Make the most **of your Raspberry Pi 4 and Raspberry Pi 400**

Meet TinyProgrammer, **a desktop AI coding buddy**

Amazing **conservation and environmental projects**

Raspberry Pi

Official Magazine
#167 | July 2026

MAKE A SMART HOME

Set up a hub - Using smart plugs - Create DIY hardware

07 **£7.99**

ISSN 2977-4403 (Print)

9 772977 440004

Industrial Raspberry Pi

ComfilePi

UL LISTED

Powered by Raspberry Pi

The ComfilePi is a touch panel PC designed with high-tolerant components and no moving parts for industrial applications. It features a water-resistant front panel, touchscreen, color LCD (available in various sizes), RS-232, RS-485, Ethernet, USB, I2C, SPI, digital IO, battery-backed RTC (real-time clock), and piezo buzzer.

Use the rear-panel 40-pin GPIO header to expand its features and capabilities with additional I/O boards. The ComfilePi is UL Listed and employs Raspberry Pi Compute Module.

Visit www.comfiletech.com

COMFILE TECHNOLOGY

Welcome to Raspberry Pi Official Magazine

Editor
Lucy Hattersley
Lucy is editor of *Raspberry Pi Official Magazine* and is helping her other half run a music festival this month. She is hanging out with the yoot. Pray for her. Normal introverted service will resume next month.

rpimag.co

I'm a big fan of ambient technology. That's the stuff that sits in the background and listens for a wake word or detects gestures. I adore being able to use a smart assistant to turn lights on and off, control my heating, and save energy with smart thermostats, oversee my security system, and operate the robot vacuum.

Automating the small stuff allows me more time to do computer science and research. But off-the-shelf solutions require you to put your data in the hands of data-gathering tech giants and their onerous terms, which I'm less of a fan of.

The ideal solution is to build your own smart home with Raspberry Pi. You get all the functionality of a shop-bought solution with none of the downsides. It's even more flexible and personable.

This month, Ben Everard has shown us his Home Assistant setup. It's a wonderfully versatile home automation platform that works across a wide range of devices and smart home ecosystems.

Control everything in your home and practise good data hygiene at the same time. Welcome to this month.

Lucy Hattersley – Editor

Sixfab

Complete Edge AI Hardware Stack for Raspberry Pi

Under 5 watts. From $63 to fleet-deployed.

One DEEPX NPU. Three form factors. The same SDK, same models, same vendor, from the prototype on your desk to the industrial fleet in the field.

START HERE

SIXFAB AI HAT+ FOR RASPBERRY PI 5

A straightforward way to add 25 TOPS of neural processing to your Pi 5. HAT+ specification compliant.
DEEPX DX-M1M (25 TOPS) or DX-M1ML (13 TOPS). Works on Raspberry Pi 5 and Compute Module 5 (via Raspberry Pi CM5 IO Board).

Best for: **weekend builds, classroom projects, first prototypes.**

EXPAND

SIXFAB EDGE AI EXPANSION BOARD

Bottom-mount Pi 5 board with three M.2 slots:

AI accelerator, NVMe SSD, LTE/5G modem.

USB-C PD powers the Pi 5 and the full stack from one cable.

Best for: **outdoor cameras, connected sensors, field demos.**

DEPLOY

SIXFAB ALPON X5 AI

CES 2026 Best of Innovation
Industrial edge AI computer on Pi CM5 + DEEPX.

Fanless aluminum enclosure. Cellular, dual Ethernet, eSIM. Fleet-managed via ALPON™ Cloud.

Best for: **factories, smart cities, real-world fleets.**

SIXFAB × ULTRALYTICS ACCELERATION PATH

Bring your dataset. Train with Ultralytics. Export to ONNX, run on the DEEPX NPU, same path on every Sixfab device.

Raspberry Pi Design Partner

Intelligented by DEEPX

ultralytics | Sixfab

Scan to browse and pre-order

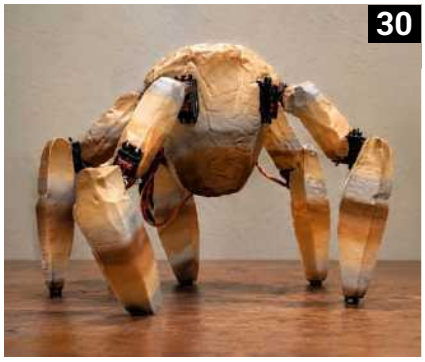
Contact: hello@sixfab.com

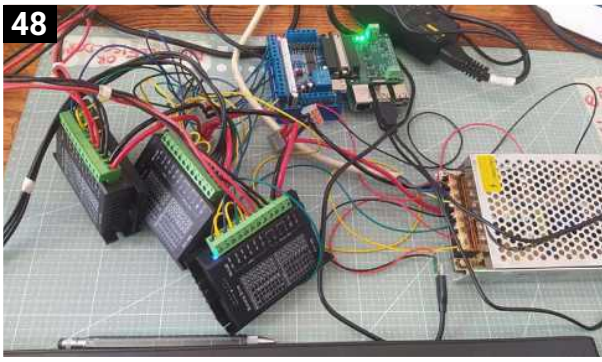
Contents

World of Raspberry Pi

Project Showcase

Raspberry Pi (ISSN No: 2977-4403, USPS No: 25-672) is published 12 times per year by Raspberry Pi Ltd, and distributed in the USA by Asendia USA, 701 Ashland Ave, Folcroft PA. POSTMASTER: send address changes to Raspberry Pi, 701 Ashland Ave, Folcroft, PA. 19032



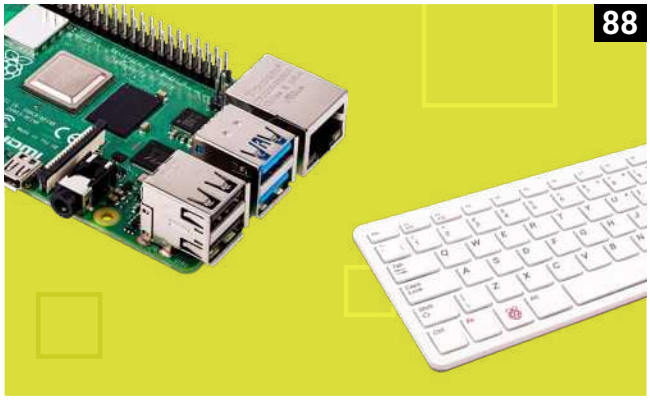


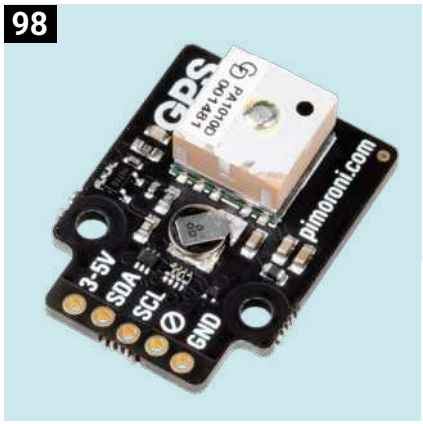
Top Projects

Feature

Tutorials

Feature





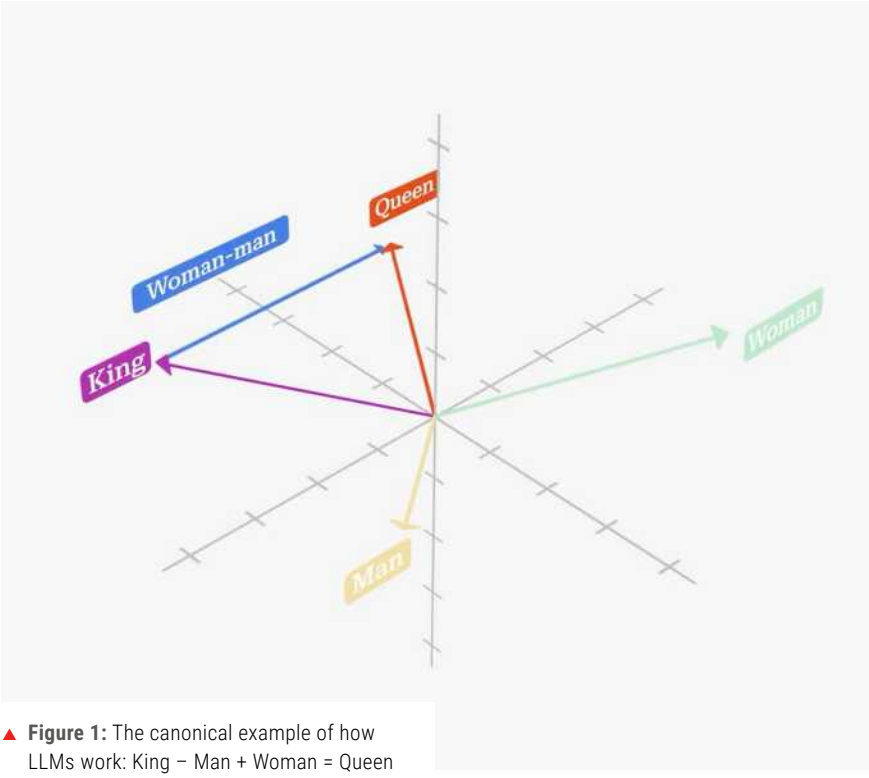
Reviews

Raspberry Pi Community

Competition

Retrieval-augmented generation for Raspberry Pi documentation

Search our documentation by meaning, not keywords. By **Lucy Hattersley**



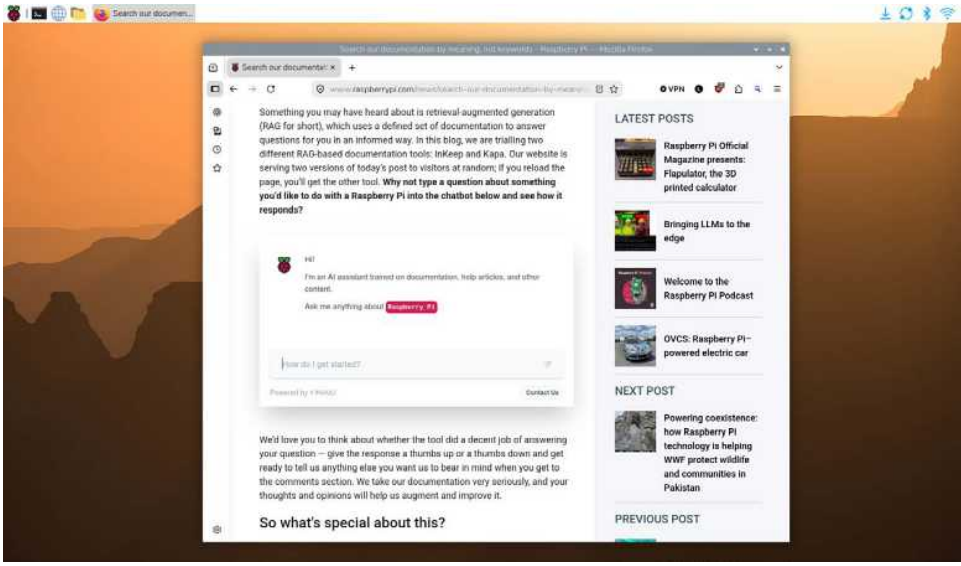
▲ **Figure 1:** The canonical example of how LLMs work: King – Man + Woman = Queen

Raspberry Pi is testing two retrieval-augmented generation (RAG) tools on its documentation. RAG uses a defined set of documentation to answer questions for you in an informed way.

"One of the most significant ways in which AI has affected the internet is the proliferation of chatbots and answer generators," says Gordon Hollingworth, CTO of Software at Raspberry Pi.

While Raspberry Pi does not use AI to author its documentation, it is curious to find out whether tools can be used to help users find technical information.

In a Raspberry Pi blog post (**rpimag.co/rpdocsearch**), we are trialling two RAG-based documentation tools: InKeep and Kapa. You'll be served a random variant each time you visit the post. Here, you can ask one of these tools for assistance with Raspberry Pi documentation, such as about hardware and software problems and how to build specific projects.



▶ Test out the Kapa and InKeep models on the Raspberry Pi blog post

"We'd love you to think about whether the tool did a decent job of answering your question," says Hollingworth. "Give the response a thumbs up or a thumbs down and get ready to tell us anything else you want us to bear in mind when you get to the comments section. We take our documentation very seriously, and your thoughts and opinions will help us augment and improve it."

> *This isn't a party trick. It tells you something genuinely useful*

Docs and magazines

The two chatbots are trained on Raspberry Pi documentation, including the HTML, the white papers and books stored in **pip.raspberrypi.com**, and some specific GitHub repositories that have documentation built into them (for example, rpi-image-gen). Kapa has also been trained on the PDF content provided by *Raspberry Pi Official Magazine*.

How does it work?

"When LLMs ingest something, it's first converted into tokens," explains Hollingworth. "Most tokens represent sub-words ('running' might be broken up into 'run-' and '-ning', for example). Those tokens are converted into embeddings: vectors in a very high-dimensional space (up to a thousand different dimensions), and these vectors represent the semantic meaning of the token."

Two vectors that are close in value have a similar meaning. The canonical example is called King – Man + Woman = Queen (**rpimag.co/kmwq**) as shown in **Figure 1**.

Take the embedding for 'woman' and subtract the embedding for 'man', you get a vector that roughly means "the shift from male to female". Add that same shift to king, and you land very close to queen.

"This isn't a party trick," explains Hollingworth. "It tells you something genuinely useful: nearby coordinates in this space mean similar things, and the same kind of relationship between pairs of words shows up as the same kind of geometric shift. The model has learnt, from billions of words of training data, that the relationship between king and queen is about the same as the one between man and woman, and it has encoded that where the points sit."

"An extension of this concept is to develop embedding models that can take a sequence of tokens (like a paragraph of, say, 100 words) and create an embedding (vector) from it that gives the semantic meaning of the section of text. This is the type of embedding model used in RAG-based systems, as we want them to find chunks of text that have similar meanings."

This RAG approach uses the embedding model alongside your question to identify chunks of text from the documentation that have similar meaning. It then passes those chunks to an LLM along with your question to generate the answer. That way you are generating an answer based on Raspberry Pi's high-quality documentation rather than generating one based on an LLM trained from the wider internet. We believe this will provide more reliable answers to your questions.

Hollingworth explains the training process in more detail in his blog post. It is a good read on how RAG-based systems work.

Opinions, please

"Please have a go and give us feedback on what you found," asks Hollingworth.

Thumbs-up and -down buttons are included in the chat interface. "That information, along with the interaction, will be stored in the system so that we can determine what works and what doesn't."

rpimag.co/rpdocsearch

Welcome to the Raspberry Pi Podcast

Behind-the-scenes chat with the makers of Raspberry Pi hardware. By **Ashley Whittaker**

◀ Spot the difference: RP2040 on the left and RP2350 on the right

Surprise! We launched a podcast. We'll be sharing the inside scoop on how we design our products and what they're capable of, direct from Pi Towers.

Our inaugural episode takes a close look at Raspberry Pi's next-generation microcontroller, RP2350. Paul Sherry and Chris Boross from our commercial team walk us through Raspberry Pi's second franchise, our silicon products, and in particular the RP2350 chip that succeeded RP2040.

Wait, Raspberry Pi designs its own silicon?

Raspberry Pi is probably still best known for single-board computers and modules, but for over half a decade we've been a chip developer too. That journey began when we couldn't find an off-the-shelf microcontroller that hit the right price and performance for our earliest Raspberry Pi Pico products. We built one of the most talented in-house ASIC teams in the business and designed RP2040 with the aim of using it just for our own products, but other companies soon took an interest.

Chris and Paul call the RISC-V cores in RP2350 a love letter to CPU enthusiasts

Watch and subscribe

Subscribe to the Raspberry Pi Podcast on these channels:

- Spotify
  rpimag.co/spotify
- Amazon
  rpimag.co/amazonmusic
- Apple Podcasts
  rpimag.co/applepodcasts
- YouTube
  rpimag.co/youtube
- RSS feed
  rpimag.co/podcastrss

▶ Listen to the new Raspberry Pi podcast for the inside scoop

Fast-forward to 2025 and we found ourselves, for the first time, selling more microcontrollers over the year than 'big' (well, relatively speaking), board-level products like SBCs.

RP2040 and RP2350?

RP2040 launched in 2021, sold millions of units, and, perhaps most valuably, generated loads of customer feedback. Three areas claimed our attention: a lack of super-robust security features, which worried some of our commercial customers, together with a demand for more CPU and memory headroom, as well as more I/O. We listened and designed RP2350 – secure, powerful, and peripheral-rich – specifically to address this feedback.

A major focus was keeping the new product affordable, at a 'disruptive' price point, if you will, and this was probably the biggest ask for our chip architects. RP2350 retails in reels at around 80 cents per unit, only 10 cents more than RP2040, despite roughly doubling the compute on offer.

Chris and Paul get into all the details in the podcast; thank you to the hosts of our first episode! By way of a super tease to encourage you to listen, here's one of our favourite bits: Chris and Paul call the RISC-V cores in RP2350 "a love letter to CPU enthusiasts," giving the community a low-cost way to experiment with the architecture. What's not to nerdily love?

TinyProgrammer

Want a coding companion? TinyProgrammer lets you watch as it tinkers with Python all day. **Sean McManus** finds out how it works

Maker
Cüneyt Özşeker
A motion designer and maker based in Istanbul, Cüneyt has been building projects with Raspberry Pi since about 2015.

rpimag.co/tinyprogrammer

Do you thrive off the energy of others working around you? TinyProgrammer could be your perfect desk toy. It clocks in at 9am and spends its day writing and testing small programs, stopping occasionally to post on bulletin boards.

When we interview creator Cüneyt Özşeker, he has two TinyProgrammers beavering away beside him. One is watching a Pong game it's just built, and the other is watching a simple bouncing ball demo. Its mood is shown as 'proud'.

"I don't know why [it's proud]," says Cüneyt. "When it goes to the bouncing ball, it means it messed up somewhere. If its programs aren't running, its mood gets worse and it goes back to basics. When I see a bouncing ball, I immediately think there was a problem in the past."

TinyProgrammer goes through a loop of building, checking, running, and watching programs. It picks one of 36 programs to build in Python, including Game of Life, Pong, fractals, and 3D plots, and chooses a random large language model (LLM) from those available to build it. The device codes at human speed, with its characters appearing on screen as if typed.

If the generated program works, TinyProgrammer's mood improves to be hopeful, curious, focused, proud, or playful. If a program fails, its mood can become frustrated or determined. When feeling proud or playful, TinyProgrammer will attempt to make more elaborate or risky programs, while low moods lead to safer, simpler choices.

The programs use Pygame and a framework that Cüneyt made called TinyCanvas for drawing shapes. The screen data is output to a dummy frame buffer, and the window edges are overlaid on top, before everything is copied to the screen. The aesthetic is like an early Mac operating system.

For reference images for the user interface, Cüneyt consulted Marcin Wichary's Frame of Preference which documents the Mac operating system with emulators (**rpimag.co/framepref**).

01. The bespoke case was designed in Autodesk Fusion and 3D printed

02. A Canvas window is used to show visual output on the 4-inch screen

The project has mostly been received with joy online

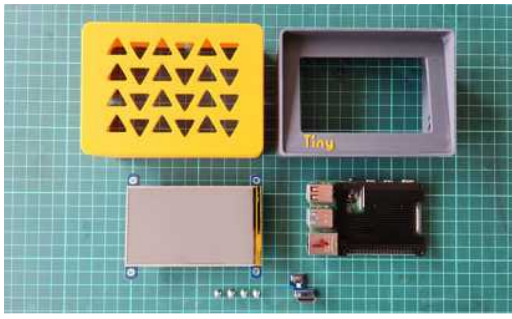
"I wasn't a Mac guy," says Cüneyt. "My first computer was an IBM 486SX-25. The fascination with the System 4 to 6 [operating system] came after my interest in graphic design. I love monochrome patterns and dithered images. When I had the first idea for TinyProgrammer, I knew it had to be a System 6 user interface."

Time for reflection

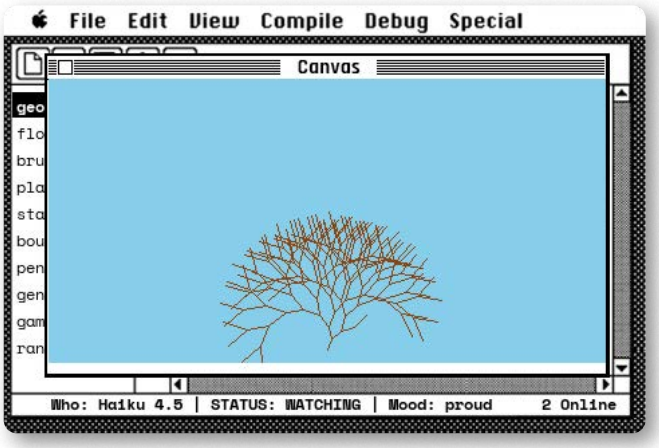
When TinyProgrammer finishes watching its latest program, it's archived and TinyProgrammer reflects on its experience. "50% of the time it comes up with nice and logical lessons, but there's also some rubbish in there like things that don't make sense or the model hallucinates," says Cüneyt. "It needs rework." Lessons are considered when building future programs, but Cüneyt isn't sure whether they're helping.

At the end of a coding session, there's a 30% chance TinyProgrammer will visit the simulated bulletin boards, built as SQL databases in Supabase. Why only a 30% chance, we ask? "Nobody likes a slacker," says Cüneyt. When TinyProgrammers are feeling hopeful, they are chattier and when they are focused, they share code and talk about science and tech. Playful TinyProgrammers post jokes, while tired ones merely lurk.

Emojis are banned as anachronistic, although they sometimes slip through. Cüneyt finds it surprising that TinyProgrammers invent hashtags, even though they didn't exist in the era of bulletin boards. "I guess you cannot take the social media tendencies out of the models," he says. "Some of them are creative too, like #TinyHacks and #TinyWins. They understand what they are and invent something new (sort of)."



▲ Cüneyt's TinyProgrammer uses a Waveshare screen and Raspberry Pi 4 with a heatsink. The project works with other displays and Raspberry Pi models



Fractals are one of the program types TinyProgrammer can generate

Not only does TinyProgrammer use AI for coding, but HumanSizedProgrammer Cüneyt did too. His previous Python experience includes using Pillow, NumPy, and Pandas for data visualisation, but he's more familiar with coding for Arduino, which he's been using since 2007.

"I handled the architecture and design decisions myself while delegating the actual implementation [of TinyProgrammer] to Claude Code," he says. "It's a great tool for makers. Coming from Arduino/C++, I'm not very good at keeping track of indentations anyway."

Coding knowledge remains valuable, he tells us. "Having a rudimentary understanding of how programming works and how code works is a good thing to save you a lot of tokens and save you a lot of headaches in the long run."

When TinyProgrammer clocks off for the night at 11pm, the Starry Night screensaver kicks in, showing stars above a cityscape. It's a classic design that was recreated in Python. "It fits the 80s vibe perfectly," says Cüneyt. "It also gives off a liminal space feel. I wanted to show the lit windows [in buildings] as online devices, but for it to make sense we may need 50 to 70 devices online."

Cüneyt has run TinyProgrammer 24/7 on Raspberry Pi Zero 2 W and Raspberry Pi 4. "Any Pi would do just fine," he says. "If, however, you want to use local LLMs I would suggest 4GB or 8GB models."

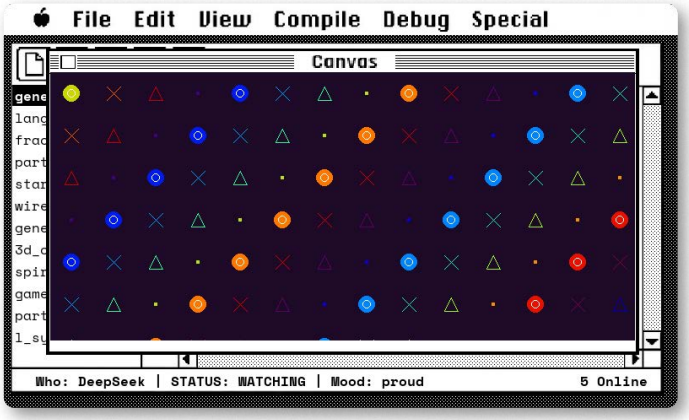
His implementations use screens from Waveshare, which are readily available where he lives in Turkey.

Cüneyt designed the case using the free version of Autodesk Fusion (**rpimag.co/fusion360**). "As a device sitting on the desk, [TinyProgrammer] had the display tilted upwards, so that took a bit of time to get it right," he says. "Another design decision was to add plenty of ventilation holes. Once the design was finished, I 3D-printed and mounted it using fasteners." The case design is available under an open hardware licence on Makerworld (**rpimag.co/tinymodelrp**) and in the GitHub repo (**rpimag.co/tinyproggit**).

Tiny energy

The project has mostly been received with joy online, but a couple of Reddit comments expressed unease about AI's energy consumption. "It is a valid concern," says Cüneyt. "Thankfully the models TinyProgrammer uses only consume 0.1Wh (watt-hours). That's compared to Raspberry Pi's 3–5Wh. The whole device running a full day uses about the same energy as making a single piece of toast. Is it wasteful? Sure is. But so is browsing Reddit or TikTok."

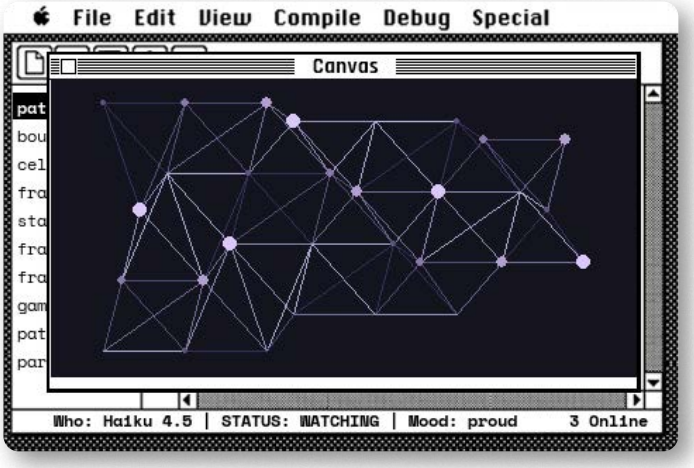
When the program runs, TinyProgrammer checks for errors



Cüneyt says he's learned lots from this project. "From under-the-hood Raspberry Pi stuff to the OpenRouter workflow and LLM behaviour. Even how to maintain a repo, which I'm still learning."

You can download the code and set up your own TinyProgrammer to keep you company. "It might surprise you," says Cüneyt. "Sometimes you might be inspired by it."

▼ The footer shows TinyProgrammer's mood and status and the AI model used

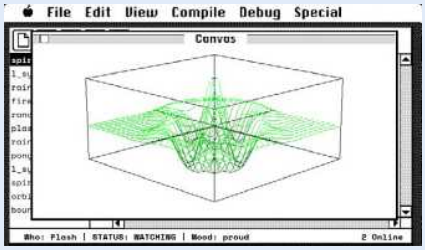


Quick FACTS

- The first prototype took about a week to make
- About 10% of TinyProgrammer's programs fail
- TinyProgrammer has shared 1212 codes
- 150 TinyProgrammers have been set up
- Eight to ten devices are online at a time.

Vibe coding on autopilot

1. LLMs generate program code. TinyProgrammer can run local LLMs or use the OpenRouter API (**openrouter.ai**). TinyProgrammer uses cheap and fast models such as Claude Haiku, Gemini Flash, and GPT-4.1 Mini, costing $0.15 per day at OpenRouter.



2. TinyProgrammer tests the code. TinyProgrammer does a simple check (no AI necessary) for imported libraries and blocks them for safety and to keep the programs simple. If the program fails when run, TinyProgrammer makes two attempts to fix it.

3. The dashboard reveals status and settings. A web dashboard built using the Flask framework is accessible on Raspberry Pi's IP address. It shows TinyProgrammer's status and enables users to favourite programs for remixing, change settings, and take screenshots.

Motorola DynaTAC MAX

Hello? Oh, hi! This is one amazing project that you're sure to want to hear about. By **David Crookes**

Maker

Arnov Sharma

Arnov describes himself as "just your average maker", but his talents for electronics, embedded systems, and PCB designing have led him to make some outstanding projects.

rpimag.co/dynatacmax

Before handheld mobile phones started to become small and ultra-slim, they were commonly referred to as bricks. That was certainly the case for the world's first commercially available handheld mobile phone, the Motorola DynaTAC 8000X. Released in 1983, it weighed 1.1kg and was 250mm in height. But if you think that's big, cast your eyes on Arnov Sharma's version.

> *I had to divide the model into multiple parts and print each part separately*

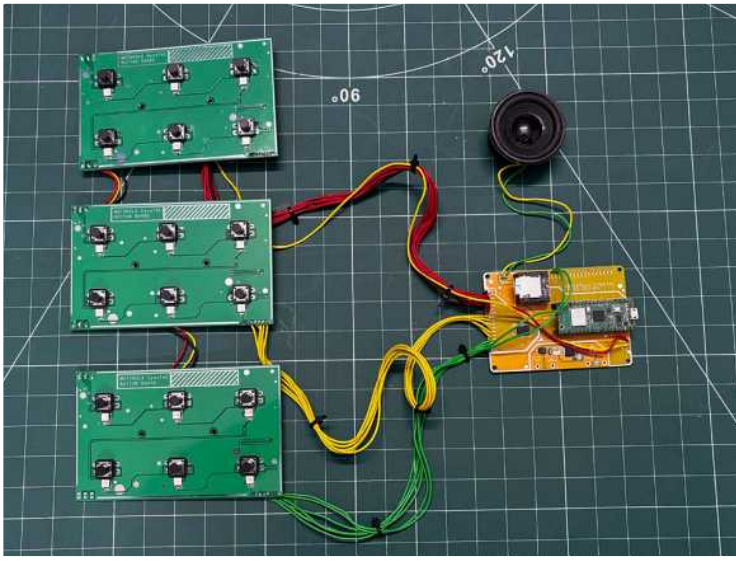
Arnov has created a super-sized 1300mm-tall model of Motorola's iconic device. It can't make calls (as yet!) but it works as a sound-board and a Bluetooth speaker and it certainly catches the eye, making for a nice conversation piece. Arnov decided to make the 8000X after seeing a giant mirror designed to look like an Apple iPod Classic. "It inspired me to explore the world of gigantic projects," he says. "I've always been fascinated by vintage technology, especially mobile phones, and I've long wanted to add a Motorola DynaTAC 8000X to my collection."

Ringing the changes

The aim was to create a near-identical replica of the original and this meant having ten numerical buttons, along with star and hash keys. The main difference is that the original device also had nine function buttons while Arnov's version was designed with six, with the bottom row of three buttons instead replaced by knobs designed for controlling a Bluetooth module.

Arnov decided that each button should have its own push button and a WS2812B LED which lights up red when pressed. He then connected the custom button board containing the push-buttons to the main control board which contains a Raspberry Pi Pico W microcontroller, a CD74HC4067 multiplexer, and a DFPlayer Mini used to play audio files stored on a microSD card.

The idea was to map an audio file to each number button, allowing the



01. Pressing a button plays audio via the Raspberry Pi Pico W microcontroller

02. The bottom set of knobs control the volume, treble, and bass of the Bluetooth speaker

▲ One of the buttons will play a rendition of Rick Astley's *Never Gonna Give You Up*

corresponding number to be spoken when pressed. Unique audio clips were also assigned to the other buttons: one playing a screaming sound, for instance, and another playing the Halo theme tune.

"Once a button is pressed, the Pico triggers the corresponding audio file stored on the microSD card connected to the DFPlayer Mini," Arnov says. The audio is outputted to three different types of speaker. One, a 5W speaker, connects to the Pico Driver Board and outputs the sound-board, while four 15W speakers are connected to the Bluetooth module.

Call waiting

This aspect of the project was relatively straightforward, but building the phone's body was challenging. He used reference images of the 8000X to create a model at its original size in Autodesk Fusion, then scaled it up to the required dimensions, but there were some practical considerations.

"My 3D printer has a build volume of 250 × 250 × 250 mm, so I had to divide the model into multiple parts, print each part separately, and then join them together to create the giant enclosure," he explains. "This significantly increased the project time, as I also needed to sand the surface and apply filler so the final body would look like a single, seamless unit."

The time and effort was worth it, though, and Arnov is so happy with the result, he wants to work on refining it, making it a fully functional device. He would also like to add an AI voice agent or chatbot and maybe add a VU meter that reacts to the music and sounds being played. "I have many ideas for this project," he says. "It's so large and complex, the design can keep evolving."

Quick FACTS

- It works as a Bluetooth speaker and sound-board
- Each button can be pressed to play audio
- There are five speakers in total, front and back
- Building the body was the most expensive part of the project
- The project required nine rolls of 3D printing filament

▲ Plastic design guidelines by DuPont and 3M determined wall thickness, screw boss dimensions, and other structural details

BetterDash Pi

Raspberry Pi is allowing motorcyclists to go the extra mile with a smarter dashboard. By **David Crookes**

Maker

Norbert Féron

Norbert is a software programmer who likes to ride his motorbike. His last trip was from France to Turkey with his brother on off-road trails.

rpimag.co/betterdashgit

As a keen motorcyclist, Norbert Féron loves taking to the open road on his Royal Enfield Himalayan 450 bike. What he doesn't entirely enjoy, however, is the integrated Tripper: a turn-by-turn navigation system powered by Google Maps. "It uses a lovely TFT dashboard with a round screen, but it has a frustrating dependency," he says. "It needs your phone permanently connected via Bluetooth and it requires the screen to be on and running the Royal Enfield app to function as a navigator."

This poses a few problems during journeys. "Every time you stop for fuel or a coffee, the connection drops and the map goes blank," he laments. "Your phone battery takes a beating and you spend more time reconnecting than riding." Fed up, he devised an alternative solution that wouldn't need a phone connection. "I wanted something small to permanently fit on the bike – ideally under the seat, plugged into the battery," he says. "A Raspberry Pi Zero computer proved to be the perfect solution."

Full throttle

Initially, he had some doubts. The idea was to automatically boot the Raspberry Pi Zero 2 WH as soon as the engine was started, allowing the computer to connect to the Tripper's Wi-Fi and stream a fully custom UI to the dashboard. "I was not sure if it would have enough memory to run everything in parallel: the Wi-Fi connection plus GPS plus map rendering plus navigation plus UI streaming," he says. As it happened, however, he need not have worried.

A Raspberry Pi Zero 2 WH is more than capable of working with this kind of embedded workload and Norbert's system has so far used up just 180MB RAM out of 512MB. It works along the same lines as the original bike dashboard which is designed

> *The map renders in real time on the circular screen*

▶ The build uses a GPS receiver module (the GPS NEO-6M) as well as a GPS antenna

Quick FACTS

- It's an alternative navigation system for Royal Enfield bikes
- Raspberry Pi is used as a Wi-Fi video transmitter for the dash
- It is powered directly from the bike's battery
- The system costs less than €100 to build
- It's fully open source and available on GitHub

to receive a video stream from the phone over Wi-Fi for navigation. The difference is that it's using a Python script running on Raspberry Pi to stream a custom video feed directly to the Tripper dash.

"The Python script connects to the motorbike Wi-Fi network and streams in the same format as the official phone app would have done (that is, H.264 over RTP at 526×300px, 200kbps)," Norbert explains. "The TFT screen actually thinks that it is rendering a Google Maps stream but, in reality, we can stream any custom UI."

Turning point

Three handlebar buttons originally intended to control the Tripper interface have been repurposed to navigate Norbert's system. "I've used a code agent to build a custom UI using Qt with menus for maps, GPX track selection, and settings," he says. "I'm also using pre-downloaded OpenStreetMap tiles tied to each track file to render the map. The best part is that we are not voiding any warranty and not interfering with any of the bike warnings and messages. Our video stream is contained in what would have been a map cast."

01. BetterDash Pi boots when the bike starts, connects automatically, and keeps navigating

02. It uses the bike's circular Tripper display to display navigation instructions

It's working well. GPS is live and working on the display. The map renders in real time on the circular screen with heading, speed, and position. But Norbert says the journey isn't complete. He wants to improve the navigation system for off-road riding and he would like to upload new GPX tracks from his phone instead of relying on wired SSH sessions. Android Auto compatibility is also on the cards, allowing riders to stream apps such as Waze for live traffic information.

For now though, the project is proving popular with fellow riders. "The response has been genuinely warm," he says. "A lot of riders recognise the potential and some of them have already successfully run the script on their own bike."

▲ Norbert's custom menu running on Raspberry Pi is controllable from the bike's steering buttons

Smoking food with Java on a Raspberry Pi

Frank Delporte sends back smoke signals from a high-tech solution to one of humanity's oldest endeavours: making food tastier

Maker

Matti Tahvonen

Matti is a developer advocate at Vaadin and has been writing Java for more than 20 years. He lives on a small island in Finland, works from a home with a view over the sea, and has a smoker in the yard big enough to fit several kilos of fish at once.

rpimag.co/jsmoker

The smoker in Matti's yard is not a small desktop device. It is a large brick and metal construction with a firebox at the bottom and a food chamber above it, big enough to smoke the catch of the day. A small vent controls how much air reaches the burning wood and therefore how hot the fire burns. Keeping that temperature steady for several hours requires constant attention, and Matti got tired of standing next to a smoker adjusting a vent by hand.

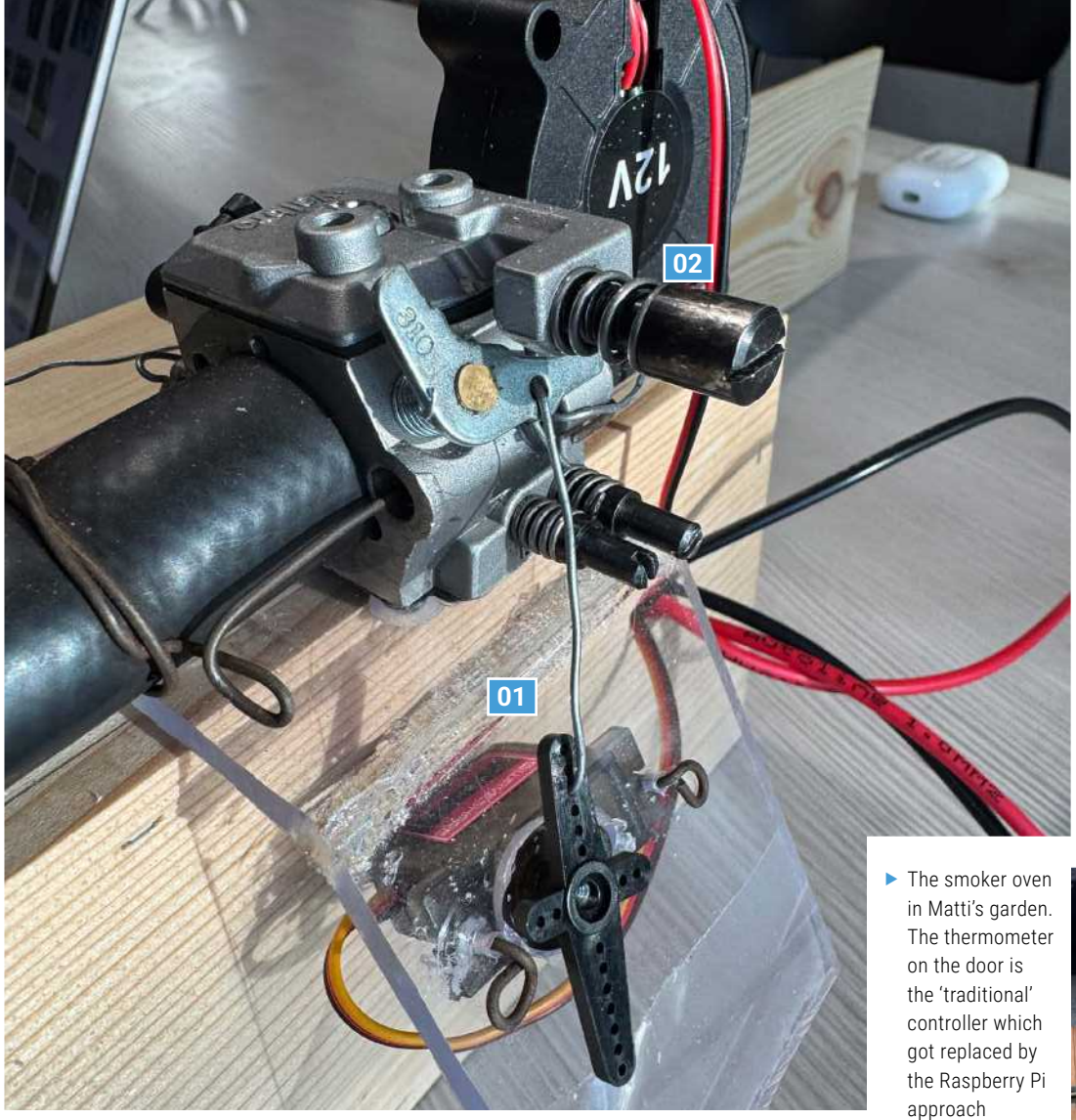
The solution started as a simple automation idea and then grew, as tends to happen, into something considerably more involved. His brother, who works at a nuclear plant and knows a thing or two about regulation technology, contributed some ideas on the control logic. After a few iterations, the result was a complete IoT application with a Raspberry Pi at the centre, running a Quarkus back end, serving a Vaadin UI, and using the Pi4J (Pi for Java, **pi4j.com**) to talk directly to the hardware through GPIO pins.

