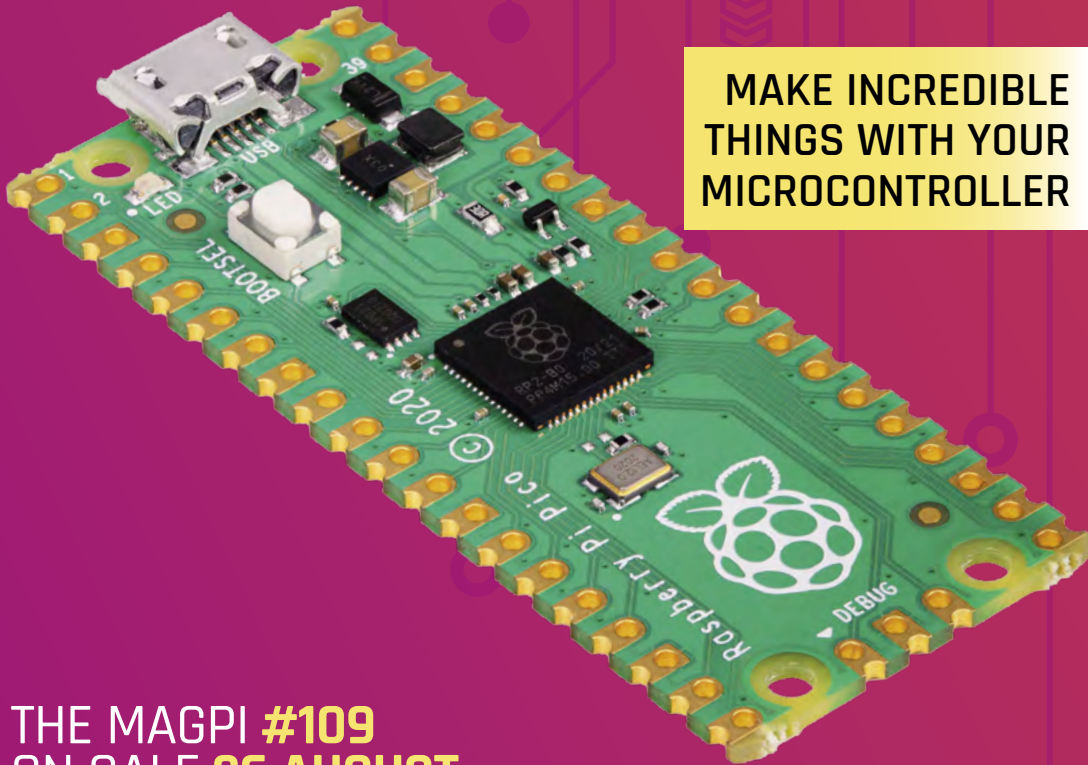
MAKE INCREDIBLE THINGS WITH YOUR MICROCONTROLLER

THE MAGPI #109
ON SALE 26 AUGUST

Plus!

Back to school with
Raspberry Pi

**10 amazing game
dev projects**

Build an
Android tablet

DON'T MISS OUT!
magpi.cc/subscribe

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

Our place in history

Computers go back a long way. By **Lucy Hattersley**

I was chatting to my in-laws recently about early computing. My mother-in-law was a programmer (but has never really used a computer). This was back when 'computer' was a women's job description.

My father-in-law also used to run simulations back in the 1970s on what turned out (after a long chat and some online digging) to be an IBM System/360 (**magpi.cc/ibm360**). He didn't personally program the machine; he wrote down instructions for the simulation on a form, another person punched the hole cards, and then the programmer ran them through the computer. The results typically came back a week or so later.

There followed a surprisingly informative chat about the history of computing covering all the usual bases: Charles Babbage, Ada Lovelace, Alan Turing, John von Neumann, and early proto-computers like the Difference and Analytic Engines, and orrery devices like the Antikythera Mechanism.

They talked with pride about Lyons LEO (Lyons Electronic Office, **magpi.cc/leo**), a computer I hadn't heard of but intend to learn more about. It was the first computer used in a commercial business setting. LEO was modelled closely on Cambridge's EDSAC, which I do know about – that was designed by Sir Maurice Wilkes, the person whose name adorns the office building I often work in.

They have parts and articles on LEO at The Centre for Computing History (**magpi.cc/tcch**) in Cambridge, which has now reopened and I can't recommend it highly enough. I'll be heading down there to take a look.

We tend to think of computing as ultra-modern and, obviously, forward-thinking. Yet there is the weight of history behind our technological toys. We had just visited Enderby's Wharf and, following a drink in Enderby House, there followed a chat about the history of underwater cabling. The first telegraph cable across the Atlantic was produced there, and much of the world's subsea communication cables were made in the area. Alcatel Submarine Networks is still based around the corner.

It often comes as a surprise to folks that most of the internet traffic isn't whizzing around in space, but bouncing along vast undersea cables. You can take a look at all the wires in this interactive map: **submarinecablemap.com**.

Learning how the mystical world of technology works helps us to ground (or even submerge, in this case) folks to reality. Rather oddly, we need to take the magic out of computing. To bring it away from the realm of "any sufficiently advanced technology is indistinguishable from magic", and into the practical world where computing becomes a tool we can control.

> **Rather oddly, we need to take the magic out of computing**
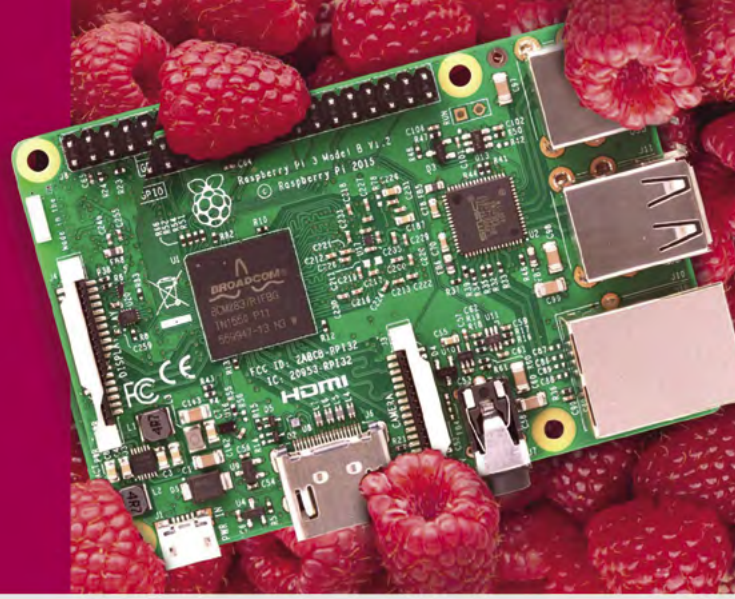
Raspberry Pi

It gave me great pride to show my in-laws Raspberry Pi and explain what each constituent part was. They were amazed by how small it was, especially Raspberry Pi Zero. Having this historical perspective helped me explain to them why Raspberry Pi was so important. It continues the UK's long tradition of designing, and building, its own computers. Raspberry Pi is vital to ensure the existence of a future generation of programmers, by putting the power of physical computing into the hands of young learners (of all ages). M

AUTHOR

Lucy Hattersley

Lucy is editor of *The MagPi* magazine and loves computers, but thinks we should have gone with 'difference engine' instead.

magpi.cc