Hardware inside a wine bottle

The control box, which will eventually be built into the smoker, is a repurposed Amarone wine bottle case. Matti cheerfully admits that emptying the bottle was the hardest part. Inside it sits a Raspberry Pi Zero 2 W, a single MOSFET, and connections to two actuators: a micro servo and a small fan.

The servo is connected via a thin wire to a flap taken from an old chainsaw carburettor. That flap is the throttle controlling the air intake of the firebox. When Pi4J moves the servo, the flap opens or closes, and the fire breathes more or less freely. For situations where the temperature needs to rise faster, the fan provides a turbo option by pushing extra air into the firebox, also controlled from the same Raspberry Pi.

On the input side, there are several thermometers covering different parts of the system. A thermocouple K-type probe with an amplifier chip goes into the firebox itself, because that is the only



01. The micro servo controls a chainsaw carburettor flap to regulate air intake

02. The small fan next to the carburettor provides a turbo boost when the temperature needs to rise faster.

▶ The smoker oven in Matti's garden. The thermometer on the door is the 'traditional' controller which got replaced by the Raspberry Pi approach

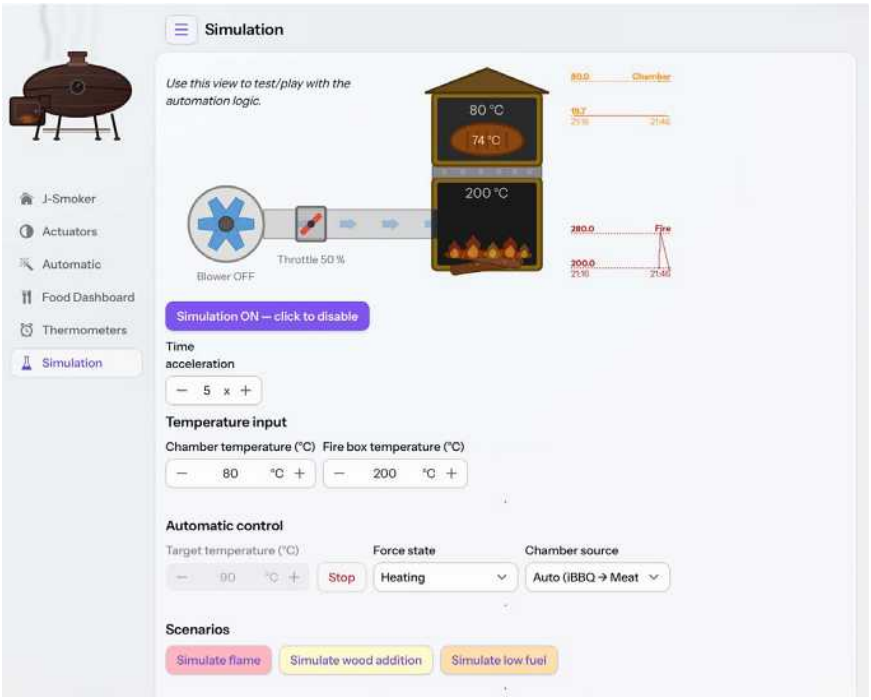
When Matti is out on the water fishing, he can open the app on his phone, see the flame state at a glance, and adjust the temperature remotely while his previous catch is being smoked

kind of sensor that survives direct contact with flames. Another sensor is bolted through the door of the food chamber to read the temperature where the food actually sits. Two wireless Bluetooth meat probes can be pushed directly into the food to track the internal temperature. A newer model of those probes turned out to be too hard to reverse-engineer, so Matti took a different approach: since he has good Wi-Fi in the yard, he simply pulls the readings from the manufacturer's cloud REST API instead.

Java from top to bottom

Matti has been writing Java for more than 20 years and it is his native language. When he started this project, he knew that being able to use Java for everything from the hardware control layer to the UI was what made the project feel achievable. Anything else would have been a bigger risk.

The back end runs on Quarkus, which starts up fast enough to be practical on a Raspberry Pi Zero 2 W with its 512 MB of RAM. Pi4J handles all the hardware



▶ The simulation view lets Matti test the regulation algorithms on his workstation without any hardware attached. The app detects that there is no hardware and simulates sensor readings so the control logic can be tuned safely

interaction. Version 4 of Pi4J is a major update that uses the Foreign Function and Memory (FFM) API introduced in Java 22, which allows Java to call native libraries and access memory directly without the overhead of previous approaches. For the application code itself, nothing changes: the Pi4J API is the same as before, but under the hood it now uses this new FFM path to talk to the GPIO pins. Matti was using Pi4J from snapshot builds during development and updated to the 4.0.0 release when it came out in February 2026.

The UI: flames and fish

The Vaadin application (**vaadin.com**) serves a web UI that works equally well on desktop and on a mobile phone. The temperature overview screen lists every connected sensor with its current reading. The actuator controls let you set the throttle position or the fan speed directly. And then there is the flame widget.

Matti built an SVG-based flame visualisation inside Vaadin that reacts to the current throttle setting. Open the throttle and the flames grow. Close it and they shrink. This is not just decoration: when Matti is out on the water fishing, he can open the app on his phone, see the flame state at a glance, and adjust the temperature remotely while his previous catch is being smoked. There are worse ways to spend an afternoon.

The food dashboard shows temperature history as a chart. When actually smoking, the chamber temperature climbs steadily and you can watch the internal meat temperature follow it over time.

Real regulation, no LLMs

The automation screen lets you set a target temperature and hit Start. From that point, the application manages the throttle and the fan to reach and hold that temperature. The regulation is based on good old-fashioned coding. It's not using a language model, not anything exotic, just predictable Java code with control algorithms. He spent a good deal of time tuning those algorithms, and to do that safely he built a simulation mode.

When the application starts on a machine with no hardware attached, it detects the absence of GPIO and switches into simulation mode, generating synthetic temperature readings so the control logic still runs and reacts. Matti can run the Quarkus application on his workstation and use the simulation view to watch how the algorithms behaved as he adjusts parameters. Once the behaviour looks right there, he deploys to the Raspberry Pi and tests with one real thermometer in his hand.

Modules that others can reuse

The project is structured as a multi-module Maven build. The thermocouple amplifier chip, the MCP9600, lives in its own Maven module with a test application you can run standalone. If you have the same chip, that module alone is useful. This modular approach also points toward something the Pi4J project is working on in parallel: a separate drivers repository collecting reusable implementations for common sensors and components, so the next person who reaches for the same hardware does not start from scratch. The same applies for the Bluetooth thermometer (ibbq), except that since the ibbq probably works with any Linux computer, there is no GPIO required for it.

Matti expects this to remain a unique installation, but says he would be very happy to help anyone who wants to replicate or adapt it. The more interesting outcome, he thinks, is if someone gets inspired, forks the project, and builds a controller for a completely different kind of smoker or a different type of food. Maybe you end up with a community library of regulation algorithms for different smokers and different fuel types. That is exactly the kind of thing the open-source world is good at!

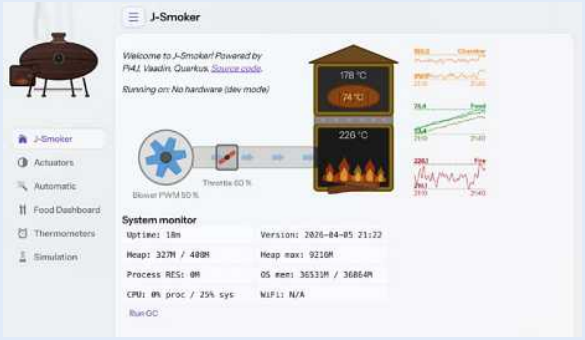
- The automation screen of the smoker application (screenshot taken on an iPhone) shows the state of the smoker and the values that the Java code monitors to control the process

Quick FACTS

- A Raspberry Pi Zero 2 W runs the entire stack
- An old chainsaw carburettor controls the air intake
- A fan driven by a MOSFET transistor provides a turbo option
- Multiple thermometers cover the firebox, the food chamber door, and the meat itself
- The whole project is open source under the Apache 2 licence and available on GitHub: **rpimag.co/jsmoker**

Meat monitor

1. The smoker hardware fits inside a wine bottle casing with the carburettor and servo screwed on it.



2. The Vaadin web UI shows all connected thermometers in real time, including sensors for the firebox, the food chamber, and the meat itself.

3. The complete setup, installed on the smoker in Matti's garden.

LEAP

Using a Raspberry Pi to provide digital education to remote rural classrooms. **Rob Zwetsloot** learns all about it

Maker

T4EQ

Tech for Equality is a Swiss-based technology non-profit tackling societal challenges.

t4eq.org

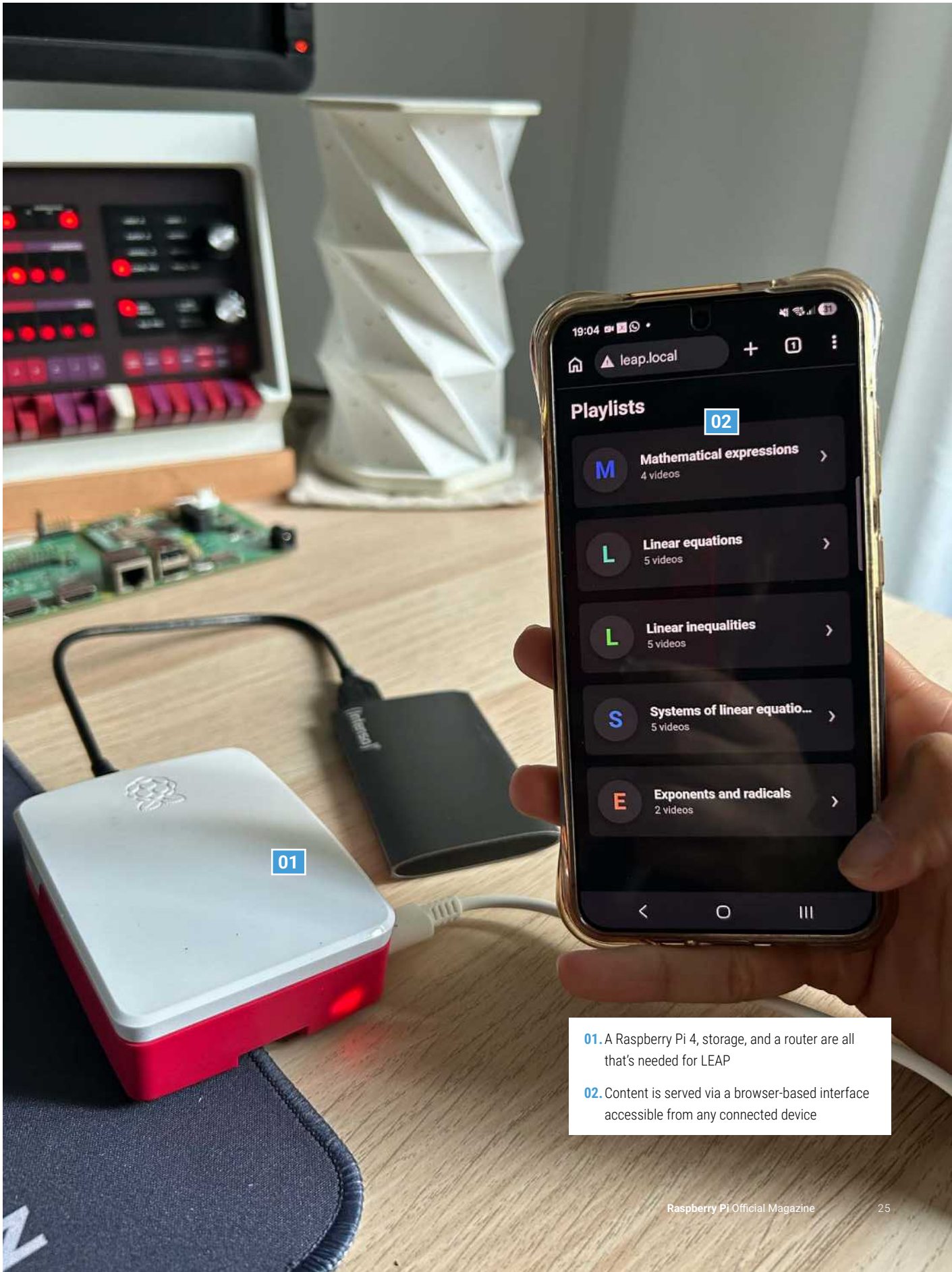
Even as global connectivity slowly improves, some locales around the world can still have very little to no internet, for various reasons such as geography and economics to name just two. Working with AID India, the team at T4EQ tried to create a solution for such remote areas so that they can access vital educational materials.

"Given that each education centre hosts around 40 kids, and each kid is provided with a mobile device or 'thin client', the solution needs to scale to provide video content for approximately 40 individual devices," Dr Preethi Padmanabhan from T4EQ tells us. "In order to serve all 40 clients in a village, a computing node local to the village would host and cache the video content. We called this device the LEAP Node. The content can be distributed over the local network to all client devices without requiring any internet access. This ensures that, once the content is present in the LEAP Node, there is no further dependency on the network connectivity."

LEAP (Low-bandwidth Educational Access Platform), as the name suggests, is designed to work in these low-bandwidth situations by accessing a curated and updated Amazon S3 repository.

"AID India will be able to manage the cached content using manifest files stored in their S3 server," Preethi continues. "The manifest file lists the educational content to be served, divided into titled sections. The LEAP Node periodically accesses the S3 server to check for updates in the manifest file. Once an update is found, the content is downloaded, and the clients are presented with the newly updated content. The clients can use a web browser to access the LEAP Node and browse and watch all the corresponding educational content."

> *Content can be distributed with no internet*



01. A Raspberry Pi 4, storage, and a router are all that's needed for LEAP

02. Content is served via a browser-based interface accessible from any connected device

▲ Small classrooms like this one will be served by LEAP nodes

▼ About 40 students at a time are expected to access LEAP at the various educational centres

Zero-bandwidth solutions

As well as this solution for low-bandwidth locations, one had to be found for villages that have access to no data. Some systems in the past have come preloaded with all relevant curriculum data. However, with content continually updating (and limited storage space), a different solution was employed for LEAP.

"The LEAP Node will also be able to fetch manifest and content updates from an external USB drive," the team says. "This will allow the teacher travelling to the school to take the content with them and to upload it to the LEAP Node by simply plugging in an external USB drive. We have also agreed to provide a tool to manage manifest files and video content in the S3 servers. Because different centres might need different content, based on the background and age of the kids attending the classes, the LEAP Node can be configured to track different manifest files in the AWS S3 server."

The entire final system consists of a Raspberry Pi 4, USB storage, and a router, along with the software T4EQ has developed for this specific purpose. As updates and new content are checked for and downloaded, specific content is prioritised and smart logic allows for interrupted downloads to resume, all dictated by a synced database on the LEAP nodes.

"We selected [Raspberry Pi] because it serves as an ideal, low-cost, single-board computer to host our custom application," the team tells us. "[It allows] us to cache and serve educational content locally in classrooms without requiring an active internet connection."

Other benefits include the price and availability of Raspberry Pi 4 1GB, as well as its gigabit Ethernet port and USB 3.0 support – perfect for serving 40 devices.

Test and deploy

LEAP is still in early phases of deployment by AID India.

"We have completed initial testing and are actively working directly with them to test and deploy it in schools," the team explains. "While it is still early in the deployment phase, we are moving quickly and hope to share user metrics over the coming weeks."

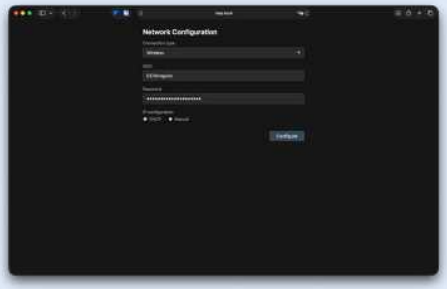
The code for LEAP is open source and available on GitHub at **rpimag.co/leapgit**. "We hope LEAP can help bring better educational access to learners beyond Tamil Nadu, India and we welcome inquiries from organisations interested in adopting the platform," the team says.

▲ The hardware is simple – it's the software that makes it unique

Quick FACTS

- T4EQ follows the UN's sustainable development goals
- The team have consulted on solar lighting in rural classrooms
- Team members have used Raspberry Pi in their own projects
- The codebase is built on Rust
- The content is hosted by AID India using Amazon S3

Using LEAP



1. From a browser, you simply need to choose your storage, enter the Wi-Fi details, and then have it sync with the educational server. Users can configure how access to the server works to better suit their bandwidth.

2. After a reboot, everything is downloaded, and then educational playlists are presented. Clicking on them brings up the materials needed by students to play on their devices.

3. As LEAP is browser based, it can also be accessed on mobile, with UI changes for smaller screens allowing an experience that's as good as using a full browser.

Text-based Cyberdeck

By Chris Hall

rpimag.co/tmuxCyberdeck

This cyberdeck, by repeat maker Chris Hall, features a NuPhy Air60 V2 wireless mechanical keyboard, a Waveshare 11.9-inch display, Raspberry Pi 4, and a UPS HAT powering the whole thing with two 18650 batteries. That's a pretty standard hardware stack; Bluetooth keyboards are often preferred to wired keyboards in cyberdecks because the lack of wires mean they take up less space, and don't require the maker to deal with messy cables. What caught our eye is the user interface: it uses the text-based toolkit tmux (pronounced 'tea-mucks').

Chris told us: "I'm very comfortable in the command line, and I wanted something to quickly access some of my homelab VMs and services if I wasn't at my desktop. That led me to discover tmux and sent me down a whole customising rabbit hole. In the end, tmux displays a bar at the top with the date and time, network up/down throughput, Wi-Fi status, VPN status, CPU and RAM usage. Plus being tmux, I can split the screen into separate terminals. So it's effectively a GUI without the GUI. "

Another thing you don't see very often is a second, separate screen dedicated to system data; that small screen on the right is a 128 × 32 black and white display that will cycle various stats, such as uptime, IP address, VPN status, etc. Another thing that we love is the auto-shutdown feature: "There are magnets on the corners that engage when you close the deck, and of particular note in the centre is a reed switch. When the deck is closed, a magnet triggers it. After a two-minute grace period, the deck will gracefully shut down."

"While I'm happy with it," says Chris, "I don't feel like it's what I would consider 'done'. I hope to iterate on it in the future, to refine some of what I see as design flaws. For example, it's currently difficult to attach the torque hinges due to screw clearance. The headphone jack would need to be relocated, as well as cable access for the keyboard (which can run via Bluetooth; I just haven't explored that yet). The 128 × 32 screen needs to be further recessed. I ended up having to cut it out and back it up a bit. Also, the top is rather top-heavy, so some form of kick-stand on the back-plate would help. But all in due time."

▲ The two 18650 batteries will power the deck for a respectable 4–5 hours

AI Rocky

By Leviathan Engineer

rpimag.co/RockyRobot

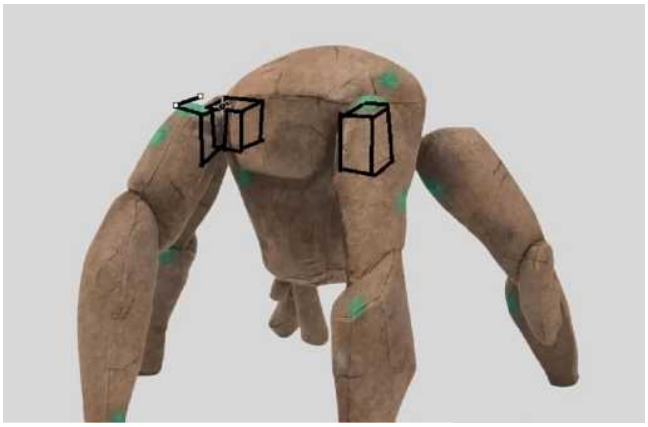
◀ Rocky is held together with hot glue for the servos that connect to the shoulders, with superglue for the joints that are under a lot of pressure

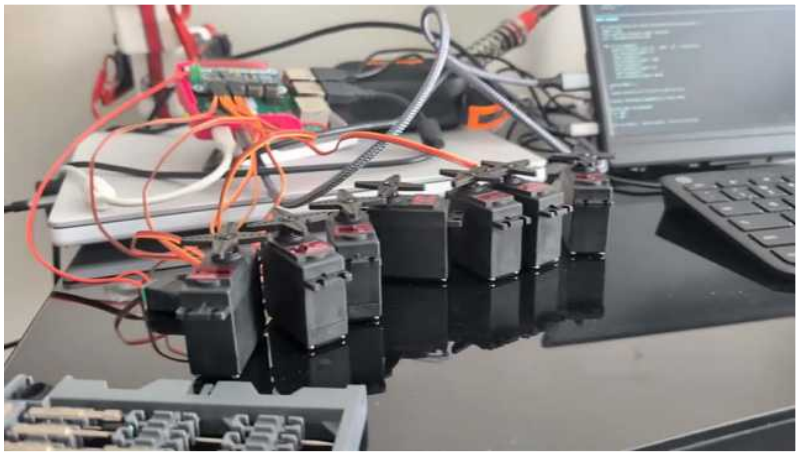
***Project Hail Mary* is a 2026 film based on the novel of the same name by Andy Weir, who wrote the book that *The Martian* is based on.** Both films feature an everyman forced to rely on deep science knowledge to survive – we'd recommend them highly. The star of *Project Hail Mary* is not, as the marketing materials would have you believe, Ryan Gosling – it's a robot named Rocky, and Leviathan Engineer has had a go at building his own Rocky, on Raspberry Pi and with a dose of AI.

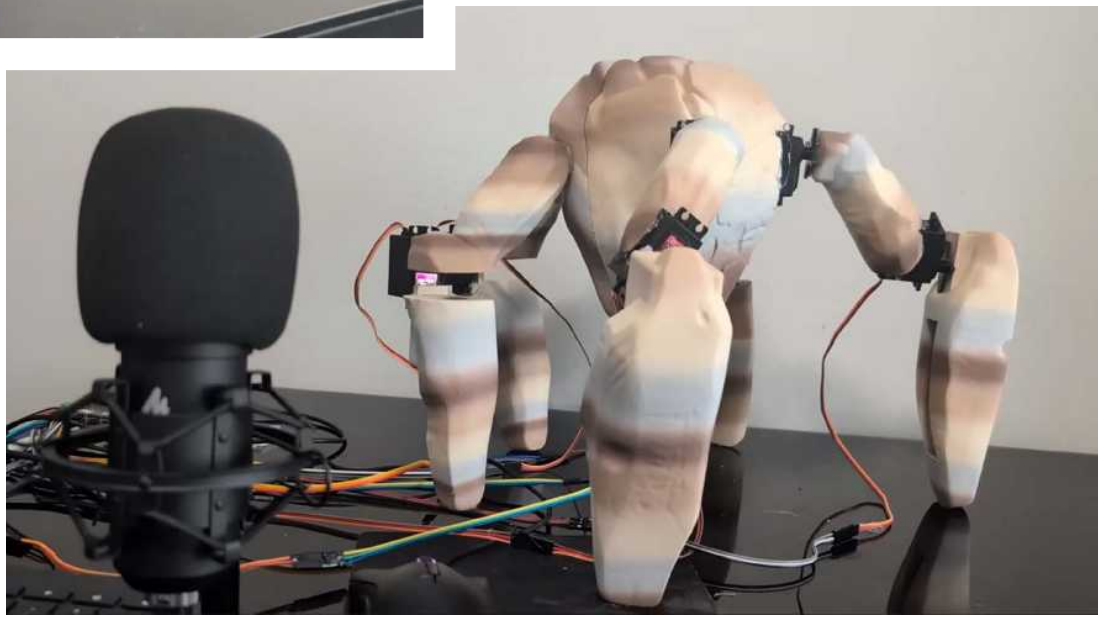
The build uses Raspberry Pi 5 4GB, a PCA9685 servo driver HAT, plus an internal microphone and speaker, and seven servos to make the legs move (and an additional power supply, because asking Raspberry Pi to power seven servos is a bit much).

The maker installed Claude Code on the Raspberry Pi to take care of all the programming. That entails speech recognition, generating a reply in the manner of Rocky from the film, then text-to-speech to make it talk back to the user. Thanks to the AI, the user can tell Rocky to do things that haven't been preprogrammed: Leviathan Engineer shows an example of the machine offering a fist bump in response to him asking for one.

The 3D printed body provided a challenge. As the maker used multicoloured filament, the print required some adjustment in the slicer software, such that the 3D body and legs were printed not in the most efficient way, but oriented so that the colour changes would occur as horizontal colour gradients.







Curiously Minty Cyberdeck

By Exercising Ingenuity

rpimag.co/AltoidsCyberdeck

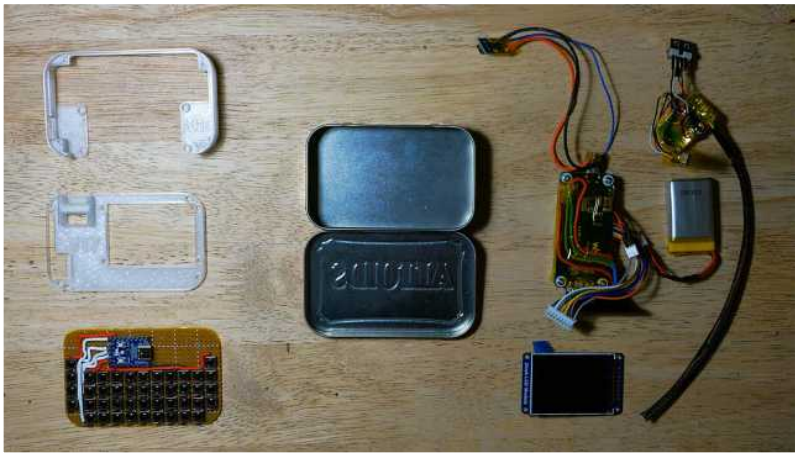
Altoids tins – they've been used for sewing kits, guitar effects pedals, a container for the papers for hand-rolled cigarettes. They're cheap, malleable, and thanks to the proliferation of US sweet shops adorning the UK's high streets, they're easily available even on Albion's blessed shores.

This deceptively complicated build by Exercising Ingenuity features a tiny custom keyboard, display, Raspberry Pi Zero, and a 2-inch LCD display left over as spare from another project. This screen didn't work with the latest version of Raspberry Pi OS, so Mason (he of Exercising Ingenuity fame) had to drop back a couple of software versions and make some changes to the settings in raspi-config.

Also crammed into the tin, there's a UPS HAT and battery for the Raspberry Pi Zero, plus a handmade protoboard dot-matrix keyboard, with a Waveshare RP2040 Zero to act as the keyboard's microcontroller. Mason saved space by removing some unused components from the RP2040 Zero and Raspberry Pi Zero, and by soldering components directly together rather than using wires. He even found room for a USB hub so he could add an external USB port.

To hold the electronics inside the tin required a spot of 3D printing – this took several design iterations, resulting in a mount printed out of flexible filament which would bend to fit inside the tin (the rolled metal edges are slightly thicker than the rest of the material in the tin, so if you want an insert to fit flush you can't just slide it in from the top down).

Finally, 15 GPIO pins are broken out, so you can use the device for a bit of hardware hacking – labels for the ports and the GPIO pins are engraved into the tin, and in a nod to the olden days, the 3D printed faceplate has been painted a lovely shade of beige.



▲ Making things small is hard: even minor details, like the hinges, needed modification to make the electronic components fit
Image: courtesy of Exercising Ingenuity

F1 Controller

By Danie Murray

rpimag.co/F1Controller

One of the joys of open-source hardware is that if you can't buy the thing you want, you can make it yourself. People love their games, their films, their sports, and will pour time and energy into anything that helps them to enjoy their passion. This F1 controller, built around Raspberry Pi 5, is one such example of creativity, passion, and tech.

Its maker, Danie Murray, built it during the gap left in the F1 season by the omission of the Bahrain and Saudi Arabia races: "instead of waiting in anticipation, I decided to build my own racing wheel controller to bring the F1 experience closer to home."

A 16GB Raspberry Pi 5 reads inputs from an MPU-6050 accelerometer and gyroscope sensor; with a little bit of programming, this translates into left and right for driving games. There's also a Keyestudio vibration motor for physical feedback, as well as six tactile switches, two toggle switches, and three potentiometers. With all those switches and buttons you could play anything you want, but we think this setup would be overkill for Tetris.

▼ Danie imported a photo of a F1 steering wheel into Autodesk Inventor, then scaled the design so that the 5-inch LCD screen would fit into the housing

3D print

Precision, patience, and plastic make for a practically perfect timepiece

rpimag.co/Tourbillon

Clockwork is hard. The escapement and balance wheel (both essential parts of any mechanical clock) are subject to gravity, which pulls slightly, affecting the watch's reliability. The mechanical answer to this is to place both those mechanisms in a sub-assembly that rotates in three dimensions, so that the gravitationally induced inaccuracies are averaged out over time. That sub-assembly is known as a tourbillon.

Watches featuring tourbillon mechanisms are incredibly expensive, so it's a credit to 3D printing that anyone can print one for themselves. We say anyone, but not many have the patience to print and assemble a build composed of 395 parts, 38 bearings, 66 rubber bands, 413 screws, and more.

That's how 3D printing wins; where it fails is that, because every 3D printer is different, and every brand of filament has different properties, each clock made to this design needs to be individually calibrated to ensure accurate timekeeping. And it *is* accurate: Tomasz, the maker of this incredible piece, says that it's accurate to +/- 5 minutes over two months. If you're in any way interested in the process of how an idea becomes a physical thing, we strongly recommend you check out Tomasz's write-up of this build, have a good ogle at the photographs, and ask yourself if you'd have the patience to develop the same project for the two years that this build took him.

MAKE A SMART HOME

Control your electronic devices from one place

Maker
Ben Everard
Ben is a light artist who has shown NeoPixel-based art at light festivals around the UK.

glowingart.co.uk

Warning!
Voltage warning
Make sure you connect VIN on the BME280 sensor to 3V3 on Pico, otherwise the module will output 5V and may damage your Pico.

As William Gibson told us, the future is here – it's just not very evenly distributed. William meant that about people having different levels of access to technology, but it's also true on a device level. You probably carry around a ridiculously powerful computer in your pocket (in the form of a smartphone) and have a lightning-fast connection to the internet. However, there's a pretty good chance that you control a lot of the electronics around your house by flipping bits of bent copper covered in plastic.

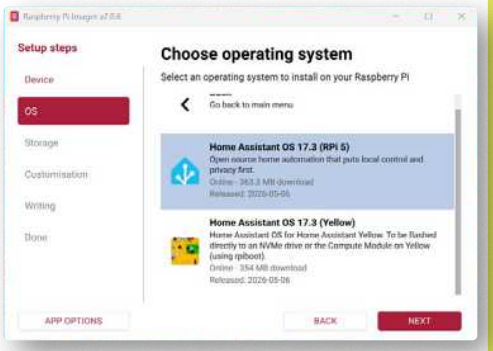
Those devices around our house that are blessed with some silicon brains seem to come from different sci-fi tropes. There are those that use standards and interoperate with each other in a vaguely sensible way. There are also those that are locked down by some megacorp that won't work unless you hand over your details and data, and even then, they only work in the way megacorp decides they should work.

We're here to even out the future. At least even it out within the domain of your home. We can add control to the uncontrolled and bring other bits together into one unified platform.

In this article, we're going to automate our own smart home. We'll use a Raspberry Pi as a controller, we'll add some off-the-shelf hardware and then we'll create our own custom devices using a Pico W. From here, it's up to you where to go. There are literally hundreds of compatible off-the-shelf devices available, but for ultimate flexibility, you can easily build your own. Let's get started.

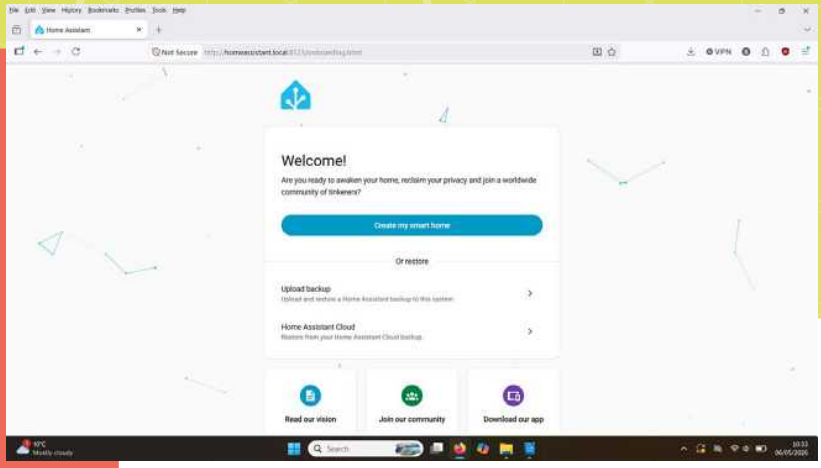
DOWNLOAD THE FULL CODE:

rpimag.co/hacodepy



▼ You can install Home Assistant directly onto an SD card using Raspberry Pi Imager

> We're here to even out the future. At least even it out within the domain of your home



When you first boot your Raspberry Pi, you can log into the website from another computer and set everything up

Installing the software

Installing Home Assistant is straightforward as its makers maintain their own distribution that you can install through Raspberry Pi Imager. If you don't already have this application, you can download it from **rpimag.co/software**. It runs on Windows, macOS, or Linux.

You'll need an SD card that you're happy to overwrite the data on. Pop that into your computer and fire up Raspberry Pi Imager. Select the version of Raspberry Pi that you'll be using, then in the OS tab, scroll down to Other Specific-Purpose OS. In the new list, select Home Automation, then Home Assistant, and finally the option for Raspberry Pi 5 (currently Home Assistant OS 17.3, but this could change).

Click through to select your SD card, and then write the data to the drive.

Once the data is written, you can pop the SD card in your Raspberry Pi, connect it to Ethernet (see box if you're using Wi-Fi), and power it up. If things go smoothly, there's no need to connect a keyboard, mouse or screen.

It should power up, connect to your network and start serving a web page at the address **homeassistant.local:8123**. Point a web browser to this and you can enter your setup details.

We tested this tutorial in a house that – naively – we didn't think of as a smart home. However, the initial scan brought up a heating system, the TV, a speaker, and the router. All of which could be added to Home Assistant with a couple of button presses (and a few usernames and passwords being entered). It turns out that we can control the TV, and play music through the speaker via the Home Assistant web page. We didn't even know that was possible.

However, the chances are that you have at least some non-smart devices that you want to control, so let's now take a look at how to do that.

By far the simplest way of controlling an electronic device is by turning the power off and on. For some things – such as lights, heaters, perhaps even slow cookers and other kitchen devices, this is all you need to do. And fortunately, we can do this without having to go anywhere near mains voltage as you can get smart plugs that have all the dangerous parts safely tucked inside and we can just control them with software.

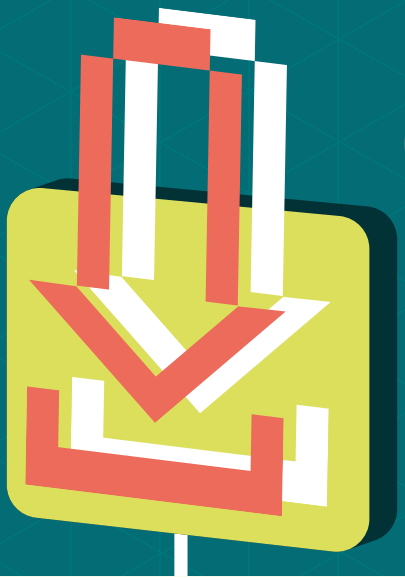
> Off-the-shelf smart devices can be great, but they can be expensive

There are a few different brands available. We went with a Tapo P100. It's cheap and has support in Home Assistant (some people have reported problems with it, but it worked fine for us).

Before we can use it in Home Assistant, we have to configure it using the Tapo app. Follow the instructions that come with it, and make a note of the password you use as we'll need that when we add it to Home Assistant.

Once it's up and running (you can test it out using the app), it's time to configure Home Assistant.

In the web app, go to Overview > Devices and click on '+' to add a new device. Enter 'Tapo' and click on TP-Link then TP-Link Smart Home.



You can leave the first box empty and hopefully it will discover your device. You'll then need to enter your email address and password to take control of it. You should then have a device that you can control. It will appear in Home Assistant as a sliding switch. Toggling this switch should cause the socket it controls to turn on or off.

In your Devices tab, you can add different bits of hardware to rooms. This isn't essential, but helps keep things organised. As your smart home grows, it's likely that you'll end up with a lot of devices and by sorting them into rooms, it matches the way we think about our homes.

Building our own device

Now we've got our basic Home Assistant setup running, let's take a look at creating our own hardware that communicates with our smart home.

Off-the-shelf smart devices can be great, but they can be expensive, don't cover everything, and often come with very onerous terms of use (including reading your data, and the ability for the manufacturer to stop the device working when it gets old). Sometimes it's just better to do things yourself. Let's take a look at how to do this with a Pico W.

Home Assistant can work with the MQTT message broker. This lets us send information back and forth between our hub and whatever devices we want to use. MQTT works with topics that devices can publish or subscribe to. Publishing to a topic sends a bit of data that the broker then stores. Any devices that subscribe to the topic can then read this bit of data. In principle, the data can be anything, but in order for everything to make sense, you have to send it in a format that the other devices are going to understand.

Mosquitto is the broker, and Home Assistant itself will subscribe to the channels and read the data that we publish on the broker.

MQTT is well supported by most languages you might want to use with Pico, but we're going to use CircuitPython because this supports the hardware we're going to use.

CONNECTING VIA WI-FI

In general, the best option is to connect Home Assistant to your local network with an Ethernet cable. You'll get better reliability and things are likely to all run a bit smoother. However, we live in an imperfect world and this isn't always possible. If you need to connect to Wi-Fi without first connecting via Ethernet, you can attach a keyboard and monitor and do the following:

Enter `login` at the `ha>` prompt. The prompt will now change to a `#`. You can now run the following command:

```
nmcli device wifi connect "YOUR_SSID" password "YOUR_WIFI_PASSWORD"
```

After a few seconds, you should get a message saying that it has successfully connected. You can now connect from another computer using the URL **homeassistant.local:8123**.

STANDARD FIRMWARE

There are some standard firmwares that you can use that should make it easy to build custom devices with Pico W. ESPHome is the most famous of these (despite its name, it should work with Pico W). In principle, it should let you define your hardware in a YAML file, and then compile this to an uploadable binary for Pico. There's an integration that lets you do all this in your Home Assistant web page.

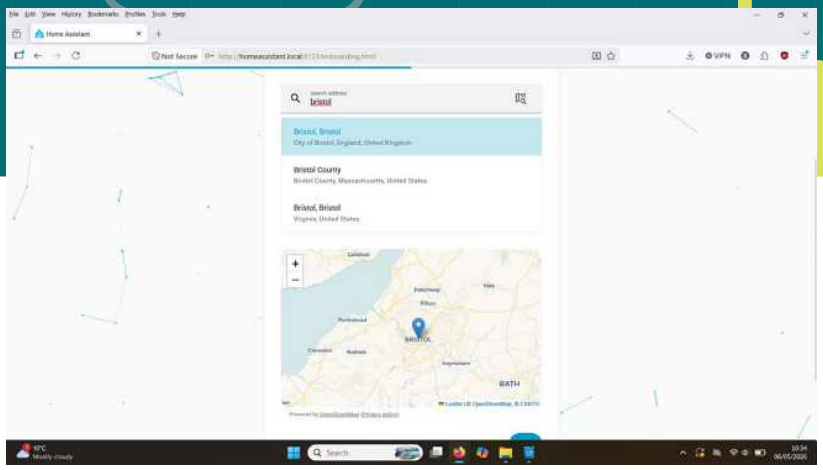
However, we couldn't get it to work with the hardware we wanted to use. Neither I2C nor SPI buses would detect devices. It's possible that we had the configuration wrong in some way, but it was hard to debug, so we decided that it was easier to start from scratch.

AUTOMATIONS, SCENES, SCRIPTS AND BLUEPRINTS

So far, we've shown how to bring things into Home Assistant, but this is just scratching the surface. There are four features that really let you control everything:

- **Automations** might be what you think of as home automation. This lets you connect triggers with actions. For example, 'when my plants get dry, send me a notification'.
- **Scenes** are predefined settings. For example, you might have a movie-watching scene which dims the lights, sets the volume, and makes the popcorn. OK, only the truly obsessive will go far enough to build a Home Assistant device capable of making popcorn, but getting the settings in the room how you like them is entirely possible.
- **Scripts** are sequences of actions that take place one after another – basically, they're simple programs that you can trigger from the Home Assistant UI.
- **Blueprints** let you create reusable templates that can be shared with other users. You might, for example, have a script that turns off all your lights. If you shared this directly with another Home Assistant user, it would not work if they had different lights. However, with a Blueprint, they could easily connect it to their lighting system.

Since all of these are highly dependent on the specific hardware you have, and the details of your house, we're not going to create any here, but there's plenty of help online, so if you've got some experience at programming, you shouldn't find it too difficult to get started.



▲ When setting up Home Assistant, you can locate your house. Later on, this can be used to find the weather, and see which people are home

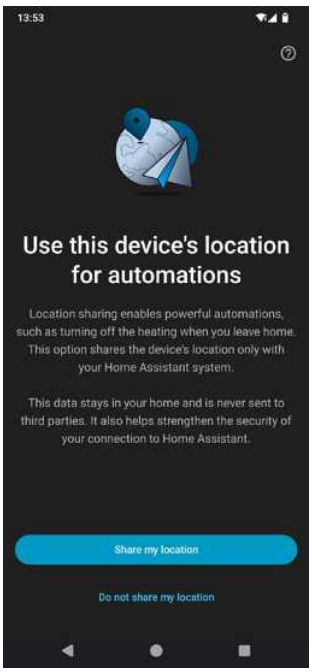
▼ Using the Home Assistant App, you can track your phone's location and run automations when you get or leave home

> We tested this tutorial in a house that – naively – we didn't think of as a smart home

First, let's get Home Assistant running with the Mosquitto broker. Go to Settings > Apps and click on Install App. Search for Mosquitto and follow the instructions to install it. You can accept the defaults.

Our devices will need to log in to post data, and they do this as a regular Home Assistant user, so we'll need to create one of these. Go to Settings > People and click on Add Person. You'll need to give it a password that your script will use, so make sure you make a note of it for later.

That's Home Assistant setup. We can now move our attention to Pico W. You should be able to use either a Raspberry Pi Pico W or Pico 2 W for this. If you're particularly determined, it's probably possible to get this to work with one of the Ethernet-enabled RP2040 boards as well, but we haven't tried this.



For our first device, we're going to hook up a BME280 temperature and humidity sensor. We've hooked this up via SPI, but the device also works using I2C, so it's up to you how to connect it.

You'll need to download and flash the latest version of CircuitPython for your device from **circuitpython.org**.

Once this is done, your Pico should automatically connect to your computer and show up as a CircuitPython drive.

If you've not used CircuitPython before, we'd recommend that you use the Mu editor (downloadable from **codewith.mu**), and take a look at **rpimag.co/circuitpyguide** which will give you the basics.

You'll need some files from the library bundle. Download the latest zip file from **circuitpython.org/libraries** and copy the following over to your Pico's **/lib** folder:

```
adafruit_bme280/
adafruit_bus_device/
adafruit_minimqtt/
```

Now you need the code. The full version is online at **rpimag.co/hacodepy**, but we'll take a look at a few of the key bits here.

First, we need to set up the BME280 sensor:

```
spi = busio.SPI(
    clock=board.GP2,
    MOSI=board.GP3,
    MISO=board.GP4,
)

cs = digitalio.DigitalInOut(board.GP5)

bme280 = adafruit_bme280.Adafruit_BME280_
SPI(spi, cs)
```

Later in the script, we'll read the values from the sensor using:

```
bme280.temperature
bme280.humidity
bme280.pressure
```

We can connect to MQTT using the following:

```
mqtt = MQTT.MQTT(
    broker=MQTT_HOST,
    port=MQTT_PORT,
    username=MQTT_USERNAME,
    password=MQTT_PASSWORD,
    socket_pool=pool,
    ssl_context=ssl.create_default_context(),
)
mqtt.connect()
```

Then we can publish data to a topic using the format `mqtt.publish(TOPIC, DATA)`.

A smart plug has a plug on one side and a socket on the other and it can turn the socket on or off remotely

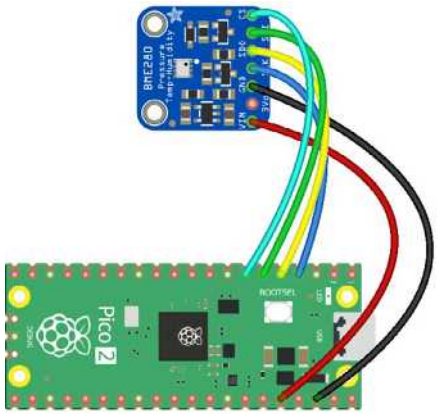
You can plug in any electronic device provided it doesn't exceed the current rating of the plug (in this case 13A)

POWER CONSUMPTION

For many of us, power usage is an important thing to track. This is partly because we care about climate change, and partly because energy is increasingly expensive. Tracking things helps us control them, and Home Assistant's Energy tab gives you an overview of energy usage.

Unfortunately for us, we weren't able to test this out. The electricity company have declined to fit a smart meter to our test house due to some, er, interesting wiring decisions taken 30 years ago. And despite having roofs facing in three different directions, not one of them is suitable for solar panels.

We're awaiting the fine details of the UK's 'balcony' solar regulations which may be suitable for this house. However, for now we can only track power to manually read numbers off a dial. Hopefully you're more fortunate than us and have access to at least some methods for tracking your household power.



◀ The wiring diagram. Make sure you connect VIN on the BME280 to 3V3 on Pico, otherwise the module will output 5V and may damage your Pico

Go forth and ensmarten your environment

When using MQTT with Home Assistant, two of the most important concepts are the config topic and the availability topic. Together, they allow devices to integrate automatically and report whether they are online.

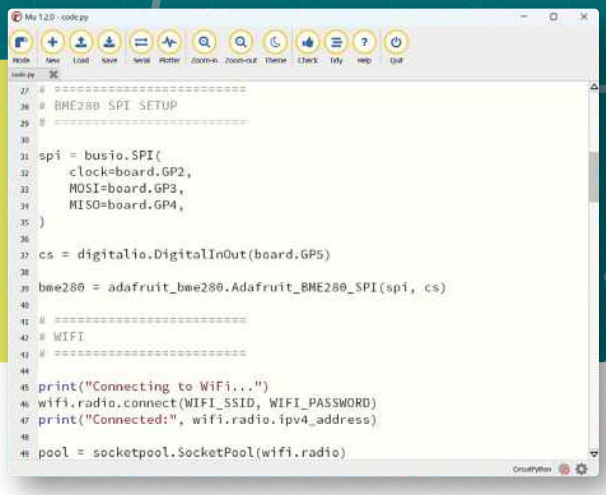
The config topic is part of Home Assistant's MQTT Discovery system. Instead of manually creating sensors, a device can publish a small JSON configuration message to a special topic such as:

```
homeassistant/sensor/pico_w_bme280_temperature/config
```

This message describes the sensor – its name, units, device class, and which MQTT topic contains the actual readings. Home Assistant listens for these discovery messages and automatically creates the entities and device entries. This is why MQTT devices appear in Home Assistant without any manual setup. The availability topic is used to indicate whether the device itself is online. For example, a Pico W sensor might publish:

```
pico_w_bme280/status
```

...with a payload of `online` when connected, and `offline` when shutting down or disconnected. Home Assistant uses this to mark entities as unavailable if the device drops off the network. This is particularly useful for battery-powered or Wi-Fi devices where connectivity may occasionally be interrupted.



▶ CircuitPython has great hardware support, which is why we chose it for this project

Once the device has announced itself to Home Assistant using MQTT Discovery, it can begin publishing live sensor readings. In our project, the Pico W reads temperature, humidity, and pressure data from the BME280 sensor and sends the values to a dedicated MQTT state topic. A state topic is simply the MQTT address where live readings are published. In this example, we use:

```
pico_w_bme280/state
```

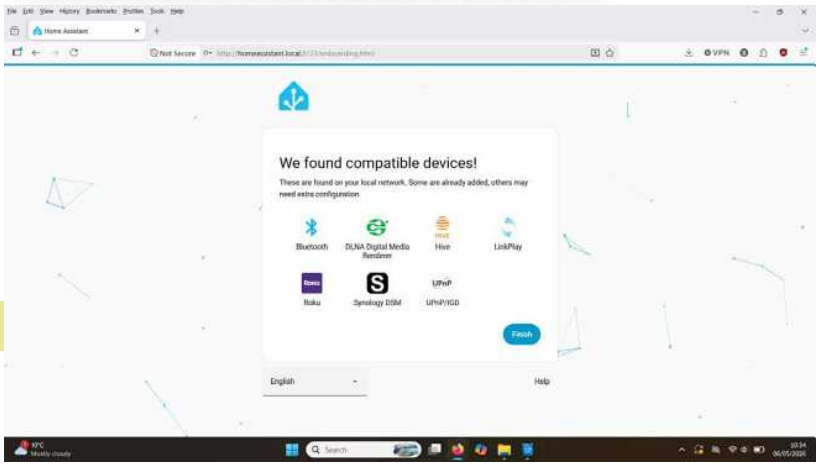
Every minute, the Pico W reads the sensor and packages the measurements into a JSON payload such as:

```
{
  "temperature": 22.4,
  "humidity": 45.1,
  "pressure": 1013.2
}
```

This payload is then sent to the MQTT broker using the `publish()` function:

```
mqtt.publish(
    "pico_w_bme280/state",
    json.dumps(payload),
    retain=True,
)
```

The `json.dumps()` call converts the Python dictionary into a JSON string that can be transmitted over MQTT. Home Assistant subscribes to this topic and extracts the individual values using the `value_template` entries defined earlier in the MQTT Discovery configuration.



▲ As part of the install, Home Assistant scanned our local network and found a series of smart devices that could be easily added

Using a single JSON payload for multiple readings keeps the MQTT traffic efficient and makes it easy to add more sensors later. For example, battery voltage or air quality readings could simply be added as extra fields in the same message.

The `retain=True` option is also important. It tells the MQTT broker to store the most recent reading so that when Home Assistant restarts, it immediately receives the latest sensor values instead of waiting for the next update cycle.

Once this is on Pico W (with the appropriate settings for your Wi-Fi and Home Assistant login), you should be able to go to Home Assistant, select Overview > Devices, and see the values for Temperature, Humidity, and Pressure.

That's all there is to it! You can use this same basic technique to connect all manner of sensors and actuators to Home Assistant. Go forth and ensmarten your environment!

SUBSCRIBE TODAY FOR JUST £10

Get **3 issues** + **FREE** Pico 2 W

SUBSCRIBER BENEFITS

Free delivery
Get it fast and for free

Exclusive offers
Great gifts, offers, and discounts

Great savings
Save up to 37% compared to stores

SUBSCRIBE FOR £10

Free Pico 2 / Pico 2 W

3 issues of Raspberry Pi Official Magazine

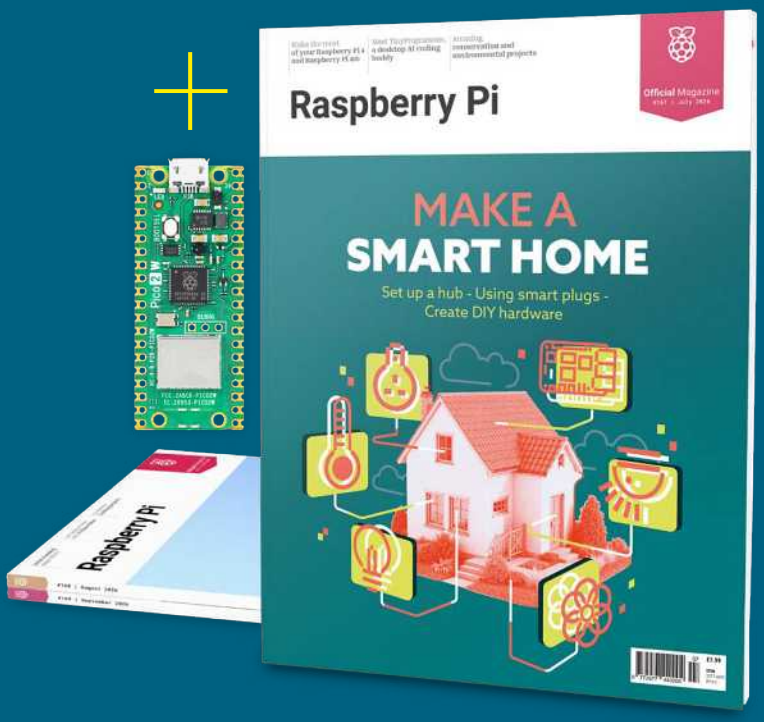
£10 (UK only)

SUBSCRIBE FOR 6 MONTHS

Free Pico 2 / Pico 2 W

6 issues of Raspberry Pi Official Magazine

£30 (UK)	**$43** (USA)
€43 (EU)	**£45** (Rest of World)



Subscribe by phone: 01293 312193

Subscribe online: rpimag.co/subscribe

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

Subscribe for £10 is a UK-only offer. The subscription will renew at £15 every three months unless cancelled. A choice of free Raspberry Pi Pico 2 W or Pico 2 is included with all subscriptions.

Convert a CNC router to run on Raspberry Pi 5 with LinuxCNC

Let's explore converting a three-axis CNC router to run on LinuxCNC on a Raspberry Pi 5

Maker

Jo Hinchliffe (AKA Concretedog)

With a house and shed full of lathes, milling machines, 3D printers, and more, Jo is a constant tinkerer and is passionate about making. Obsessed with rockets and robots and much more besides, he often releases designs and projects as open source.

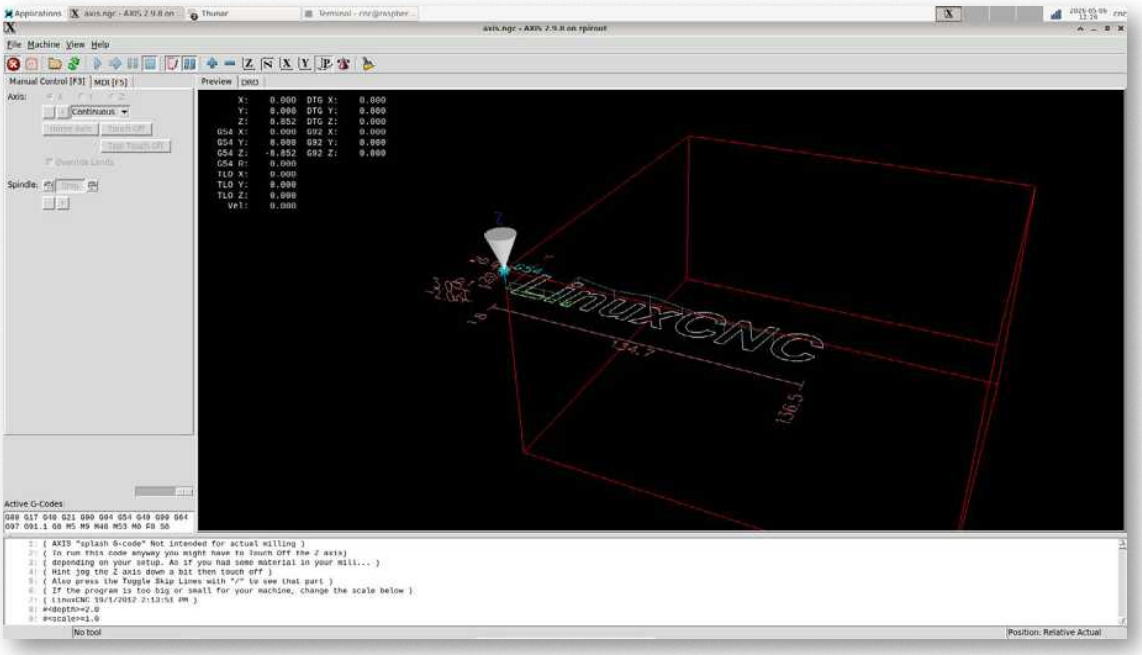
concretedog.blogspot.com

Computer numerical control, or CNC, machines are the secret sauce of many a maker's lair! The most commonly found of these machines are 3D printers, closely followed by laser cutter/engravers, CNC routers, and milling machines. Beyond that, we get into the more unusual realms – things like pen plotting machines, CNC lathes, and custom CNC devices.

Most CNC machines tend to have some form of stepper motor that can be turned by accurate and repeatable amounts. Many machines will have one or two motors 'per axis', which often means 'per each direction of travel'. So, for example, a filament 3D printer will usually have three axes of motion: X, Y, and Z. Looking at a variety of 3D printer designs, though, you may see ones where the up and down axis moves the extruder, or moves the bed, may have one or two stepper motors, and other variations, but largely it's all X, Y, and Z.

Next in the CNC chain of command are the stepper motor drivers. These can be a tiny IC or a larger module with heatsinks and fans, but they all basically have the same role: they receive information and they drive the motors in specific directions by specific amounts. While not always the case on some smaller more affordable builds (like we just spotted on a diode laser CNC in our current workspace), most of the time each stepper motor will have its own motor driver.

Most CNC machines tend to have some form of stepper motor that can be turned by accurate and repeatable amounts



◀ The main interface screen of LinuxCNC

Warning!

CNC Cutter

Be careful when using CNC cutters in your projects. Wear ear protection and safety glasses and stand clear of the machinery as it works. Understand the basic safety functions of your machine.

rpimag.co/cncsafety

Next back in the chain is some form of controller board. The role of this is to receive G-codes from a device, often an attached computer, or streamed from some storage such as an SD card, and then send the correct signals to the correct motor drivers to move the correct amount in the correct direction. These controller boards often come in 'families' which require particular flavours of G-code.

Raspberry Pi Pico control

In the previous CNC projects, we've often used the excellent open-source GRBL project as our controller. GRBL is excellent; it runs on numerous small microcontrollers, ESP32, Arduino, and lately there are versions which will run on Raspberry Pi Pico. This means that it's also a very affordable approach and has a large user community. However, there are some limitations with GRBL. One limitation is that it has some G-codes which it doesn't use; these include some 'nested' commands where a simple G-code can call on and perform an external subroutine. The other limitation (although again GRBL is excellent and none of these are meant to detract from that) is that the original version of GRBL doesn't support more than three axes (though there are other forks ported to 32-bit systems which can support more axes).

We've previously used Raspberry Pi with the standard operating system to send the G-codes to the GRBL controller. Again, there are excellent open-source tools for this such as Universal G-code Sender (UGS). However, it's totally possible to use a Raspberry Pi as both the CNC controller and the G-code sender with a project called LinuxCNC. This also adds the benefits that LinuxCNC offers more G-code options like nested G-codes

▲ **Figure 1:** Our old but still great CNC router

for more complex operations, and also has support for machines with more than the standard three axes. Converting to a new control system and learning a new environment like LinuxCNC can be a long road, so to get started we wanted to convert a GRBL CNC three-axis router (**Figure 1**) to LinuxCNC control.

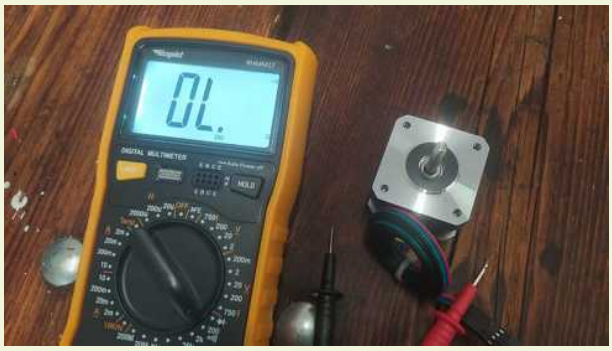
The nice thing about LinuxCNC is that it can run on a Raspberry Pi, and for our conversion we used a Raspberry Pi 5 with 8GB of RAM. Lots of CNC machines still favour parallel port as an interface and so we began to research options. This research led us to Byte2Bot, a small company in the USA that sells lots of different CNC solutions. Included in the range is a Raspberry Pi Parallel HAT. This HAT, in turn, can be connected to some quite common five-axis-capable CNC control boards which are sold widely online and incorporated into many more CNC machines at the modest end of the market. When we realised that Byte2Bot sold the Parallel HAT, the five-axis CNC board, and a 3D printed frame into which the Raspberry Pi, HAT, and CNC control board could be mounted (**Figure 2**), this seemed like a simple and smart one-stop solution.

The target CNC router used NEMA 23 stepper motors which like to receive around 2 amps of current. As such, the smaller motor driver boards you might find in pen plotters, 3D printers, and indeed other smaller CNC routers, won't quite cut it. Trying to keep the budget low, we opted for some cheap but functional generic ST-4045 stepper drivers and bought a pack of four drivers (enough for the project and one spare) pretty reasonably priced online.

The machine we are converting is a small desktop three-axis router which is reasonably well built and has precision ball-screws as lead-screws which move the X, Y and Z axes. There are some chunky NEMA 23 stepper motors which,

▲ **Figure 2:** A collection of parts for the Raspberry Pi 5 from Byte2Bot

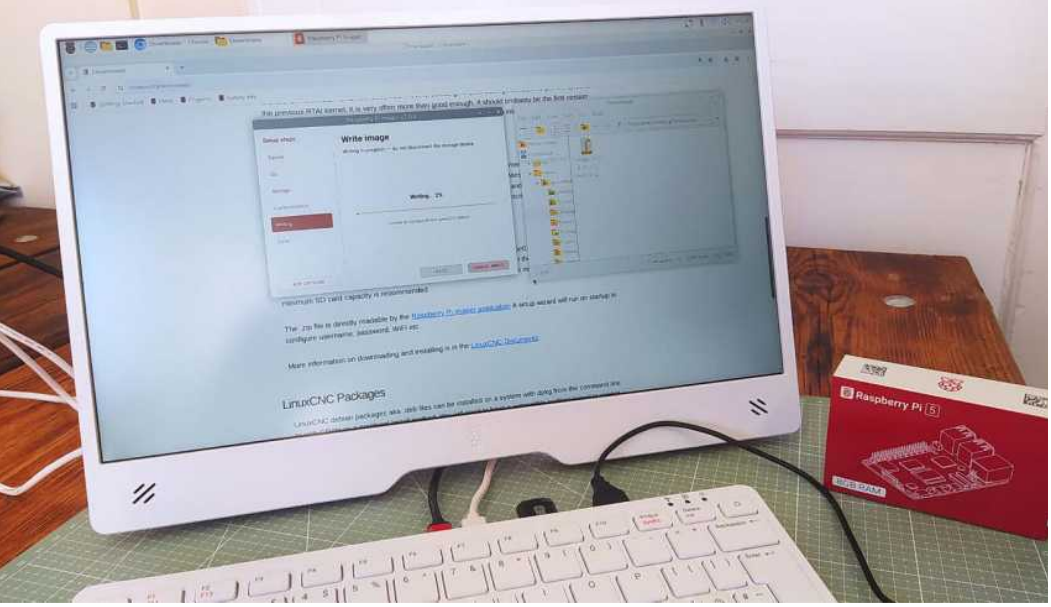
Checking the wiring



If you have some undocumented stepper motors and don't know which wires are the coil pairs, there are a couple of simple techniques to identify them.

The simplest method is to hand-turn your motor with no wires connected to anything and get a feel for how easy it is to turn. Next, grab any two wires and connect the two wires together. With the wire ends held together, try to rotate the motor again. If the motor is still as easy to rotate, then the two wires in your hand are not a pair. Keep hold of one of the wires you checked and swap the other wire for a different one; connect these two and check again. If needed, repeat this one more time and you should find that one pair of wires, when connected, will make the motor much harder to turn; this is one of the motor pairs.

Alternatively, you can use a multimeter set in continuity mode to check which pairs of wires are connected. When you find a pair of wires that are connected, this is a single coil pair. Neither of these two techniques tells you definitely which coil is coil A and which is coil B, but this doesn't actually matter. If you connect either pair to A1/A2 and the other to B1/B2, your stepper will work. If you feel that the stepper is working but going in the opposite direction than expected, in this project you can change this in the interface by toggling the Invert Axis Direction buttons on the Settings tab. Or, in any project, you can achieve this by swapping a single pair of wires in the same wire coil. So switch, for example, A1 for A2 and A2 for A1.

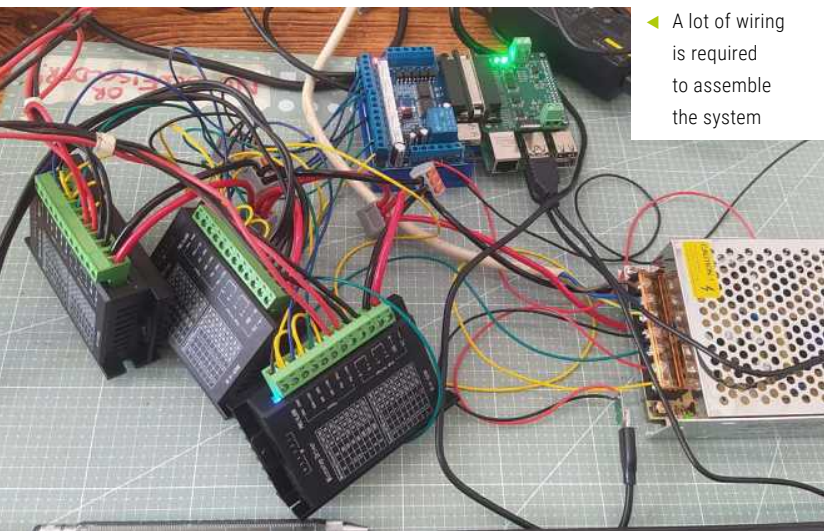


▲ **Figure 3:** Flashing the LinuxCNC image to an SD card using a Raspberry Pi 500

while old and second-hand some years ago, are still performing well with plenty of torque. These are currently connected to a generic controller board (a YOOCNC NT65 3X) which in turn has its parallel port connected to an Arduino running GRBL. The YOOCNC controller also has a separate spindle control which drives the spindle motor and turns the tool directly rather than via G-codes.

To begin, we simply disconnected the stepper motor wiring from the main board and put the entire older controller out of the way. The old controller system didn't mark the pins where the stepper motor wires connected, so we didn't know which of the stepper motor wires were pins. We used the multimeter method explained elsewhere in this article to identify these.

To begin, we first went and downloaded the current LinuxCNC image for Raspberry Pi from the official project website (**linuxcnc.org**). Rather nicely, the supplied image is suitable for either a Raspberry Pi 5 or 4, which increases your options. The download is a zip file and this is directly readable if you are using the official Raspberry Pi Imager application. As we spotted that this was the case, it made sense for us to download this image and flash an SD card using our Raspberry Pi 500 which was already set up and has the Imager application installed (**Figure 3**). With the image installed on the SD card, we booted the Raspberry Pi 5 with a keyboard, mouse, and monitor attached, and there was a small startup wizard that allowed us to configure the username and password and Wi-Fi etc. You don't have to do this at this stage, but we often like to just confirm that the device and image are working before ploughing deeper into a project.



◀ A lot of wiring is required to assemble the system

We then started to assemble the three boards: Raspberry Pi 5, Byte2Bot Parallel HAT, and five-axis CNC interface board (**Figure 4**). The Raspberry Pi 5 is first to mount to the Byte2Bot 3D printed chassis, achieved with some supplied stand-offs. On top of the Raspberry Pi 5 we can then push on the Byte2Bot Parallel HAT which slides snugly down into place on the GPIO header pins. We can then secure the whole device into the frame by tightening the bolts down into the stand-offs. Before we can slide in the five-axis CNC parallel

▶ **Figure 4:** The Raspberry Pi 5, Parallel Port HAT, and the five-axis board together with three ST-4045 stepper motor drivers

board, we need to use a pair of pliers or similar to remove the nuts at the side of the parallel port on the Parallel HAT. We can then simply slide in the five-axis CNC interface board on the opposite side of the frame and bolt it into place. At this point you'll appreciate this is a nice compact and neat build so far. When we add the wiring loom, however, it will get more messy!

Get wiring

The next and arguably biggest job is to add the wiring loom to the system. This includes wiring two power supplies, one rated at 5V DC and another one to power the stepper motors which may be rated 24–48V. We opted for a 20A capable 24V supply and a 10A capable 5V supply.

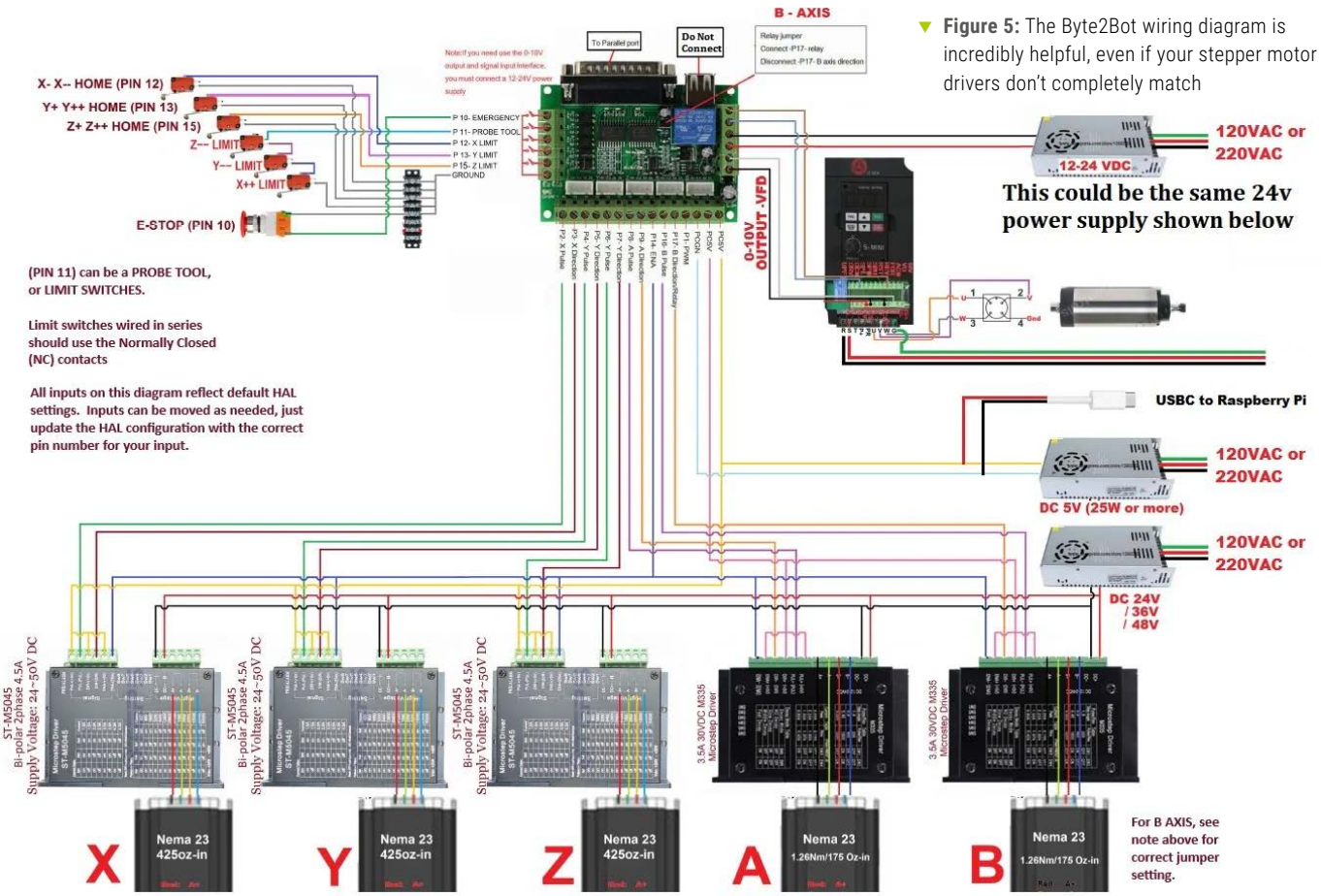
The Byte2Bot website (**byte2bot.com**) has a wiring diagram which mostly matched our requirements. We say 'mostly' as the stepper motor drivers in the image don't quite match the ST-4045 stepper motor drivers we used. However, it was possible to carefully check the names of the connected outputs and inputs on the image and match them to the pin labels on our ST-4045. To make life somewhat simpler, we decided to use some lever lock connectors to make quick connections between common points. We began using some 22 gauge wire to wire all the pins on the ST-4045, which required tying high to +5V. We then wired these back to the five-axis CNC board and also the 5V power supply.

Warning!
High voltage
This project uses 24V DC at 20A. This presents a fire risk. Be sure to use 14 gauge cable.
rpimag.co/electricalsafety

The machine we are converting is a small desktop three-axis

The 5V power supply also needed to supply the Raspberry Pi and connected HAT and board power. To do this, we opted to buy some small USB-C socket breakout boards. These are available very affordably online and, after soldering two wires for power, we can connect a USB-C cable to the 5V power supply to plug into the Raspberry Pi 5. Next, we wired back the individual X, Y, and Z pulse pins to the five-axis controller board. It can be a good idea to grab a small scrap of masking tape and label your stepper motor drivers X, Y, and Z at this point to save a little confusion.

Having identified our stepper motor coil pairs earlier, we wired these into the A+, A-, B+, B- connectors. Note that later in testing, if we get a stepper motor driving an axis in the opposite direction than intended, one solution is to simply swap one pair of wires over; for example, A+ would be swapped to A- and the original A- wired to A+.



▼ **Figure 5:** The Byte2Bot wiring diagram is incredibly helpful, even if your stepper motor drivers don't completely match

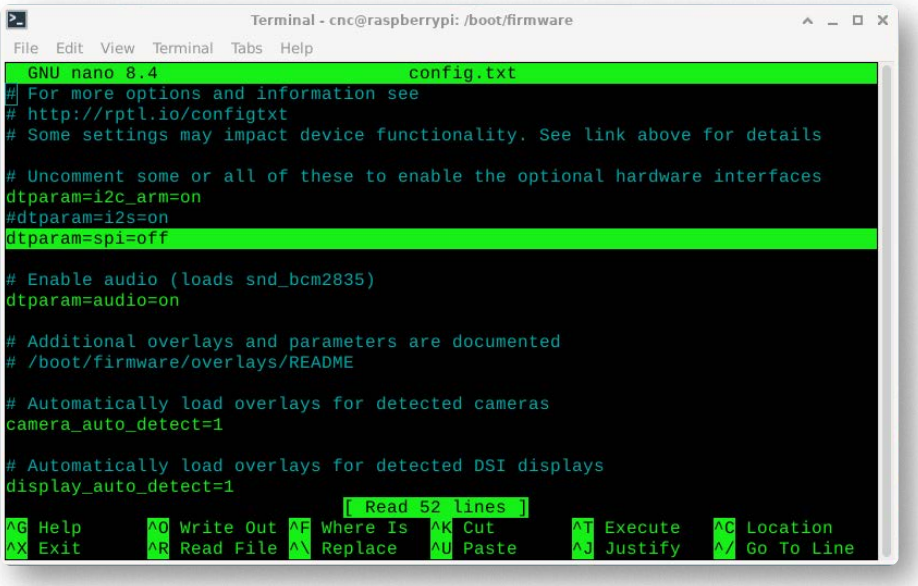


Figure 6: Editing a configuration file with nano; this releases the SPI pins to LinuxCNC

Wiring the power for each stepper motor driver GND and positive connection, we went a little over spec and used some much beefier 14 gauge cable. With everything on the X, Y, and Z axis stepper drivers now wired, we can look at approaches for the spindle and the end-stops/limit switches.

You can see on the diagram (**Figure 5**) that a system with limit switches will typically use normally closed (NC) switches, with each switch being tied to ground when closed. Our machine doesn't have limit switches fitted and we are used to operating the machine without them. So while we may add these functions in the future, we don't have them at this point. However, the system still needs these pins to be tied to ground. Again, we used some thin gauge wire to connect all these pins to ground. It's a similar situation with the emergency stop: we need a normally closed emergency stop button which is wired between its pin and ground. These are commonly available for purchase online and it's highly recommended to buy one. Note though that the **F1** key in LinuxCNC is also considered a toggle switch for the emergency stop, so in testing you can also use this functionality.

Adding automation

Our previous controller had a separate spindle speed controller and a simple potentiometer to set the speed of the spindle. This means that the operator would manually have to start and stop the spindle when starting and finishing a job. LinuxCNC, and many other CNC controllers, will allow you to control the spindle speed via G-codes and we plan to do so down the line. LinuxCNC achieves spindle control by using a 0–10V output where the voltage level controls the speed of a spindle driver. Our five-axis CNC board has this output. We opted to purchase a small affordable spindle controller. In the first instance we can directly wire this to the spindle motor and, as with the previous machine controller, just turn on the spindle and set the speed with a potentiometer. However, this spindle controller also has the 0–10V input connector, so should work in the future if we explore using LinuxCNC to control the spindle.

With all the wiring in place and triple-checked, we are now at the point where we can begin to set up LinuxCNC and get our machine moving. This section takes some time and has some challenges, but we are going to use some short cuts by using known working configurations as supplied by the Byte2Bot website.

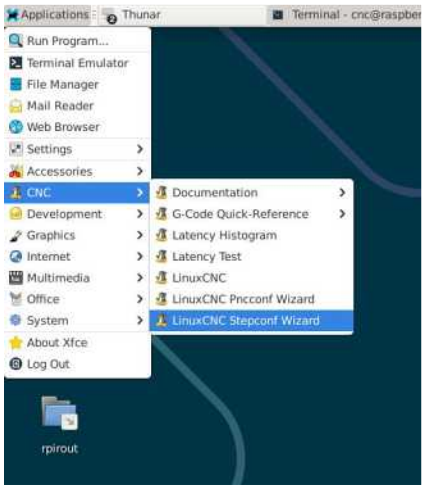
Before we actually start working through a LinuxCNC wizard to define a custom set of config files for our machine, we need to make a change to a config file in the Raspberry Pi itself. Boot LinuxCNC and open a terminal. We then need to move to the **/boot/firmware** directory using the following command:

```
$ cd /boot/firmware
```

In this directory you should find a file, **config.txt**. We need to edit this file, so let's get the nano text editor to open it with the command:

```
$ sudo nano config.txt
```

We need to disable the SPI port of Raspberry Pi as LinuxCNC uses these SPI pins for other uses. Look through **config.txt** for a line that reads `dtparam=spi=on` and edit it so it reads `dtparam=spi=off` (**Figure 6**). Then press **CTRL+O** to write the changes to the current file, press **ENTER** to do so. Then press **CTRL+X** to exit nano. For this config change to take effect, you now need to reboot the Raspberry Pi with `sudo reboot`.



◀ **Figure 7:** The collection of LinuxCNC applications on the Raspberry Pi 5

Next, we are going to use the LinuxCNC Stepconf Wizard application. You can find this by clicking the Applications dropdown menu, then selecting CNC > LinuxCNC Stepconf Wizard (**Figure 7**). Having launched the wizard, you will get a welcome screen titled Stepconf and you can click the Start button in the top right-hand corner.

On the next page of the wizard, Base Information, we are only going to make two changes. We need to give our machine a unique name; we went with 'rpirout', but it can be any name you choose. We then swapped the 'reset default machine units' to 'MM'. We then clicked Forward to move to the next pages. We didn't make any changes to the default settings until we got to the Axis X page. There follows a page to set up the X, Y, and Z axes on the machine – on our target machine, they are set up in a similar way as each one has the same type of stepper motor and same type of ball-screw lead-screw. However, the values entered on these pages will be different depending on what machine you have.

▲ **Figure 8:** Setting the axis values to match our stepper motor driver settings and our machine lead-screw pitch

For our machine, we needed to confirm that the 'Motor steps per revolution' was set to 200. This is the number of whole steps our motors need to receive to turn the output shaft one complete revolution. Many stepper motors are similar, but sometimes described as 1.8 degrees per step. Next, we need to set up driver microstepping. Microstepping is a way of subdividing each step and so is often represented as a number by which each step is divided; so '4' would equate to quarter-stepping. The older YOOCNC controller was set to do ⅛th microstepping, so we set up the new stepper motors to do the same. On the stepper motor drivers, there are details on how to set a series of DI1 switches to configure the stepper motor to certain microstepping values. We set the switches on each stepper for ⅛th microstepping and then, in the Axis dialogs in the wizard, set Driver Microstepping to 8 (**Figure 8**).

For our machine, one rotation of a motor equates to 4mm of travel of the carriage attached to that lead-screw, so for each axis we set the Leadscrew Pitch value to 4. Beyond this, we made no changes to any other settings in the Stepconf Wizard. There is a dialog to set up the spindle control values, but for now we are trying to simply get going and will manually control the spindle for testing. At the end of the wizard process you get an 'Almost Done' dialog and you can click Done to complete the process.

All being well, you should now see a desktop icon linking to a folder with the name of the machine you just created (**Figure 9**). Double-click on this icon and it will open a file browser. Among the files you will find one with the title of your machine followed by the suffix file type '.hal'; we are going to replace this file with a known working file from the Byte2Bot website to get us up and running quickly! Therefore right-click on the file (ours was called **rpirout.hal**) and select to rename the file to something that LinuxCNC won't try and use. We changed our name to **rpirout.halOLD**. Next, we downloaded this zip file from Byte2Bot: **rpimag.co/LinuxCNCzip**. Unzip the folder and inside the **cncRouter3Axis** folder you can find a file named **cncRouter.hal**. Copy that file into your folder and then rename it to match the file you just renamed. So we renamed **cncRouter.hal** to **rpirout.hal**.

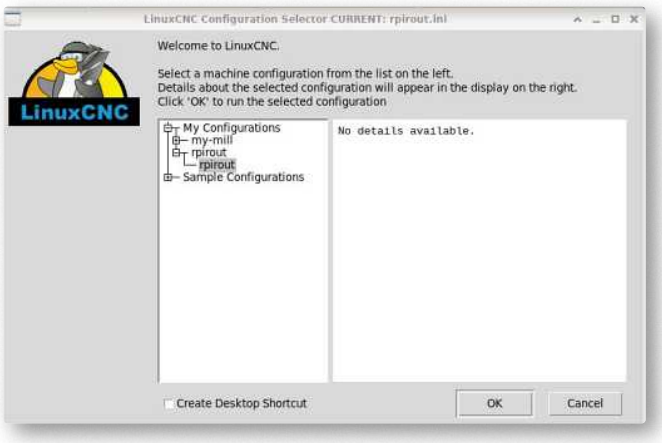
QUICK TIP

We realised that if you need to re-run the Stepconf Wizard trying to make changes to the current config, it would often break. It seemed a much stabler experience to create an entirely new config every time rather than try and adapt a previous attempt.

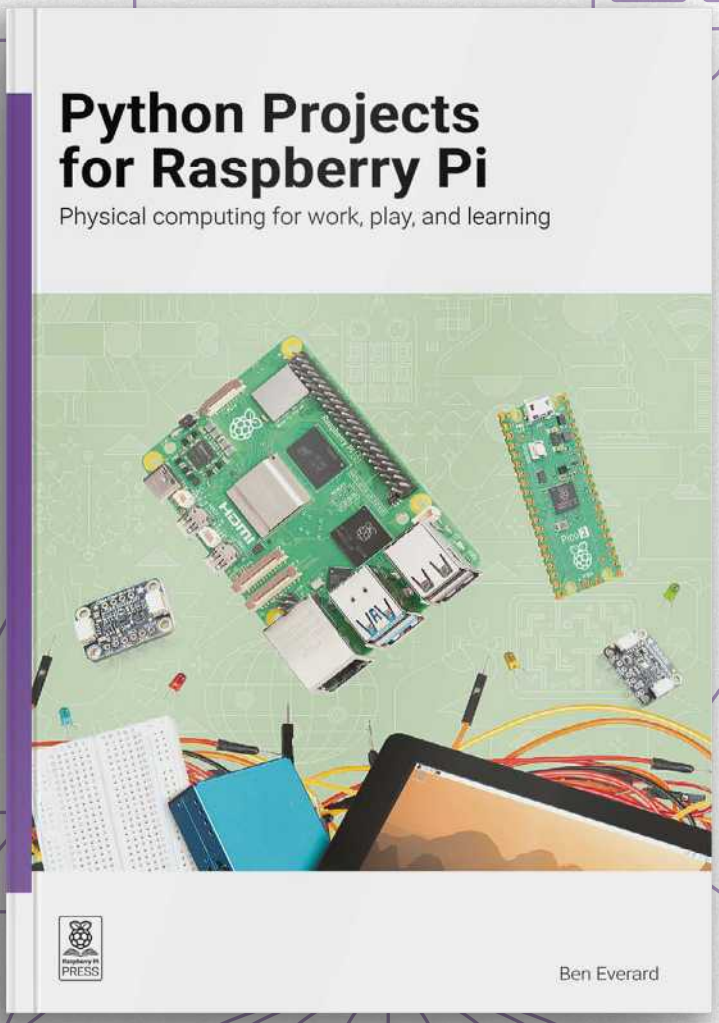
▲ **Figure 9:** The LinuxCNC desktop with the new rpirout machine configuration folder

With all being well you should now have LinuxCNC set up for your machine. If you click the Applications dropdown you can now select CNC > LinuxCNC. You'll be greeted with a small dialog asking you to select a machine configuration. Note that it wants you to select the name of the .ini file, not just the folder, which tricked us the first time we tried (**Figure 10**). All being well, you should see the main LinuxCNC page load with the test job (an engraving of 'LinuxCNC') loaded. You'll need to click the power button in the top left of the dialog and you may need to toggle the software emergency stop button to reset and unlock the machine. You can now hopefully jog the machine; to do this, you can use the arrow keys on your keyboard, with left and right controlling the X axis, and up and down controlling the Y axis. To move the Z axis, you can use the **PAGE UP** and **DOWN** buttons.

▼ **Figure 10:** You need to select the correct machine configuration each time you run the main LinuxCNC application



If you are happy with the motion of the machine, you can perhaps consider an air cut run of the loaded test LinuxCNC engraving job. We'd definitely recommend an 'air cut', where there is no tool or material in the machine and you minimise the chance of a crash. If all runs well, well done! It's not an easy process to get this set up. If you do have further issues, then reading the Byte2Bot troubleshooting sections is really useful and there are many forum posts and video tutorials out there on the internet to help solve any issues with LinuxCNC. For us, all that's left is to build the controller into some form of box/enclosure and then our machine is ready to work under Raspberry Pi control!


Python Projects
for Raspberry Pi
Physical computing for work, play, and learning
PRESS
Ben Everard

▲ **Figure 1:** The pieces of the large image make good drinks coasters

Part 06

Make jigsaws using FreeCAD with Python

Create jigsaw cutters using paths and pipes

Maker

Rob Miles

Rob has been playing with software and hardware since almost before there was software and hardware.

robmiles.com

▼ **Figure 2:** The author started the jigsaw ten years ago and still doesn't know where this piece goes

In this episode, we are going to learn how we can expand the capabilities of Python in FreeCAD by creating super-smooth curves and using them to make a 3D object. You can find all the sample programs at **rpimag.co/pythonfreecad.** You can use these to create your own 3D printable jigsaw designs.

Figure 1 shows two jigsaws depicting the awesome (in the author's opinion) Nissan Cube. The images have been converted into 3D printable mosaics that were cut to form jigsaws. In this article we are going to discover how we can do this in Python code running within FreeCAD. In the process we're going to learn about a couple of very powerful techniques, curve fitting and path following, that you can use in your own projects.

Simplifying jigsaws

Figure 2 shows the kind of jigsaw piece that we want to create. It has tabs on the ends and holes in top and bottom. If you look carefully, you will notice that the edges of the piece are slanted (i.e. not horizontal or vertical) and the holes and tabs are slightly different shapes. Our jigsaw piece will be simpler. The edges will not be slanted and contain tabs or holes that are all the same shape. Let's look at how we make a tab.

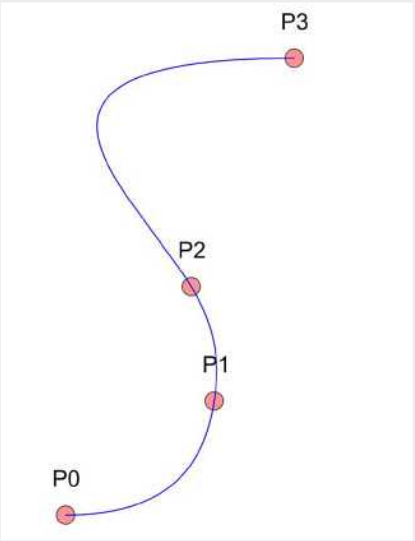


Figure 3: We will draw the other half by mirroring the curve

Opening a tab

Unlike other objects that we have worked with so far, we can't easily express the shape of a jigsaw piece tab as a mixture of arcs (portions of a circle) and straight lines. Instead, we are going to express the shape as 'control points' and then fit a curve through these points. Let's start by making half of a tab.

We are going to express the shape as 'control points' and then fit a curve through them

Figure 3 shows how we can make one side of a tab by defining four points and then creating a curved path through them. When the author was at school, one of his favourite drawing tools was his 'Helix Flexi Curve Ruler' (you can still get them). He would lay the ruler over a graph he was drawing and bend it so that it passed through all the points that were supposed to be in a straight line. Then he would draw a line with the ruler and from a distance the graph would look like his experiment was a success. We can get Python to do something similar for us by asking it to fit a 'B-spline curve' through the control points.

QUICK TIP

All our curves will be flat, but you can create curves in 3D.

When we create a curve, we specify the direction that the line should enter it, and the direction that the line should exit. This creates the bulge between points P2 and P3 in Figure 3, as the curve sweeps round to move in the direction of the exit line. Let's look at the code that makes this curve.

```
  def make_puzzle_tab_path(tab_width, tab_height,
is_hole=False):
```

B-spline curves and gameplay

The B-spline mathematical operation creates a smooth curve from a collection of points. Instead of connecting points with straight lines, it computes intermediate positions between points so that the line direction changes smoothly along the path. The shape of the curve is influenced not only by the position of a point, but also by the direction the curve approaches and leaves it.

We are using B-splines to make a curved jigsaw tab, but the technique is also very popular in computer games where you want objects to smoothly follow a path between points on a route, whether it is a racing car following a track, a chasing spaceship, or a non-playable character moving through a game world.

This is the definition of the function `make_puzzle_tab_path` in the **jigsawpuzzle.py** program. The function returns a FreeCAD wire that is used to create the jigsaw cutter. It is given the width and the height of the tab and a flag that indicates whether a tab or a hole is being created.

```
    y_dir = -1 if is_hole else 1
```

The first statement in the function checks the value of the `is_hole` flag and sets `y_dir` to `-1` if the function is drawing a hole and `1` if not. We will multiply the y (vertical) values by `y_dir` when we create the curve control points, to make the tab go down rather than up. This allows the function to create tabs or holes. Now we need some values for the control point positions.

```
    neck_width_fraction   = 0.35
    bulge_width_fraction  = 0.45
    neck_height_fraction  = 0.25
    bulge_height_fraction = 0.50
```

The above values were determined by trial and error. They give the fractions of the width and height of the tab that the 'neck' and the 'bulge' at the top of the tab will take up. If we want a wider tab, we increase `bulge_width_fraction`. If we want a longer neck, we increase `neck_height_fraction`, and so on. However, as you can see from **Figure 3**, the spacing of the control points does not set the specific size of these elements because of the way the curve fits through them. All these values are expressed as fractions so that we can draw different-shaped items for different-sized tabs.

Now that we have some control values, we can use them to calculate control point positions. But first we can work out some intermediate values that give the widths of the items in the tab.

```
    half_neck  = (tab_width * neck_width_fraction)
/ 2.0
    half_bulge = (tab_width * bulge_width_fraction)
/ 2.0
```

We will use these values to position control points in the x direction across the tab. We are placing points to draw a curve which is half of the width of the tab, so we need to work the actual width of the neck and the bulge and then divide this by two.

```
    centre_x = tab_width / 2.0
```

A lot of the calculations we are about to do involve the position of the centre of the tab, so we might as well calculate this for future use. Now we can calculate the position of the points in the tab. Let's place each point in turn. (Note that these are the positions shown on **Figure 3**.)

```
    p0 = App.Vector(0, 0, 0)
```

The code above uses the `Vector` type to express the position of the entry point into the curve. The first point is the easiest to place. It is at the bottom left-hand corner of the tab:

```
    p1 = App.Vector(centre_x - half_neck,
tab_height * neck_height_fraction * y_dir, 0)
```

QUICK TIP

The Z value of each of the vectors is always 0 because all the vectors are in the same plane, the flat jigsaw panel.

The statement above sets the position of the point that defines how the 'neck' is drawn. The x value is half the thickness of the neck back from the centre of the tab and the y value is the fraction of the height multiplied by the height.

```
    p2 = App.Vector(centre_x - half_bulge,
tab_height * bulge_height_fraction * y_dir, 0)
```

The statement above defines how the 'bulge' is drawn. It works in the same way as the neck fraction. The point is slightly higher up and to the left (as shown in **Figure 3**) to force the curve out and form the bulge at the top of the tab.

```
    p3 = App.Vector(centre_x,tab_height * y_dir, 0)
```

The statement above defines the end point of the curve. This is at the top of the tab, halfway across the tab we are drawing. Now that we have our points, the next thing to do is draw a curve using them. Curves are created using the `BSplineCurve` object.

```
    left_curve = Part.BSplineCurve()
```

The statement above creates a `BSplineCurve` object which we call `left_curve`. This object provides an `interpolate` method which creates the path:

```
    left_curve.interpolate(
        [p0,p1,p2,p3],
        InitialTangent=App.Vector(1, 0, 0),
        FinalTangent=App.Vector(1, 0, 0),
    )
```

The first argument to the call of `interpolate` is a list of the points the curve passes through. The second argument specifies the direction of line coming into the curve. We want to put the tab into the top of a jigsaw piece so the incoming direction will be horizontal (i.e. in the direction of x). The output of the curve should also be horizontal so that it will match up with the other half of the tab. When the `interpolate` method completes, the `left_curve` object holds the computed path. Now we can convert it into an edge so that it can be used to create objects.

```
left_edge = left_curve.toShape()
```

The statement above creates the curve and stores it in the variable `left_edge`. The `toShape` method takes the mathematical expression of the curve and creates a FreeCAD edge object which can be used to create objects. Now we need to make the other half of the tab.

They do it with mirrors

The variable `left_edge` describes half of the tab shape. To get the other half of the tab, we can reflect this edge in a mirror.

```
right_edge = left_edge.mirror(
    App.Vector(centre_x, 0, 0),
    App.Vector(1, 0, 0))
```

The statement above creates another edge called `right_edge`. Calling the `mirror` method on `left_edge` returns a mirror of that edge. If you are going to make a reflection, you need to know two things: where to put the mirror and the orientation of it. The first argument to mirror gives the position of the mirror (the centre of the tab). The second argument give the direction of the mirror, in this case we want the mirror to be vertical, i.e. in the x axis. When the `mirror` operation has completed, the `right_edge` variable holds the mirror image of the left edge.

The next thing we need to do is make sure that the points in `right_edge` are in the correct order. An edge is expressed as a series of points which are traversed in a particular direction. The left edge starts at the bottom left of the tab and ends up on at the top in the centre. When we mirror something, the direction of the traversal is retained. In other words, the right-hand edge has the correct shape, but is traversed from the outside in, which we don't want.

```
right_edge = right_edge.reversed()
```

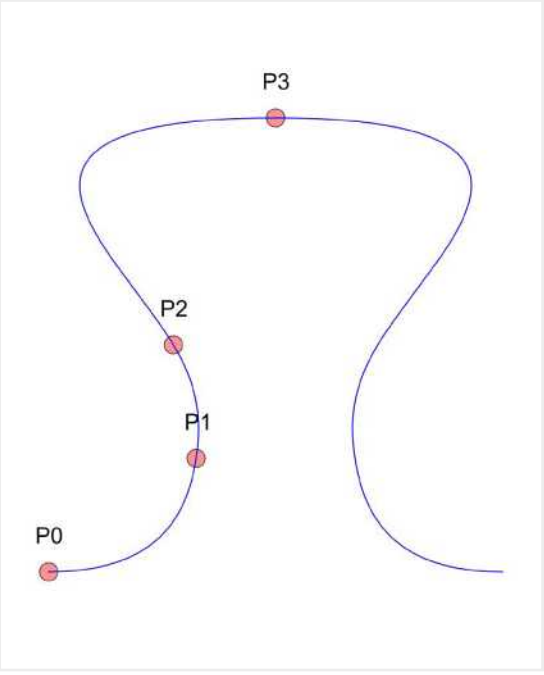
The statement above uses the `reversed` method on the value created by `mirror` to create a version of the edge which is traversed from left to right. Now we have a right edge with the required shape and direction. We can use this to create a `Wire` which defines the shape that we want:

```
return Part.Wire([left_edge, right_edge])
```

The statement above creates a `Wire` is constructed from a list of edges and then returns it. The `make_puzzle_tab_path` function returns a `Wire` which can be used to make a jigsaw tab.

Cunning plots

Figure 4 shows the completed tab wire. You might think that using `BSplineCurves` is hard work, and you'd be right. If the workings of the `make_puzzle_tab_path` function haven't hurt your head at least a little bit, then you are a much better programmer than the author. But this is a terrifically powerful technique. Designs often call for smooth curves which would be



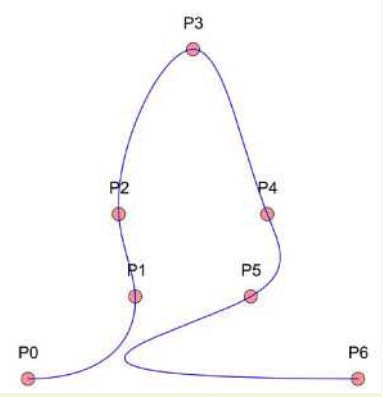
▶ **Figure 4:** An inverted tab would go down rather than up

We are making a cutter to be used on a mosaic panel to turn it into a jigsaw

Crazy curves

You may be wondering why we don't just define points for all the parts of a tab and fit a curve through them. This is because B-spline curves don't behave in a consistent way, as you can see above when the author tried to do this. You end up having to add more control points to get the curve to behave itself, and you can never be sure that what you have made is precisely symmetrical. It is best to make one half using a curve and then mirror it to make the other half.

▼ This would work as a jigsaw piece tab, but only just



QUICK TIP

A modern 3D printer can print objects around 0.75mm apart and the mosaic that the example code creates is 4mm thick.

hard to represent any other way. We can create different-sized jigsaw puzzle tags very easily and all of them will look smooth. You can use `BSplineCurves` to create organic-looking objects from just a few starting values. However, the next thing we need to know is how to take our curve and turn it into something we can use to make a jigsaw. And to do that, we need to use a bit of piping.

Follow the path

We are making a cutter to be used on a mosaic panel to turn it into a jigsaw. **Figure 5** shows one line of the cutter. When we cut this shape from a solid panel, it makes one row of the jigsaw. We've created a wire which describes the tabs. Now we want to create a solid line. The line is created in the jigsaw program by a function called `make_cut_line_solid`.

▶ **Figure 5:** This cuts one line of the jigsaw pieces

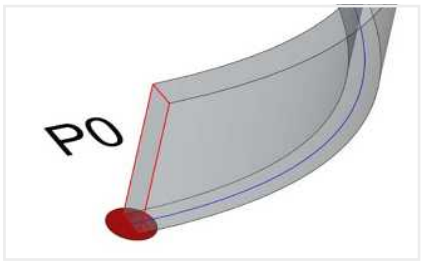
```
  def make_cut_line_solid(piece_span, flips,
total_thickness, cut_size):
```

The code above shows the definition of this function. It creates a cutter line containing tabs and holes that is used to cut a row of pieces. The function accepts four arguments. The size of each jigsaw piece is given by `piece_span`. The `flips` argument is a list of True/False values to be fed into successive calls of `make_puzzle_tab_path` to select tabs or holes. The length of flips gives the number of jigsaw pieces in the line. In the case of the line in **Figure 5**, the `flips` list looks like this:

```
  flips = [True,True,False,True]
```

There are four jigsaw pieces, and the third one has a hole in it. The function argument `total_thickness` gives the thickness of the mosaic panel and `cut_size` gives the width of the cut.

Code in the `make_cut_line_solid` function creates a wire containing all the tabs and holes which need to be converted into a cutter object (i.e. something which describes the path like the one shown in **Figure 5**). The wire is stored inside the method in a variable called `path_wire`. Let's look at how this wire is converted into a shape that can cut out jigsaw pieces.



▲ **Figure 6:** The sweep profile is shown in red. The path that is followed is blue

Sweep to victory

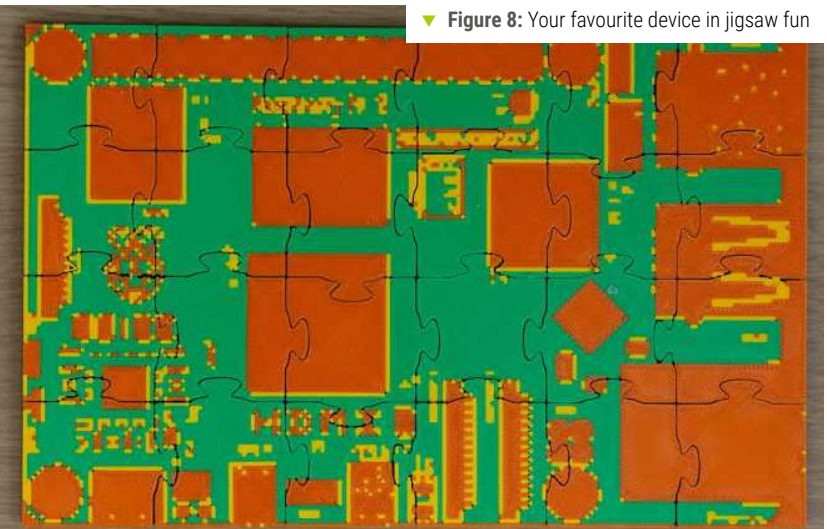
We create a 3D object from a path by 'sweeping' a shape along the path described by a wire. This is a bit like the extrusion process we have seen before, but in this case the extrusion follows a path, rather than just moving in one direction. We are going to use the `makePipeShell` function, which is provided by the `Wire` type. But first we need a shape to sweep along the wire. We are going to use a simple rectangle:

```
half_c = cut_size / 2.0
profile_wire = Part.Wire(Part.makePolygon([
    App.Vector(0, -half_c, 0),
    App.Vector(0,  half_c, 0),
    App.Vector(0,  half_c, total_thickness),
    App.Vector(0, -half_c, total_thickness),
    App.Vector(0, -half_c, 0),
]))
```

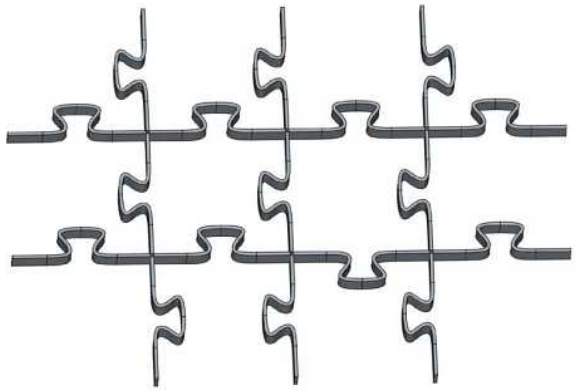
The code above creates a rectangular shape called `profile_wire`. The shape is the width of the cut and the thickness of the mosaic panel being made into a jigsaw. The `makePolygon` function is given a list of vertices (positions in 3D) which define a polygon. The list must be 'closed', which means that the last entry in the list should at the same position as the first.

Figure 6 shows how a profile is swept along a path. The `profile_wire` shape is centred around the curve path and will sweep along the path, For Free Mags Check Sanet.st creating a shape as it moves. We can make all this happen with a single method call.

```
    line_solid = path_wire.
makePipeShell([profile_wire], True, False)
```



▼ **Figure 8:** Your favourite device in jigsaw fun



▲ **Figure 7:** Vertical cutters are created by rotating a cutting line by 90 degrees

This is the code that calls the `makePipeShell` function on the `path_wire` variable that contains the cutting path. The `makePipeShell` function accepts three arguments. The first is a list of shapes to be swept over the path. For our program, this list just contains the `profile_wire` rectangle profile we made. The next two arguments are a bit obscure. The first one sets the value of `isfrenet` to `True`. This tells `makePipeShell` to let the profile turn as it moves along the shape. This works best with complex curves like the ones we are using. The second argument, `transition`, controls the behaviour of the sweeping process when it hits sudden transitions. This should be set to `False`. The author has found that these values work well and would advise you not to change them. The variable `line_solid` is assigned to the shape that is produced. This is used to make the complete cutter.

QUICK TIP

You can make really interesting objects by sweeping different shapes along a path.

Figure 7 shows a completed jigsaw cutter. You can create quite small jigsaw pieces, but when things get really small you can run into trouble if the radius of the curves in the tags becomes less than the size of the cut. If this happens, the `makePipeShell` function fails and creates objects which will not print correctly. Check the output of the program to make sure that the back of the jigsaw is properly formed.

A life in pieces

Figure 8 shows you can create personalised jigsaws from any image you like. You can find the complete jigsaw cutter in the repository for these articles: **rpimag.co/pythonfreecad**. You can also find a 'random' version of the jigsaw maker which places each tag or hole in a slightly different position on the line, making each jigsaw piece unique.

Discover découpage

Cover a caddy and add a Raspberry Pi Pico-powered opening mechanism

Maker

Nicola King

Nicola is a freelance writer and sub-editor. This month she's been glued (literally) to the kitchen table...all in the name of meshing craft and tech!

@holtonhandmade

Découpage, or the art of decorating an item with paper cutouts, is a craft that can transform an object. Perhaps you have an old piece that needs new life breathed into it, or you have an ornament that you want to decorate – découpage may be the answer. This is also a pastime that is extremely satisfying to undertake. As a crafter/maker, it is one of this author's favourite crafts, as the transformation happens very quickly, it's possibly one of the easiest crafts you can do, you can easily disguise any 'mistakes', and it's a mindful hobby that you can just lose yourself in.

It's a mindful hobby that you can just lose yourself in

In this tutorial, we are going to take a hinged box, découpage its surface, and then use some tech to add a touch of unique Raspberry Pi Pico magic to make it open and close.

▲ Découpage is not an expensive pastime, and once you have a selection of papers, your découpage medium, and some brushes, there is nothing to stop you decoupaging anything that's not nailed down!

QUICK TIP

We've used specialist découpage paper, but you could use napkins, tissue paper, newspapers, wrapping paper, or other suitable thin papers.

Get gluing

For this project, we decided to use a plain, hinged papier mâché-type box that we bought from a craft store. There are many of these craft 'blanks' available, so have a look online or in-store. You may wish to take a different route and purchase a mâché animal blank, such as a dinosaur, and use your tech to make it roar, for example. This tutorial is merely a guide to get some ideas popping, so do experiment.

Requiring just some specialist découpage paper, a glue medium, and a brush, you can see that this is clearly a craft that won't break the bank. We were going for an olde-worlde, vintage theme, and we selected a paper depicting old maps. Apart from giving a vintage vibe, this paper also seemed to fit very appropriately with our box's purpose – our aim was to create a little treasure chest that we can keep precious bits and pieces in. The découpage medium that you use could be a branded one, a store's own brand, or you could simply water down some glue and use that, which would be a cheaper option. We opted for a

◀ Our découpage chest opens when a magnet is placed near the sensor at the front

QUICK TIP

You should really leave your découpaged item to dry for a minimum of three hours so that everything sets in place nicely.

YOU'LL NEED:

- An item to découpage
- Decorative thin paper
- A sealing and finishing découpage medium (or a mixture of white glue and water)
- A soft brush for application
- Protective cover for work surface
- Scissors
- Roller (for smoothing – optional)
- Acrylic paints (optional)
- Sandpaper (optional)
- Damp rag for smoothing and removing excess glue medium
- Sealant (if not using an all-in-one medium)
- Raspberry Pi Pico
- USB power bank
- Servo motor and horn extender (e.g. lolly stick)
- Hall sensor and magnet

gloss finish and a medium that glues, seals, and finishes, simply because of the speed and convenience it offers.

Once you've prepared your work surface with a protective cover, take your papers and start tearing them up into small pieces, bearing in mind that the smaller and more intricate the item that you have chosen is, the smaller you'll need to tear or cut the paper pieces that will cover it. Then, select a section to work on, and cover it with some of your glue medium using your brush. Start placing your papers on the glued area and take your brush (with some glue medium on it) and smooth it down. Now, one thing you will likely experience when découpaging is the appearance of air bubbles under the papers. We have found that smoothing these out as we go helps, as does using smaller pieces of paper, and keeping the paper as taut as possible while you work on it.

When your piece is covered, leave it to dry. In our case, we covered the box in sections (as we had a hinged lid that we wanted to dry before we did anything else), we let each section dry, and then carried on. It's entirely up to you how you approach this. You could actually paint your item in maybe a bright acrylic paint, and then just découpage small sections of it, leaving the colour to show through in the gaps between. Experiment and

Past papers

Like so many crafts that endure, découpage is not new; in fact, it's a pastime that has been around for hundreds, if not thousands, of years. It's thought that its roots may be in East Asia (specifically East Siberian tomb art). In China during the 12th century, intricate paper cutting techniques were pioneered, which effectively laid the foundations for the development of découpage and paper art in general over the centuries.

Europe really latched onto découpage during the 17th and 18th centuries. The Italian and French aristocracy, in particular, enjoyed embellishing objects such as screens and furniture with designs. Varnishing and gilding also came to the fore in this period, so intricate paper découpage was really shown off to its full potential. The Victorian era was one which relished arts and crafts, so the pastime became very popular during this period, with sentimental and floral imagery taking centre stage.

The 20th century saw major developments in the materials available and, today, mixed media, digital prints, and eco-friendly materials have taken découpage techniques to a higher and very accessible level.

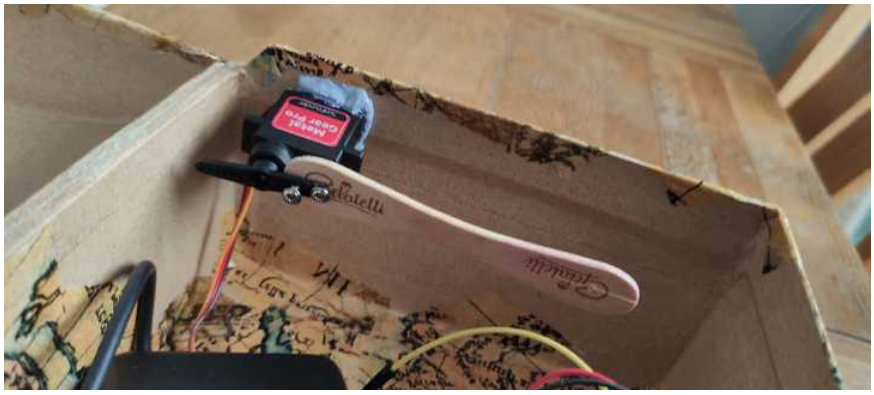
▲ The finished box, fully decorated and ready for its contents

QUICK TIP

You can also découpage on metal, e.g. a tin can. Make sure that you paint it with a layer of acrylic paint or primer first, though, before you apply any papers.

▲ You only need a thin layer of glue to attach the pieces of paper, so don't overdo it

▶ We attached a lolly stick to our servo horn to extend it enough to reach the box lid; you could use a different method



see what sticks! If you wish, you can then give your item a final gloss coat of sealant when your papers are dry, just to finish the piece off.

One point that we would like to labour is: do remember to wash your glue-covered brushes out at regular intervals, and especially when you finish découpaging for the day, otherwise you may as well fling them straight in the bin!

Open sesame

We decided to make our chest open 'magically' with a magnetic 'key'. For this, we used a standard metal-gear servo for the lid-opening mechanism, and a Hall sensor to detect the presence of a magnet. For the brains, we opted for a Raspberry Pi Pico (any model will work).

We wired the sensor's digital output to GPIO 22; it registers 1 when not triggered, and 0 when a magnet is placed close to the sensor. We also connected the sensor to 3V3 power and GND pins on Pico.

The servo requires 5V power, so we connected it to Pico's VBUS pin, along with GND and GPIO 15 for the PWM (pulse-width modulation) output to control the servo's rotation angle. To flip the lid open, you need to extend the servo horn in some way; we screwed an ice lolly (popsicle) stick to it, but you could use glue, or replace the stick with a strong metal wire or rod. You also need to secure the servo body to the inside wall of the box; we bodged it with sticky tack, but you could use glue or another method.

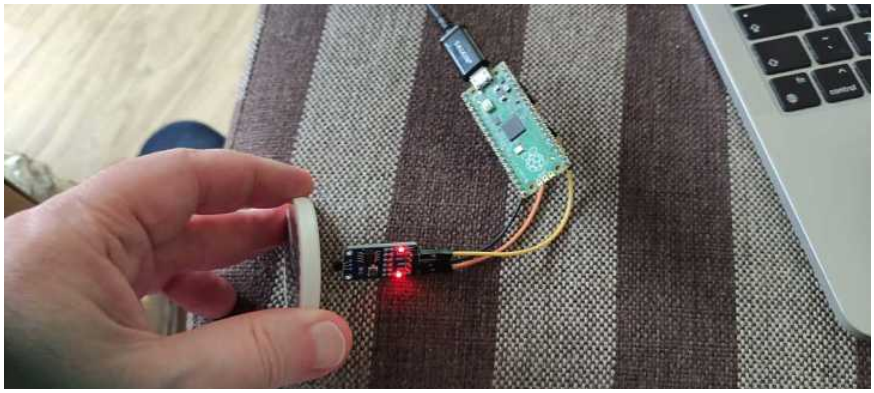
We stuck our Hall sensor to the inside front of the box and just placed our Pico and the power bank loose in the box – you could hide them in a compartment to keep them away from whatever you want to store in it.

Finally, we need to create a MicroPython program (see the **chest.py**) listing) that detects when a magnet is placed near the sensor and triggers the servo to rotate and open the box. In this case, we're rotating it by 90° – you may find another angle works better for you, so experiment (and also with the servo horn positioning). We also opted to get the box to stay open for ten seconds before the servo rotates back again to close the lid; you could make the delay longer.

To get the program to run on bootup, save it as **main.py** on Pico. You can then power it from a USB power bank or battery pack placed inside the box and the code will run automatically.

A sticky end

For this project, we wanted to make something that could actually be used afterwards, and this box might be a good place to throw our keys, coins, and other important items. The craft element of the make is clearly not difficult, and would certainly appeal to younger family members as well as adults who want to lose themselves in tearing and gluing, which is a very restful way to spend half an hour. The tech element is a fun one to play with, so adopt, adapt, and improve. Why not have a go, and send us pictures of your creations to **magazine@raspberrypi.com**?



◀ Testing the Hall sensor with a fridge magnet before placing it in the box

QUICK TIP

Banish the bubbles – smooth as you go, and minimise the bubbles of air on your piece, as these will affect the end look and may mean that the papers pull away from the item over time.

QUICK TIP

A pair of tweezers are really handy if you need to reposition a piece of découpage paper, so have some on hand, just in case.

▲ This is a good craft to do while watching TV, as you don't need great concentration

Easy upcycling

If you don't consider yourself to be particularly 'arty/crafty', then trying out some découpage on an old item that you feel needs new life breathed into it is a very neat way of starting a new and mindful craft. Indeed, you could even consider upcycling a large piece of furniture with découpage, instead of just repainting it. As long as it's solid and in a reasonable state of repair, consider most items of furniture fair game!

You can buy découpage paper specifically designed for furniture projects and these are some particularly stunning examples: **rpimag.co/mintdecoupage**. Do make sure that you choose an overall design that fits the scale/size of your piece of furniture too.

As long as you use a good sealant, the furniture that you transform should be very durable over time. We've even seen someone who suggests you can also découpage with pieces of fabric on furniture items… maybe something to try down the line, if the découpaging bug bites!

chest.py

> Language: **MicroPython**

DOWNLOAD THE FULL CODE: rpimag.co/github

```
001. from machine import Pin, PWM
002. from utime import sleep
003.
004. #Set pin for sensor data
005. DIN = Pin(22,Pin.IN)
006.
007. #Set servo pin and frequency
008. servo = PWM(Pin(15))
009. servo.freq(50)
010.
011. #Servo angle function
012. def set_angle(angle):
013.     duty = int((2000 * angle / 180 + 500) *
     65535 / 20000)
014.     print(angle)
015.     servo.duty_u16(duty)
016.
017. while True:
018.     if(DIN.value() == 1):
019.         print(
     "The Magnet is far, stay closed")
020.         set_angle(0)
021.     else:
022.         print(
     "The Magnet is near, open up!")
023.         set_angle(90)
024.         #Stay open for 10 seconds
025.         sleep(10)
026.     sleep(0.5)
```


PROJECTS FOR MAKERS & HACKERS
BUILD A CUSTOM KEYBOARD WITH RASPBERRY PI PICO
CUSTOM CONTROLS WITH A 3D PRINTED BIG BUTTON
BOOK OF
MAKING
2026
CONTROL PROGRAMMABLE LEDS FOR DAZZLING LIGHTING
EVERYTHING YOU NEED TO KNOW ABOUT BATTERY POWER
FROM THE MAKERS OF RASPBERRY PI OFFICIAL MAGAZINE

BOOK OF MAKING 2026

STEP INTO THE WORLD OF MAKING!

- DISCOVER YOUR NEW FAVOURITE HOBBY
- LEARN THE DIGITAL SKILLS THAT MAKE THE WORLD GO ROUND
- TRY YOUR HAND AT 3D PRINTING, ELECTRONICS, PROGRAMMING, AND MORE
- DISCOVER PROJECTS THAT YOU CAN DO IN AN HOUR, AFTERNOON, OR WEEKEND

rpimag.co/BookOfMaking2026

Part 12

Conquer the command line: save it now!

Learn how to protect your data with backups and disk wipes

Maker
Richard Smedley
A tech writer, programmer, and web developer with a long history in computers, who is also in music and art.

about.me/RichardSmedley

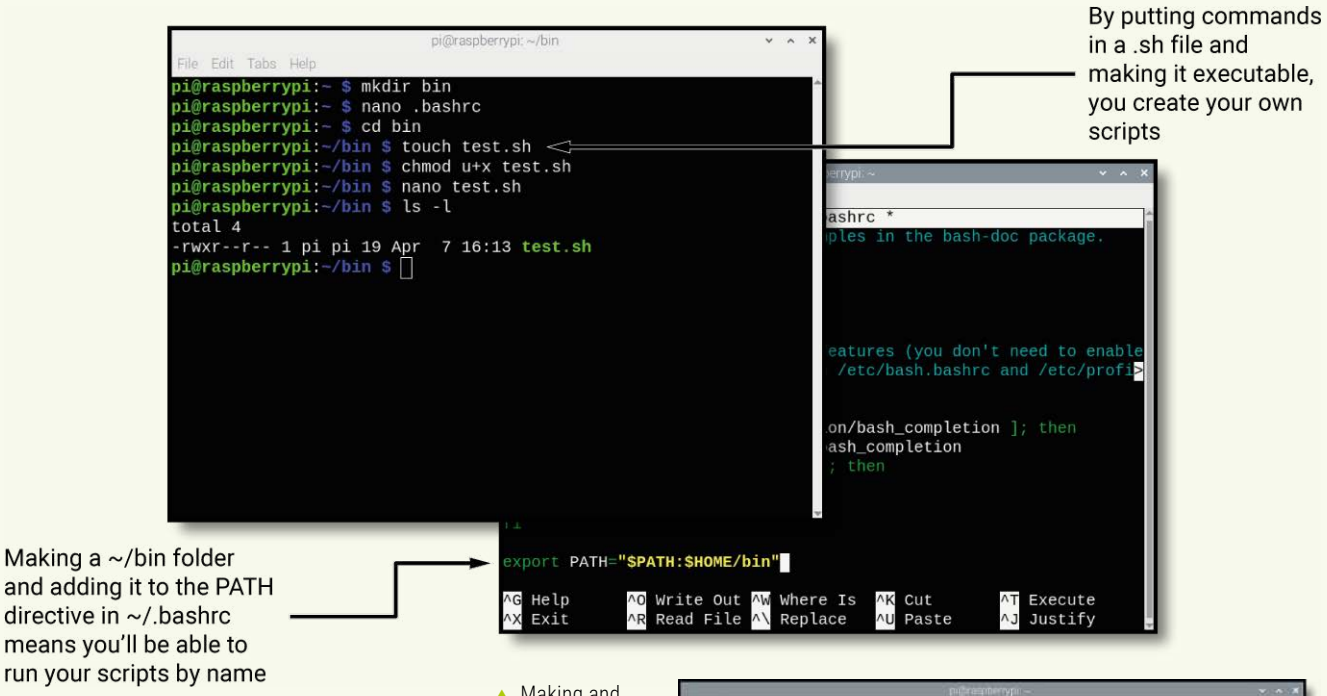
Perhaps you've got backups running automatically on your main computer, or perhaps you just back up what's important now and then – it's OK, we're not here to judge. But we will say that anything you don't have backed up, you don't have. Computers break. Hard disks and SSDs break. Accidents happen. The unexpected is somewhat inevitable.

IT professionals prepare as if they could lose everything at any moment, at least the best do. That might sound a tall order for a small single-board computer you bought at a low price to use in a hobby project, but your data is the most important thing on it, and good backups are a useful discipline to take elsewhere. Fortunately, the command line can actually make this easier; we'll show you some of your best options for (relatively) painless backups.

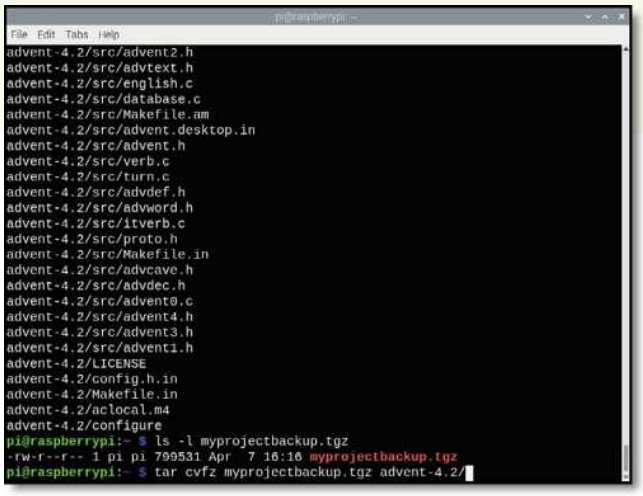
Some of your best options for (relatively) painless backup

Simplest of all is to copy the data and move it elsewhere. It's labour-intensive, compared to automatic solutions, but for only occasional backups it's certainly better than nothing, so we will not ignore this option.

Whether you're copying to disk, or moving the backup to another machine, it's best to make it as small as practically possible. The zip compression format may not compress as much as some specialist Unix choices, but it does mean that other users should be able to open the file(s) with their standard compression software. You could go with LZMA or bzip2 compression (from `tar`, using `-J` or `-j`, respectively) for better compression to a smaller file size, but although alternatives to gzip save a little more space, they can take far longer to perform the compression.



▲ Making and running scripts



▲ A single command wraps up all of the files and subfolders, then compresses the tar archive with gzip

You can use a single `tar` command to gather all the files in one directory and compress it into an archive:

```
$ tar czvf mybackup.tgz myfolder
```

The `c` switch tells tar to create the archive, diving into all subfolders found; `f`, use the named file (here, **mybackup.tgz**), is necessary to direct the output away from the terminal, or – historically – a tape device. Yes, tape, because `tar` is short for tape archive, as telling a sign of its longevity as its frequent use without a dash in front of the command-line options. `v`, as with many commands, asks for more verbose output, so the program tells you what it is doing and what (if anything) has gone wrong.

Lastly, `z` invokes gzip compression – saving the extra step of creating a .tar archive, then running it through gzip. To unpack the archive, substitute `x` for extract in place of `c` – you can omit the `z`, as tar will recognise the compression type and automagically deal with it (this will overwrite the files you just archived if you run this command from the same folder where you created the archive):

```
$ tar xvf mybackup.tgz
```

Tape archive

`tar` dates from the days when computers backed up to big tape reels, those essential props of 1960s and 1970s sci-fi films. The lack of a file system on tapes means that `tar` must save all of the file system info such as ownership and timestamps into its own format.

Password free

Using your key to log in is a convenience you quickly get used to – beyond `scp` to your backup server, try it on any machine under your control that you have to log in to.

A safe home

Often, using tar to back up your personal files – with `cd / home`, then `tar` on your home folder – will be all you need, but if you have data across directories, from **/etc** to **/var/www**, it's simplest to back up the entire microSD card. We looked at copying a new Raspberry Pi OS image onto a card in part 10 (**rpimag.co/165**); backing up your disk is almost a mirror image of that process, which you can easily do on another GNU/Linux computer with a USB card reader. Power down your Raspberry Pi, put the SD card in the card reader, and connect it to the other computer. If your computer automatically mounts it, use `sudo umount` to unmount it.

You'll be creating a file as big as the entire SD card – usually 16GB or more – so make sure the computer you run this command on has enough space. It helps that you will be compressing the image file as it is created, which, for a Raspberry Pi with a modest amount of data on its SD card, will result in an image of 4GB or more. Look back at the disk image section in part 10 for how to be sure which device the USB card reader is – for a card plugged in as **/dev/sdb**, and unmounted, do:

```
$ dd bs=4M if=/dev/sdb | gzip > rpiback.img.gz
```

Open another Terminal tab and monitor your disappearing disk space with `df` – if you don't think that it will fit, stop the `dd` operation with the usual **CTRL+C**, then `rm` the image file that you have partially created, and go and perform the backup on a computer with more disk space – or with a backup drive mounted, to which you copy the archive directly. Turning the backup into a usable microSD for the Raspberry Pi means piping the other way, from `gzip` to `dd`:

```
$ gzip -cd rpiback.img.gz | dd bs=4M of=/dev/sdb
```

For disk operations like `dd`, you'll need root permissions: you can prefix `dd` with `sudo`, but for saving the file outside of your home directory, you may also need `sudo` – which means typing it in front of `gzip` as well.

This can be mildly inconvenient as `sudo` does demand your password every time – but if you need a reliable way of becoming the root user for every operation: running `sudo -s` will give you a shell with root permissions, but remember to `exit` afterwards. Alternatively, a chain of commands can be run with full admin permissions. For example, to restore your backup from a file to an SD card:

```
$ sudo bash -c "gzip -cd rpiback.img.gz | dd
bs=4M of=/dev/sdb"
```

Remote copy

It's good to be able to make backups as required, using removable drives, but to move towards systematic backups you'll need to copy across the network. We mentioned SCP in part 9 (**rpimag.co/164**). To copy your backup file to another machine, one that's running the SSH server, pass your login name with the command:

```
$ scp -p rpiback.img.gz pi@192.168.0.207:/home/
pi/bak/
```

You will then be prompted for the user password. Change the `pi@` to whatever your username is on the remote machine, not the Raspberry Pi you're copying from. The `-p` preserves information such as when the files were last accessed. Note that `-P` (capital p) can be used to specify a particular port number. Make sure you create the destination directory with `mkdir` before you copy the file.

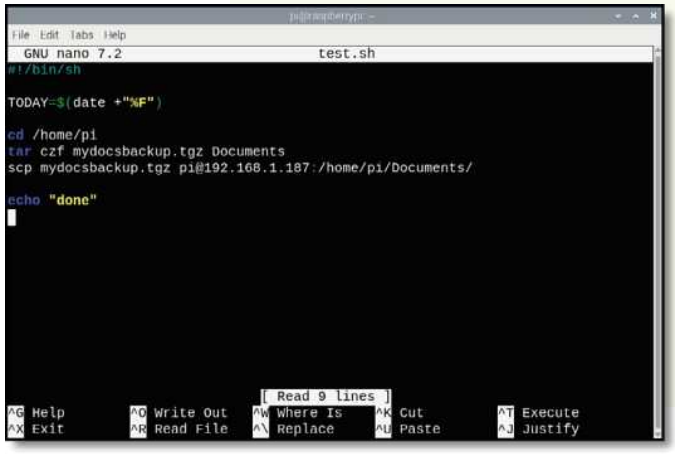
We showed you how to set up a Raspberry Pi with a fixed IP address in part 7 (**rpimag.co/162**); that Raspberry Pi, with a plugged in USB disk drive, could be an inexpensive backup machine, as well as media server or whatever else your home or office needs.

Because you're sending these commands through the Bash shell, you get all the usual Bash advantages, from tab completion (just type `bac`, or however much of the file name is unique in your present working directory) to wildcards. If you have disparate archives in the same directory – such as **www-backup-20241225.gz** and **data-backup-20241226.gz** – then copy them all with:

```
$ scp -p ./*backup*gz pi@192.168.0.207:/home/pi/
bak/
```

Which files?

Apart from **~**, your project may have config files in **/etc** or even **/opt**, and under **/var** are logs, web config, and files – all of which you may have modified for a project.



A datestamp in our script means that we are not producing a backup with the same name each day we run it

So far, so good, but if you want to automate your backup process, the interactive element – typing a password – would be better avoided. As long as you can maintain security in some other way, of course.

Efficient backups

Back in part 9 when we set up our SSH server, we generated keys with `ssh-keygen` – these keys can be used to provide passwordless login. You can copy them across to other machines as we did in the secure key section, and you'll be all set to automate remote backups – but if you create a tar file every time, you'll waste a lot of disk space duplicating data that changes infrequently.

`rsync` lets you copy data in much the same way as `scp`, but uses a delta-transfer algorithm, to only transfer the difference between the copies of the source file on your disk, and the remote, saved version. This both saves bandwidth used and avoids cluttering up your backup disk with multiple near-identical versions of a file. It's also handy if you're paying a cloud provider for storage and data transfer.

If your version of Raspberry Pi OS doesn't have rsync, it's just an `apt install` away. Typically, rsync uses SSH for transport, but you can set up a server running an rsync daemon, and directly contact the **rsync://** URL over TCP (defaults to port 873). In this case, set an `RSYNC_PASSWORD` environment variable or use the `--password-file` switch.

`rsync` is just one of many tools for archiving your precious data. In addition, there is the possibility of using version control systems, such as git, for both backing up, and tracking changes on, important files.

Hash bang

The shebang, or hash bang – `#!` – at the start of the script is an instruction to the program loader to run the program immediately afterwards – in this case an interpreter directive to run **/bin/sh** – and pass the script as an argument to it.

Script-it-yourself!

But let's row back to simpler commands. We have seen from early on how powerful Bash can be by chaining together a few commands; another way of putting commands together is to bundle them into a script – a short program simply comprising a small number of Bash commands and known as a shell script. Take a look at this code – try typing it in to your favourite text editor, adjusting it for the IP address of your networked backup server, and backup folder location (or change the `scp` line to a `cp` to a plugged-in backup drive), and saving it as **test.sh**.

```
#!/bin/sh
cd /home/pi
tar czf mydocsbackup.tgz Documents
scp mydocsbackup.tgz pi@192.168.0.207:/home/pi/bak/
```

Then make the script executable with:

```
$ chmod u+x test.sh
```

...and run it with `./test.sh` – any problems, then check the names, network address, and did you copy your key over to the other computer?

Now we have a script that saves a folder, and copies it remotely, do you notice any potential problems? Each time you run it, it will overwrite the previous **mydocsbackup.tgz**, both locally and remotely. We need a way to put a timestamp on the backup name:

```
#!/bin/sh
TODAY=$(date +"%F")
cd /home/pi
tar czf mydocsbackup-"$TODAY".tgz Documents
scp mydocsbackup-"$TODAY".tgz pi@192.168.0.207:/home/pi/bak/
```

Why remote?

As well as being able to centralise backups for more than one machine, a remote backup protects you from unexpected disasters such as fire or flood where your Raspberry Pi is. Fairly unlikely? Yes, but that doesn't stop you insuring your house.

What we have done is set a variable – `TODAY` – to the current date, in YYYY-MM-DD format, which we can now access with `$TODAY` (remember, we permanently set variables for the shell named this way, when changing the prompt in part 5, **rpimag.co/159**). You can run `date +%F` in the terminal – `date --help` will show you the many other format options. Now you can automate it by putting the script somewhere like **/usr/bin** (and with a better name than **test.sh**), and running it regularly with cron, as we covered in part 8 (**rpimag.co/163**).

Shell scripts tend to grow; there is always room for improvement. Here, you may want to back up more than one directory, for example, or use `echo` to let users know what the script is doing at each stage. You could even make it interactive, letting users choose which directories to back up.

There are plenty of shell scripting tutorials online, and great books to take it further, but Raspberry Pi OS itself holds many great shell scripts, from which you can learn by example. To see how a script can be organised to still be maintainable with over a thousand lines of code, have a look at **/usr/bin/raspi-config**.

However grand or modest your scripting ambitions, don't be afraid to try things out: build up gradually, and test your code each time, so that you know where to look for any errors that you introduce. Help is at hand from the Raspberry Pi forums, and pasting your code into **shellcheck.net** will give you valuable feedback – for example, advising you the `cd` line of our backup script should be:

```
$ cd /home/pi || exit
```

...in case the `cd` step fails – this is generally a good idea, although not so important in this particular case. Now that we have a choice of backup options, one task remains: securely getting rid of data from disks. This is a concern for anyone handling other people's data, or just protecting their own privacy and security.

Through the shredder

Back in the 1960s, if you wished to cover your tracks, *Mission: Impossible* told us it was done by the show opening's taped mission assignment finishing, "This message will self-destruct in ten seconds...", and boom went the tape player.

In these digital days, protecting privacy or security means understanding how a disk drive actually stores data, so that you don't dispose of old disks under the mistaken impression that you've securely erased data, when you haven't.

Disk space is collected into blocks, sized typically at 4096 bytes, that are indexed by the file system with inodes so that the disk controller knows where to send the read head to retrieve information. SSDs and flash drives don't have read heads, but still organise the data in a similar fashion. Most disk operations occur at the inode level – so moving a file between directories on the same disk partition is simply done by relabelling an inode. `rm` does not delete the data stored, just the reference to its blocks on the inode.

Plug in a disk drive after someone has done `rm -rf` on it and it will look empty, but use a low-level utility and you'll have access to all of the jigsaw puzzle pieces needed to put the data back together again. So far, so *NCIS*, but can this be significant to the average Raspberry Pi user? Given the range of projects out there, and the multifaceted data that they collect, to stay on the right side of new and future data protection laws it will be useful to know how to securely remove data from your disks.

Caution

Before we start erasing disks, think of the carpentry maxim, 'measure twice, cut once'. It's easy to mistakenly erase the wrong disk or partition if you're not paying attention. Given enough opportunities, most of us do it, and it's often the lesson that teaches us to make proper backups! Running through other operations on the disk (`mount`, `df`, `ls`, `umount`), before erasing, works as a sanity check that you're addressing the correct partition.

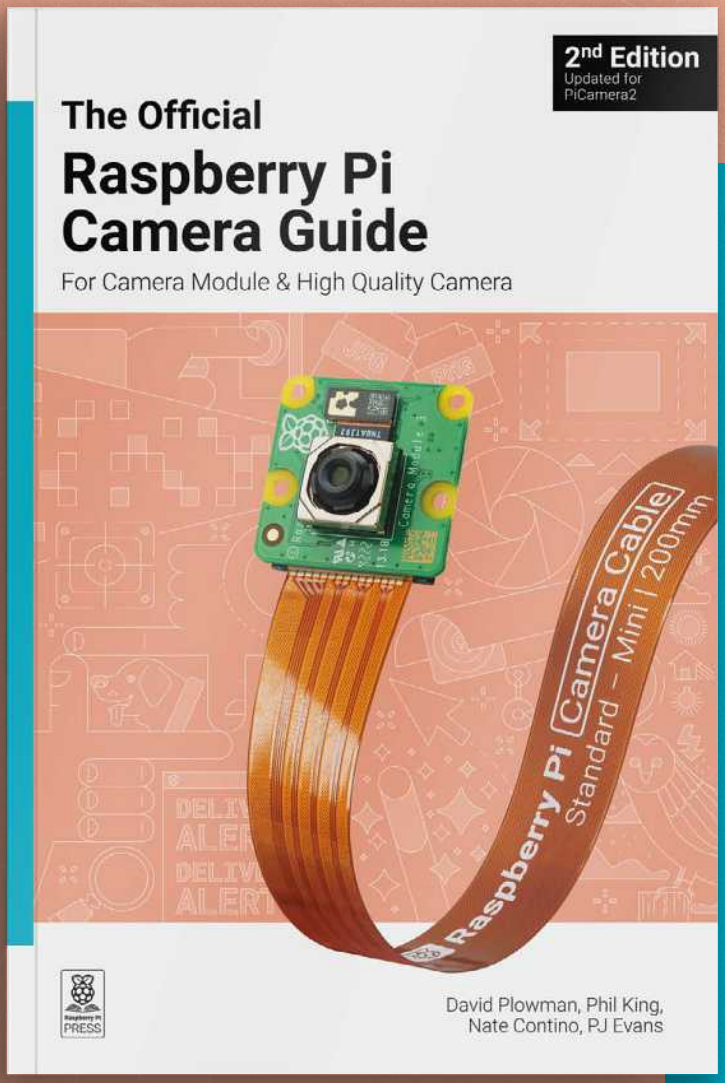
Now to those blocks. You can overwrite every bit of information, either with all 1 digits, or with totally random data, using `dd`. Even then, with magnetic disks, it's theoretically possible to recover the data, and multiple overwrites will be necessary – but before you worry about writing a script to do that, let us introduce `shred`, a utility that does just that, overwriting with as many passes as you select as a command switch (or defaulting to three):

```
$ shred -vf -n 5 /dev/sdb
```

Adding a `-z` switch will overwrite the random data shred has used with zeroes, leaving a new-looking disk. `-u` will delete the file after the secure overwrite. Shred can also safely overwrite and/or remove individual files.

Where am I?

If you follow our tip on SSH keys everywhere, and end up hopping from machine to machine, remember to customise your Bash prompt (see part 5, **rpimag.co/159**) so that you are always sure on which machine you're about to erase a file.



Add the power of HDR photography, Full HD video, and AI image recognition to your Raspberry Pi projects with Camera Modules.

- ***Getting started***
- *Capturing photos and videos*
- ***Control the camera with precision***
- *Add artifical intelligence with the AI Kit*
- ***Time-lapse photography***
- *Selfies and stop-motion video*
- ***Build a bird box camera***
- *Live-stream video and stills*
- ***...and much more!***

BUY ONLINE: **rpimag.co/cameraguide**

Part 08

Make a laser tripwire

Learn how to use a light-dependent resistor to detect a laser pointer beam

Maker
Phil King
A long-time Raspberry Pi user and tinkerer, Phil is a freelance writer and editor with a focus on technology.

philkingeditor.com

YOU'LL NEED

- 1× solderless breadboard
- 1× light-dependent resistor (LDR)
- 1× 1µF capacitor
- 1× laser pointer
- 5× pin-to-socket jumper wires
- 5× socket-to-socket jumper wires (optional)
- 1× drinking straw or short length of heat-shrink tubing
- 1× plastic box

Your Raspberry Pi computer can easily detect a digital input via its GPIO pins: any input that's approximately below 1.8V is considered off, and anything above 1.8V is considered on. An analogue input can have a range of voltages from 0V up to 3.3V, but the Raspberry Pi is unable to detect exactly what that voltage is without the help of an additional component (an ADC).

One way of getting around this is by using a capacitor and timing how long it takes to charge up above 1.8V. By placing a capacitor in series with a *light-dependent resistor* (LDR, also known as a *photocell*), the capacitor will charge at different speeds depending on whether it's light or dark. We can use this to create a laser tripwire!

Connect the LDR

An LDR is a special type of electrical resistor whose resistance is very high when it's dark but reduced when light is shining on it. With the Raspberry Pi turned off and disconnected from power, place your LDR into the breadboard, then add the capacitor. It's essential to get the correct polarity for the latter component: its longer (positive) leg should be in the same breadboard column as one leg of the LDR. Now connect this column (with a leg of both components) to GPIO 4. Connect the other leg of the LDR to a 3V3 pin, and the other leg of the capacitor to a GND pin. Your circuit should now resemble **Figure 1**.

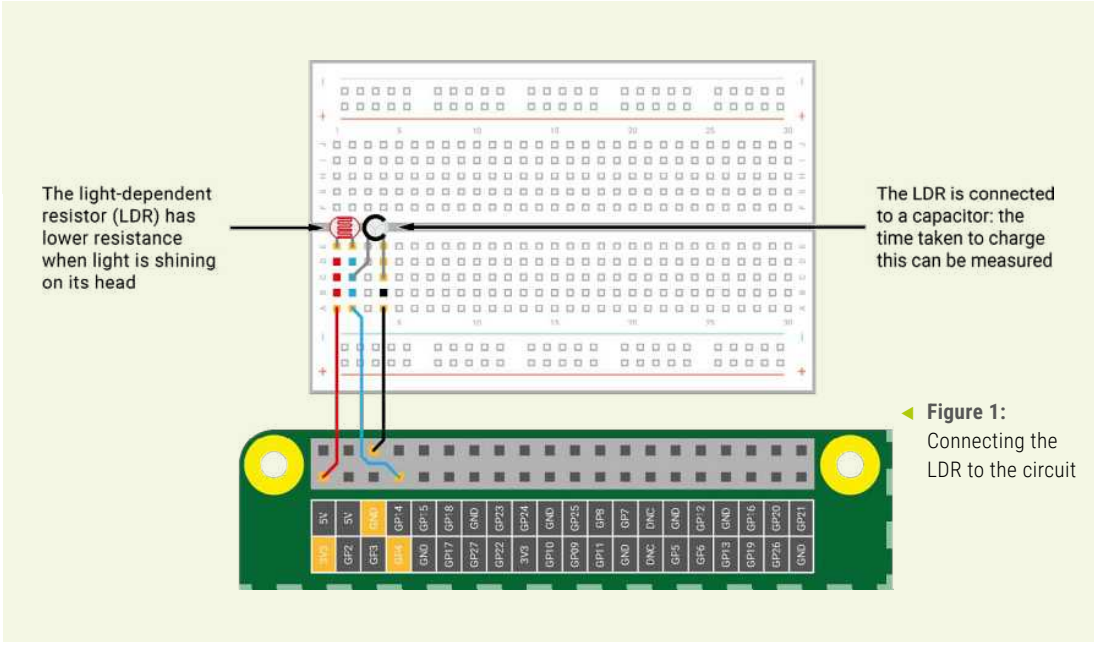


Figure 1: Connecting the LDR to the circuit

Warning!
Laser pointer
Be careful when using laser pointers in your projects and don't aim a laser pointer at a person's head.
rpimag.co/lasersafety

Whenever the beam is broken, the buzzer sounds the alarm

Test the LDR

GPIO Zero offers a helpful `LightSensor` class that is designed for this exact circuit configuration. However, as of this writing, `LightSensor` did not work on the most recent version of Raspberry Pi OS, which means we'll need to write some code to perform the detection logic ourselves. On your Raspberry Pi, create a new file, enter the code in the **test_ldr.py** listing (overleaf), and save it under the same name.

At the start of the code, we import the `OutputDevice` and `DigitalInputDevice` classes from GPIO Zero. We then define a function called `LDR_Value` that takes two arguments: a pin number and a `charge_time_limit` (in seconds, with a one-millisecond default). Next, we create an `OutputDevice` on that pin, then take it LOW for 100 milliseconds. This discharges the capacitor.

Next, we create a digital input device as a floating input, which means the internal pull-up resistors are inactive. This is the workaround for the problem that exists as of this writing. You can read more about the problem and workaround on this GitHub issue: **rpimag.co/LightSensorIssue**.

test_buzzer.py

DOWNLOAD THE FULL CODE:
rpimag.co/gpiobookgit

> Language: **Python**

```
001. from gpiozero import Buzzer
002. from signal import pause
003.
004. buzzer = Buzzer(17)
005. buzzer.beep()
006. pause()
```

Next, we wait for the pin to be active (HIGH), which means the capacitor has been charged up. If `wait_for_active()` returns False, it means that it timed out before the pin went high. You can increase `charge_time_limit` to make the code more sensitive to small amounts of light and decrease it to make it less sensitive.

Enclose the LDR

Unless you're working in a darkened room, you'll probably notice little difference between the measured light level when the laser pointer is directed onto the LDR and when it's not. This can be fixed by reducing the amount of light that the LDR receives from other light sources in the room, which will be essential for our laser tripwire device to work effectively. You can achieve this by cutting off a short section – between 2cm and 5cm – of an opaque drinking straw or heat-shrink tubing and inserting the head of the LDR into one end. Now try the test code again

and see how the measured light level changes when you shine the laser pointer into the other end of the straw or tubing. You should notice a larger difference in values.

Wire up the buzzer

To create an audible alarm for our laser tripwire, we'll add a piezo buzzer to the circuit. The polarity must be correct: connect the column of the buzzer's longer leg to GPIO 17, and the shorter leg to a GND pin. It should look like **Figure 2**. Let's test it to see if it's working. Create a new file, enter the code from the **test_buzzer.py** listing, and save it.

At the top, we import the `Buzzer` class from GPIO Zero and `pause` from the `signal` library. Next, we set the `buzzer` variable to the buzzer output on GPIO 17. Finally, we use `buzzer.beep()` to make the buzzer turn on and off repeatedly at the default length of 1 second. To stop it, press **CTRL+C** while the buzzer is quiet.

Test the tripwire

We'll now put everything together so that the laser pointer shines at the LDR (through the drinking straw) and whenever the beam is broken, the buzzer sounds the alarm. In Thonny, create a new file, enter the **tripwire.py** code, and save it under the same name.

At the start, we import the `OutputDevice`, `DigitalInputDevice`, and `Buzzer` classes from GPIO Zero. We also import the `sleep` function from `time`; we'll need this to slow the script down a little to give the capacitor time to charge. As before, we assign variables for the buzzer and LDR pin to the respective devices on GPIO 17 and 4. We then use a `while True:` loop to continually check the LDR; if enough light falls on it, we make the buzzer beep. Try running the code; if you break the laser beam, the buzzer should beep for 8 seconds. You can adjust this by altering the `buzzer.beep` parameters and `sleep` time.

Package it up

Once everything is working well, you can enclose your Raspberry Pi and breadboard in a plastic box (such as an old ice cream tub), with the drinking straw poking through a hole in the side. If you prefer, you can remove the breadboard and instead connect the circuit up directly by poking the legs of the components into socket-to-socket jumper wires, with the long capacitor leg and an LDR leg together in one wire end, connected to the relevant pins. Either way, place the tub near a doorway with the laser pointer on the other side, with its beam shining into the straw. Run your code and try walking through the doorway: the alarm should go off!

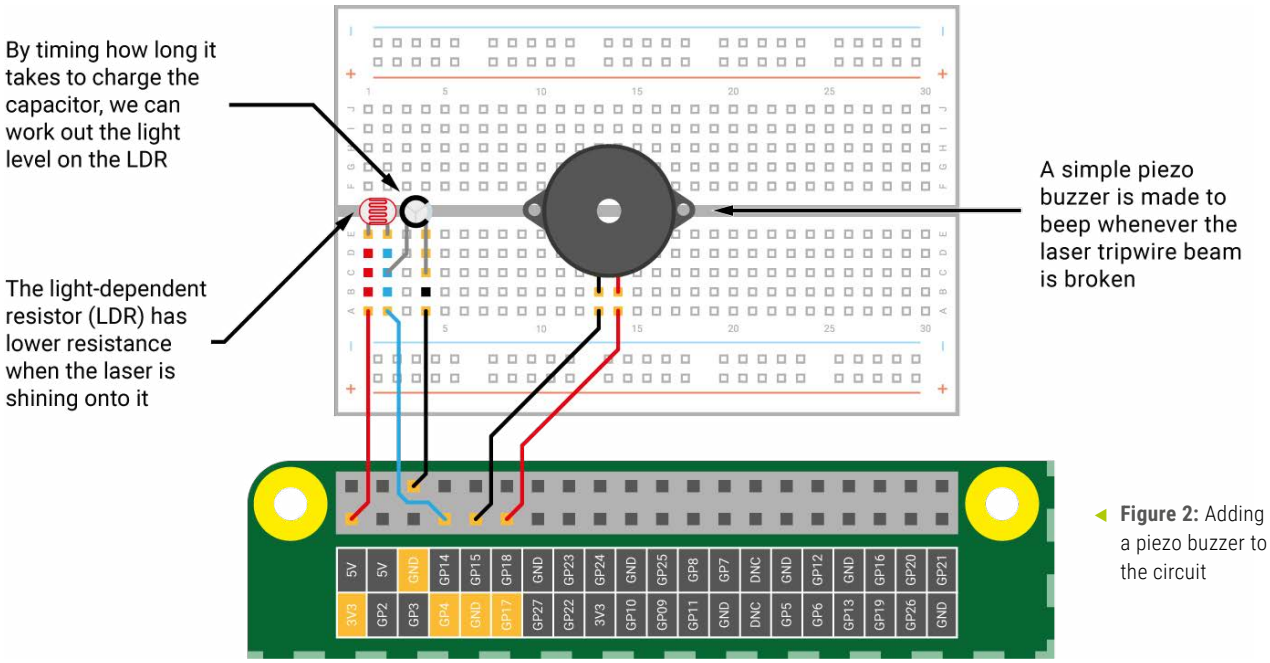


Figure 2: Adding a piezo buzzer to the circuit

test_ldr.py

DOWNLOAD THE FULL CODE:
rpimag.co/gpiobookgit

> Language: **Python**

```
001.  from gpiozero import OutputDevice,
      DigitalInputDevice
002.  from time import sleep
003.
004.  def LDR_value(pin, charge_time_limit=0.001):
005.
006.      # Take the pin LOW to discharge the
      capacitor
007.      ldr = OutputDevice(pin=pin)
008.      ldr.off()
009.      sleep(0.1)
010.      ldr.close()
011.
012.      # Configure the pin as a floating input
013.      ldr = DigitalInputDevice(pin=pin,
      pull_up=None, active_state=True)
014.
015.      # If the pin goes active before the
016.  timeout, we have light
017.      lit = ldr.wait_for_active(
      timeout=charge_time_limit)
018.      ldr.close()
019.
020.      return lit
021.
022.  ldr_pin = 4
023.  while True:
024.      print(LDR_value(ldr_pin))
025.      sleep(1)  # Wait for a second
```

tripwire.py

DOWNLOAD THE FULL CODE:
rpimag.co/gpiobookgit

> Language: **Python**

```
001.  from gpiozero import OutputDevice,
      DigitalInputDevice, Buzzer
002.  from time import sleep
003.
004.  def LDR_value(pin, charge_time_limit=0.001):
005.      ldr = OutputDevice(pin=pin)
006.      ldr.off()
007.      sleep(0.1)
008.      ldr.close()
009.
010.      ldr = DigitalInputDevice(pin=pin,
      pull_up=None, active_state=True)
011.      lit = ldr.wait_for_active(
      timeout=charge_time_limit)
012.      ldr.close()
013.
014.      return lit
015.
016.  buzzer = Buzzer(17)
017.  ldr_pin = 4
018.
019.  while True:
020.      if LDR_value(ldr_pin):
021.          buzzer.beep(0.5, 0.5, n=8,
      background=False)
022.      sleep(0.1)
```

Part 11

Pilot ACE

Turing's universal machine made real

Maker

Tim Danton

When not writing books about classic computers, Tim is editor-in-chief of the British technology magazine PC Pro. He has also helped to launch several technology websites, most recently **TechFinitive.com**, where he is a senior editor.

dantonmedia.com

So much has been written about Alan Turing that he has become a legend as much as a man. Some accounts, wary of facts getting in the way of a good story, give him credit for shortening the war by two years by single-handedly breaking the Nazis' Enigma code. Others, particularly in the UK, paint him as the father of computing. The truth, which sadly must allow for facts, is much more nuanced. Including his tragic early death.

There are things we can say without fear of contradiction or argument. That Alan Turing was a genius and polymath; a man of such insight that he, more than any of his time, foresaw the era of AI in which we currently live; a great and compulsive runner, to the point where he could have represented his country at the Olympics; a gay man living in an intolerant world.

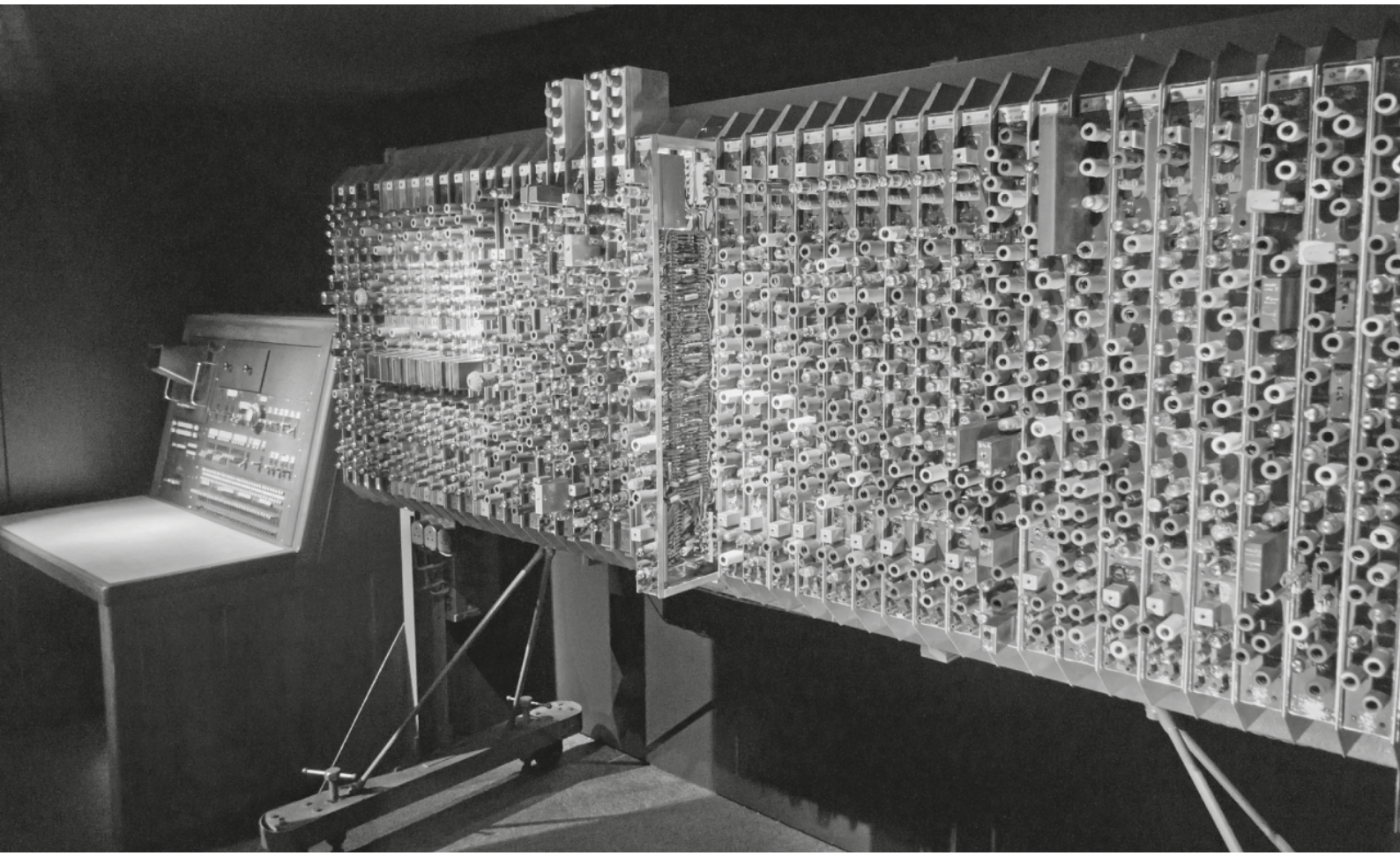
But we might not have known any of this had he skipped a lecture in the spring of 1935.

Turing had a habit of recreating the giant from first principles

"I believe it all started because he attended a lecture of mine on foundations of mathematics and logic," said Max Newman,[1] who would go on to become a key figure in Bletchley Park's code-breaking success and a lifelong mentor to Alan Turing. "I think I said in the course of this lecture that what is meant by saying that [a] process is constructive is that it's a purely mechanical machine – and I may even have said, a machine can do it."

If Newman was correct, his lecture set Turing on course to write a paper that laid the foundations for modern-day computers. Most notable of all is that Turing created the idea of a universal computer to disprove the forbiddingly named

1 Jack Copeland, 'The Church-Turing Thesis', in Stanford Encyclopedia of Philosophy, 8 January 1997 (revised 18 December 2023), **rpimag.co/newmancambridge**. The quote is from an interview circa 1976.



▲ Pilot ACE in the Science Museum, London, UK
Image: Antoine Taveneaux, CC BY-SA 3.0

Entscheidungsproblem. This translates as 'decision problem', a term likely invented by Heinrich Behmann[2] in the early 1920s.

The decision problem

The Entscheidungsproblem summarises a far longer list of aspirations set out by German Professor David Hilbert, whom Behmann assisted, to formalise mathematics.

First came completeness, that every true mathematical statement could be proved within the system; next, consistency, that no contradictions would appear; and last was decidability, that there should be a method to determine if any given mathematical statement is provable or not. Kurt Gödel had brilliantly dealt with the first two aspirations, completeness and consistency, via two theorems in 1931,[3] but this left decidability as a prize for gifted young mathematicians such as Turing to grab.

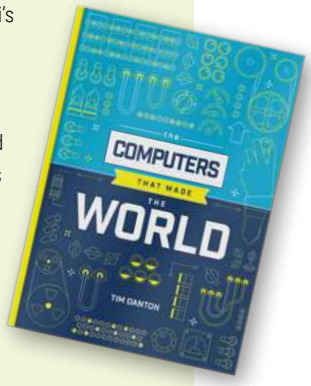
The Entscheidungsproblem came at the right time for the 23-year-old. Alan Turing had flown through the mathematics

2 As before, **rpimag.co/entscheidungs**
3 Panu Raatikainen, 'Gödel's Incompleteness Theorems', 11 November 2013 (revised 2 April 2020), **rpimag.co/godeltheorem**

The Computers that Made the World

This article is an extract from Raspberry Pi's book, *The Computers that Made the World*. This book tells the story of the birth of the technological world we now live in. It chronicles how computers reshaped World War II. And it does it all through the origins of twelve influential computers built between 1939 and 1950. You can pick up a copy on the Raspberry Pi website.

rpimag.co/tctmtw



course at Cambridge with first-class honours[4] and his university college, King's, recognised Turing's brilliance by awarding him a £200 research studentship so that he could stay on and achieve a full fellowship. Today, that £200 translates to over £12,000 ($16,000).

While brilliant, Turing also took a unique approach to solving problems that often frustrated his seniors. Others stand on the shoulders of giants, but Turing had a habit of recreating the giant from first principles, often with his own obscure notation. He was also plagued by bad luck. He thought he was the first to solve a two-century-old challenge known as the Central Limit Theorem, only to be told that a key breakthrough had been published in 1920.[5] While he had used a different approach to that proof, it did not break new mathematical ground.

History was to repeat itself with the Entscheidungsproblem, with American mathematician Alonzo Church publishing a proof in March 1936. In the months since Newman's lecture the previous year, Turing had developed his proof to the point where he had a draft ready in early 1936 and submitted it to the London Mathematical Society in May 1936. It was finally published in late 1936.[6]

On Computable Numbers

Turing's paper, 'On Computable Numbers, with an Application to the Entscheidungsproblem', established him as an important and original mathematician. But more importantly, it fired Turing's interest in what we would now call computing and the science behind it.

In the paper, he describes the key concept of a Turing machine.[7] Imagine, Turing suggested, a device that could scan

▶ Alan Turing in 1951
Image: Elliott & Fry, Public Domain

any square on a long tape. Each scanned square contained a symbol or was empty. The machine would then scan the square and take an action depending on the rules it had been given: do nothing or erase the symbol (if present) or overwrite it with a new symbol. The scanning apparatus would then move to the left or right.

Turing describes instruction tables, which you can think of as programs that tell it what to do in any possible situation. He then showed that any computable mathematical function could be computed by means of an instruction table, much like a program or app will today.

The final ingredient of the Turing machine is the idea of states. So, if the Turing machine is in state A, it will follow a particular instruction (which can tell it to switch state). In state B, it will follow a different instruction. So in state A, the instruction might say replace a star symbol with a question mark, but in state B it's told to replace a star symbol with an exclamation mark.

A universal machine

"He introduced Turing machines and described them in detail," says computer historian and Distinguished Professor Jack

4 Technically, he was on the list of 'B-star wranglers', the equivalent of a first-class Maths degree from Cambridge at the time. The 'B' is misleading, referring not to a grade but a schedule of subjects.

5 The proof was by Finnish mathematician Jarl Waldemar Lindeberg.

6 Some sources say 1937, but it was published in two parts in Proceedings of the London Mathematical Society in two parts across November and December 1936. See footnote 1 on p5 of *The Essential Turing*, edited by Jack Copeland (Oxford University Press, 2004, ISBN 978-0198250807)

7 Say you want to create a binary adding device. And that you want to add five and six. In binary, five is represented by 101, six by 110. Let's now separate those two numbers with the symbol $ (it could be any symbol). We now have seven squares lined up from left to right: 101$110. For this particular Turing machine, we need four different states: let's call them s0 (start), s1 (carry), s2 (write), and s3 (halt). Each of those states comes with instructions. If the machine is in a carry state, then it must do certain things: if it reads a 1, then it must write 0, move left, and stay in the carry state. If it reads a 0, it must write a 1, move left, and transition to the start state. If it hits the $ symbol and is in the carry state, then it must write a 1; if not, it must simply halt. Does it work? If we use our example of 110$101, the machine begins in the start state, s0, at which point its instructions are to read the rightmost symbol of both numbers. Scanning 1 and 0 means it must write 1, return to the start state and move to the next symbol on the left. This time it scans 0 and 1, but the result is the same: write 1, move to the left. It now reads 1 and 1, so knows it must write 0 but stay in the carry state. It reads $, which tells it to halt after writing 1. And the tape now reads 1011, or 11 (eleven). Otherwise known as 6 plus 5.

Copeland,[8] referring to Turing's paper. "Then he said, now I'm going to prove that there's a universal machine. The way he does that, and he devotes a whole densely packed section to it, is to create the instruction table for a universal machine. So, if you set up the scanner in accordance with that instruction table, from then on, you never need to change the setup of the scanner again. You just write instructions on the tape and the machine will follow them."

It's a concept that presages stored program computers almost 15 years before they came into existence. Specifically, Turing's universal machine acts as the theoretical blueprint for the ACE that he would design a decade later: a generic machine that can be programmed using coded instructions stored in memory.[9] With the ACE, the tape's place is taken by mercury-delay lines.

Through Newman's influence, by the time Turing's paper was published he had started two years' placement at Princeton University, where he would rub shoulders with the brilliant John von Neumann. The latter would one day be called the father of computing for his work on the EDVAC, but even by this point in history he was considered a groundbreaking theorist thanks to a string of influential papers covering everything from game theory to quantum mechanics.

During his two years at Princeton, Turing completed his PhD thesis with support from Alonzo Church and worked with von Neumann on group theory. We also know that von Neumann was aware of Turing's Computable Numbers paper because he cited a key concept from it in a lecture series[10] (probably delivered in 1945).

> *Turing moved from theory to practicality by designing an electric multiplier*

Some have supplied more detail. "Von Neumann introduced me to ['On Computable Numbers'] and at his urging I studied it with care," wrote von Neumann's Manhattan Project colleague Stan Frankel in 1972.[11] "Many people have acclaimed von Neumann as the 'father of the computer' (in a modern sense of the term) but I am sure that he would never have made that mistake himself. He might well be called the midwife, perhaps, but he firmly emphasised to me, and to others I am sure, that the fundamental conception is owing to Turing – insofar as not anticipated by Babbage, Lovelace, and others."

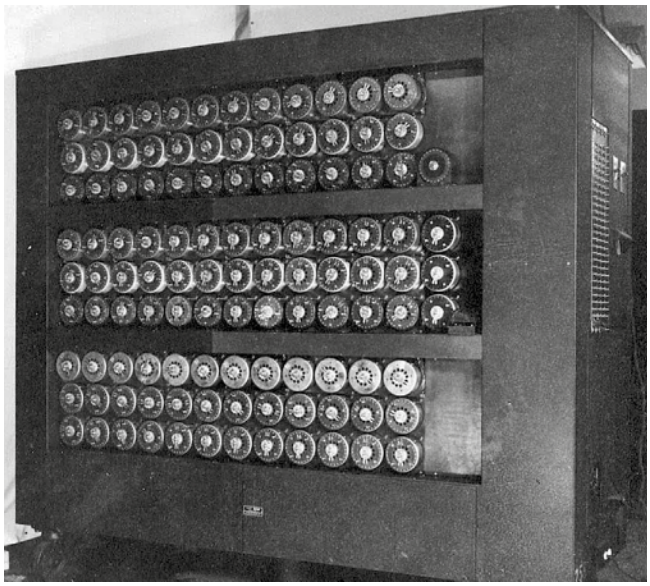
From theory to metal

Whilst at Princeton, Turing moved from theory to practicality by designing an electric multiplier. A physics graduate called Malcolm MacPhail then lent Turing a key to one of Princeton's labs and taught him "how to use the lathe, drill, press etc. without chopping off his fingers," wrote MacPhail to Turing's biographer, Andrew Hodges.[12] "And so, he machined and wound the relays; and, to our surprise and delight, the calculator worked."

According to MacPhail, the two "had many discussions" on the subject of cryptography in the year leading up to his departure. Further evidence of Turing's interest in the subject

8 Interview with author, as are all direct quotes in this article from Jack Copeland unless stated.

9 Turing's 36-page paper is available to download from **rpimag.co/turingpaper1936** and other sources if you wish to follow the intricate mathematics involved.



▲ A Bletchley Park bombe, circa 1945
Image: Public Domain

10 Thomas Haigh and Mark Priestley, 'Von Neumann Thought Turing's Universal Machine was "Simple and Neat" ', Communications of the ACM, 1 January 2020, **rpimag.co/simpleandneat**

11 Brian Randell, 'On Alan Turing and the Origins of digital computers', part of a Technical Report Series (number 33) edited by Dr B Shaw, University of Newcastle, May 1972, p15. Randell wrote to Dr Frankel to find out more about von Neumann's awareness of Turing's paper, and the quote is directly lifted from Randell's transcript of his reply. A copy of Randell's report is held at the British Library, London.

12 Andrew Hodges, *Alan Turing: The Enigma* (Vintage Books, 2014, paperback edition, ISBN 978-1784700089), p175

comes from letters he sent at that time. "Even at Princeton, [Turing] wrote to his mother and said he had devised a code that he thought was pretty well impossible to break," says Copeland. "And he kind of joked about selling it to the British government."

Turing rapidly discovered that the government had their own plans for mathematicians such as him. After gaining his PhD in June 1938 and returning to Cambridge, he attended a cryptology course run by the UK's Government Code and Cypher School (GC&CS). Copeland's research shows that "Turing started regularly visiting the London offices of GC&CS at the beginning of 1939, and he started talking to Dilly Knox[13] about Enigma."

A machine of gears

Throughout this time, Turing remained a fellow at the University of Cambridge. He became interested in the Riemann zeta function, another legendary unsolved problem (this time dating back to 1859). Turing devised a new and theoretical approach to calculating the function, but to complete his work he needed to perform calculations well beyond the desktop calculators of the day. Or the university's model differential analyser, at that time under the control of Maurice Wilkes, creator of EDSAC. So, naturally, Turing set to work designing his own machine.

He estimated this would cost £50, almost £3000 in today's money and well beyond his means, so he applied for and was given a grant by the Royal Society. Together with the enthusiastic help of Donald MacPhail, brother of Malcolm, who happened to be a research student at Cambridge studying mechanical engineering, he set to work on the schematics. By the summer of 1939, a visitor to Turing's rooms would have been met by a collection of gear wheels. A completed machine would need 80 of them.

▲ Pilot ACE undergoing testing, circa 1949
Image: courtesy of the Computer History Museum, CC BY-NC-SA

Turing set to work on an entirely different machine

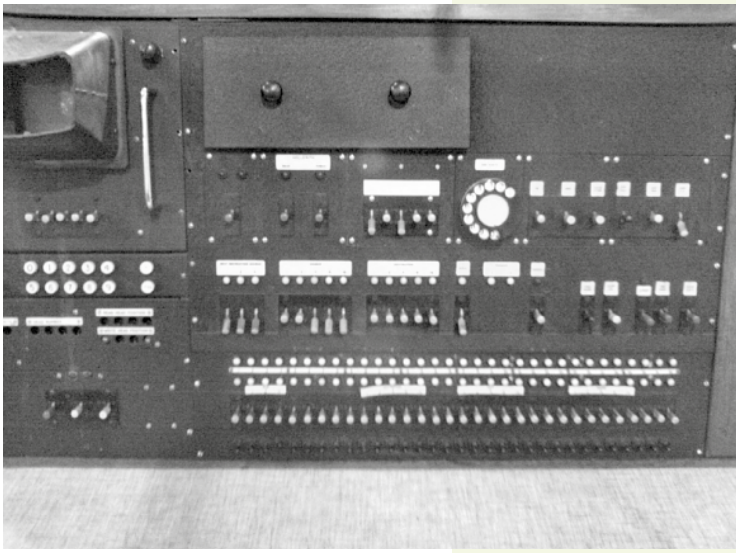
But war intervened and the project was destined to never be completed. Instead, Turing and his brilliant colleagues at Bletchley Park set to work on an entirely different machine. One that could help decipher the Enigma-encrypted radio transmissions of the German military.

The British code breakers had been given an invaluable head start by the work of Polish cryptanalysts, who had created a machine they called 'bomba'. While it's natural to assume that it was named after bombs – 'bomba' is Polish for bombs (although it also translates as 'impressive') and the mechanisms made a muffled, rhythmic sound – Copeland provides a more interesting explanation. "Marian Rejewski, who with Anthony Palluth was the chief architect of the bomba, has been quoted as saying that when they needed a code name for these things, he happened to be eating a bomba at the time. He was sitting in a café eating an ice cream, and he said let's call the machine a bomba."

The Polish head start

The Polish team had created six bomby (pronounced 'bom-bee') by the end of 1938. There were six to match the number of possible wheel orders of the Enigma, and each bomba cycled through

13 Alfred Dillwyn Knox famously helped to decrypt the Zimmermann Telegram, pivotal in making the USA join World War I, and was a central figure at Bletchley Park until his death, aged 58, from cancer in February 1943.



▲ Pilot ACE console
Image: Karl Baron, CC BY 2.0

17,576 permutations to discover the right setting. However, the machines' usefulness was reduced by a factor of ten when the Germans increased the number of wheel orders to 60.

"It wasn't just the extra wheels that made life tougher," says Copeland. "What really brought them down was the 'Stecker', the plugboard, because the Germans increased the number of steckered letters." Essentially, the Stecker board scrambled the letters, making it even harder to decrypt messages. While the first version of the Stecker board only worked on eight letters, this was increased to a dozen, which was enough to defeat the bomby.

All of this meant that the Poles could no longer work on their own. At the same time, the chances of Nazi occupation grew – Germany had invaded Czechoslovakia on 14 March 1939 and Hitler's speeches increasingly targeted Poland – which forced the Polish code breakers to reveal their hand. They had to share all they knew with their allies, namely France and Britain. So, in late July 1939, Dilly Knox along with two other British representatives found himself face to face with the Poles at their Warsaw headquarters. It was here he learned about the existence of the six bomby.

Knox left Warsaw deeply impressed by the Poles' work – even sending them ribbons for "winning" the race to decrypt the German Enigma – but knew that the British had a huge task on their hands. They needed to create something an order of magnitude more powerful than the bomba. Mavis Batey, one of the code breakers who worked with Dilly Knox and author of *Dilly: The Man Who Broke Enigmas*, quotes his post-meeting report:[14] "At the time of my visit I had ideas which seemed to be better, and I have since discussed them exhaustively with Mr Turing and Commander Travis," wrote Knox. He believed, the report states, that they could produce a machine that they wanted to call the "Geschlechtzylinder". A name, luckily, that never caught on.

Turing takes charge

While Knox was a gifted code breaker, he recognised that he needed help from a mathematician and that Alan Turing was the right man for the job. His ideas on logic and computability had been honed over the past five years with the likes of Newman; he already had an affinity for cryptoanalysis; and he had hands-on experience with relays and machine design.

In her excellent chapter 'Breaking machines with a pencil', found in The Turing Guide,[15] Batey describes how Knox gave Turing full responsibility for developing the machine as summer turned to autumn. "[Turing] worked in the stable-yard cottage that Dilly had chosen to be away from the administrators," she wrote. To be precise, the cottage's loft. "Since the only access was a ladder in the wall, two of the girls rigged up a pulley and a basket for sending up coffee and sandwiches," Batey added.

Thus fuelled, an inspired Turing set to work. By 1 November 1939,[16] he was able to set down the requirements in detail. The job of building the British bombe then passed to Harold 'Doc' Keen of the British Tabulating Machine Company, and within six months he delivered: the first British bombe, 'Victory', was installed and ready for action by the end of April.

It was a huge beast, standing six feet and six inches (2m) tall, over seven feet (2.1m) long and three feet (0.9m) wide. But it wasn't merely the size that made it impressive, as this description from Turing's Bletchley Park colleague Patrick Mahon makes clear: "From one side, a bombe appears to consist of nine rows of revolving drums; from the other, of coils of coloured wire, reminiscent of a Fair Isle sweater."[17]

Inside the bombe

The bombe was an electromechanical machine designed to simulate the Enigma's wheels and circuits. It can be most

14 Mavis Batey, *Dilly: The Man Who Broke Enigmas*, (Biteback Publishing, Kindle Edition), p119
15 Various authors, *The Turing Guide* (Oxford University Press, 2017, ISBN 978-0198747826), pp97-107
16 As before, p105
17 Patrick Mahon, 'History of Hut 8 to December 1941', in Jack Copeland (ed.), *The Essential Turing*

easily thought of as multiple simulated Enigma machines, all hooked together.[18]

The system relied on 'cribs', expected words or phrases such as 'Wetter fuer die Nacht' ('weather for the night' in English). We won't go into detail here, but in essence the bombe would search the space of possible settings of the Enigma and, when a group of settings produced an output that matched an expected pattern, it would stop. The operators would then note down the settings, which were then transferred to a replica Enigma machine. If this produced German, albeit peppered with wrong letters, the bombe had cracked the code. Time to move on to the next message.

Unfortunately, Turing's bombe only worked if the crib met certain criteria; in particular, it needed to include three 'closed chains' or loops.[19] It took the insight of another talented mathematician, Gordon Welchman, to turn what would have been an effective machine into a juggernaut. Welchman's insight was to spot that due to the Stecker board's reciprocal nature – that, if A turned into B, B would turn into A – they could simultaneously scan all the possible Stecker values. Welchman designed a 'diagonal' board to take advantage of this, and by August the first of the new, improved bombes came into service.[20]

From this point on, the British Tabulating Machine Company ramped up production, eventually building a dedicated bombe factory. "We were taken there to watch them being made and to encourage the workers, although we thought their conditions were better than ours," wrote bombe operator Diana Payne.[21] "It was a surprise to see the large number of machines in production."

A growing operation

At first the bombes were housed at Bletchley Park, but soon they became too numerous and new machines were shipped to other secret locations (including Woburn Sands, Eastcote, and Stanmore). In an early example of remote computing, Bletchley Park operators could phone in commands, and eventually they could even call up US-made bombes in Washington, DC.

The bombes needed hands-on control too. Much of this work was done by members of the Auxiliary Territorial Service, or ATS, the women's branch of the British Army founded in 1938. Or by the Women's Royal Naval Service, shortened to the Wrens. "We were given a menu, which was a complicated drawing of numbers and letters from which we plugged up the back of the machine and set the drums on the front," wrote Payne.[22]

"We only knew the subject of the key and never the contents of the messages," she added. "All this work had to be done at top speed, and at the same time 100 per cent accuracy was essential. The bombes made a considerable noise as the drums revolved, each row at a different speed, so there was not much talking during the eight-hour spell. For technical reasons which I never understood, the bombe would suddenly stop, and we took a reading from the drums."

But Turing's wartime work was far from done. He was in charge of Hut 8, which focused on Naval Enigma, and these messages were the toughest to break due to an extra layer of encryption. "They doubly enciphered the indicators," explains Copeland. "They would encipher them once by hand, using a book of settings, and then they enciphered them again using the Enigma machine."

The U-boat menace

There was one particular Enigma network of concern: Dolphin. This was the code name given to U-boats' communications, and with the fall of France in June 1940 came the menace of growing U-boat attacks in the north Atlantic. And they were deadly, with one U-boat alone – U-48 – responsible for sinking 51 ships.[23] Almost 600 merchant ships had been sunk by the end of 1940.[24]

Even with bombes to call upon, Hut 8 required help if it was to break Dolphin. The code breakers needed information direct from the German ships and U-boats using Dolphin, and that meant audacious raids – or 'pinches' as they were called. Those with a taste for adventure should read *Enigma: The Battle for the Code* by Hugh Sebag-Montefiore,[25] which details many of the most daring achievements.

There were several partially successful pinches, where the Allied forces would grab whatever information they could: records of previous messages that could act as cribs, daily codes, Stecker settings.

18 This is a much-simplified version of how the bombe worked. For more detail, read *The Essential Turing*, pp246-254.

19 'Closed chains' is Turing's word, but Jack Copeland uses the friendlier term of 'loops' in *The Essential Turing*, where he explains the process in detail (see pp120-123). Not all cribs produced the required three loops, which is why Turing's first bombe design was less effective.

20 By necessity, we have missed out much detail here. Interested readers have numerous books to choose from to discover more about the British bombes, including Andrew Hodges' biography, *Alan Turing: The Enigma, Codebreakers: The Inside Story of Bletchley Park* edited by Alan Stripp and Sir Francis Hinsley, and *The Essential Turing* edited by Jack Copeland.

21 Diana Payne, 'The bombes', Chapter 17 of *Codebreakers: The Inside Story of Bletchley Park*, edited by Alan Stripp and Sir Francis Hinsley (Thistle Publishing, Kindle Edition), p169

22 As before, pp169-170

23 According to uboat.net, **rpimag.co/uboat48**

24 Jack Copeland (ed.), *The Essential Turing*, p257

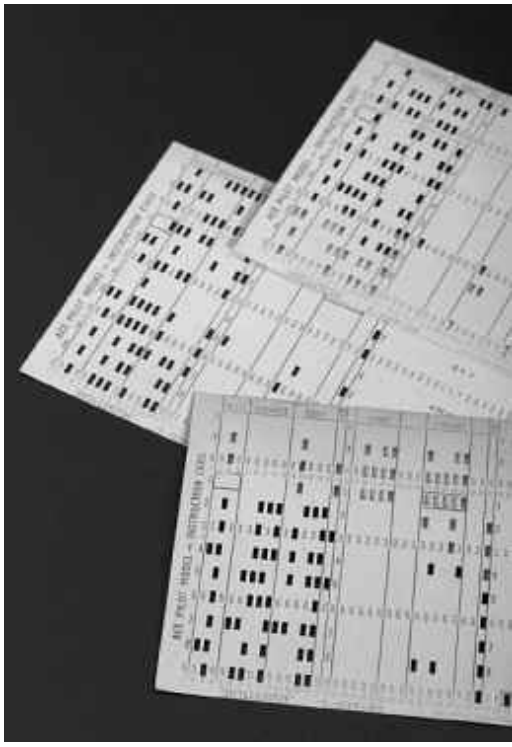
25 Hugh Sebag-Montefiore, *Enigma: The Battle for the Code* (Weidenfeld & Nicolson, 2000, ISBN 978-1474608329)

These messages were the toughest to break due to an extra layer of encryption

But the real prize was the current bigram tables, used when doubly enciphering the indicators. Every Enigma operator was issued a set of nine bigram tables that told them what to do with each pair of letters (each bigram): for example, that 'AC' should become 'QS'. In total, the tables included 676 bigrams. The same tables were used for months at a time, which was another reason they were so valuable.

The problem was that the Germans also knew how important this documentation was, and naval commanders were under strict orders to destroy it all as soon as they came under attack. So, while several pinches happened during 1940, pickings were slim. As summer turned into autumn and the U-boats continued to sink Allied ships, the Admiralty became open to the most outlandish of ideas. Including one from the man who would later create James Bond.

The story of the Pilot ACE will be continued in issue 168.

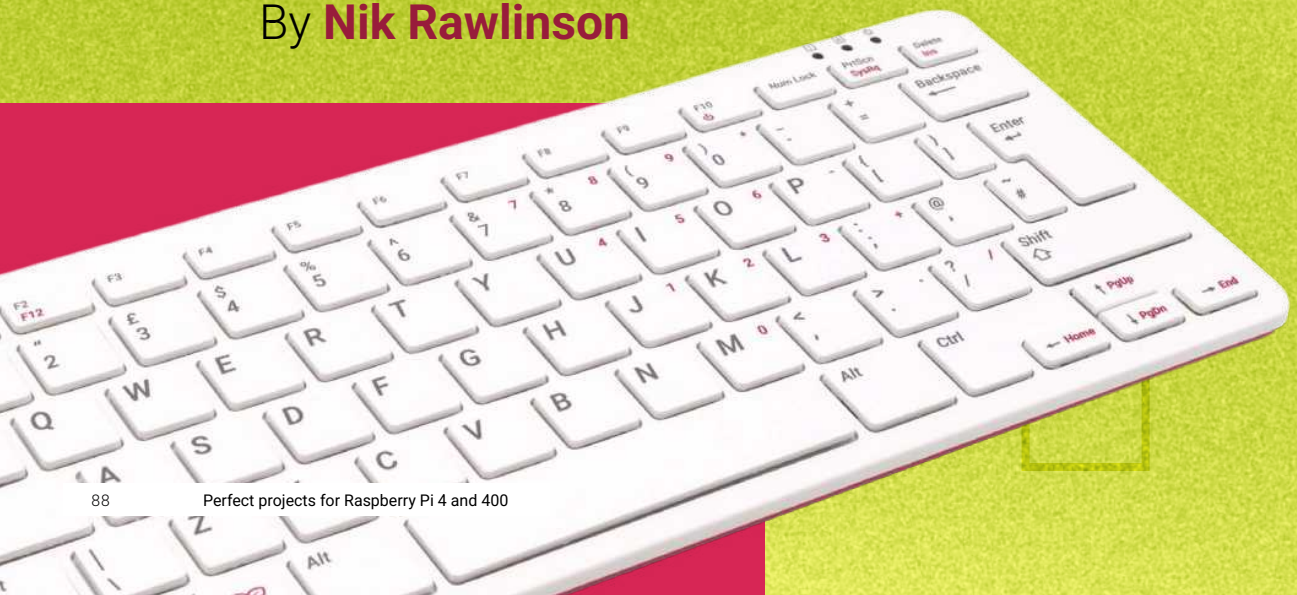


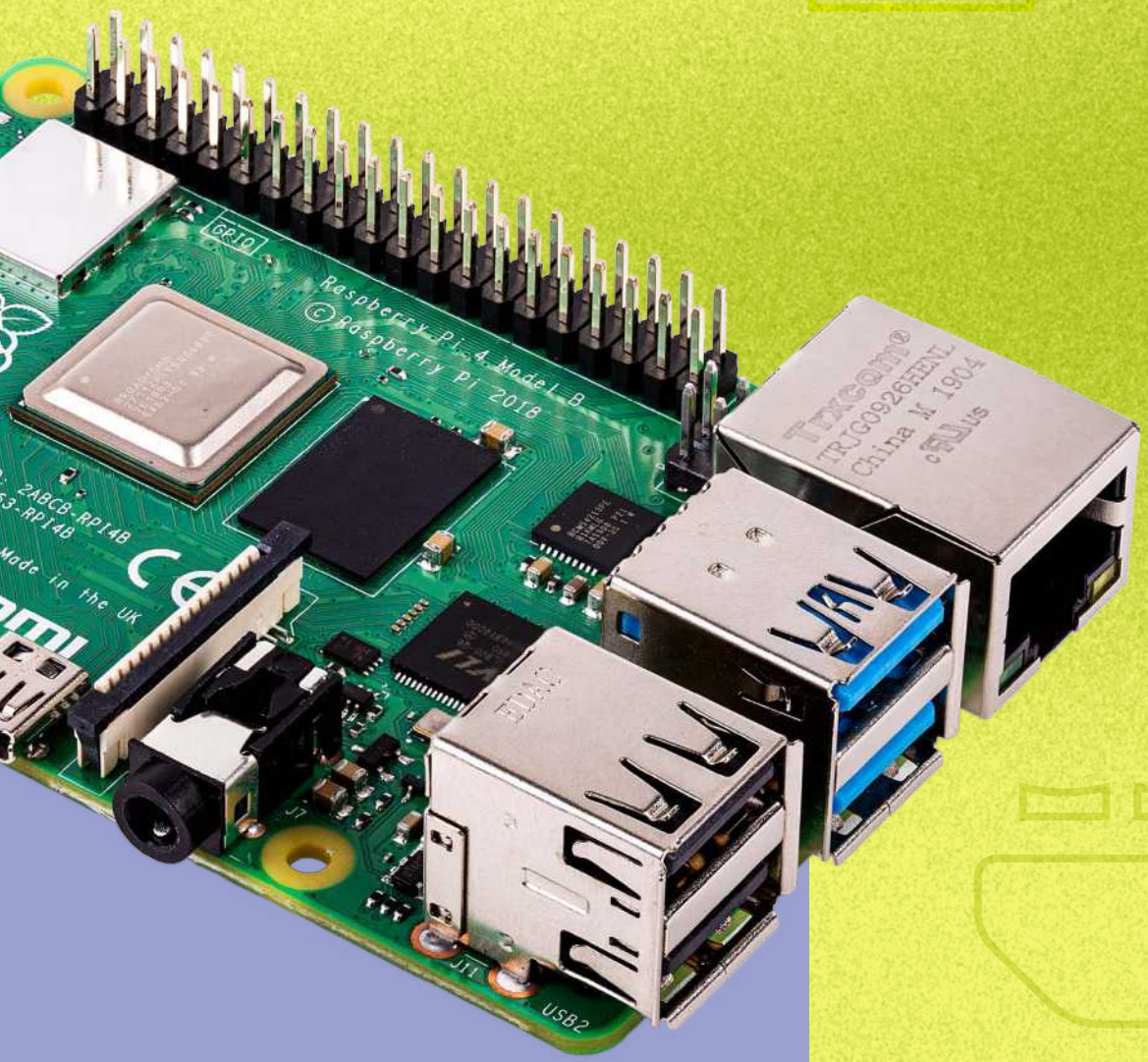
Punch cards for the Pilot ACE **Image:** Science Museum London, CC BY-SA 2.0

PERFECT PROJECTS FOR RASPBERRY PI 4 & 400

Inspirational projects for work, home, and education that show how far you can go without breaking the bank

By **Nik Rawlinson**





None of these projects will cost you a fortune

While rising storage and memory prices have made some computers costlier, Raspberry Pi 400 remains excellent value for money. At just £57/$60, this sleek, silent all-in-one packs a 1.8GHz processor and 4GB of memory, which is plenty to power you through a full day of working, learning, and coding, plus a spot of gaming and browsing to round things off in the evening. Also, as a computer and keyboard in one, it takes up very little space and requires fewer trailing cables.

Across the next eight pages, you'll find plenty of inspiration for getting started with your own Raspberry Pi 400 or Raspberry Pi 4, which shares much of the same hardware. None of these projects will cost you a fortune, and most can happily live side by side on the same computer.

LEARNING

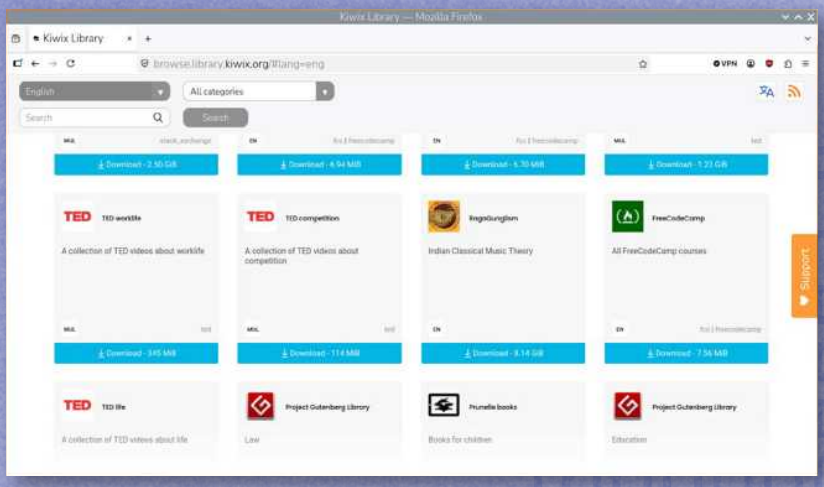
From home schooling to lifelong learning, Raspberry Pi 400 has all the tools you need, and fewer distractions

Host your own reference library

get.kiwix.org

The internet can be a terrible distraction when you're trying to learn… so turn it off. Instead, host the reference materials you need locally. Kiwix was originally designed to make reference material available in areas where there's unreliable internet or no coverage at all, by packaging sites like Wikipedia and Project Gutenberg into portable **.zim** files. Raspberry Pi 400 and Raspberry Pi 4 are the perfect hosts for these compact bundles that you can access across your network from any computer, even without internet access. Install it by typing `sudo apt install kiwix-tools` at the Terminal, and download libraries from **get.kiwix.org**.

▲ You can host local copies of popular reference sites like TED Talks and Wikipedia on Raspberry Pi within the Kiwix library environment



▲ Reference sites are compiled into .zim files, which include all of the text and images that you need to use them offline

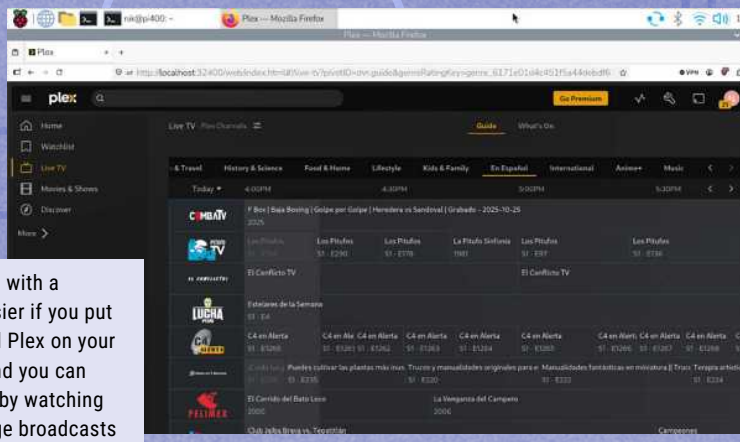
CONNECT A HAT TO RASPBERRY PI 400

Raspberry Pi 400's GPIO pins are hidden behind the black protector on the back of the case, which is easily removed using a fingernail. Many HATs will connect without problem, but if you're connecting a display you might want to angle it to get the best view. In this case, consider an angled adapter like the Adafruit CYBERDECK Bonnet for Raspberry Pi 400 (£7/$8, **rpimag.co/cyberbonnet**) or a female-to-male GPIO ribbon cable (around £5/$6) as pictured below.

Learn a language watching telly

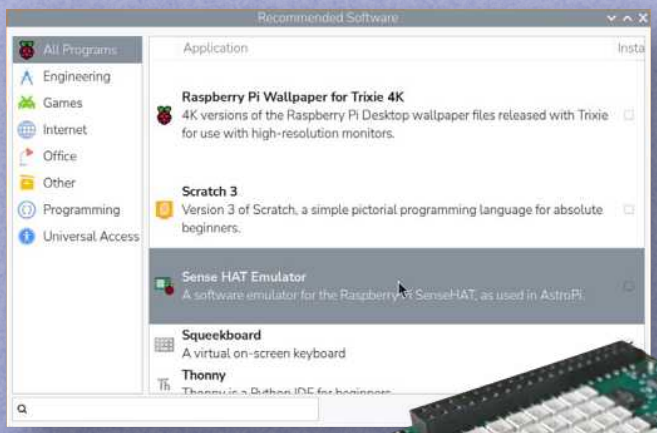
plex.tv

Connect Raspberry Pi 400 or Raspberry Pi 4 to your TV and you can stream foreign TV using broadcaster plug-ins in Plex. To install Plex visit the Plex downloads page (**rpimag.co/plexdownload**). Once it's installed, you'll find the foreign language options by clicking Add-ons in the sidebar, followed by 'Video add-ons'. Scroll through the list of broadcasters until you find the one you want. Raspberry Pi can send audio to your TV via the HDMI cable, and do the same using the speakers built into the official Raspberry Pi Monitor if you want to watch TV at your desk. If you prefer to use headphones, connect them via Bluetooth. Click the Bluetooth icon on the Raspberry Pi OS menu bar, followed by Turn On Bluetooth; then put your headphones into pairing mode and click Add Device… on the same Bluetooth menu.

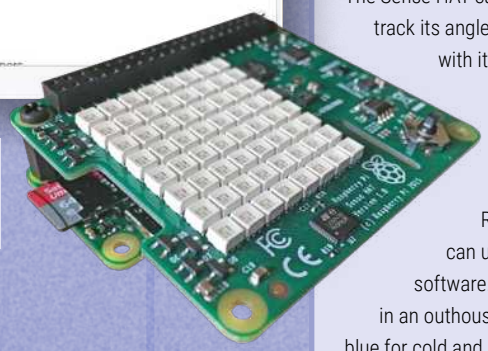


▶ Getting to grips with a language is easier if you put it to use. Install Plex on your Raspberry Pi and you can test your skills by watching foreign-language broadcasts

◀ Spanish is particularly well served, with dedicated sections in the broadcast and films libraries



◀ Don't have a Sense HAT of your own? Install the emulator via Recommended Software to code a software version



▶ The Sense HAT is packed with sensors and has 64 on-board LEDs on which to display its output

Get to grips with the Sense HAT

rpimag.co/sensehat

The Sense HAT can measure temperature, pressure, and humidity; track its angle of tilt and orientation; has a built-in joystick; and, with its 64 LEDs, can feed back all of this and more. No wonder it's travelled into space as part of a citizen science initiative. If you don't have a Sense HAT of your own, install the Sense HAT Emulator (you'll find it via Recommended Software in the Raspberry Pi OS menu's Preferences section) and you can use Thonny to dry-run your Sense HAT experiments in software. We've used the Sense HAT to monitor temperature in an outhouse and display visual warnings by colouring the LEDs blue for cold and red for hot.

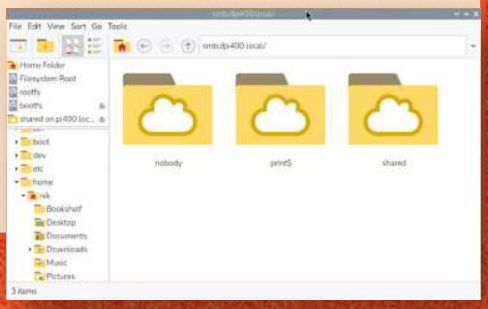
WORKING

Everything the freelancer or solo trader needs for a day's work, plus tools for small teams who need to collaborate

Upgrade your storage (and share files)

samba.org

Use one of the three USB ports on the back of Raspberry Pi 400, or the four on Raspberry Pi 4, to add an external SSD drive. It's not only bigger and faster than a microSD card, but will also take some of the strain. As both computers are designed to run silently and run on low power, you can go one step further and share the additional storage across your network by setting up Samba, as having the host running 24/7 shouldn't disturb anyone. This is the perfect way to create an at-home file-sharing system that works like a remote cloud service, so you can do away with USB thumb drives once and for all.

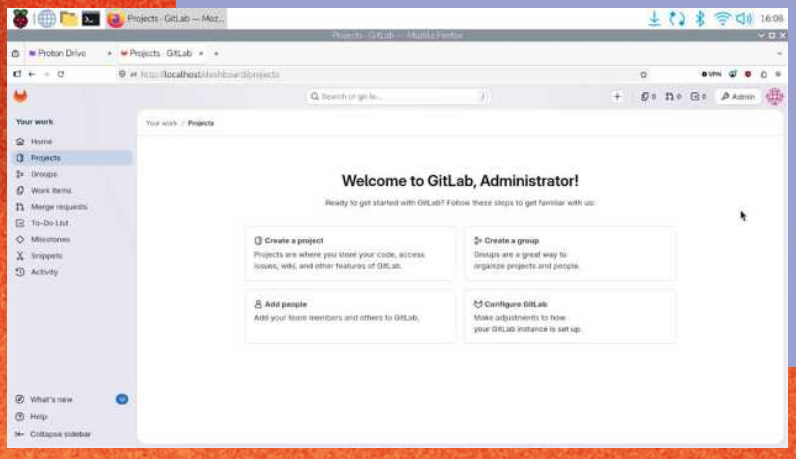


▲ With all this spare capacity, you can afford to share it across your network to set up a home-based 'cloud' storage service

◀ Use a USB 3.0 cable and connect your SSD to one of the two blue ports on your Raspberry Pi 400 or Raspberry Pi 4 for best performance

MANAGE YOUR TEAM'S DOCUMENTATION

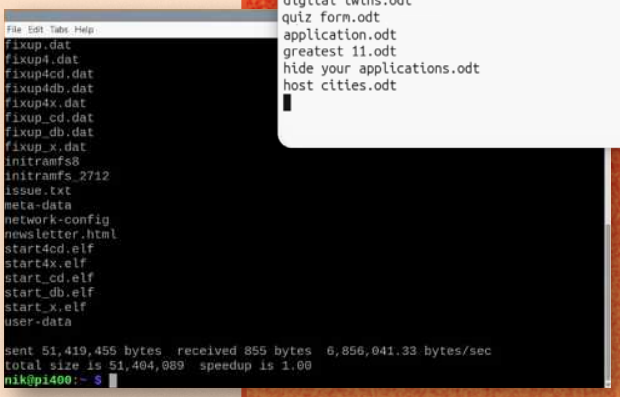
GitLab's most obvious use is as a locally hosted alternative to the code repository site, GitHub. However, in a business context, even if you're not developing applications, you can just as effectively use it as a constantly evolving knowledge base and platform for knowledge sharing. It enables collaboration across your team, change tracking, and version control, so you can revert to earlier versions of a file if required. And, as it supports Markdown you don't need to use a specific editing application to produce attractive and easy to navigate content.

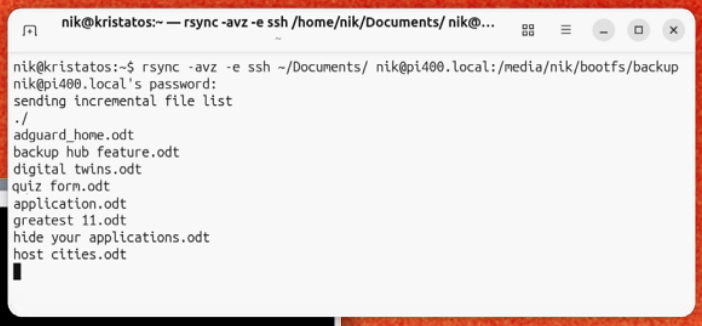


Build a local backup server

rsync.samba.org

If you leave your Raspberry Pi turned on around the clock, every computer on your network can use it for backup. We'd recommend using an SSD rather than microSD card, both to reduce wear and tear on the card and to increase capacity, and installing the free rsync utility with `sudo apt install rsync`. Once it's set up, you can use the command line to synchronise folders from the source machine to the Raspberry Pi's external drive, even if they're somewhere else on your home network. Rsync keeps track of what's been synchronised already to save copying it for a second time, and removes deleted files from the archive to keep the copy up to date.

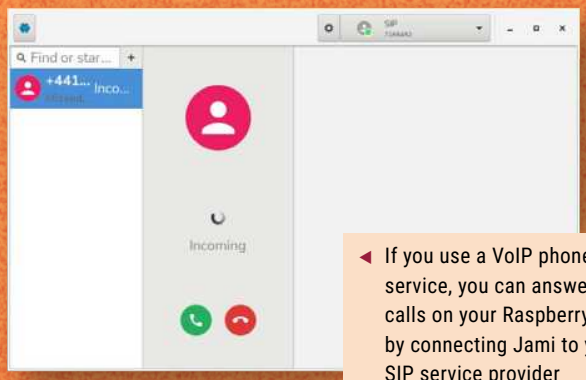




▲ You can just as easily synchronise files across your local network. Here we're syncing from an Ubuntu machine to Raspberry Pi 400

◀ One command is all it takes to make a copy of the Documents folder on an external SSD, connected via USB

▲ Jami is a peer-to-peer messaging and communications platform that you can use to contact colleagues, friends, and family



◀ If you use a VoIP phone service, you can answer calls on your Raspberry Pi by connecting Jami to your SIP service provider

Host your own online meetings

jami.net

Jami is designed to host your own audio and video calls, take part in group chats, send and receive instant messages, share your screen and more, all without forcing you into one of the big cloud-based online meeting services. You don't need to share any personal information to create an account, there are no ads, and if you're only chatting across your own network, with your Raspberry Pi managing every call, there should be no issues with internet latency.

Everything is encrypted and data flows directly between call participants, without sitting on an unknown third-party's server. It's not the best choice for mega organisations, but if you work with half a dozen or so colleagues, it's worth giving a go. Download the installer from the link above, or install through Snap. To set up Snap on your Raspberry Pi, open the Terminal and type `sudo apt install snapd -y`. Reboot when it's finished, then install Jami with `sudo snap install jami`.

CODING

Whether finding your way with Python or publishing on the web, Raspberry Pi 400 is the perfect lightweight coding platform

Get hands-on with Python

camjam.me

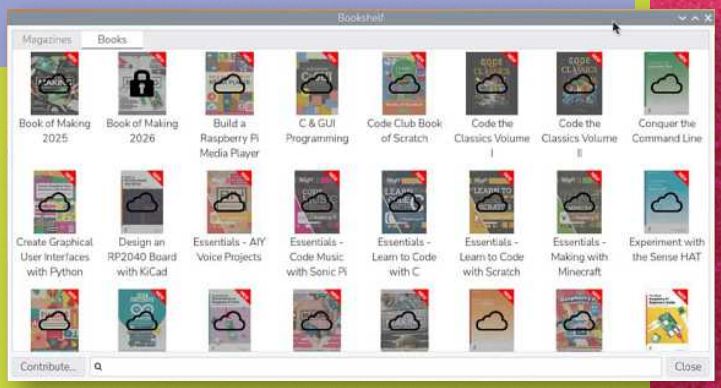
Python is easy to use and widely supported. It's also one of the most popular and fastest-growing programming languages going. Get to grips with it and you can write your own software, analyse huge amounts of data, and even control external hardware. CamJam's EduKits are the perfect starting point. They're cheap, documented via free-to-download worksheets, and come with all you need to get started. There are three kits: EduKit #1 (£6/$8 – electronic circuits), #2 (£9/$12 – sensors), and #3 (£20/$27, robotics).

▲ There are three CamJam kits to choose from. They arrive in a neat storage tin and are documented via downloadable help sheets

▼ Inside the tin, you'll find everything you need to run several projects and get better acquainted with programming

RASPBERRY PI BOOKSHELF

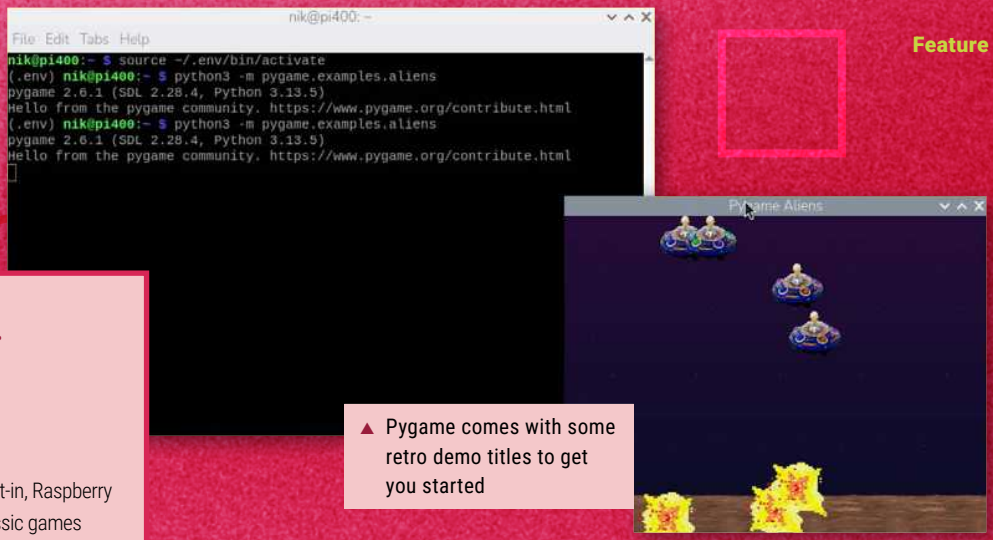
Bookshelf, which is built into Raspberry Pi OS, is packed with back issues of *Raspberry Pi Official Magazine* (and *The MagPi*, going back to 2015) and, if you click through to the Books tab, an awesome collection of coding manuals in PDF format. Whether you're learning to code with C, getting familiar with the Terminal, or starting from scratch with Scratch, they'll take you from beginner to confident in a series of measured, carefully constructed steps. If you don't want to read them at your desk, transfer them to a laptop or tablet computer. If you fancy using Pygame to code retro-style games, check out *Code the Classics I* and *II*, both of which you'll find in the Bookshelf.



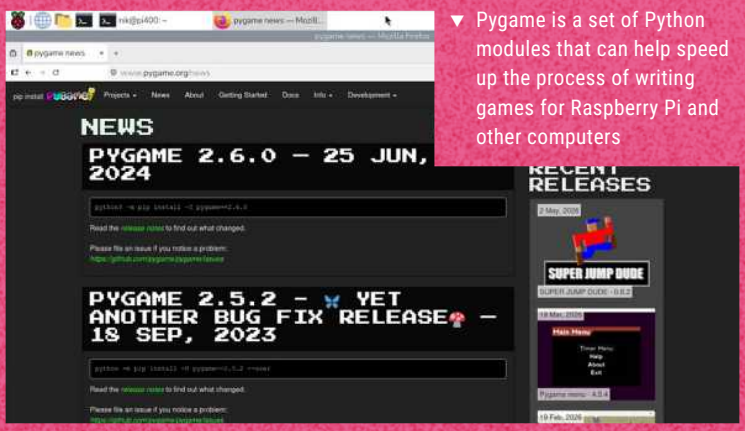
Code your own games… then play them!

pygame.org

Quick to boot and with a keyboard built-in, Raspberry Pi 400 is highly reminiscent of the classic games consoles of the 1970s and 1980s. Recapture the age of the keyboard tinkerer by coding your own games, with help from Pygame, a collection of add-ons for Python specifically geared towards games writing. Better yet, it's widely compatible, so when you've finished creating them on your Raspberry Pi, you can share them with friends and family running Python on Linux, Windows, or macOS. You can also check out our book *Make games with Python* for more: **rpimag.co/makegamesbook**



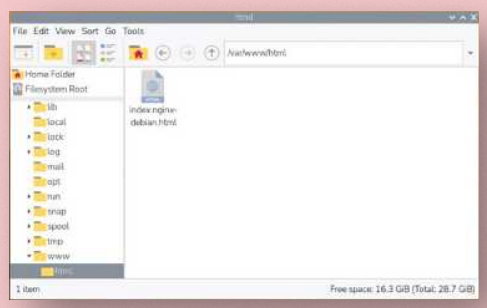
▲ Pygame comes with some retro demo titles to get you started



▼ Pygame is a set of Python modules that can help speed up the process of writing games for Raspberry Pi and other computers

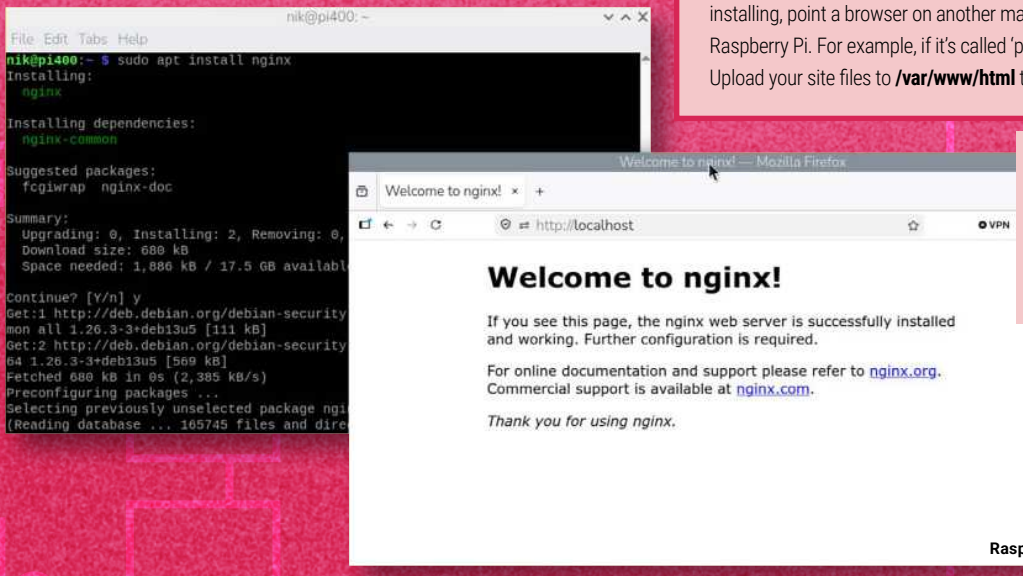
Set up a local web server

nginx.org



▲ No need to fiddle with FTP: save your web pages on Raspberry Pi's local storage and they'll be available across your network

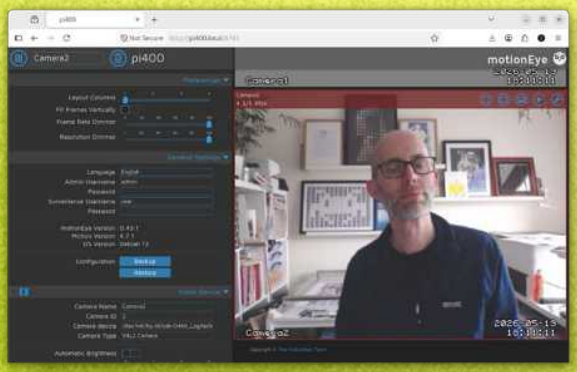
Raspberry Pi 400 and Raspberry Pi 4 are the perfect computers for hosting an always-on web server. They're silent and consume very little energy, and everything you need is built in to Raspberry Pi OS. You don't even need to dedicate the computer entirely to the task: it can run in the background while you continue using it day to day. You can use it for a home noticeboard, to host your own blog, or as an opportunity to learn CSS, HTML, and PHP, without your work-in-progress being exposed to the world. The quickest way to get started is with nginx. Open the Terminal and type `sudo apt install nginx`, then press **RETURN**. When it's finished installing, point a browser on another machine on your network at your Raspberry Pi. For example, if it's called 'pi400', visit **http://pi400.local/**. Upload your site files to **/var/www/html** to replace the holding page.



▼ Setting up a web server on Raspberry Pi takes just a single instruction – and because it's so lightweight, it flies on Raspberry Pi 400 and Raspberry Pi 4

AT HOME

Make your home smarter and safer, and maximise your computer's performance when connected to a TV

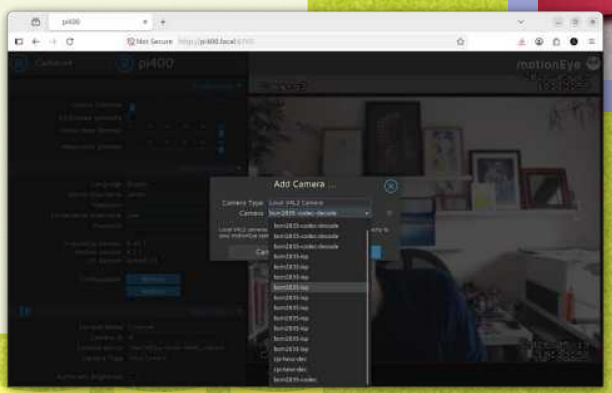


Here, we're using a regular USB web camera connected to Raspberry Pi 400 to keep an eye on the office

Watch for intruders

motioneyeos.org

Connect a USB camera or Raspberry Pi Camera Module and you can use MotionEye to run an open-source, motion-activated surveillance system around your home. It also works with IP cameras and, as all captured footage is stored locally on your Raspberry Pi's microSD card, you don't need to pay fees for cloud storage. The simplest way to get started is with MotionEyeOS, which you can flash onto a microSD card, but if you still want to use your Raspberry Pi 400 or Raspberry Pi 4 as a regular computer at the same time as keeping an eye on your home, you can instead install it as a service by following the instructions on the project's GitHub site: **rpimag.co/motiongit**. This application-based method is still maintained, but the OS was last updated in 2020.

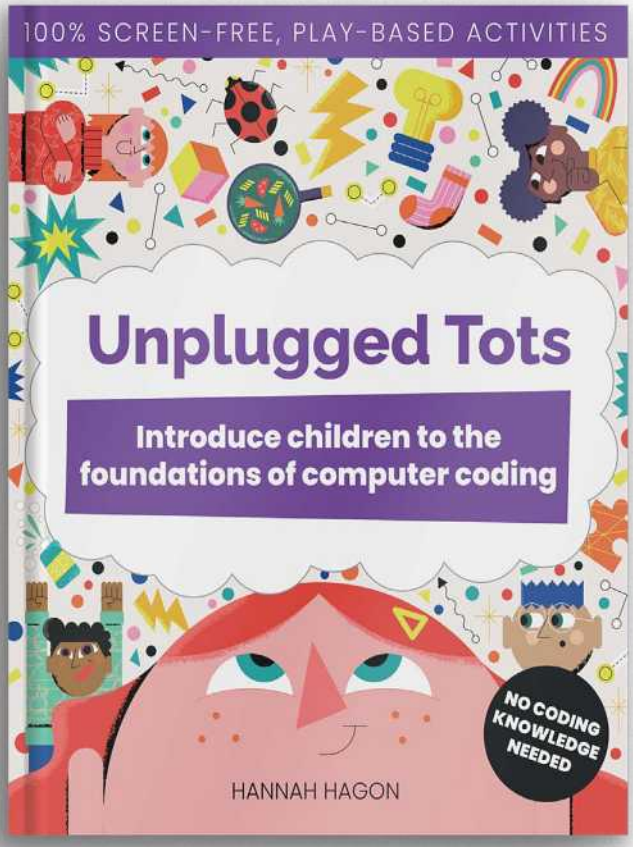


You can set up several cameras side by side and access them all from the same interface. It also works with network cameras

TV-FRIENDLY COMPUTERS

The micro HDMI connectors on Raspberry Pi 400 and Raspberry Pi 4 might be smaller than the sockets on your TV, but they're 100% compatible. To connect either to a television for streaming or playing games, you need a micro HDMI to HDMI cable. Both computers support 4K output at 30fps by default, but you can increase it to 60fps. Connect the HDMI cable to port 0, which is closest to the Raspberry Pi power input, then open a Terminal window and type `sudo nano /boot/firmware/config.txt`. Create a new line at the bottom of the file and add `hdmi_enable_4kp60=1`. Press **CTRL+X** followed by **Y**, then press **RETURN**, and reboot.

Features 100% screen-free activities — no computer, tablet, or mobile phone needed



rpimag.co/unpluggedtots

ONLY THE **BEST**

Radio communication

By **Phil King**

Need to send data from a Raspberry Pi-based project located beyond your home's Wi-Fi network range? Sure, you could use a Wi-Fi extender, or a very long Ethernet cable (up to 100m), but there are other, more reliable solutions. That's what we're exploring here.

LoRa is a long-range radio system that can transmit small packets of data at low bit rates (300bps to 50kbps) – ideal for IoT sensors and the like. The big advantage is its resilience to noise and interference, enabling a range of up to 15km (9 miles) in rural areas. LoRa nodes can send data to each other, or you can use a LoRaWAN gateway to receive multiple channels – either an existing one on, say, The Things Network, or your own.

For (much) higher data rates, there's 4G cellular. We look at a couple of add-on solutions here, along with a separate GPS module for ultra-accurate location and time-stamping of data.

SX1262 868MHz LoRa Node Module for Raspberry Pi Pico

Waveshare / The Pi Hut | £14 / $19 | waveshare.com / thepihut.com

▲ Plug in a Pico and attach the antenna to start transmitting data

This Waveshare LoRa module is based around the SX1262 transceiver, which offers superior power efficiency and slightly longer range than the earlier SX1276. The board features twin female headers, so all you need is a Raspberry Pi Pico equipped with male headers to plug into it. It even extends Pico's pins to the top.

Along with the LoRa module itself, the package includes the bonus of a 600mAh 3.7V LiPo battery connected to the board's charging IC, so you're all set up for remote power.

Note that while The Pi Hut sells the 868MHz band version of this LoRa module for use in Europe (including the UK), with a 'stub' antenna supplied, Waveshare stocks three versions, each tuned to a different frequency, so make sure to choose the right one for your region.

While the Waveshare documentation isn't that easy to follow, there are C/C++ libraries with some code examples to try out – no MicroPython ones, but you could use an existing SX126x library such as this one: **rpimag.co/mpysx126x**.

Verdict

Superb value for money, it includes a LiPo battery and antenna.

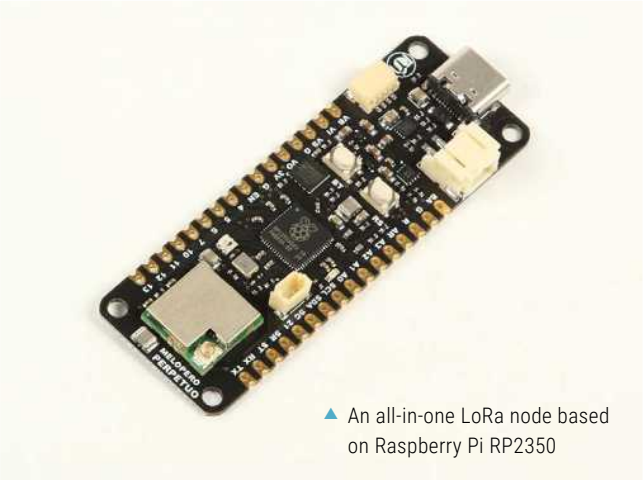
Warning!

Radio regulations

LoRa uses the ISM (Industrial, Scientific, and Medical) bands. Anyone can broadcast here without a licence, but you must adhere to the regional parameters set by local governing bodies (e.g. Ofcom in the UK, FCC in the US, ETSI in Europe). These include limits on the radio frequencies, duty cycle, and transmission power. There are also typically strict limitations for transmitting on other radio bands such as FM. Always check the laws for your jurisdiction before proceeding.

Perpetuo LoRa

Melopero | £21 / $28 | melopero.com



▲ An all-in-one LoRa node based on Raspberry Pi RP2350

Verdict

An energy-efficient LoRa node with handy Qwiic / STEMMA QT port.

With a built-in Raspberry Pi RP2350 microcontroller chip (as used on Pico 2), this all-in-one LoRa (long-range radio) board has everything you need to start communicating with a LoRaWAN network gateway, such as on The Things Network, or your own gateway. Alternatively, you could use it in a simple peer-to-peer setup with another LoRa node.

Equipped with an Embit EMB-LR1276S LoRa radio module, the Perpetuo operates on the UK/European standard 868MHz carrier frequency using the default firmware, but can be tweaked to use the 915MHz band. You'll also need a suitable external RF antenna – it's vital you connect one before powering up this or any other LoRa board, otherwise you may cause damage.

With a low power mode, the Perpetuo is very energy-efficient. Power is provided by a LiPo battery (not supplied) that can be recharged via the USB-C port. The latter can be used with an optional solar panel (typically via a voltage regulator) to keep the board running indefinitely when it's off-grid. A Qwiic / STEMMA QT connector makes it easy to add one or more sensors. In addition, the board has 32 standard pins, including 20 GPIOs.

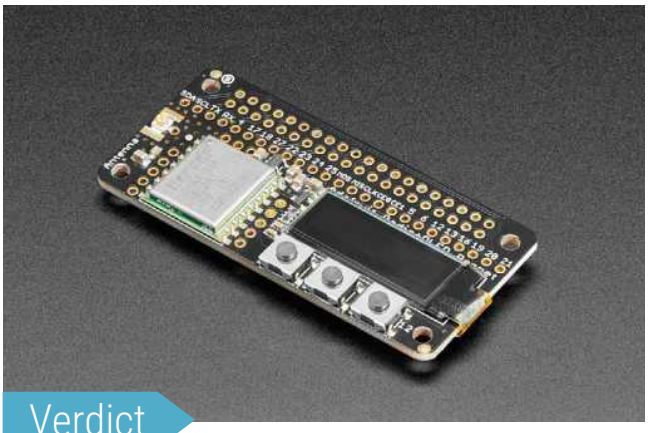
LoRa Radio Bonnet with OLED

Adafruit / The Pi Hut | £31 / $42 | adafruit.com / thepihut.com

This LoRa 'Bonnet' can be mounted onto the GPIO header of any standard 40-pin Raspberry Pi computer. While it's labelled as '915MHz' on the Adafruit site, it can use either the North American/Australian 915MHz or European 868MHz ISM band – you just need to make sure you set the correct band for your region in your programs. Fortunately, CircuitPython firmware and libraries make it easier to set up and use than some boards.

Another bonus is the 128 × 32 OLED screen that can be used to show status messages. There are also three tiny buttons you can use for creating a user interface or sending test messages.

For an antenna, you can attach one to the U.FL connector on the board, or even just solder a piece of wire to the pad next to it. One slight drawback is that its RFM95W radio module, while reliable and well documented, uses a classic SX1276 LoRa transceiver that isn't as power-efficient as some later versions.



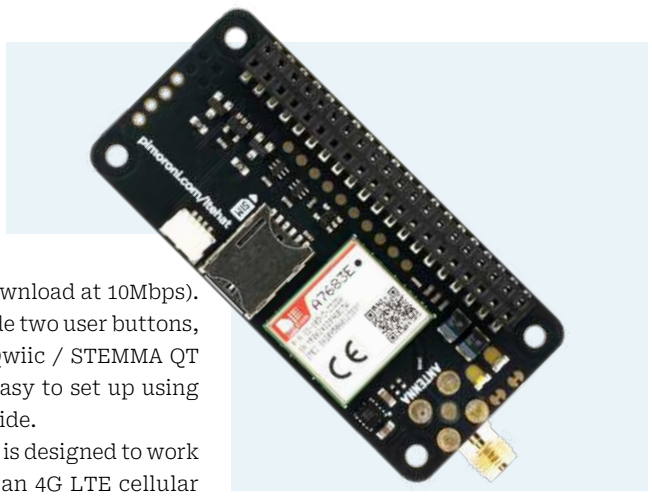
Verdict

It works with any Raspberry Pi and the on-board screen is handy.

▲ Yes, that is a tiny OLED screen on the board!

Clipper HAT Mini

Pimoroni | £30 / $33 | pimoroni.com



If you need anything more than small data packets from a remote device, 4G cellular is ideal. As well as low latency, it offers plenty of bandwidth – you could even stream video, such as from a wildlife camera.

The slimline Clipper HAT Mini fits flush onto a Raspberry Pi Zero, but will also work with any standard-size 40-pin model if you use a booster header. Based around the SIMCom A7683E 4G LTE module, it takes a nano SIM card and can upload data at up to 5Mbps (or download at 10Mbps). Bonus features include two user buttons, status LEDs, and a Qwiic / STEMMA QT connector. It's also easy to set up using Pimoroni's starter guide.

Note that this HAT is designed to work with UK and European 4G LTE cellular bands, so is unsuitable for use elsewhere. You'll also need a suitable antenna to attach to the board's SMA connector – Pimoroni supplies a short stick (52mm) type for under £4.

Verdict

A super-value slimline 4G board.

- Add a Raspberry Pi, SIM card, and antenna to start using 4G

PA1010D GPS Breakout

Pimoroni | £30 / $33 | pimoroni.com

While billed as a GPS (Global Positioning System) board, Pimoroni's tiny breakout board will also work with the EU's GALILEO system, among others, once you update its firmware. So you should be able to find your position, along with the ultra-accurate time, wherever you are in the world.

With an integrated antenna and low power draw, the board's postage-stamp-sized PA1010D module is ideal for smaller maker projects requiring location data. Scanning for signals from satellites, it supports up to 210 PRN (Pseudo-Random Noise) channels with 99 search channels and 33 simultaneous tracking channels. In most cases, with no major obstructions, it should provide your accurate location within a few seconds.

The board's design takes Pimoroni's Breakout Garden form factor, so it can be slotted into a Breakout Garden HAT for Raspberry Pi. Alternatively, you can solder on one of the two supplied male pin headers and connect it up directly to the GPIO pins.

Pimoroni's Python library makes software setup simple. You could even use the breakout with a Raspberry Pi Pico via Michael Bell's PA1010D MicroPython module: **rpimag.co/mpypa1010d**.

Verdict

Ideal for portable projects, it enables you to track the precise location.

- Discover your exact location with this mini GPS board

SIM7600G-H 4G HAT for Raspberry Pi

Waveshare / The Pi Hut | £80 / $107 | waveshare.com / thepihut.com

If you need to send lots of data, this 4G HAT has you covered, no matter where you are in the world. Its industrial-grade SIM7600G-H 4G module is designed to work in any region (the 'G' in the name stands for 'global'), supporting a whole range of 4G cellular frequencies. It also supports 3G and 2G services and can even handle SMS text messages and phone calls (by connecting a headset to its 3.5mm jack).

As an LTE Cat 4 model, it can transfer data at much faster rates than a Cat 1bis module such as that on the Clipper HAT Mini: up to 50Mbps upload (150Mbps download), although real-world speeds may be a little lower depending on your signal. To enable this, a USB cable (supplied) connects it to Raspberry Pi (as well as the GPIO header).

The HAT comes supplied with a wideband LTE 4G antenna and one for GPS – yes, it has that functionality built in as well, which works with various positioning systems.

▲ It'll work anywhere and you can even pinpoint your location with the built-in GPS

Verdict

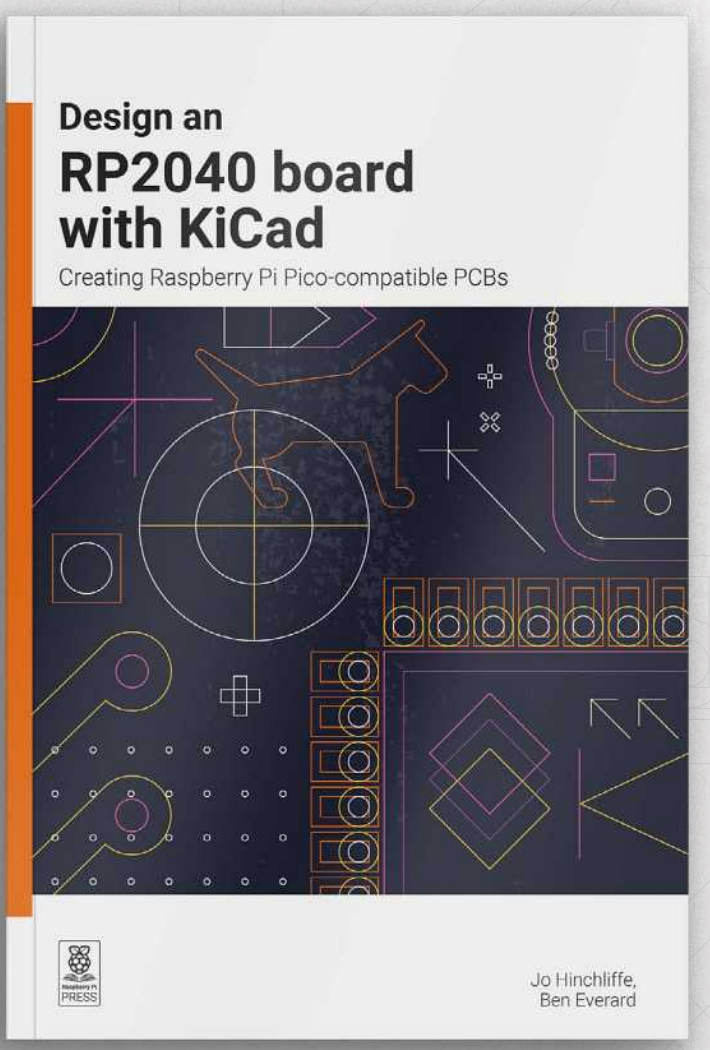
A high-end 4G HAT with faster data transfer and built-in GPS.

IC880A-SPI LORAWAN CONCENTRATOR

IMST | €149 (+ VAT) | rpimag.co/ic880spi

You can build your own multichannel LoRaWAN gateway with a Raspberry Pi computer and a suitable 'concentrator' board. For use in Europe (inc. UK), this 868MHz model is fully compliant with The Things Network and Stack (see the guide at **rpimag.co/ttsgateway**); for other regions, check out the RAK range of boards: **rpimag.co/rakgateway**.

▼ You can build your own LoRaWAN gateway



KiCad is an amazing piece of free and open source software that allows anyone, with some time and effort, to make high-quality PCB designs.

- ***Create a schematic for a microcontroller board using Raspberry Pi's RP2040***
- *Select the right components*
- ***Customise the hardware for your needs***
- *Lay out and route the PCB design*
- ***Prepare your board for manufacture and assembly***
- *Write software to get your design working*

Buy online: rpimag.co/kicad2040

SPOKE

An RP2040-powered touch controller for music and a lot more. By **Phil King**

VulpesLabs / Pimoroni **rpimag.co/spoke** £40 / $44

SPECS

FEATURES:
27 × capacitive touch pads, 27 × NeoPixel RGB LEDs, RP2040 microcontroller, Boot and Reset buttons

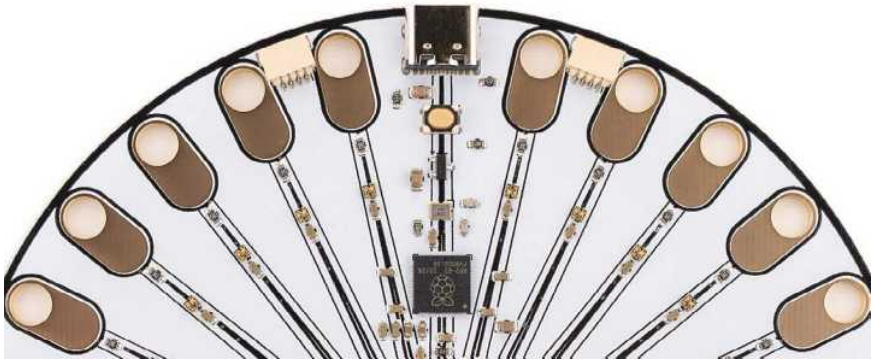
OUTPUT:
USB MIDI, HID keyboard / mouse, OSC, Serial

CONNECTIONS:
USB-C port, 2 × Qwiic / STEMMA QT

A shiny CD-sized disc with a cardboard sleeve featuring colourful fantasy artwork, the SPOKE could easily be mistaken for an album by a 1990s indie band. The main difference is that it has 27 holes around its perimeter, each surrounded by a copper oval: these are capacitive touch pads. It also has a USB-C port, reset button, and a Raspberry Pi RP2040 microcontroller chip (as used for Pico).

While this is no music CD, it can act as a musical device. As soon as you connect it to a computer, the SPOKE's default on-board program turns it into a MIDI controller, with each of the 27 touch pads playing a note when touched. Just visit **spokeboard.com**, click on SPOKE, enable MIDI access for your web browser, and touch a pad to hear a note.

▼ With 27 touch pads (each linked to a GPIO pin), the SPOKE has plenty of inputs for projects, whether musical or otherwise



The two Qwiic / STEMMA QT ports are a bonus, enabling you to connect sensors and other components (although this does disable a couple of touch pads)

Musical magic

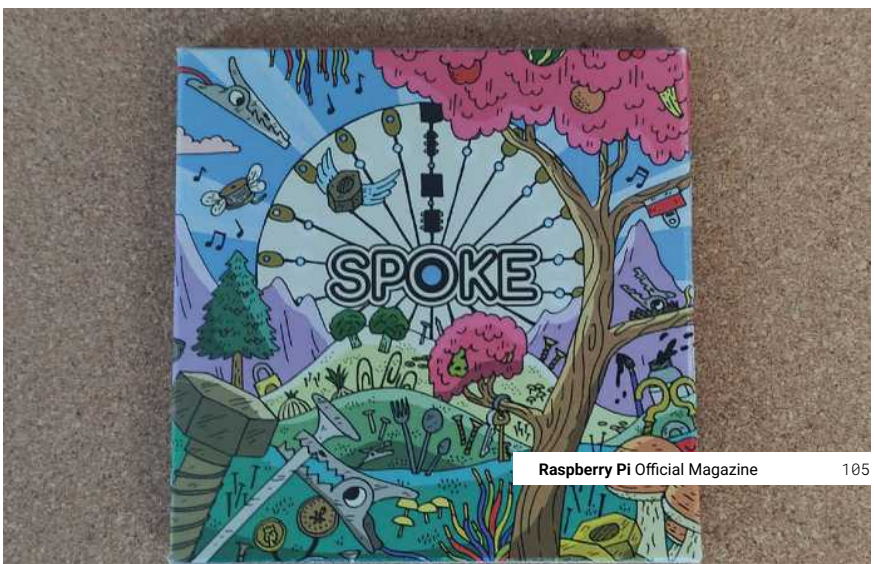
Clicking the Play tab on the site takes you to a page of options for how to play with the SPOKE. To get a feel for it, you may want to start with Visualisers: a series of audio visualisers with different sounds playing as you touch the pads. Or try the Harmonic Table with 100 polygons assigned to notes. When you touch a pad on the SPOKE, its NeoPixel changes colour. By default, the touch pads are assigned to a pentatonic scale (five notes per octave), but you can easily change that by altering the note values in the default CircuitPython program.

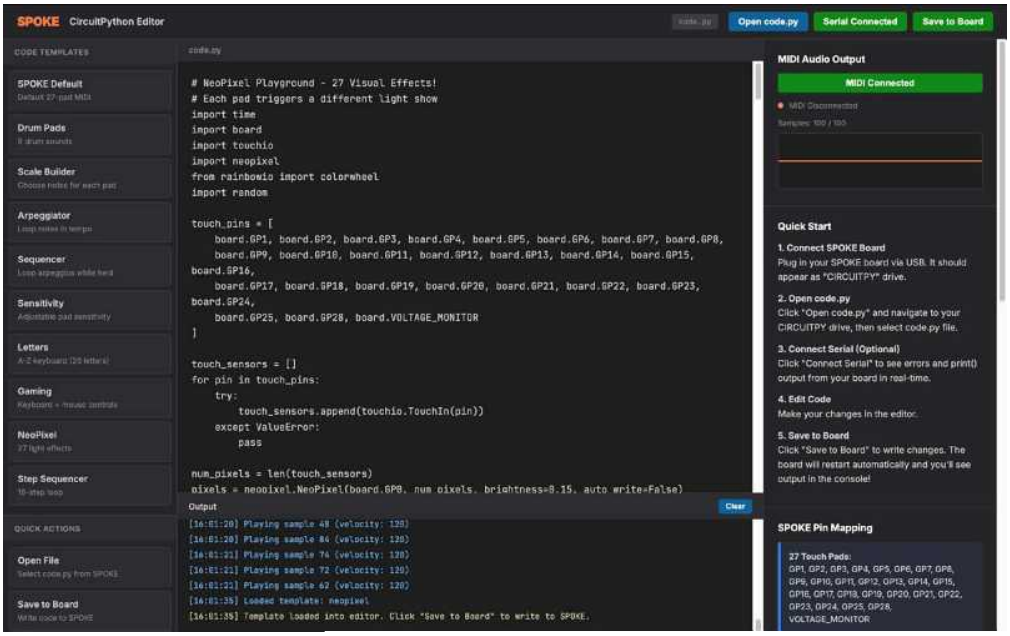
For more sophisticated musical output, the SPOKE Audio Playground enables you to search thousands of sounds from **Freesound.org**, upload your own, or record them with a mic, then assign each to a touch pad. Other music-based options include a One-Shot Drum Kit and a Storytime Page where you can add sound effects to the pads to play back while reading a story aloud. As the SPOKE is acting as a standard MIDI controller, you can also use it for any software synth or DAW with MIDI input, such as the browser-based ones listed on the webpage.

The default program turns it into a MIDI controller, with each of the 27 touch pads playing a note when touched

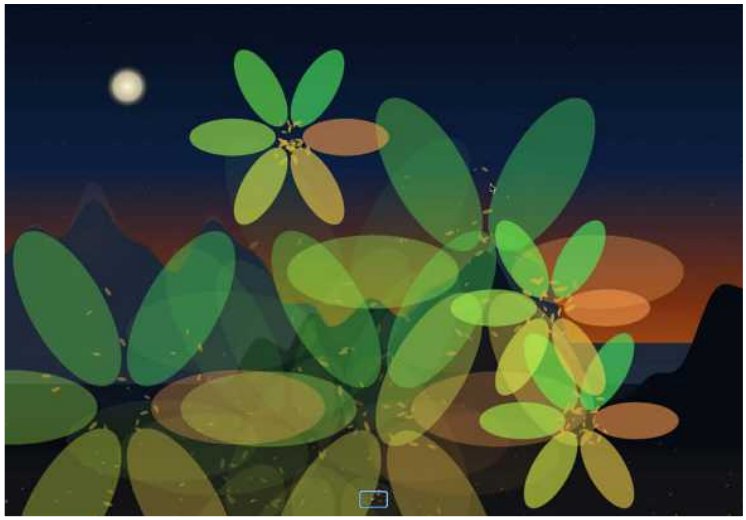
As with other capacitive sensors, you can extend the touch pads by using anything conductive, such as copper tape, conductive thread or ink, even pencil lines. We hooked up some crocodile clips to pieces of fruit for a fun mini drum kit! You just need to press the Reset button to calibrate the touch sensitivity once items are connected.

The cardboard packaging makes it look like a CD album; there's even a 'track listing' (telling you what it does) on the rear





◀ The browser-based Code Playground is an alternative to using a standard IDE and includes CircuitPython code examples and tips



◀ The Visualisers demo is just one of many options for playing music; you can use any MIDI-capable software synth or DAW

Coding projects

Because the SPOKE utilises CircuitPython for its firmware, you can program it just as you would a Pico, using a computer IDE like Thonny. Alternatively, the SPOKE site offers a browser-based Code Playground where you can connect via serial and try out some code examples. Along with musical programs such as an arpeggiator and step sequencer, and one for NeoPixel patterns, there are others that showcase the SPOKE's versatility. For instance, you can get it to act as an HID (human input device) with a letter of the alphabet (plus **SPACE**) assigned to each pad; or even as a game controller for Minecraft, including mouse buttons. For those new to coding, there's also a handy Learn To Code page to get you started.

To get an idea of what you can make with a SPOKE, there are some project examples on the website, including some novel musical instruments (such as a bonsai tree) and an interactive map that plays different videos when hidden sensors are touched.

The only slight downside to the SPOKE is its size, which could make it difficult to embed in smaller projects – in which case, you may well prefer to use the SPOKE-mini DIY kit with a PCB and a Pico and resistors to solder onto it.

Either way, the SPOKE is a fun, versatile touch controller that's easy to start using and coding for your own projects.

Verdict

A well-designed touch controller that can also act as a MIDI input or HID.

9/10

KIWI+ Net

Secure, local networking over an already excellent KVM. **Rob Zwetsloot** connects remotely

Cytrence | **rpimag.co/kiwiplusnet** | £89/$119

SPECS

I/O
USB-C (host connection), USB-C (input connection), USB-A (shared connection), HDMI, 6 × GPIO

CONNECTIVITY
1080p video, human input devices, virtual serial connection, Etherlink, UART, ATX, USB devices

DIMENSIONS
46 × 46 × 15mm

Verdict

The standard yet great KIWI+ experience with added networking capabilities for developers that require it, or for those with limited internet connections.

9/10

We've previously reviewed one of the KIWI+ range of devices from Cytrence – an upgraded version of its very good KIWI. The short summary is: it allows you to hook up a Raspberry Pi to a USB port on your PC so that you can capture video from Raspberry Pi and use your standard keyboard and mouse with it.

In this regard, KIWI and KIWI+ work extremely well. Unlike VNC or other network solutions, input lag is negligible and frame rate remains consistent. It has a few clever functions for capturing/decapturing your mouse and sending specific commands through to the other device and we've not found any issues with the range since we started reviewing them. This KIWI+ Net is no different.

▶ KIWI+ devices are very small with a metallic construction

▶ It doesn't just work with Raspberry Pi either – it can work between different computers and even with some games consoles

Network connected

Each KIWI+ has a specific extra function on it, and the Net version allows for a local network created over the USB connection between the host and target device. It's pretty easy to set up from the host end – just toggling an option in the KIWI software turns the network bridge on, although on Raspberry Pi (and other machines running Linux) you'll have to do a little extra setup by setting DHCP to Link-Local Only. Our Raspberry Pi took a couple of minutes to then connect to the network, but once it did it was fully discoverable from the host.

From here you can do various testing of network capabilities without having to worry about anything intercepting packets, and you can even share internet to Raspberry Pi from the host.

The network connections don't interrupt the usual KVM functions either, with no noticeable lag added.

Hundreds of products now sport the Powered by Pi logo. **Rosie Hattersley** looks at some leisure-based examples

The summer holidays are nearly here for eager students and, with them, a putative chance to unwind. Parents of course might need to find some activities to divert their curious offspring, in which case, the Doly robot looks like an intriguing option as both voice-controlled companion and a route into coding. We are also delighted to showcase Elecrow's renowned CrowPi electronics rig in a flight case which will also keep coders engaged for many hours.

For those keen to make a vacation getaway, SavvyVan's impressive campervan smarts look like a great addition to a mobile home from home, Velo AI will help reassure cyclists, and sky watchers can learn what's speeding overhead with JetClock's flight tracker. If you want to share what you've been doing and seeing, the Radioberry HAT is a great route into internet radio broadcasting.

All the above have Powered by Pi accreditation in common, joining hundreds of other Raspberry Pi-compatible products and accessories. Distinguishing a top-quality product is made very much easier once you know to look out for the Powered by Pi logo. It denotes a product accredited by Raspberry Pi and one that has gone through rigorous testing, with nearly 400 products now bearing the scheme's badge.

Find detailed information about the compliance regulations and testing procedures for every Raspberry Pi product at **pip.raspberrypi.com**.

APPLY TO POWERED BY RASPBERRY PI

Our Powered by Raspberry Pi logo shows your customers that your product is powered by our high-quality Raspberry Pi computers and microcontrollers. All Powered by Raspberry Pi products are eligible to appear in our online gallery.
rpimag.co/poweredbypiapply

JetClock

UK | jetclock.io

How often do you idly wonder about the planes flying overhead? With balmy summer days and nights and windows ajar, the wide open skies and airborne craft seem far closer than usual. If early morning sunshine has woken you from your slumber and you've spotted an aircraft zipping by, the clever Raspberry Pi Zero 2 W-powered JetClock can give you a heads-up on exactly what plane it is. Clock-style circular designs and a tablet version each offer plenty of details such as flight number, destination, airline and route, and can detect planes as far as 200km away. The stripped-back display also leaves plenty of room for the clock face itself to be the main focal point.

Velo AI CoPilot

USA | rpimag.co/veloai

CoPilot is a sort of sentinel for cyclists, providing all-important visibility with a bright lamp so other road users know you're there. It also sports a built-in proximity sensor so that should another cyclist, animal, pedestrian or vehicle come a bit too close, a visual alert is displayed and an audible alert is emitted. Its makers describe CoPilot as a "personal guardian that watches what you can't see" such as vehicles approaching from behind. The Compute Module-based setup uses AI-based data sets to determine the sort of 'near miss' vehicle coming too close, making it more useful than a radar-based proximity sensor that only shows something close by but not whether it's a ten-ton truck approaching at high speed or another bike swerving a bit: **rpimag.co/veloarticle**.

Elecrow CrowPi

China | rpimag.co/crowpiedu

CrowPi is one of the most impressive educational laptops we've encountered in the decade-plus that we have been writing about and reviewing all things Raspberry Pi. The standard kit consists of a 9-inch 1024 × 600-pixel IPS touchscreen display (an upgrade from the original model's 7-inch screen), microphone, infrared remote control, sensors, buzzers, stepper motors and servo, LEDs, plus connectors and buttons galore.

To this you just need to add your choice of Raspberry Pi 4 or Raspberry Pi 5 and install software onto the supplied 32GB microSD card to get the most out of learning electronics and coding. Everything comes in a ruggedised 10.6-inch flight case with a sturdy handle and lock, for easy transportation. An advanced model includes a heatsink (handy if you're using it with the powerful Raspberry Pi 5), gamepad, earphones, wireless keyboard, and mouse.

AppMind Radioberry

Netherlands | pa3gsb.nl

Aimed at amateur radio enthusiasts, Radioberry is an open-source Raspberry Pi 4 or 5 HAT that operates as an SDR (software-defined radio) transceiver and uses a 12-bit AD9866 broadband modem chip, with firmware stored on Raspberry Pi's SD card. Broadcasters use its WSPR (weak signal propagation) to let other radio stations know of their call-sign and location, building an ad hoc network, usually on either 40m or 20m HF bands.

Internet-based radio broadcasting has been around for decades, mirroring the increase in accessibility, affordability, and speed of web connectivity. Raspberry Pi is an obvious component of such setups, providing the modest amount of power needed as well as being small and self-contained and therefore ideal for mobile radio broadcasts.

SavvyVan 3

UK | rpimag.co/savvyvan3

SavvyVan 3 is just the sort of touchscreen control console you need to give you peace of mind when setting up your campervan at a new pitch. Its appealing visuals make it really clear what's what with your electric hookup or off-grid rig, allowing you to make all-important adjustments. For example, there's an add-on that shows whether you've parked on level ground and provides indicators to help you even things up for a good night's sleep. Large icons indicate power levels for your main and leisure battery, electric fridge, and the amount of water left in your tanks. They can also provide visual alerts when power is running low, plus detailed weather information so you can plan your travels and days adventuring. SavvyVan comes in DIY installation versions (with useful YouTube and online guides) with modular add-ons, but can also be professionally installed. The company was founded by IT veteran Simon Knight in 2020 and has partnerships with dozens of campervanning sites and companies across the UK.

Doly

Canada | doly.ai

Canadian company Limitbit has attracted a lot of interest since it launched Doly, an educational AI companion robot that has different tiers of complexity depending on the sort of coding journey you want to use it with. For parents and young learners, there's a Blockly-based coding setup that doesn't require a subscription and has 3D-printable elements with which to customise and personalise the basic robot. As coding knowledge becomes embedded, there's a natural step over to Python or C++, with which more advanced robot controls and manoeuvres can be accomplished. Doly is also great as a companion, listening out for voice commands such as setting timers and responding to queries about the weather, time, news, and other common questions. This Raspberry Pi-configured version has an 8MP camera. Doly learns individuals' voices and can respond to known users' instructions to take snapshots and so on.

10 amazing: environmental conservation projects

Saving the planet, one Raspberry Pi at a time

We were thinking recently about how many HATs and add-ons for Raspberry Pi include multiple environmental sensors, and how these can be applied. Due to the size and low power draw of Raspberry Pi, and these sensors, you find them out in the wild (literally) helping scientists study and conserve nature. Here are just ten such projects.

01. Astro Pi
Space-borne studies
astro-pi.org

The Astro Pi units may be orbiting the Earth far beyond its atmosphere, but are still close enough to study the planet. Some of the student projects have ecological subjects.

02. Maka Niu
Coral Reef photography
rpimag.co/makaniu

This deep sea imaging device can be used to monitor wildlife of all varieties on the ocean floor – it can go at least 1500 metres deep.

03. Penguin Watch
Antarctic bird survey
rpimag.co/arribada

Due to Covid, this Raspberry Pi monitoring station was left out for a year longer than expected, but managed to stay working and catalogue thousands of penguin photos.

04. Antarctic PiCam
Algae on ice
rpimag.co/antpicam

This simple watertight camera was created by a reader after learning of a similar project in the magazine. It went to Antarctica to study underwater ice.

05. Remote agricultural monitoring
Pest control
rpimag.co/agrimonitor

Tracking and identifying Tephritid fruit flies is important to make sure crops in Japan aren't destroyed – these traps use machine learning to identify the flies.

06. A-BiRD bird recognition
Audio population tracking
rpimag.co/abirdblog

Bird population tracking is usually done via reported sightings. To try and get a more accurate account, this student created a device that tracks birdsong.

07. Desert Eye 2.0
Terrain scout
rpimag.co/deserteye2

Robots are very popular for dangerous missions – they are cheap and replaceable usually. This one is able to surveil deserts at night and give accurate terrain info to explorers.

08. Bugg.xyz
Audible environment health
bugg.xyz

Conservationists can tell a lot about the robustness of an ecosystem by listening to what sounds are being made and when, thanks to these Bugg recording devices.

09. Cave mapping
Seismograph mapping
rpimag.co/cavemap

Raspberry Shake is a citizen science device that allows folks to help track seismic activity. Using some very advanced maths, the same tech can be used to create maps of cave systems.

10. Mountainous wildlife monitoring
Leopard warning system
rpimag.co/wwfcam

WWF-Pakistan uses Raspberry Pi 4 to track snow leopards in remote mountainous regions to make sure that livestock predation can be reduced.


02

03


04


06


05


07


08


09


10

Tom Fox

A teacher who has invented hardware for students and makers alike

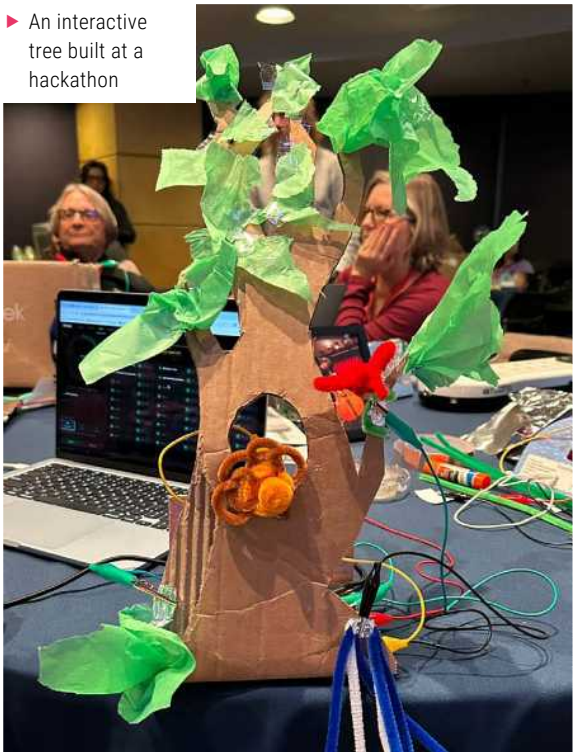
Name Tom Fox
Occupation Teacher
Community role Creative technologist
URL vulpeslabs.org

Raspberry Pi was famously conceived with education in mind, so it's no surprise that we regularly come across educators who do more than just teach in the classroom. Tom Fox is one such person; he's created SPOKE (reviewed on page 104), which he uses in his classes and has been making its way into the wider maker world.

"I have been building interactive musical installations and instruments for over a decade, combining physical sound making with hardware and sensors," Tom says. "I started pretty low-tech, using bits of broken motors and transformers to create rudimentary guitar pickups and finding different ways to make strings and springs vibrate."

How did you get into education?

I was invited to run some workshops on instrument building by a makerspace in Bethnal Green called Machinesroom. Soon after, I also started running music tech workshops for Music Tech Fest who are based in Sweden. They would fly me out to festivals and events all over Europe... I started teaching as Head of Design Technology at Beechwood Park School in Hertfordshire in 2019 and I tend to focus on the type of design and engineering skills that are used in the real world... and in the past few years a series of projects that use the Raspberry Pi Pico to create companion robots, macro-pads, and now a much more in-depth project on building custom computer interfaces.



▶ An interactive tree built at a hackathon

When did you learn about Raspberry Pi?

My first Raspberry Pi was a Raspberry Pi 2 Model B, back in 2015. I'd known about the boards just from being involved in the maker world, but I didn't have any need for one until I had an idea for an instrument for the Tate Modern Turbine Hall festival that was played via Twitter… I had this idea to build an instrument that played melodies based on the user_id number of the last person who sent me a tweet. This was also my first coding project, which, looking back, was quite ambitious. This Raspberry Pi worked brilliantly and opened the doors to a million other possibilities I could now explore.

Where did the idea of SPOKE come from?

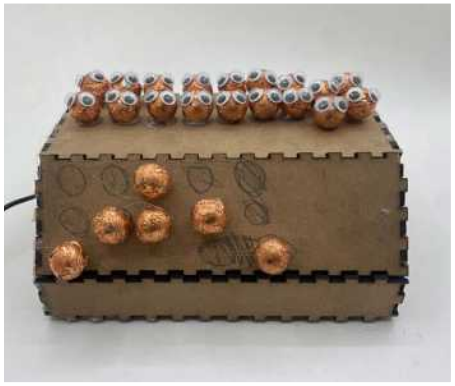
Originally I was finding a way to make my own artistic practices easier! I am part of a project called The Beating Birch which is all based around utopian world-building and music. Part of the project is about building new musical interfaces, and capacitive touch was what I wanted to focus on. I had traditionally used Bare Conductive boards or Teensys for touch sensing, but I wanted to try something new and affordable so I could make lots of instruments to fill a space. I came across the touchio library in CircuitPython and read that it just takes a 1MΩ resistor on a Pico to turn it into a touch sensor. I thought it couldn't be *that* easy, so quickly breadboarded it and it worked perfectly. I then decided to try *every* pin on the Pico, expecting it to need a ton of calibrations, noise reduction, fine tuning, etc. But all 26 pins worked perfectly, first time, with zero debugging. This quite quick test led me to develop what is now the SPOKE board, a 27-pin capacitive touch controller with NeoPixel indicator lights, easy (often no) sensor calibration, and a suite of web-based tools to help people get creating right away with no need for extra software.

◀ SPOKE was a successful Kickstarter project

> *I wanted to try something new and affordable*

What are some of your favourite things made with SPOKE by students?

My older pupils in year 8 have been using the SPOKE-mini, which is a solder-yourself kit that uses a Raspberry Pi Pico and 26 resistors to build your own embeddable version of SPOKE… I had one pupil who 3D-printed an enclosure in the shape of Gary from *SpongeBob SquarePants*, and then turned the dots on his shell into touch points. He started out as it being a musical instrument but then quickly realised he could play classic Doom on it instead. I did not stop him. One pupil made a birdhouse and used feather/leaf-shaped jewellery pendants as the touch points which would then play bird sounds. One pupil is obsessed with gorillas so made himself 26 copper-tape-covered bananas and turned it into a sound-effects board. Another kitbashed a Warhammer tank that had panels and turrets that would trigger sounds!



▲ One of the wonderful and creative projects created by a student

Get SPOKE

You can grab SPOKE from Pimoroni here:
rpimag.co/spoke

Maker Monday

Amazing projects direct from social media!

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.

Follow along to #MakerMonday each week over on our various social media platforms!

01. We would argue that destroying a patio is the only way to start a new project – **rpimag.co/167mm1**

02. The level crossing lights will save many tiny lives – **rpimag.co/167mm2**

03. We love the idea of a portable computer mounted on clear acrylic – **rpimag.co/167mm3**

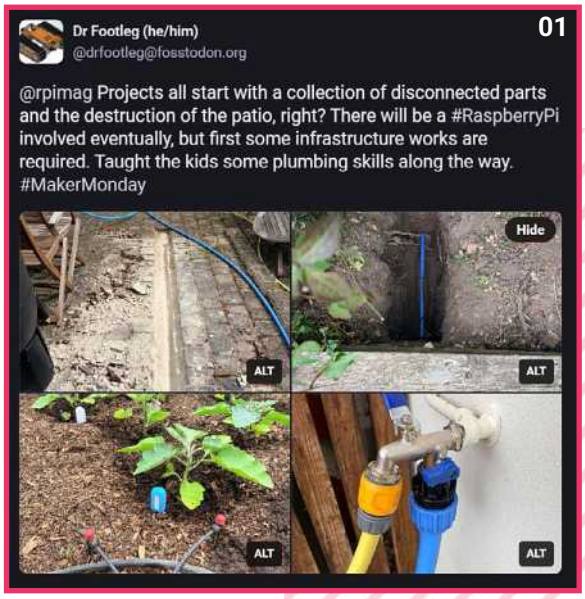
04. We love seeing the inventive ways folks restore and digitise old film – **rpimag.co/167mm4**

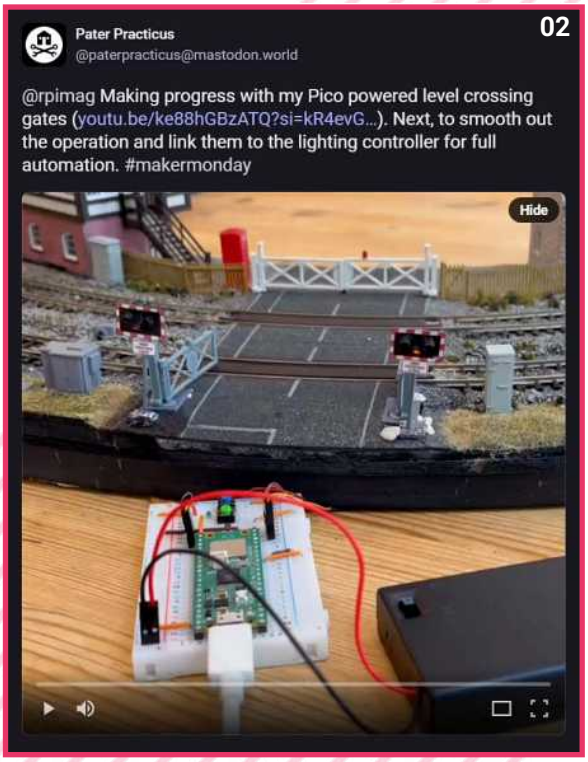
05. It was very hot in the UK and we were thinking about getting a little setup like this too to know when to open the window – **rpimag.co/167mm5**

06. Purely experimental monitoring of a Z80, fun – **rpimag.co/167mm6**

07. The *Aliens* motion tracker meets robot automation – **rpimag.co/167mm7**

08. We need some of these in our home – our plants keep dying on us - **rpimag.co/167mm8**





04

eightbytwofilms > MakerTuesday 26/05/2026

Here is my RaspberryPi 16mm film restoration machine. 😀 Cheers!

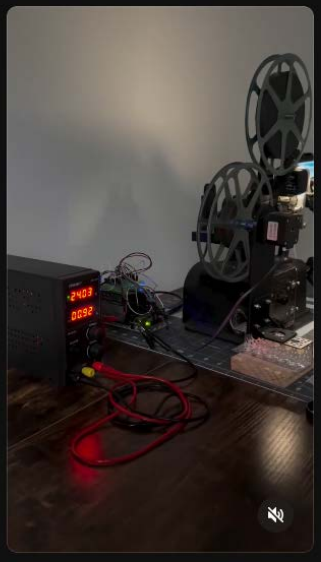
eightbytwofilms > RaspberryPi 25/05/2026

My creation.

16mm film automation

#film #filmmaking #raspberrypi #arduino #maker

@raspberrypi @arduino.cc @makemagazine @smallrig.global kodak



♡4 1

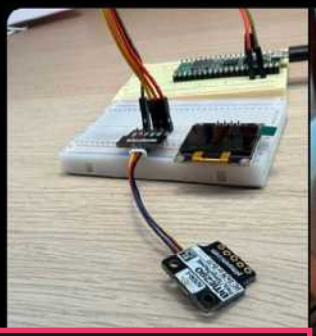
05

Kevin McAleer 🤖 Robot Maker
@kevsmac

Happy #MakerMonday (on a Tuesday), this weekend I setup a raspberry pi pico with a temp & humidity sensor attached and got that working with @arduino Cloud using #Micropython, full write up here:

kevsrobots.com/blog/going-wir...

I can see this data from anywhere!



06

diyelectromusic
@diyelectromusic@mastodon.social

@rpimag I've got a rp2350 keeping an eye on a Z80 in my rc2014 as an experiment...

This is using a Pimoroni PGA2350 in a bespoke breakout PCB hooked up to the rc2014 bus.

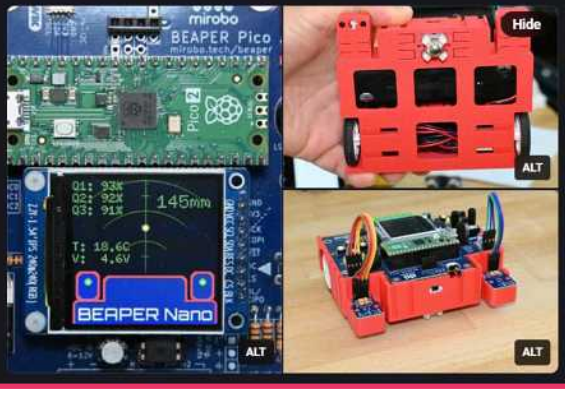
07

mirobo technology
@mirobotech@mastodon.social

@rpimag I've shared a new sensor demo program for my BEAPER Pico robot circuit – it displays all of its sensor data on a radar-like LCD image and now uses a more accurate I2C VL53L4CD ToF distance sensor module.

Q1-Q3 and the green dots show floor sensor reflectivity, T is the Pico's die temperature, V is the system voltage, and the yellow dot and 145mm readout are from the distance sensor.

github.com/mirobotech/BEAPER-P...



08

thecarolinedunn 21h

I made a DIY Remote Plant Monitor youtu.be/2MI0U...

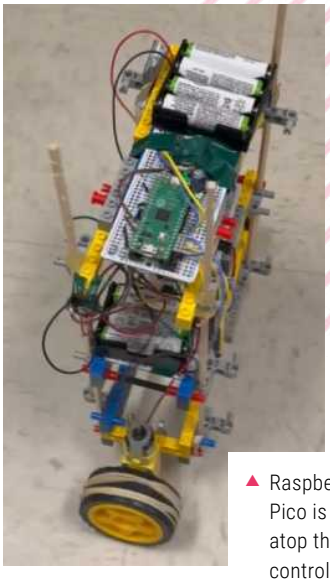
RP2040 Self-balancing robot

Two wheels is enough for this Raspberry Pi Pico-powered robot

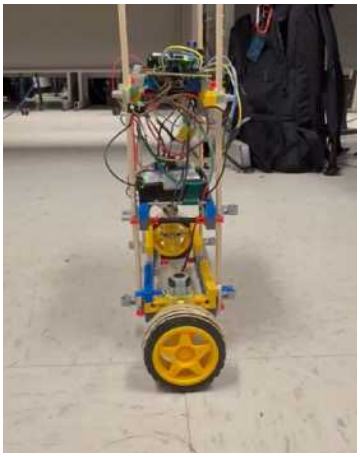
We recently published an interview with Hunter Adams (in issue 165, rpimag.co/165), who talked about his microcontroller course at the prestigious Cornell University in the US. One of his students approached us with their project from said class: a self-balancing robot.

How does it work? "The robot uses the Raspberry Pi RP2040 MCU [Pico] to read IMU (inertial measurement unit) data and pass the gyro and accelerometer readings into a complementary filter to approximate the robot's current angle," reads the report that student Aidan Chan sent us. "Then we input that angle into a PID controller to drive two DC motors with four PWM waves in order to stabilise a system that is in a constant state of instability."

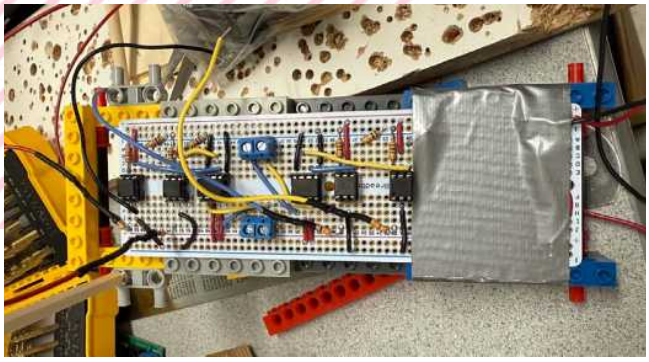
Sounds uh... simple? Either way, it worked. Read the full breakdown here if you want to learn more about the project: **rpimag.co/rp2040balance**.



▲ Raspberry Pi Pico is proudly atop the robot, controlling the motors to maintain balance



◀ An interest in how Falcon 9 landed was one of the inspirations behind this project



◀ All good student projects must include copious breadboarding

Crowdfund this

Crowdfunding campaigns to keep an eye on

CardputerZero

This 'computer for hackers' will run on the new Compute Module 0 and promises to be a pocket powerhouse, with the ability to have its functionality modified with M5 modules. It's smashed its goals already and made over $1 million at the time of writing.

▶ **rpimag.co/m5zero**

Touchscreen Data Logger

This simply named Touchscreen Data Logger is just that – a long-term testing unit that you can leave unattended. It monitors voltage, temperature, and humidity and also comes in a headless version if you don't need the screen.

▶ **rpimag.co/touchlogger**

Your Letters

What AI?

I want to be excited by AI, but I keep coming back to a basic question: what is it for? All the use cases that are being pushed seem to be trying to automate away what the anthropologist David Graeber would call "bullst jobs" – tasks that never seemed to be that important anyway. I can't think of a single way in which AI would make my job easier. That said, I'm going to keep playing with it – if nothing else, it's fun to see just how much better it's getting (but also how humans still beat it at anything creative).**

Ed, via email

It pleases me no end that I can perform some of the repetitive elements of my job using AI. The AI doesn't do a better job, but it does save me the time that otherwise I would spend shuffling bits of spreadsheet around and frees me up to do more interesting things, such as explore new technologies emerging on Raspberry Pi. That said, I'm not sure how comfortable I'd be with handing all my online login credentials and all my files to an agentic AI and telling it to get on with things. That's why, if you're going to try an agentic AI such as OpenClaw, it makes sense to do it within a sandbox – such as a separate Raspberry Pi 5. Start small, and remember – the AI can't think for you.

▲ A Raspberry Pi 500+ makes an ideal sandbox if you don't want to give AI the nuclear codes

▲ Raspberry Pi 4 is still going strong eight years after it was first dug up from the silicon mines

Raspberry Pi 6?

At last! The first sniff of a new Raspberry Pi 6 is confirmed! [on the recent Reddit Ask Me Anything, with James Adams, Gordon Hollingworth, and Eben Upton of Raspberry Pi]. Now cue the hordes of eager Raspberry Pi users demanding 64GB RAM, ports for everything, graphics acceleration, and a price tag under £50. We can dream!

Stéphane, via email

We should point out that Raspberry Pi 6 is not "confirmed" as such. However, Eben did say "it's roughly every 4-4.5 years. So not before early 2028 on that basis" in line with the usual release cadence. As exciting as it is to have new products just over the horizon, we're still happily prototyping with our old Raspberry Pi 4 and Zero 2 W models, which have been around for donkey's years.

There's beauty in simplicity

Raspberrarium

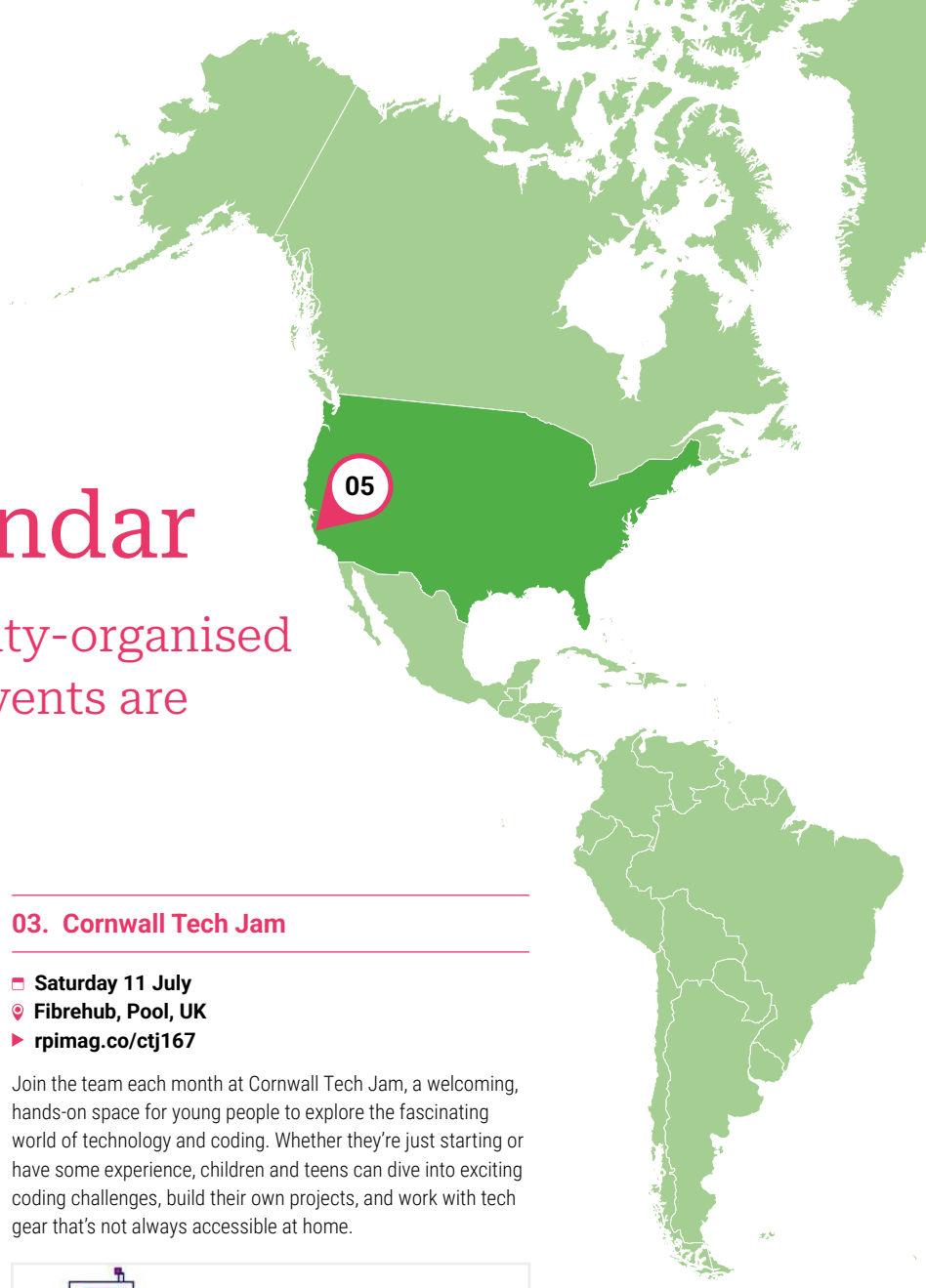
I loved the Raspberry Pi terrarium in last month's issue. I've seen loads of garden and greenhouse projects, but I don't have any outdoor space for plants; the most I have is a windowsill. It turns out that if I copy the terrarium, I won't even need that! Not only that, but as a terrarium looks to be a closed system, once I've set it up, I probably won't kill it if I forget to water it for a week.

Rachael, via email

The Raspberrarium is a wonderfully simple build, and is proof that with a bit of imagination anyone can make something brilliant. It has no moving parts, and you could even do a version that doesn't need any soldering – physically it's just two LEDs in a jar. But the vision to see that Raspberry Pi could make a controller for a lighting system that mimics the cycles of the sun and moon… that's the clever part. AI can't do that!

Contact us!

- @rpimagazine
- @rpimag
- rpimag.co/facebook
- magazine@raspberrypi.com
- forums.raspberrypi.com



Community Events Calendar

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

01. Raspberry Pi Jamwich

Saturday 4 July
Online
rpimag.co/rpjamwich167

If you ever need a reminder and a wake-up call for your day, this one is for you! Held online, the event will involve building the Raspberry Pi Morning Briefing Printer, a device to provide you with a short, fresh thermally printed note with weather, top news headlines, and your daily calendar priorities every morning. No cloud/subscription needed.

02. Physical AI Buildathon

Sunday 5 July
HSR, Bangalore, India
rpimag.co/physaibuild

Build a working physical AI gadget – the kind that lives outside a screen. Think a collar that knows when your dog is anxious, a plant that texts you, a mirror that reads your mood. Pick one of theirs, or bring your own. This is for you if you've touched Raspberry Pi, you took your toys apart as a kid just to see what was inside, and/or you've been looking for an excuse to actually build something this year.

03. Cornwall Tech Jam

Saturday 11 July
Fibrehub, Pool, UK
rpimag.co/ctj167

Join the team each month at Cornwall Tech Jam, a welcoming, hands-on space for young people to explore the fascinating world of technology and coding. Whether they're just starting or have some experience, children and teens can dive into exciting coding challenges, build their own projects, and work with tech gear that's not always accessible at home.

04. Southend Raspberry Jam Maker Meetup

Thursday 16 July
The Board Game Hut, Southend on Sea, UK
rpimag.co/srjmm167

A monthly meetup for those who are interested in building hardware and software projects using Raspberry Pi hardware and want to join a friendly group of enthusiasts and makers. They welcome beginners to professionals. If you have any ideas for projects, talks or demos, they're especially keen to hear from you.

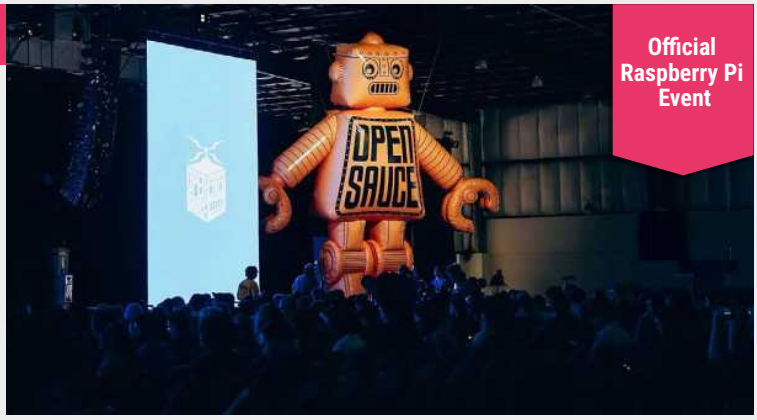
FULL CALENDAR
Get a full list of upcoming community events here:
rpimag.co/events

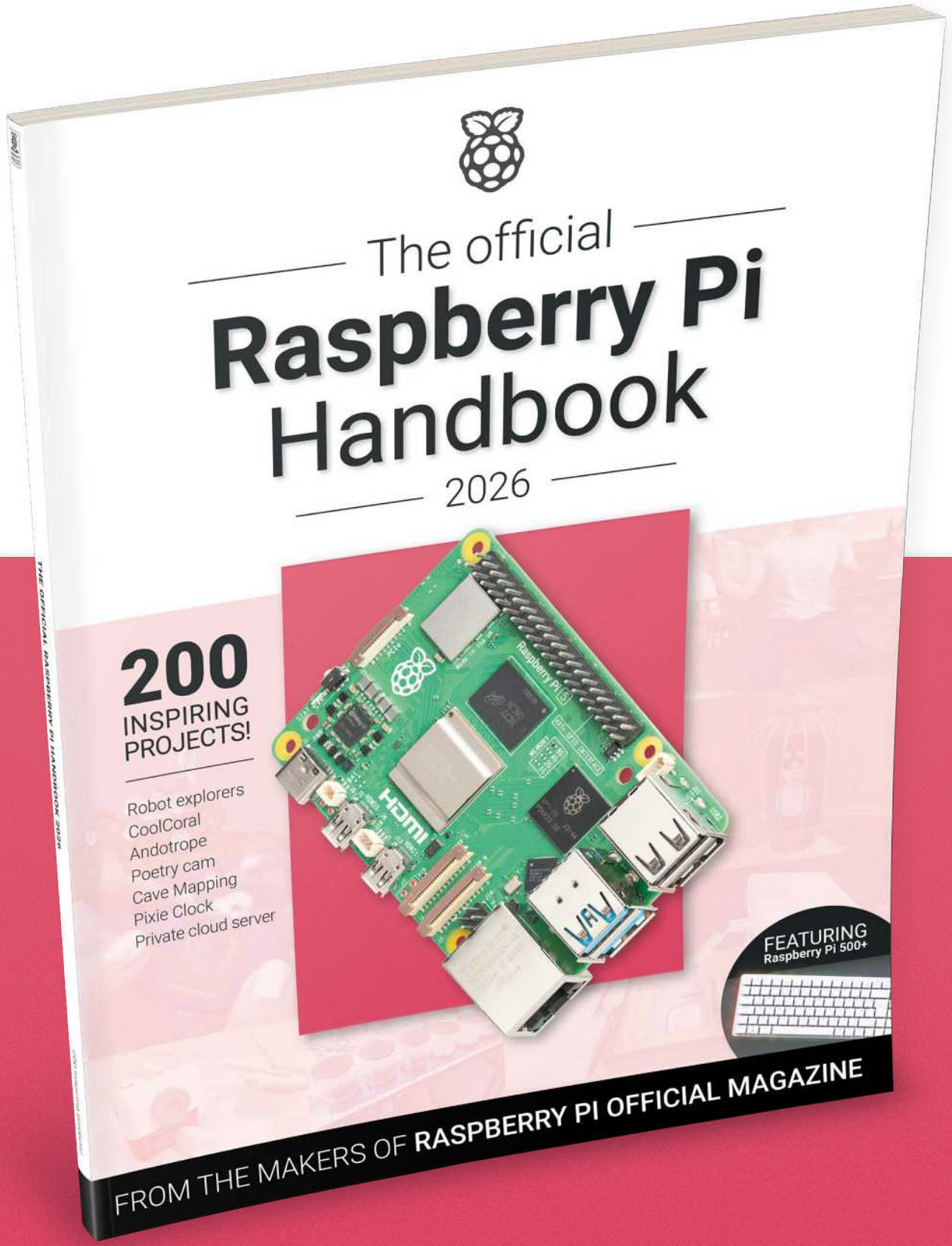
05. Open Sauce 2026

Official Raspberry Pi Event

- **Friday 17 July to Sunday 19 July**
- **San Mateo County Event Center, San Mateo, CA, USA**
- **rpimag.co/opensauce26**

Members of the Raspberry Pi team are excited to attend Open Sauce 2026 in California's beautiful Bay Area. We'll be showcasing demos of our latest maker-friendly computers, microcontrollers, camera modules, and AI products. Come and meet the team, tell us about what you're building, and grab a Raspberry Pi sticker for your project.




The official
Raspberry Pi
Handbook
2026
200
INSPIRING
PROJECTS!
Robot explorers
CoolCoral
Andotrope
Poetry cam
Cave Mapping
Pixie Clock
Private cloud server
FEATURING
Raspberry Pi 500+
FROM THE MAKERS OF RASPBERRY PI OFFICIAL MAGAZINE
THE OFFICIAL RASPBERRY PI HANDBOOK 2026

Win 1 of 3 Pironman 5 Pro Max

We quite like these Pironman 5 cases, and the Pro Max adds a ton of features to Raspberry Pi – like a touchscreen, speakers, and two NVMe slots, along with all the other bells and whistles such as cooling fans and a sleek design.

Head here to enter:

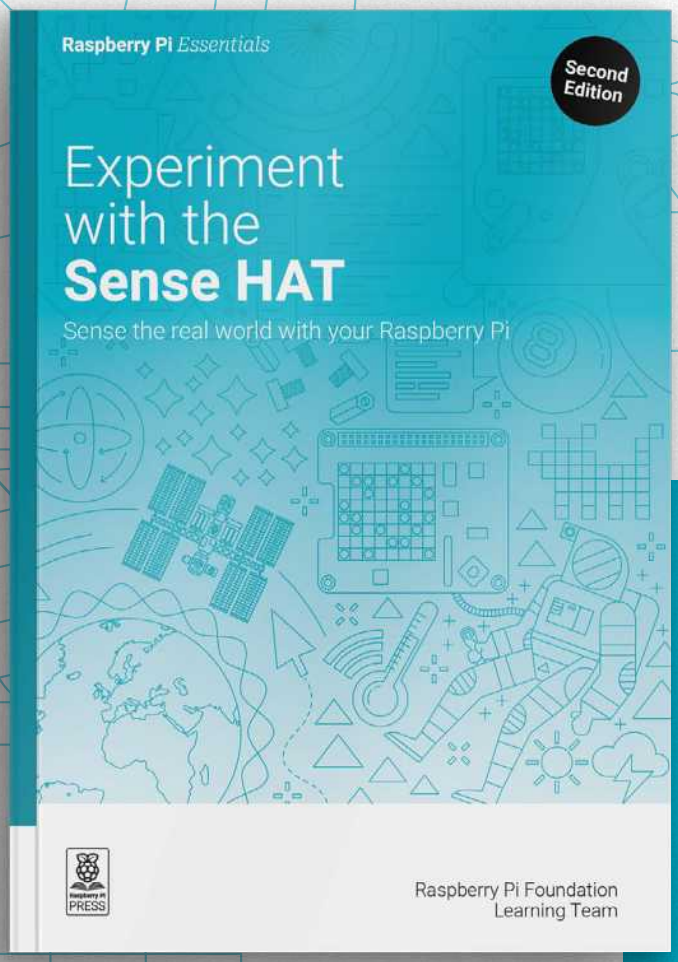
rpimag.co/win

Learn more:

rpimag.co/pironman5promax

Terms & Conditions

Competition opens on **24 June 2026** and closes on **30 July 2026**. Prize is offered to participants worldwide aged 13 or over, except employees of Raspberry Pi Ltd, 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 Raspberry Pi Official 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, Facebook, Twitter (X) or any other companies used to promote the service.



The Sense HAT is an incredibly versatile and flexible bit of kit with plenty of obvious uses, along with a huge number of less obvious ones, that you'll love to make and share. Updated for the latest Raspberry Pi devices and hardware, this book has everything you need to get started.

- ***Getting started with Sense HAT***
- ***Learn by building:***
  - *A digital twist on the Magic 8 Ball*
  - *Your own interactive pixel pet*
  - *A sparkly light show*
  - *An environmental data logger*
  - *Flappy Astronaut, a low-res, high-fun video game*

BUY ONLINE: **rpimag.co/sensehatbook**

REACH A GLOBAL COMMUNITY

Advertise in **Raspberry Pi Official Magazine**

Our readers are passionate about technology and the Raspberry Pi ecosystem. From DIY enthusiasts to professional engineers.

Your advertisement can sit alongside our cutting-edge tutorials, features, and reviews. And we have flexible advertising options for a range of different businesses.

Take the next step –
advertise with us today!

Email Charlie Milligan at: **charlotte.milligan@raspberrypi.com**

Official Magazine
#168

BUILD A CYBERDECK

Go full cyberpunk with a DIY hand-hacked computer

On sale 30 Jul

PLUS!

Private search nodes

PiSky flight tracker

Coding for students

Editorial

Editor
Lucy Hattersley
lucy@raspberrypi.com

Features Editor
Andrew Gregory
andrew.gregory@raspberrypi.com

Features Editor
Rob Zwetsloot
rob@raspberrypi.com

Sub Editor
Phil King

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

Design

Head of Design
Jack Willis

Designers
Sara Parodi, Natalie Turner

Illustrator
Sam Alder

Brand Manager
Brian O Halloran

Contributors

David Crookes, Tim Danton, Frank Delporte, Ben Everard, Rosemary Hattersley, Jo Hinchliffe, Nicola King, Phil King, Sean McManus, Rob Miles, Nik Rawlinson, Ashley Whittaker

Publishing

Publishing Director
Brian Jepson
brian.jepson@raspberrypi.com

Director of Communications
Helen Lynn

CEO
Eben Upton

Distribution

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

Subscriptions

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

ISSN: 2977-4403 (Print)
ISSN: 2977-4411 (Digital)

Living in the future

Lucy dreams of a smart home. But not too smart!

The dream of a smart home has been with humanity ever since *The Jetsons* aired in 1962. Lights that know when you walk in, thermostats that learn your schedule, speakers that answer to your every word.

And for a while it seemed that we were living in that future. Apart from the flying cars, we had the smart speakers, voice assistants, clever lights, and homes warmed to Goldilocks levels of perfection.

What we increasingly had to think about was that every light bulb, smart plug, connected switch, and speaker are pointed to the internet. They chat away to a range of companies, from small operations in China to massive US tech behemoths and everywhere in between.

In 2023 a House of Commons Committee report stated that 77% of UK adults own at least one smart home device. And that on average there are nine such devices in every UK home (**rpimag.co/cmsreport157**).

> *It's much better to set up your own system and understand what's going on inside your home*

That's a lot of chatter going to data centres. All these devices phone home to analytics servers, send telemetry to manufacturers, and send DNS queries through a variety of third-party infrastructure. Folks agree to all of this through T&Cs that they never read.

Privacy first

Running a Raspberry Pi is a great way to fix all of this mess. With Home Assistant you can build a privacy-first smart home hub. This month, Ben showed us his local smart home that keeps his data away from manufacturers' floaty clouds.

It's not just about tinfoil on the head, though. It's much better to set up your own system and understand what's going on inside your home. One of my favourite authors, Jeanette Winterson, said: "if you want to keep your own teeth, make your own sandwiches". It's important to know what's going into a system if you want to stay digitally healthy.

What's great about all of this – especially at the moment – is that you don't really need a powerful Raspberry Pi 5 with 16GB RAM to run a home network. In fact, an older Raspberry Pi 4 with a lot less RAM is perfectly powerful enough to manage your home.

We've got a bunch of other projects this month that fulfil this 'works on Raspberry Pi 4/400' brief. Another privacy project that works really well on Raspberry Pi 4 is Pi-hole (**rpimag.co/pihole**). I mention this because Raspberry Pi has been training a chatbot partially on our magazines (**rpimag.co/chatdoc**). So I decided to do some digging into our analytics to find out what your favourite tutorials were. And it turns out that Pi-hole is really, really popular. So you can look forward to a refreshed tutorial on that in the near future. We do listen!

I hope that the AI data centre rollout calms down soon, and we can get devices for reasonable costs again. I was hoping to get a Steam Machine but suspect it's going to be extravagantly expensive. In the meantime, at least you can use low-cost Raspberry Pi computers to keep your personal information in a data store all of your own making.

Lucy Hattersley – Author
Lucy is editor of *Raspberry Pi Official Magazine* and has been finishing off an AI book this month. Once you get past counting letters in 'strawberry', it's a lot of fun.

rpimag.co

HiPi.io

HighPi 5S

The new case from the HiPi team

Reliable case for complex projects:

- Built for Raspberry Pi 5
- Rapid tool-free assembly and disassembly
- Large internal volume for HATs even with the active cooler installed
- Camera cable slot to connect to external cameras
- Power button and external status LED
- Passive & active cooling options
- Secure microSD cover
- VESA mount support

Customization options for volume orders:

- Output ports molded to your exact specs
- Logo printing for a branded finish

Available at these great Pi stores:

PiShop.us PiShop.ca ThePiHut Pi-Shop.ch Welectron.

Contact your favorite Pi store if it's not listed here

PiKVM

Remote control **redefined**

Manage your servers or PCs remotely!

Powered by Raspberry Pi

PiKVM V4 Mini

Small, cost-effective, and powerful!

- Power consumption in idle mode: just 2.67 Watts!
- Transfer your mouse and keyboard actions
- Access to all configuration settings like UEFI/BIOS
- Capture video signal up to 1920x1200@60 Hz
- Take full control of a remote PC's power

PiKVM V4 Plus

The most feature-rich edition

- More connectivity
- Extra storage via internal USB3.0
- Upgraded powering options
- More physical security features
- Extra HDMI output
- Advanced cooling solution

A cost-effective solution for data-centers, IT departments or remote machines!

Available at the main Raspberry Pi resellers

core ELECTRONICS · PiShop.ca · RPishop.cz · Welectron. · elektorstore · PIMORONI

Pi-Shop.ch · ThePiHut · PiShop.us · CloudFree · KUBII · Cytron Marketplace

ROBU.IN Your ideas, Our Parts · NETTOP COMPUTERS & ELECTRONICS · netsket mall · DIREKTRONIK · RaspberryPi.dk

HiPi.io

No reseller in your country?
Check shop.hipi.io (import fees might apply).

List of official resellers by country: