Reanimate a **skeleton for Halloween**

Bring a dead parrot **back to life with AI**

Survive in style **with a cyberdeck computer**

Raspberry Pi

Official Magazine
#158 | October 2025

Raspberry Pi 500+

HIGH-SPEC — HACKABLE
AND **HERE AT LAST!**

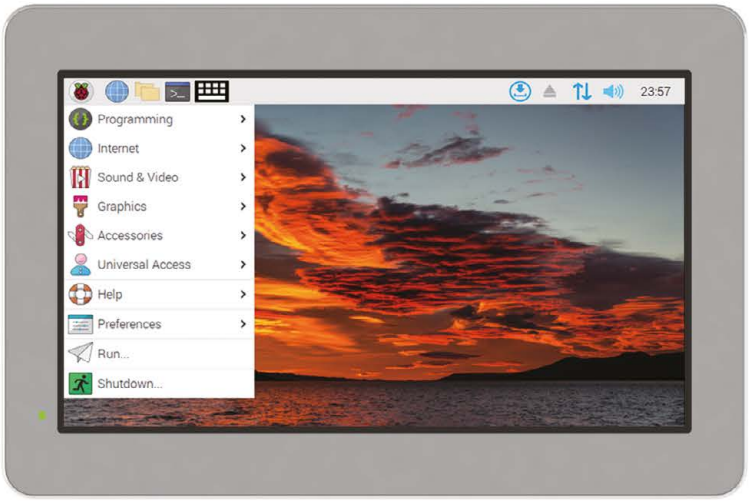
Exclusive engineering interview!

Mechanical keyboard
16GB RAM & 256GB SSD

10
£7.99
ISSN
2977-4403
(Print)

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
This month, Lucy has been digging out the anorak and fielding complaints from her cat. Summer was fun while it lasted.

rpimag.co

At last! I can finally talk about the incredible new computer that Raspberry Pi has been working on.

I am writing this on my brand new Raspberry Pi 500+ in Pi Towers, Cambridge. I've got a matching one on my desk at home and am syncing the two together. These are my computers now.

Everything I write is done with a Raspberry Pi 500+. The keyboard is amazing. An absolute joy to type on. The whole computer is self-contained within the keyboard design, and it's incredibly fast. Thanks to the 2.4GHz Arm CPU, 16GB RAM, and an M.2 NVMe SSD, it skips along without missing a beat.

Even if 500+ was running Windows or macOS, it would be swift. But because my 500+ is running Raspberry Pi OS based on Linux – the best operating system – it runs like the wind. There are no rainbow wheels and spinning hourglasses here.

We can't wait to share 500+ with you. It really is the best computer we've used in a long time. It is packed with all the maker goodness of Raspberry Pi, along with a keyboard designed with coders and engineers in mind, with expandable GPIO and RGB lights.

For a long time now, I've espoused that Raspberry Pi is every bit as good as a grown-up computer. That was then: this is now.

It's better – and it's ours!

Lucy Hattersley – Editor

The parts we sell help us reach for the stars

Since first venturing beyond the safety of a cave, we have been natural explorers. Discovering what's beyond the next hill, horizon, and star, is a drive that is innately us.

The parts and services we provide help fuel the next era of exploration, and like you, we can't wait to see what's out there.

Discover millions of parts at digikey.co.uk

DigiKey

we get technical

Contents

World of Raspberry Pi

Project Showcase

Raspberry Pi is published monthly by Raspberry Pi Ltd., 194 Cambridge Science Park, Milton Road, Cambridge, England, CB4 0AB. Publishers Service Associates, 2406 Reach Road, Williamsport, PA 17701 is the mailing agent for copies distributed in the US. Periodicals Postage paid at Williamsport PA. Send address changes to Raspberry Pi c/o Publishers Service Associates, 2406 Reach Road, Williamsport, PA 17701.

Top Projects

Feature

Tutorials

Feature

PCBWay

FULL-SERVICE ELECTRONICS MANUFACTURER

PCB Fabrication / CNC | 3D Printing / PCB Assembly / OEM I EMS

CUSTOMIZED TECHNICAL PARTS & PCB MANUFACTURED

WHY CHOOSE PCBWAY?

PCBWay offers a wide range of services including PCB fabrication, PCB assembly and even CNC machining for over a decade, and has earned a distinguished reputation globally in the industry.

Hassle-free ordering

The digital quote-to-order platform puts you in the back seat. Upload a Gerber file and receive feedback soon.

On-time shipping

We maintain a 99% on-time delivery rate, working in three shifts to ensure your packages arrive fast.

Quality assurance

Our technicians have been working strictly to high standards. All the boards will go through the most stringent tests.

Customer support

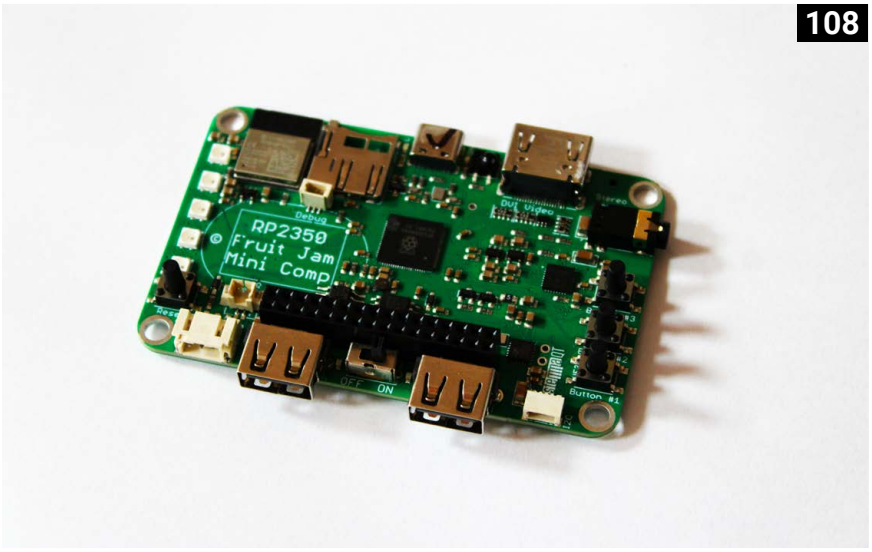
The customer service teams work inshifts to provide 24-hour support. You can always contact a live customer service person.

Scan the QR code for more details!

CONTACT US:

www.pcbway.com

service@pcbway.com



Reviews

Raspberry Pi Community

Competition

Win 1 of 5 Raspberry Pi 1TB SSD

WIZnet

W6300

The Ethernet Solution for Faster, Smarter Networking!

Need ultra-fast, stable Ethernet?
W6300 delivers up to 90 Mbps with powerful IPv4/IPv6 dual stack support!

Key Benefits

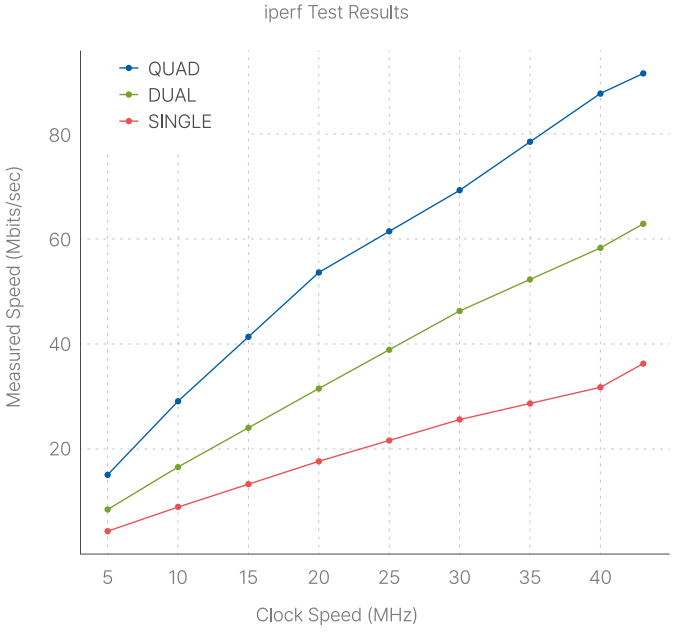
- **High-speed QSPI Interface**
  SINGLE, DUAL, QUAD modes supported
- **8 Independent Sockets**
  Multi-device connectivity made easy
- **Low Power & Wake on LAN**
  Efficient and remote-friendly
- **Seamless IoT & Industrial Integration**
  POS, cameras, automation & more

Where to Use W6300?

- **Smart Home & IoT**
  Network sensors, smart APs, home automation
- **Industrial & Factory Automation**
  POS, network printers, LED displays
- **Security Systems**
  DVRs, CCTV, access control systems
- **Embedded Servers & Cloud Devices**

Performance Test Results

- iPerf3 Performance
  Up to 90 Mbps on RP2040/RP2350 in 43MHz QUAD mode
- Optimized speed across different QSPI modes



WIZ630io 

W6300-EVB-Pico (W6300 + RP2040) 

W6300-EVB-Pico2 (W6300 + RP2350) 

Smaller Touch Display 2 released

New 5-inch variant of Touch Display 2 announced. By **Lucy Hattersley**

▲ Testing zoom in the slideshow application

Raspberry Pi has released a smaller variant of Touch Display 2, its touchscreen solution for Raspberry Pi. The new variant features a smaller, 5-inch, display with a simplified form factor.

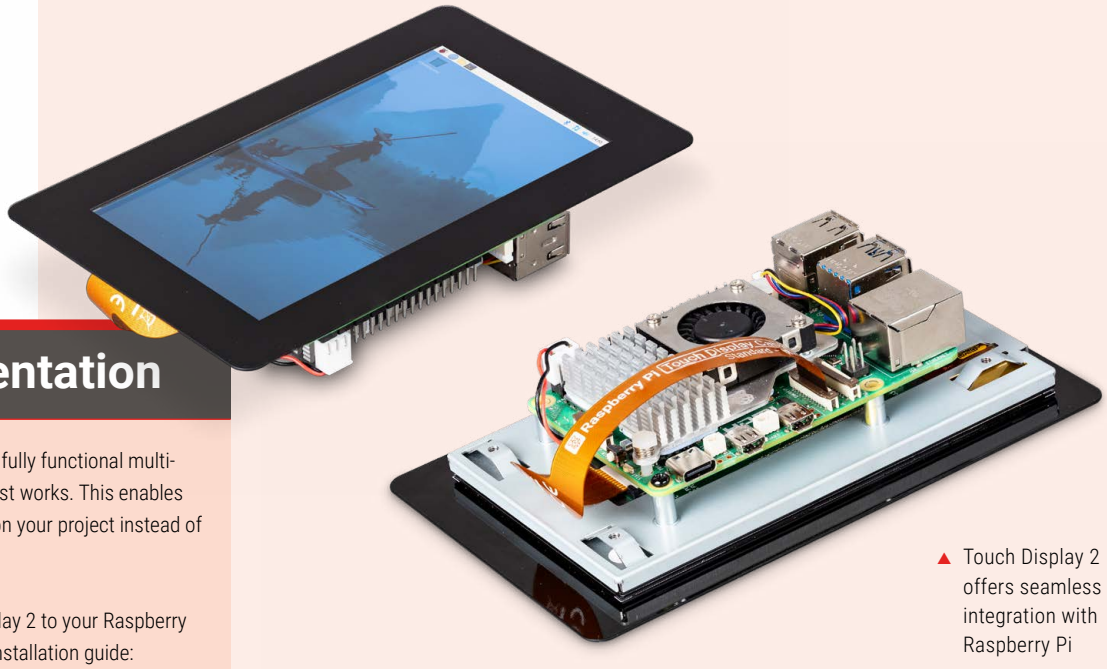
"Last year we launched the refreshed 7-inch Raspberry Pi Touch Display 2 as a successor to our original 2015 Touch Display, offering a simplified form factor and higher 720×1280-pixel resolution at the same $60 price point. Today we're excited to announce a new 5-inch variant, available to buy now from Raspberry Pi Approved Resellers. It shares the same resolution and easy setup as the 7-inch variant, at the low price of $40," says Gordon Hollingworth, Chief Technology Officer, Software at Raspberry Pi.

Raspberry Pi Touch Display 2 allows seamless integration with the rest of the Raspberry Pi ecosystem. "Its capacitive touchscreen works out of the box with full Linux driver support," explains Hollingworth. There is "no manual calibration required, no hunting through device trees, and no wrestling with incompatible touch controllers".

A demonstration

To help guide newcomers into using Touch Display 2, Hollingworth has created a touch display slideshow app, built using Claude Sonnet 4 and Cursor. "To illustrate our new 5-inch display's capabilities, I decided to create a simple slideshow application using AI-assisted development," he says. "This seemed like a perfect opportunity to explore and demonstrate both the hardware's multi-touch features and modern development workflows.

"Not everyone thinks AI is the future of software engineering, but I find it important to understand how technology advances, so this year I've been dipping my hand into coding with AI."



▶ Touch Display 2 now comes in a smaller, 5-inch display, variant

▲ Touch Display 2 offers seamless integration with Raspberry Pi

Documentation

Touch Display 2 is a fully functional multi-touch display that just works. This enables you to concentrate on your project instead of hardware hassles.

Connect Touch Display 2 to your Raspberry Pi by following the installation guide: **rpimag.co/touchdisplay2doc**.

Hollingworth has shared the AI prompt used: "I would like to create a simple application running on the Raspberry Pi remote device which has a touch panel attached. The application should display images from a local directory as a slideshow. Touching the display should stop the slideshow and allow the user to manipulate the position and be able to zoom in using standard gestures."

> *Touch Display 2 allows seamless integration with the rest of the Raspberry Pi ecosystem*

You can read more about this code development process on Hollingworth's blog post: **rpimag.co/touchdisplay2blog**.

"I hope my quick demo has given you an idea of how easy it can be to develop interactive applications with our multi-touch displays," he writes. "Touch Display 2 offers a straightforward way to integrate a high-quality user interface into countless applications, whether those are personal builds, research projects, or commercial solutions, and our new 5″ variant provides an even more compact option if you're targeting the smallest form factors."

Both 5-inch and 7-inch variants of Raspberry Pi Touch Display 2 are available to buy now from our worldwide network of Approved Resellers. As ever, we'll enjoy seeing what you do with it.

Key features

Except for its size, the specification of the new 5-inch variant is almost identical to that of its bigger sibling:

- 5-inch diagonal display
- 62.1mm × 110.4mm active area
- 24-bit RGB 720 × 1280 (RGB) pixels
- True multi-touch capacitive panel, supporting five-finger touch
- Fully supported by Raspberry Pi OS
- Powered from the host Raspberry Pi
- All necessary cables, connectors, and mounting hardware included

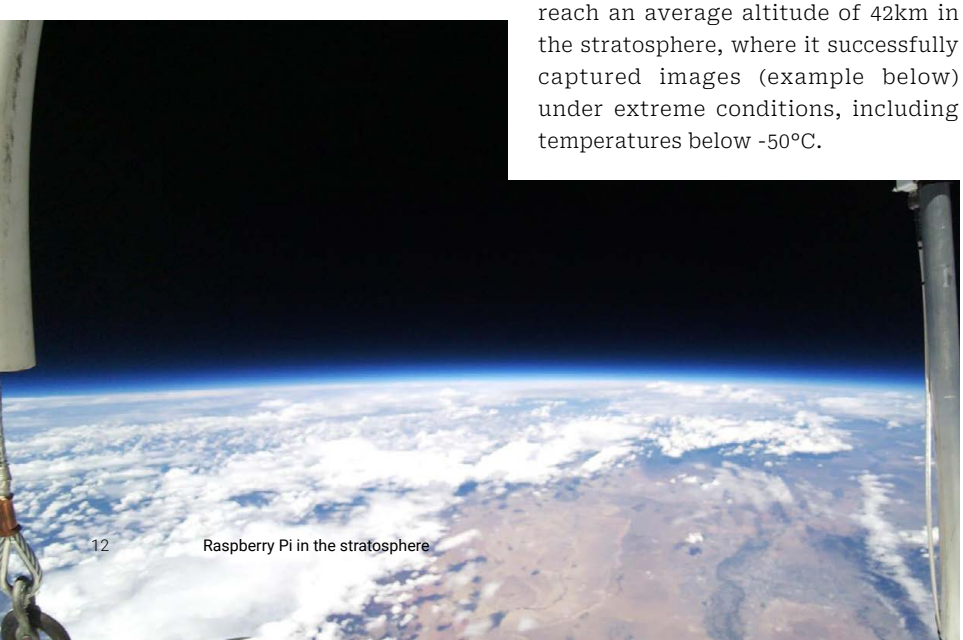
Raspberry Pi in the stratosphere

Project Trinidad. By **Ashley Whittaker**

Elvis Andrés Ayala, an electronics engineer who is passionate about developing accessible technology for space exploration, wrote to us to share the progress of his project Trinidad, which has been made possible thanks to Raspberry Pi hardware.

Extreme conditions

Last year, Elvis had the opportunity to participate in a NASA scientific mission, EMIDSS (Experimental Module for the Iterative Design of Satellite Subsystems), during which he launched the first version of the Trinidad system: an image acquisition module based on a Raspberry Pi Zero 2 W and a Raspberry Pi Camera Module 3, housed in a 3D-printed enclosure. This device was able to reach an average altitude of 42km in the stratosphere, where it successfully captured images (example below) under extreme conditions, including temperatures below -50°C.



◀ The view from the stratosphere

Sensory upgrade

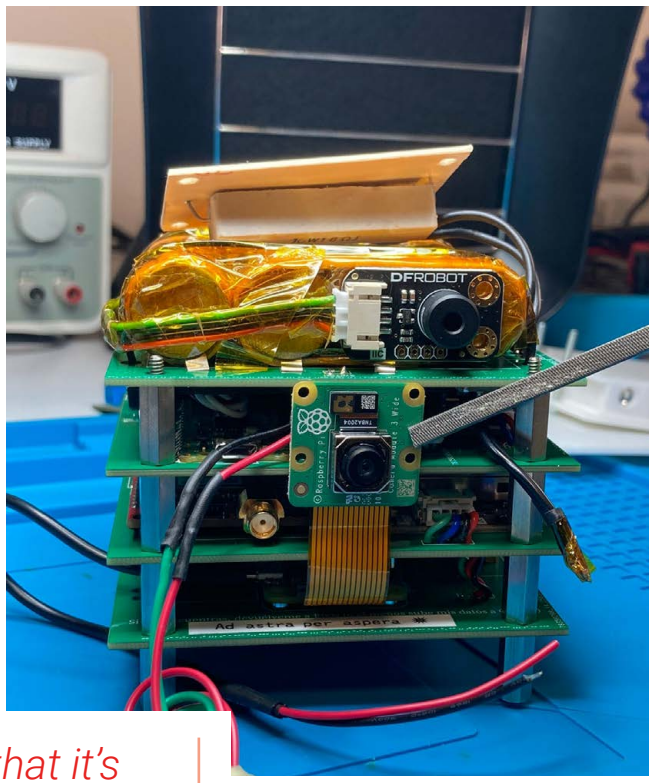
This year, Elvis had the opportunity to develop Trinidad Version 2 – an improved system that integrates new sensors and enhanced capabilities. As well as capturing images, Version 2 can collect relevant environmental data from the stratosphere. This updated version is scheduled to launch, like the first, aboard a stratospheric balloon from NASA's base in Fort Sumner, New Mexico.

Affordable aerospace systems

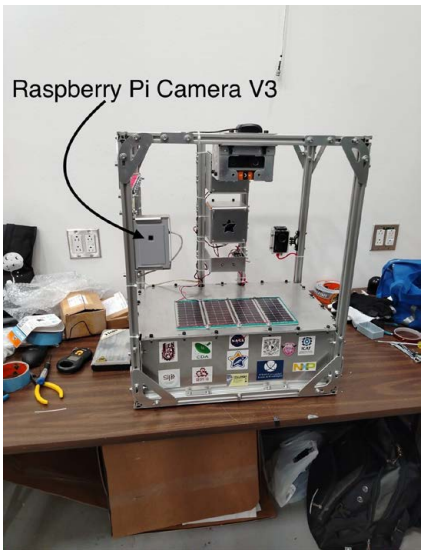
Elvis's goal was to demonstrate that it's possible to create functional, low-cost aerospace systems using commercially available components. It is his hope that systems like Trinidad can become powerful educational tools for universities and schools, opening the door to new opportunities for students and enthusiasts who dream of reaching space.

Cue more CubeSats

If you're interested in more projects like this, we've featured loads of Raspberry Pi–powered CubeSat missions, including GASPACS (Get Away Special Passive Attitude Control Satellite) and its record-making orbit: **rpimag.co/cubesat**.

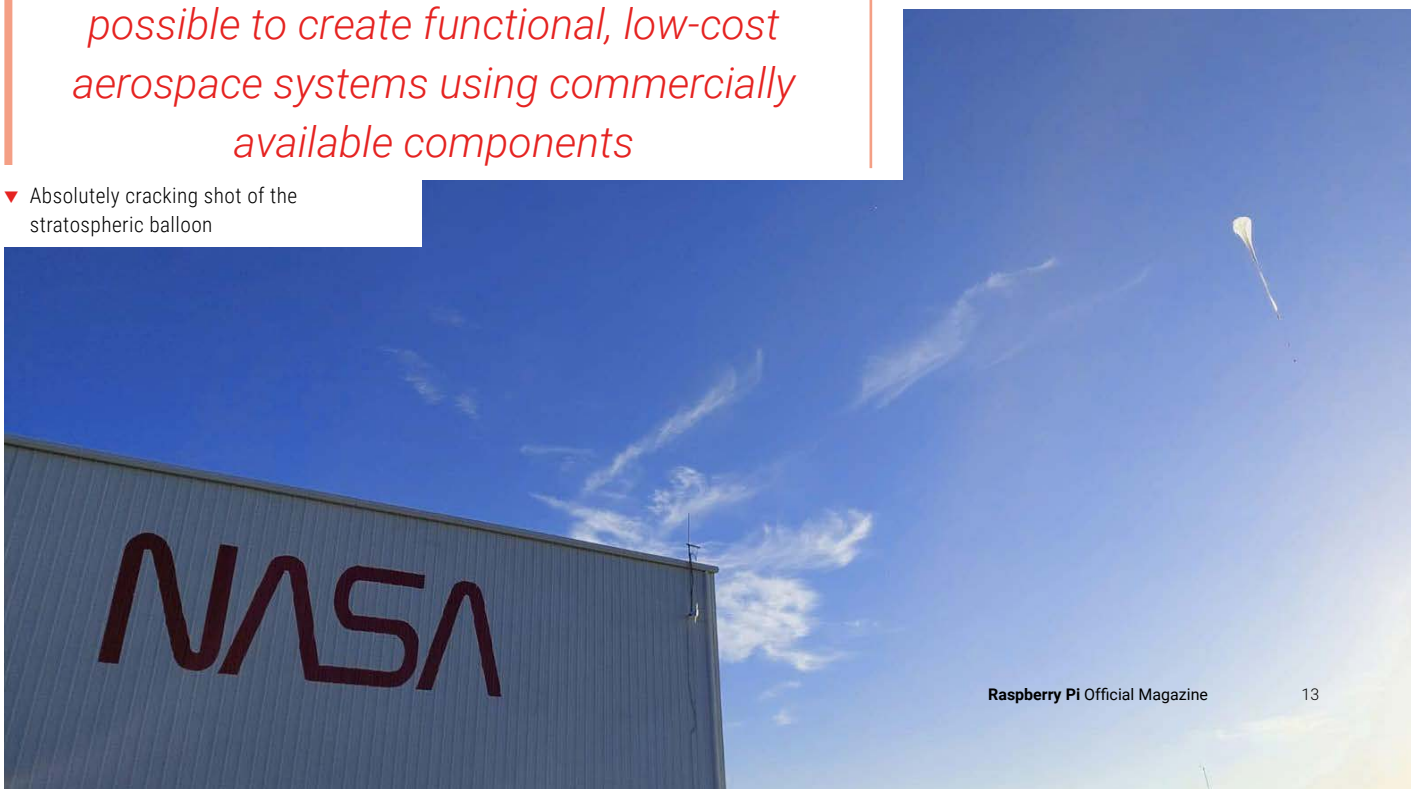


▶ Trinidad Version 2, with added sensors



▼ Version 1 of Trinidad

Elvis's goal was to demonstrate that it's possible to create functional, low-cost aerospace systems using commercially available components



▼ Absolutely cracking shot of the stratospheric balloon

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**

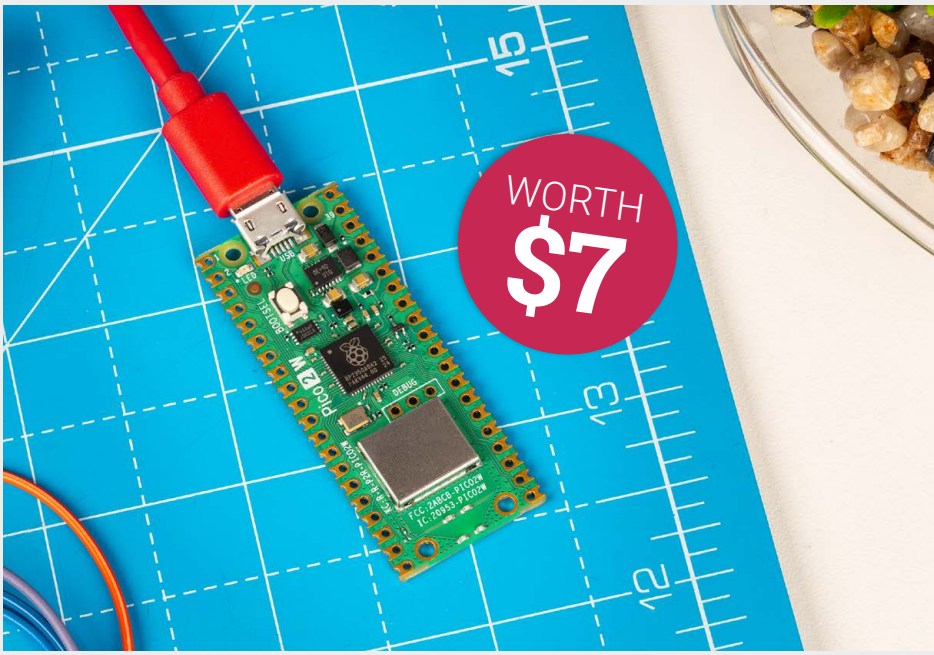
Subscribe for £10 is a UK-only offer. The subscription will renew at £15 every three months unless cancelled. A choice of free Raspberry Pi Pico 2 W or Pico 2 is included with all subscriptions.

SUBSCRIBE TODAY AND GET A

FREE Raspberry Pi Pico 2 W (or Pico 2)

Subscribe in print today and get a **FREE development board**

- A brand new RP2040-based Raspberry Pi Pico 2 W / Pico 2 development board
- Learn to code with electronics and build your own projects
- Make your own home automation projects, handheld consoles, tiny robots, and much, much more



Free Pico 2 or Pico 2 W. Accessories not included. This is a limited offer. Not included with renewals. Offer subject to change or withdrawal at any time.

Buy now: **rpimag.co/subscribe**

SUBSCRIBE on app stores

From **£2.29**

Available on the App Store

GET IT ON Google Play

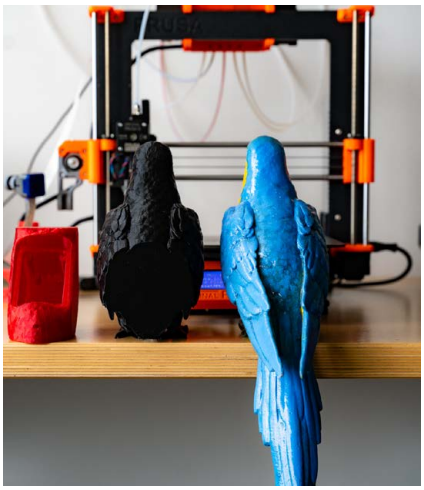
Locoloro AI Parrot

A talking parrot with some Raspberry Pi smarts proved a real pull for a Polish pop-up restaurant, learns **Rosie Hattersley**

Maker
Paweł Skiba
Paweł oversees RapidLab, a Polish Internet of Things design house that often incorporates Raspberry Pi in its creations.

rapidlab.io

Parrots may not be a common sight in Poland, but the city of Gliwice is home to an Ecuadorian restaurant named after the bird ('loro' in Spanish), and its owners thought a talking parrot could make for some effective marketing. The Locoloro street food restaurant sells a fairly traditional range of empanadas and other South American snacks and dishes, which visitors to the city will remember thanks to its distinctive parrot logo. Using Raspberry Pi and AI to bring the iconic bird to life seemed like a fun but memorable way of piquing customers' interest and entertaining them while they queued.

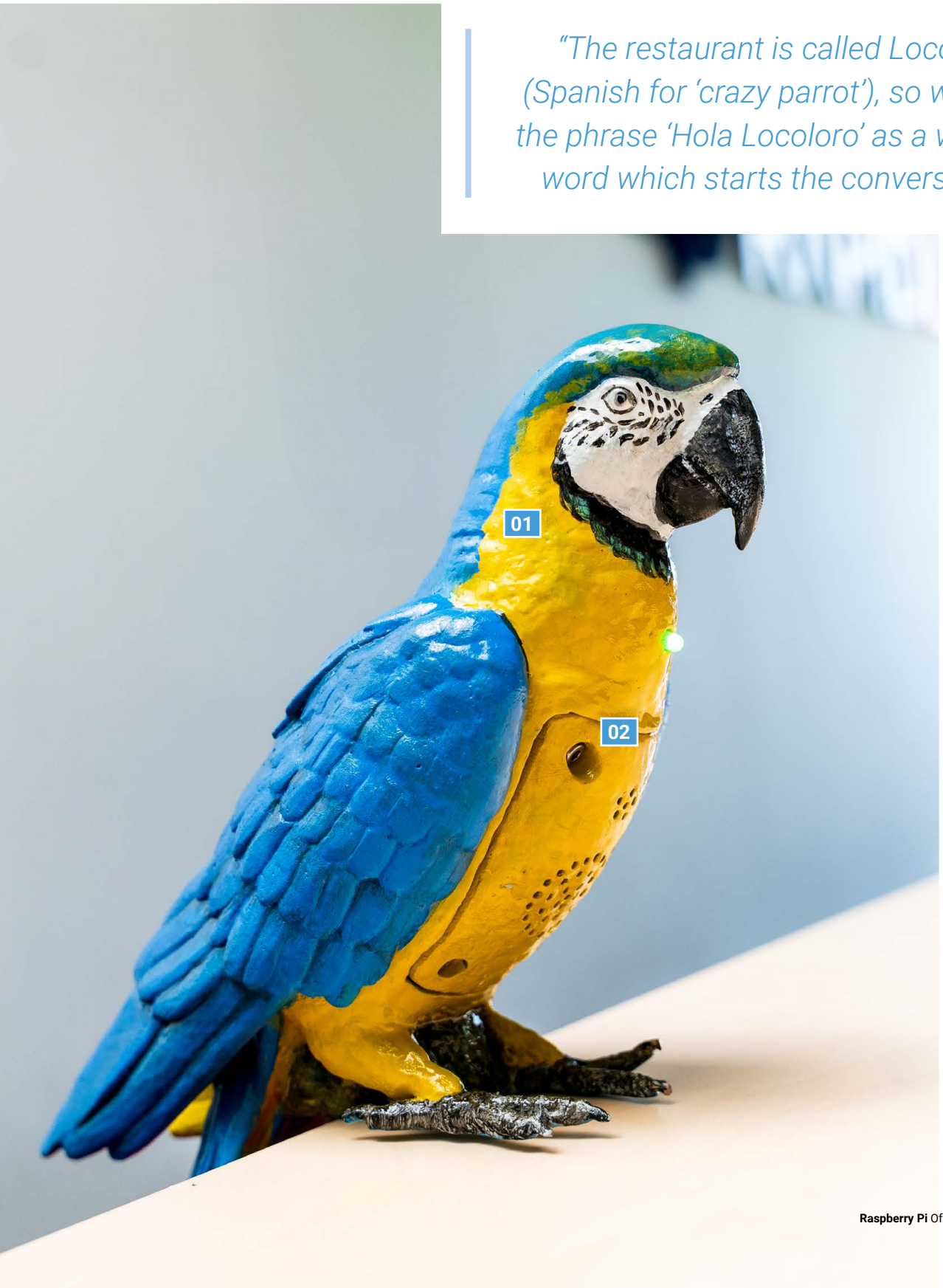


▲ The parrot was made from metal and based on a plastic version from a third-party

Talking point

Gliwice is a small but thriving city in Upper Silesia, Poland with plenty of hospitality options, but a harsh climate. The attractive medieval city attracts plenty of visitors in an area largely known for coal mining and, more recently, chemical and engineering firms. Restaurateurs vie for attention. "Imagine a restaurant serving Ecuadorian food in the middle of a Polish city. In this kind of business, there is a need to distinguish the restaurant among many others," explains RapidLab's CTO and founder Paweł Skiba. "The owner came up with this idea of having a parrot talking to the customers. But how to find a colourful bird talking 24/7, accepting rain and snow while not feeling bad at all?"

"The restaurant is called Locoloro (Spanish for 'crazy parrot'), so we used the phrase 'Hola Locoloro' as a wake-up word which starts the conversation"



01. A metal parrot based on a plastic toy version starts chatting with customers at an Ecuadorian restaurant in Gliwice, Poland

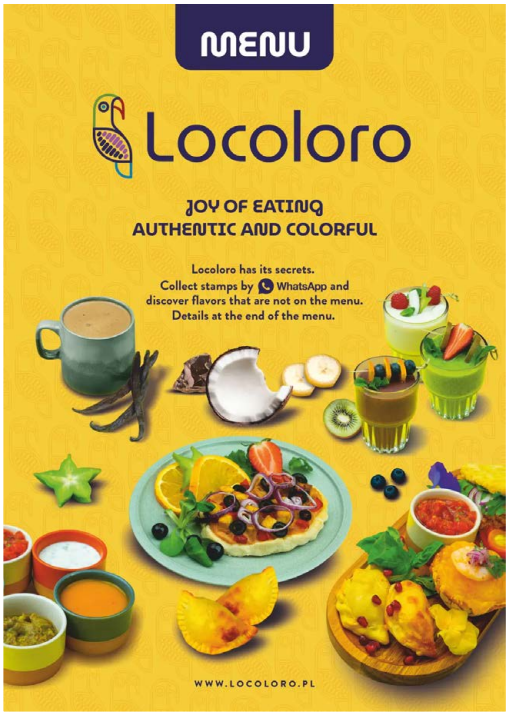
02. Raspberry Pi 5 controls the speaker, microphone and ElevenLabs voice generator, which responds to calling "Locoloro" – crazy parrot!

The Internet of Things design house is based in Poznań, nearly 300km away, but RapidLab's website gives the distinct impression that its engineers and designers are quirky with a slightly madcap approach to projects, so it was little surprise to learn the talking AI parrot project didn't faze them. "A bird in a cage is not something people like to see these days, is it?", states Paweł, rhetorically, whereas "building a parrot-shaped device equipped with a human-like voice is the kind of challenge we always accept."

RapidLab builds custom hardware for companies from startups upwards, with Raspberry Pi a frequent inclusion. Paweł cites the power to quickly deliver high-quality products within a reasonable timeline and acceptable price level, "especially in early stages when time-to-market is a crucial factor for our customers' success".

That's the way to do it!

A vibrant blue and yellow parrot conceals a small microphone attached to Raspberry Pi 5 (prototyping having first been done using Raspberry Pi Zero 2 W and a cloud-based API) that listens out for specific words and phrases and can process everything locally. The RapidLab team created a metal facsimile of the original 3D parrot they'd bought from a third party, enlarging the interior and adding holes so it could accommodate components such as Raspberry Pi and AI HAT+, speakers, microphone, LED, and 5G modem. "It was quite a lot of components even when we were using Raspberry Pi Zero. The box is hidden from guests' eyes anyway," observes Paweł, "so in the final version we decided to move the controller to an additional waterproof enclosure and connect it to the parrot by wires."



▶ The interactive parrot idea links to the Ecuadorian restaurant's existing logo

They then trained the AI parrot using a tiny Edge AI model to avoid data and transfer costs as well as latency issues. "The restaurant is called Locoloro (Spanish for 'crazy parrot'), so we used the phrase 'Hola Locoloro' as a wake-up word which starts the conversation." The parrot then connects to ElevenLabs' voice generator and recites one of several sentences that the AI generates. Switching to Raspberry Pi 5 resulted in smoother operation, and meant the parrot can potentially show off additional features. The final production-version that RapidLab handed over to the restaurant owners features 5G (rather than Wi-Fi) and "made the parrot completely independent".

Locoloro has proven rather a hit, both at RapidLab's HQ and at the restaurant in Gliwice – so much so that the owners are keen to have more interactive animals dotted around their various locations in Poland. Paweł and team are keen to create more chatty characters, too: "We can design a talking humanoid, hedgehog, dinosaur, or a smiling potato if requested."

You can view the parrot in action on RapidLab's blog: **rpimag.co/aiparrot**.

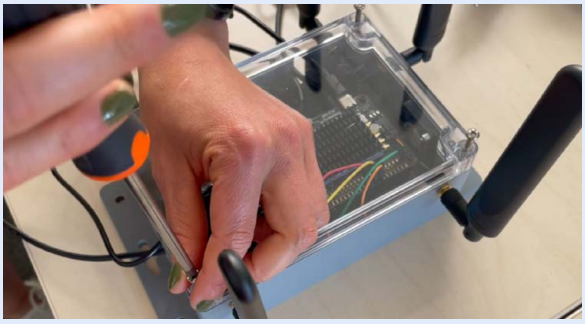
Quick FACTS

- At first glance, Locoloro looks like a standard plastic toy
- In fact, RapidLab made their own, slightly oversized version from metal
- The robust steel bird should be able to repel assaults from unfriendly customers
- The prototype sometimes commented on its makers' conversations
- The RapidLab team rather miss Locoloro's amusing asides

▼ The chatty parrot became a firm favourite with its creators at RapidLab

Small talk

1. Ideally, Raspberry Pi, AI HAT, 5G module, microphone, and speaker, plus wires, and an LED, all need to go inside the parrot (or other animal) case. Don't look too closely where the wires protrude from this one.



2. If your talking parrot is going to be concealed behind some strategic shrubbery, you can house everything inside a case and wire it to the parrot toy.

3. Specify a wake word or phrase that will prompt the interaction with the parrot. You can use ElevenLabs' voice generator to conjure up amusing interactions.

Vertical Runner

A cyberdeck that looks as good in real life as it does in a film script.

By **David Crookes**

Maker

James Reeves

James is an American filmmaker known for his futuristic-hesh style. He is currently pursuing becoming a more versed ethical hacker.

rpimag.co/verticalrunner

Many Raspberry Pi projects are created for use in the real-world. Not Vertical Runner. This one was designed as a film prop for a short science-fiction thriller that James Reeves was writing with his wife. It was set to star a lonely, introverted, skilled hacker looking to access rare cars and it was going to be shot at night.

"I wanted a really strong title sequence showing our main character coding, printing, and assembling and I wanted something unique as her main hacking tool," James says. "It needed to be an extension of the main character and look great on film at night, perhaps in a phone booth, around a corner, or adjacent to a neon sign."

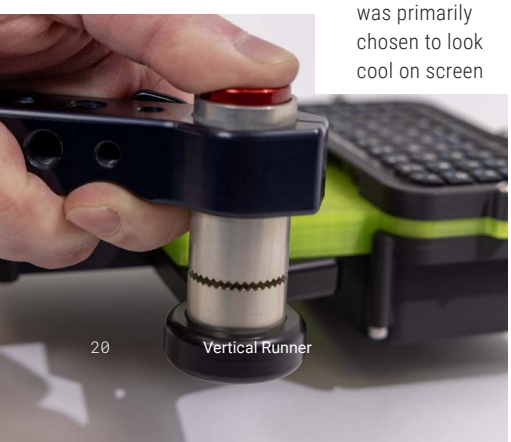
James also wanted his device to double as a principal light source for the character, exposing the main character's face. "It kind of just spiralled from there," he reveals. As time went on, James learned about ethical hacking, cybersecurity, and Linux. The end result was a handheld computer that actually worked.

Hacker training

Initially, James had little understanding of what a hacker would do. To learn, he built a hacking lab using two Raspberry Pi 4 computers – one acting as an "attack" machine and the other as a "victim". He played around, getting comfortable with Raspberry Pi OS before hitting upon a configuration.

The Vertical Runner, he surmised, should comprise a Raspberry Pi 4 8GB, a Waveshare 3.5-inch touchscreen LCD, a Fly Way Bluetooth 3.0 keyboard, and a BrosTrend Dual Band Wi-Fi adapter, all powered using a Vilros battery pack. "It also needed to be an extension of the main character – someone agile, flexible, determined, resourceful and punk," James says.

For this reason, he mixed practicality with aesthetics. "The screen had to work as a key light," James explains, figuring a vertical screen would work best. "This orientation lent itself to the lighting plane shape I wanted for the main character's jaw line," he adds. But while it also looked cool and futuristic, it wasn't without its difficulties.



▼ The aluminium grooved grip with threaded holes was primarily chosen to look cool on screen

James Reeves working on a night shoot for one of his films

> *The main challenge was getting the entire screen to correctly display vertically*

"The main challenge was getting the entire screen to correctly display vertically," he explains. "I could get the display to work, but it was only displaying a 4:3 aspect ratio regardless of the orientation and it would not fill in the bottom 200 lines of pixels. Finally, I just decided to try my own lines of code for the **config.txt** file and after about two hours of doing that, I got it to work."

Quick FACTS

- The device was designed as a film prop
- James wanted it to be agile, flexible, and punk
- It evolved into a fully fledged cyberdeck
- It doubles as a key light for the main character
- A strap enables the device to be easily worn

Punk aesthetic

James then sought to perfect the look. "My friend, Michael, owns and runs a 3D printing business called Precision Additive and we started putting concepts together," James says. "We went through three iterations before the final one and added an unusual grip which I salvaged from a rotational handheld camera rig."

The grip was included to allow an actress to use the device in different positions. "It could be rotated for use as a self-defence weapon too," James says. "The handle also allowed me to use small lighting devices or car mount systems." By producing a real cyberdeck, James was able to create something believable.

"I still intend to use it a film prop, but I've created a fully-functional penetration-testing mobile computer which works fantastically well," James says. "It could have a better Wi-Fi monitoring/injection adapter, but I'm really happy with it. I'm now looking forward to starting the build of another cyberdeck."

01. A dual-band Wi-Fi dongle (with antenna) allows the cyberdeck to run in monitoring or injection mode

02. The screen is mounted vertically and connects to a Raspberry Pi 4 8GB computer

A honeycomb backing is used to help prevent the device from becoming too warm

Instant Camera

Print stickers in a flash with this photogenic camera. By **David Crookes**

Maker

Spacerower

Spacerower is a maker behind some great-looking, minimalist-designed Raspberry Pi hardware including the Compad V2 cyberdeck.

rpimag.co/instcamzero

Warning!

Burn Risk

This project includes a thermal printer (EM5820) which becomes hot during operation and may cause burns if touched immediately after use.

The first ever commercially available instant camera was released in 1948. Created by Edwin Land, it was manufactured by Polaroid, sidestepping the need to spend ages processing traditional film in a darkroom.

At first, the Land Camera would output sepia-tone prints. Two years later, it was printing in black and white and, by 1963, it was surprising consumers with colour. But even though the popularity of instant prints waned in the 1980s, there's still no mistaking the joy of being able to see an image form before your eyes in double-quick time.

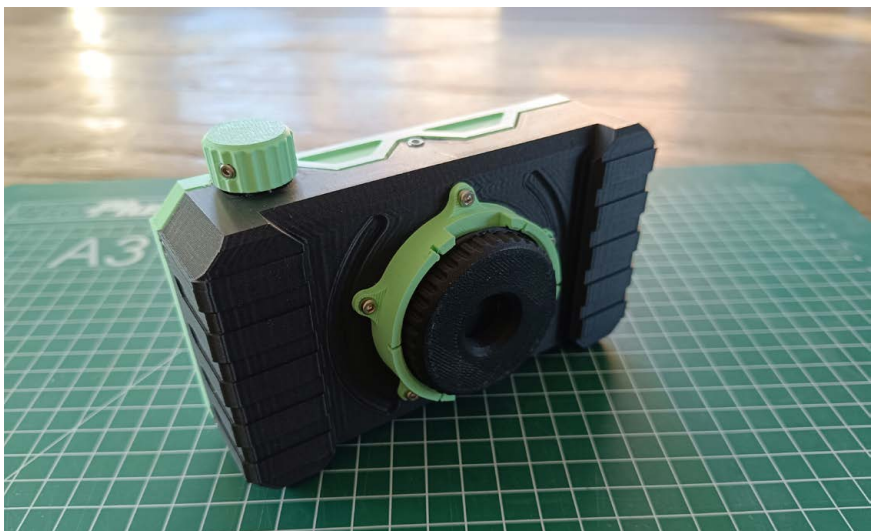
It's why, despite having a pick of amazing digital cameras – including those on phones – this project is proving so eye-catching. Not only does it dazzle with its clean, colourful design, it allows photographers to take quick snaps, preview the shots, and print them onto sticker paper in much the same manner as an old Game Boy camera.

Flash to the future

Produced by maker and developer Spacerower, who has asked us to use this Reddit username, the Instant Camera was apparently a difficult build. That said, the components are straightforward enough.

Aside from including a Raspberry Pi Zero computer, the project incorporates a Raspberry Pi Camera Module and a 240×240px LCD screen, along with the all-important thermal printer capable of printing the stickers. The project also includes a 1200mAh 2S lithium-ion battery, some LEDs, an assortment of power circuits, and magnets

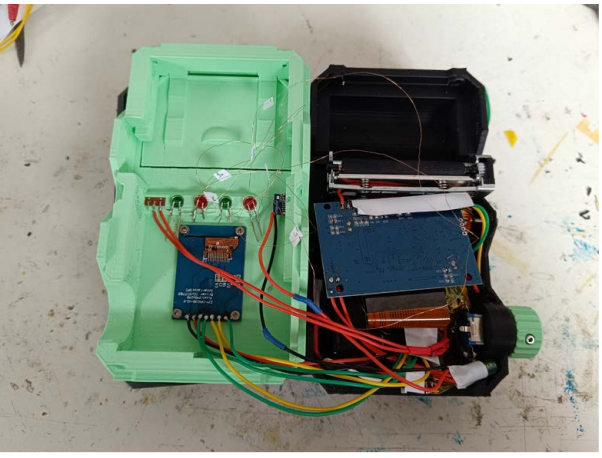
For the thermal printer, Spacerower chose the relatively inexpensive EM5820. It has a 60mm/s print speed and, as expected, prints in black and white without the need for an ink cartridge or



▶ You can find the 3D printer files at **rpimag.co/instcamzerostl**

01. The inexpensive LCD screen allows the user to view a preview of the images

02. Fitted over a rotary encoder, the multifunction knob controls the camera



◀ Spacerower says the wires and PCBs only barely fit inside the case

toner. "It can be quite easily interfaced, either through USB or serial," the maker says. "These printers are also quite nice for other projects: when you take them apart, they consist of just a PCB and the printer module, and are quite compact."

All of the components are encased in a 3D-printed outer shell, printed using PLA (Polylactic Acid). For good measure, Spacerower has added a way of protecting the Camera Module by rotating the black lens, inspired by the Peep Hole Cover created by Tjsangster on Thingiverse (**rpimag.co/peepholecover**). It looks as good as a product you'd buy in a shop.

> *It looks as good as a product you'd buy in a shop*

Image processing

Bringing the project together is code written in Python which takes input from a green button positioned on top of the camera. "A short press on the green button takes a picture," Spacerower says. "By rotating it, you can adjust the brightness of the picture, and a long press prints the picture onto the sticker paper."

Images are captured using the Picamera2 library, and Spacerower has used Pillow – the successor to the Python Imaging Library – to crop, rotate, and convert the pictures into black and white. By using the Python ESC/POS library to connect the printer, nice crisp images can be committed to the thermal sticker paper.

There is some scope for improvement, though. "Image processing can be slow," Spacerower admits, noting that a more powerful processor would have helped. Even so, it's much faster than the traditional method and a heap more fun as well.

Quick FACTS

- The camera prints onto thermal sticker paper
- Photos can be adjusted before printing
- It makes use of Python-based libraries
- The build cost less than £100
- The 3D printer files are available to download

Ubo Pod AI assistant

Slick smart home options paired with voice controls in a powerful Raspberry Pi package impress **Rosie Hattersley**

Maker
Mehrdad Mazjoobi
Mehrdad is an open source evangelist specialising in voice-based AI devices, often featuring Raspberry Pi.

rpimag.co/ubopodks

As Google, Facebook, email clients, and any number of persistently connected devices hoover up our details and interactions while 'summarising' and providing 'assistance' crafting responses, many of us are keen to safeguard access to our innermost thoughts, dashed-off notes, and utterances. A device that provides centralised control over a customisable range of smart devices, and can be voice controlled but, crucially, doesn't share its owner's data, has clear appeal. Mehrdad Mazjoobi believes he's come up with just such a device – or one with potential to offer such tools – and recently launched a Kickstarter to support its further development after garnering interest at various trade shows over the past two years. Ubo Pod is "an open-source, hackable, multi-modal personal AI assistant" primed with a 5MP camera, 32GB to store data locally, and an LED status ring. A microphone and speakers are incorporated for spoken commands and interactions – conversational AI. Popular Raspberry Pi smart home apps can be added via an API.

Hands-off approach

A decade ago, almost every new product we came across seemed to be touch-enabled. The coronavirus years put paid to many touchy-feely devices, but advances in speech recognition have brought hands-free, voice-based control to the fore. With a PhD in electrical and computer engineering focusing on hardware and security, Mehrdad has been developing voice-based products for the past decade. He built BitLock, a Bluetooth bicycle lock – crucially, also operating hands-free – as well as a Raspberry Pi-based, contact-free elevator control panel which he developed while working at a voice AI startup during lockdown.

Ubo Pod came about because Mehrdad is "personally concerned with the amount of data we are feeding large AI companies with little understanding of its impact" and believes users should have primary ownership, control, and protection over their personal data, including chat histories with AI agents and models trained on it. Mehrdad is proud that Ubo Pod doesn't share details of your searches and activity with 'carefully selected' and supposedly trustworthy commercial entities.

More oomph, please

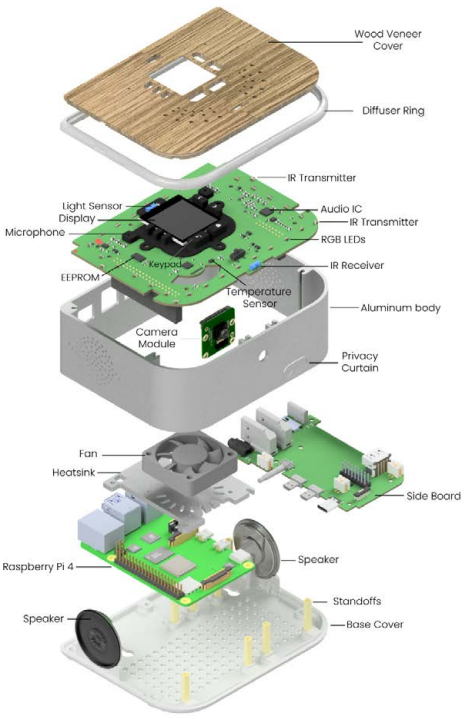
Looking for all the world like a fancy radio with its curved wood chassis, Ubo Pod sits unobtrusively on a desk, runs on either Raspberry Pi 4 or 5, and includes a "collection of interconnected open-source subsystems and libraries" that combines

01. Ubo Pod is a 'user-hackable' voice-controlled AI assistant and Internet of Things (IoT) hub

02. It features Raspberry Pi 4 or 5 and an on-device AI accelerator, keeping user data private

03. A demo video shows how Ubo Pod can control a headless Raspberry Pi: **rpimag.co/ubopodkvm**

Mehrdad is proud that Ubo Pod doesn't share details of your searches and activity with 'carefully selected' commercial entities



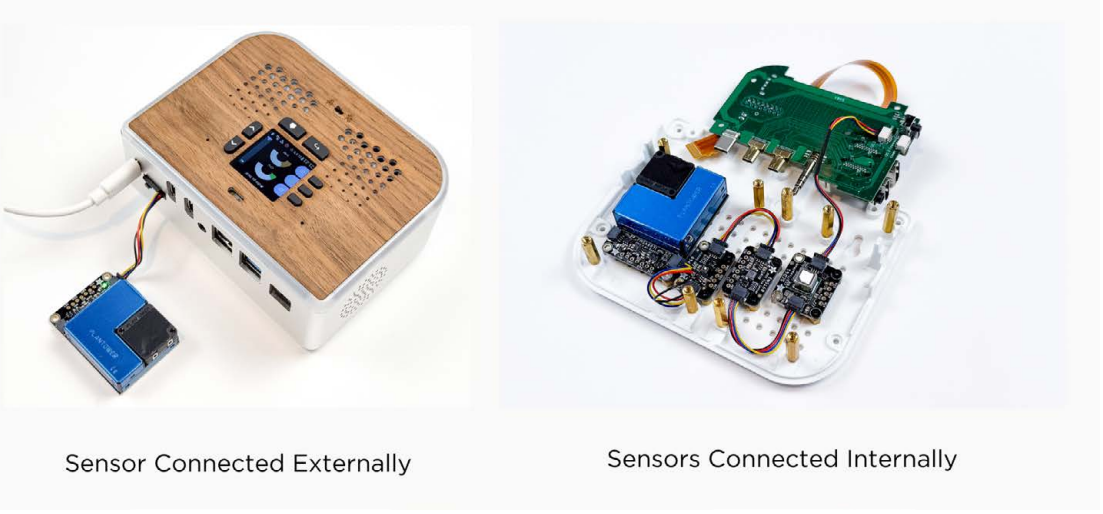
▲ The Developer Edition can accommodate plenty of hardware add-ons

smart home integration with useful voice-activated features. It comes pre-installed with open-source 'conversational AI' and vision features, to which owners can add apps and custom features using gRPC (remote procedure call) APIs. While hardware nerds will enjoy this, Mehrdad intends to focus on developing an even more polished experience for average consumers.

Mehrdad chose Raspberry Pi as the basis of Ubo Pod due to "its popularity, modern kernel/software, driver support, backward compatibility, long-term availability, extensive documentation, and community support", plus the ease with which he can update to newer versions thanks to the modular design. Mehrdad acknowledges the relentless drive towards more processing power, "especially in the LLM era" (large language models underpin machine learning, on which AI is based). "Raspberry Pi 5 with 16GB will be a great option for running larger models, but Raspberry Pi 5 with 4GB RAM will still be able to run most 3B models (such as Gemma 3 1B)," he states. Support for OpenWakeWord and an API for Porcupine by Picovoice for local wake word detection will be added shortly.

The original design was little larger than a Raspberry Pi, but Mehrdad eventually enlarged the case to accommodate full HDMI connectors, rear ports and a more polished wood veneer design. The aesthetic came about because the original 3D-printed design "looked cheap", prompting Mehrdad to cover it up with a walnut exterior that then became central to Ubo Pod's design. It also gave him a bit more wriggle room to accommodate components and extend Ubo Pod's potential functionality. A multi-camera setup with the option to add a CSI camera and processing video streams from up to five sources at once, all connected via Raspberry Pi, is currently being tested. This will use a Google Coral or Hailo AI accelerator along with Moondream to add context to the image analysis.

Because it is open source, you can even contribute to Ubo Pod's design and features: **rpimag.co/uboapp**.



◀ Ubo Pod originally had a five-way joypad, but individual buttons proved more intuitive

▼ Mehrdad's voice-controlled elevator panel invention was a boon during the hands-off pandemic years

Quick FACTS

- Mehrdad spent lockdown designing hands-free, voice-activated setups
- These included a voice-controlled Amazon locker and barcode reader
- He now works for voice-control developer Picovoice
- He calls his zeal for AI privacy "ambitious but worth the shot"
- He always travels with an Ubo Pod "as a sort of nerdy travel buddy"

Build an Ubo your way

1. You can use your own case, components, and power supply or buy a predesigned kit from Tindie which also includes a custom circuit board, heatsink, 5MP camera, and cables: **rpimag.co/ubocase**.

2. STEMMA sensors, an additional front-facing camera, AI Hailo accelerator HAT, and other custom components can be easily added to the Raspberry Pi 4- or 5-based Ubo Pod setup.

3. The LCD details connected devices. The GitHub repo shows how to set these up, add a custom circuit board, and has a video guide on running the Raspberry Pi headless: **rpimag.co/ubopcb**.

Bog Body theatre prop

It's not every day a maker emails **Rosie Hattersley** with a tall tale about Raspberry Pi bringing a mythical creature back to life

Maker

Sean M Tracey

Sean runs a company that claims to be able to build anything – often with the help of Raspberry Pi.

mitchelltechnologies.co.uk

rpimag.co/bogbody

We last encountered Sean Tracey engaged on an international trade mission somewhere between London and San Francisco, stealthily trying to set up a Raspberry Pi cluster computer in an anonymous-looking briefcase with the aid of some rather whiffy spray paints without attracting the attention of the conference hotel staff. This issue, he's left the latex special effects to the theatrical pros while he takes on the task of animating a fake 1000-year-old corpse with a little help from Raspberry Pi.

Deadly serious

Sean helped attract attention to a previous company he worked for by animating a cutout of Elvis Presley and placing it in the office window, where it would follow people as they walked by and block their view of what was going on inside. Whether this had any bearing on his recent project is a moot point, but earlier this year Sean was approached by a theatre company about building a life-sized animatronic mummy that could rise up from a table unassisted. It would need to reach out for an actress on-stage in what, quite evidently, was going to be a horror-themed play. The titular bog body was to be a life-sized animatronic mummy that could rise up and then reach out for an actress on-stage.

> *Sean was approached by a theatre company about building a life-sized animatronic mummy that could rise up from a table unassisted*

The project wasn't as easy as it might sound, says Sean, "especially when there are so many unknowns, which is how things tend to be when working in theatre productions". Practical issues included how much load certain types of motors could take and the materials that would make up and support lifting a full-sized

01. Bog Body showcases the folklore behind corpses preserved for centuries, with the twist that this one appears to be alive

02. Two stepper motors are controlled wirelessly by 8GB Raspberry Pi 5. They enable the latex-clad skeleton to silently sit up and reach towards the actress

latex-covered plastic skeleton weighing around 8kg from resting to bolt upright in less than 30 seconds. The whole thing also needed to be collapsible (for ease of transportation and storage) and able to be up and running on stage in less than ten minutes.

Having used Raspberry Pi in approximately 80% of his previous creations, Sean immediately knew his best bet would be using one to run some custom Python code that could hook into the theatre's systems sat at the heart of the project. "Honestly, I don't think it would have been possible to build this kind of project without it."

Use your illusion

Being "a big believer in having more capacity than you think you might need", Sean set up Raspberry Pi 5 (8GB) knowing it would provide reliable networking capabilities, could operate silently – critical for maintaining the eerie atmosphere of the play – and could easily drive two motors via a stepper motor HAT, With reliable networking capabilities, and the ability to drive at least two motors simultaneously. "It's always better to have more power and not need it than to need it and not have it!"

Importantly, Sean remembered the Elvis project and corrected an error. The cutout was supposed to move in lock-step with the curious passer-by, but precision of movement that stepper motors offer came at the cost of speed. "You could set the motors to move an exact amount, but at no more than a few revolutions per second." Sean resolved this with some clever gearing and C++ coding. "For the mummy, precision over speed was exactly what I needed, so having almost the exact same motors I used a decade ago powering the raising of the mummy was perfect!

Libby Morris, who Sean describes as "an incredible London-based horror artist who's built more terrifying things than I could ever imagine!" – built up the body's latex skin in layers to look as though it had deteriorated over a long period of time. The two of them had less than a month to bring the bog body to life, complete with 14 3D-printed parts and counterweights carefully geared to control the speed at which it rises up and extends its finger.

As well as writing code to control the bog body, Sean had to work out how to connect everything and make it work with the Theatre Control Panel (TCP), a popular piece of software used in theatres for controlling sound, video, and lighting cues. "Thanks to the huge ecosystem of Raspberry Pi HATs and software packages, bringing all of the parts together that we would need to bring the mummy to life, it took less than two days after we had the work package signed-off," he says.

Bog Body opened to much acclaim at Camden Fringe Festival in August and has since been on tour in the UK.

▲ Actor Jen Tucker encounters the bog body in the dress rehearsal

▼ Bog Body cast and prop on-stage at Camden Fringe Festival

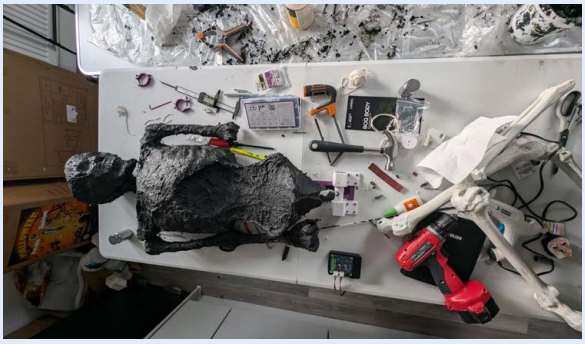
▼ Sean shows off the cadaver before it's hidden under a sheet until the very end of the play

Quick FACTS

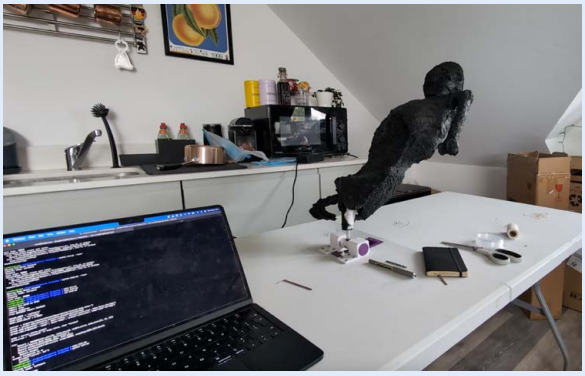
- Bog bodies are naturally mummified and preserved, often in peat bogs
- Several dozen have been unearthed in the UK and Ireland
- The pH levels of bogs effectively pickle the bodies, preserving them
- Sean's bog body is animated using counterweights and stepper motors
- Raspberry Pi 5 controls them in a similar fashion to a grandfather clock

Looking lively

1. Theatrical horror artist Libby Morris covered a life-size plastic skeleton in latex skin. Sean then drilled holes where it would need to pivot so it could be raised.



2. A steel bar hidden beneath the body and attached to geared counterweights causes the bog body to smoothly and silently sit upright and fall backwards.



3. Raspberry Pi 5 controls the NEMA stepper motors remotely using an API Sean wrote for the project. Once the play has finished touring, he hopes to post the code for animating mummies on his website.

Flight sim radio

This is Romeo-Papa-Oscar-Mike-One-Five-Eight, come in virtual control, **Rob Zwetsloot** speaking

Maker
Leo
An A-level student with a passion for computers and aviation.

Way, way back in issue 141 (rpimag.co/141) we featured a fun, very low-cost project by maker Leo where he used recycled cardboard and a Raspberry Pi Pico to create an analogue throttle for flight sims. It was simple but effective. Leo is now back with something a bit more complex – a switch panel for controlling the radio in flight simulators.

"I thought it would be really cool to have a physical interface that mimics real aircraft controls for various simulator actions," Leo says. "Off-the-shelf solutions were very expensive, so I knew I'd have to make my own... Originally, I was planning to use a wooden box from a local craft shop, but after seeing an autopilot panel shown by VIZIX on YouTube (**rpimag.co/autovizix**), I saw that 3D-printing the panel can result in a tidy surface finish by printing the panel face-down on the bed, while allowing me to add all the slots, holes, and mounting points I would need for the various components. This approach was successful as the Creality Ender 3 V3 SE I was lucky to receive at Christmas last year did a great job."

> *I thought it would be really cool to have a physical interface that mimics real aircraft controls for various simulator actions*

Open-source sim

The system is designed for X-Plane 12, a highly detailed flight simulation that you can connect to with MobiFlight, open-source software that lets you connect to flight simulators so you can add custom – extremely custom – hardware using microcontrollers over USB.

"Once the parts arrived, I set to work prototyping a mount for the seven-segment displays in FreeCAD," Leo explains. "I knew that I needed six digits exactly in each frequency window on the panel, but the displays are eight digits long, so I would hide the extra two digits

01. All the flips and switches have a function

02. The rotary encoder has two modes – push it in for more 'coarse' frequency skipping

outside of the window. As I was going to screw the displays into the back, I did try a method where a 3D-printed peg would friction-fit into the hole instead. Because of the orientation I had to print the pegs, however, they were too fragile to be viable and could break during assembly. With the final panel printed, I started to add the components, testing them as I went using the FOSS MobiFlight Connector to interface with the simulator."

For this build, Leo used a single rotary encoder instead of the dual concentric encoder on a real aircraft radio – the dual ones are much more expensive, and pushing down the encoder switches between the coarse and fine-tuning functions.

Quick FACTS

- Leo's grandfather introduced him to flight simulators
- The project uses a Raspberry Pi Pico 1
- MobiFlight also works on Microsoft Flight Simulator
- Autopilots are often referred to as MCO or FCU
- You can create entire cockpits for flight sims

▼ The radio replicates the functions of the in-game one, without needing to move the mouse

Dial it in

"The two seven-segment displays show the active and standby frequencies, just like the real aircraft," says Leo. "The rotary encoder changes the frequency in 5kHz steps. Holding down the encoder while turning changes in 1MHz steps. The two square 'KD2' push-buttons select either of the two radios in the virtual aircraft... The switch between the windows can be pushed left to swap the standby frequency and the active frequency, and pushing it right selects a preset frequency. The switch marked 'POWER' changes the display and KD2 brightness between two levels – when the panel was designed, this was not the plan, hence the switch isn't marked 'BRIGHTNESS'."

According to Leo, all the components are hooked up to GPIO pins on a Raspberry Pi Pico, then configured via MobiFlight: "As I'm using X-Plane as a flight sim, this happens via the DataRef and Command systems. Because I'm using MobiFlight, I can easily adapt this to various other major simulators."

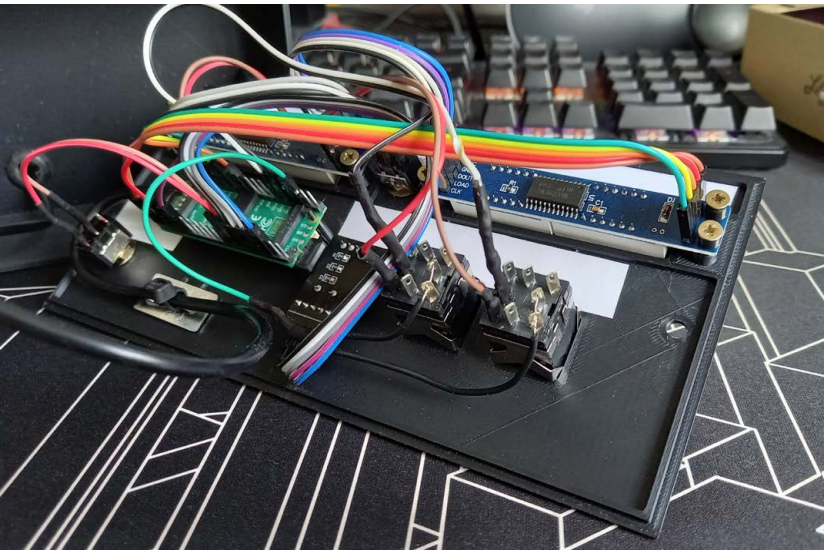
Autopiloted

While the encoder is a bit finicky, the full project works very well, and seems to have given Leo more ideas for projects:

"I plan to make an autopilot panel using the same hardware," he tells us. "I hope to make this either the same width or the same height so they will sit together nicely. I might also try and set this up with FlightGear, an open-source flight sim."

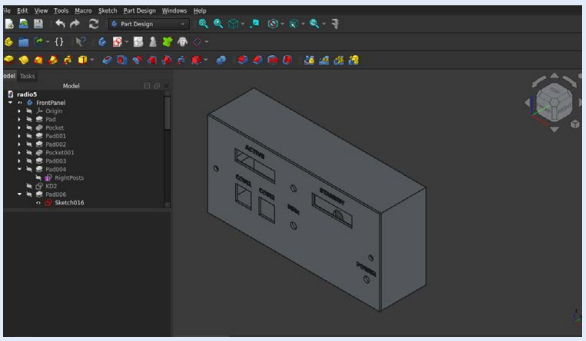
And that cardboard throttle? We might see a 3D-printed one in the future.

▼ The cabling is very tidy, and you can see a Pico H connected to the various components

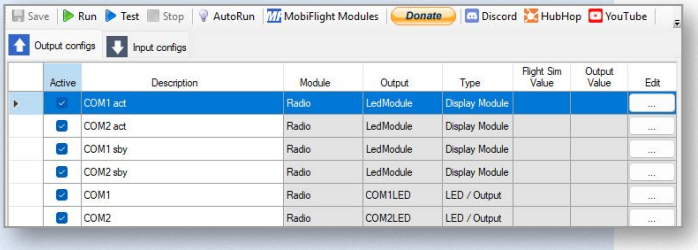


▶ The cardboard throttle was an ingenious and simple solution

Customised controls



1. Designing a box in KiCad is very easy – you can even do simple boxes in Tinkercad if you want to do it cheaply for 3D printing.



2. MobiFlight allows you to configure the inputs for specific GPIO pins so they connect straight to the flight sim without having to create fake keyboard outputs.

3. Now you can control the part of the aircraft without mouse controls, leading to a more tactile experience.

Yashica IR camera

This brave build uses technology to take features away. By **Andrew Gregory**

Maker

Malcolm Wilson

A tinkerer since he was a teenager, Malcolm now works in the automotive industry.

rpimag.co/camerahacksmj

In the olden days, we'd point our cameras at scenes, press a button, then have to wait to get the film developed so we could see if the photos we'd taken were any good. The process was slow, and it forced the user to be selective and deliberate with what they were doing. Nowadays our cameras are our smartphones, and we hardly even look at the photos we take, still less print them out and have them around the house.

One maker who wanted to go back to the way things were is Malcolm Wilson. Malcolm's a prolific camera hacker, adding 3D-printed bits, Raspberry Pi Camera Modules, and copious amounts of Python code to get things to work just so. This is his latest build: it's based on a Raspberry Pi Camera Module 3 NoIR, and it's just the thing if you like taking weird infrared photos that make it look like you're wandering in a post-apocalyptic landscape. "[Photography] was a hobby that I picked up during the pandemic to stay creative", says Malcolm. "I started out shooting mainly film and then branched out into modifying cameras. I've always loved the look of Aerochrome film and wanted to experiment with different ways of achieving the same look without spending thousands on a converted speciality camera."

Quick FACTS

- Malcolm took up photography as a hobby during the pandemic
- No cameras were harmed in the making of this project…
- … as the donor camera was already broken when Malcolm acquired it
- This build uses a Raspberry Pi Zero 2 W and a Raspberry Pi Camera 3 NoIR

◀ The changing colours of autumn leaves look great under IR light

▶ It's perfect for making moody, melancholy montages

▼ This build requires a variety of skills to get right

It was my first time using the Camera Module 3 and I was pleasantly surprised by the image quality and the speed of the autofocus

01. This camera is built into the body of a Yashica Electro 35 film camera – as used by Peter Parker in the 2012 film, *The Amazing Spider-Man*

02. The screen on the back tells you how many photos you've taken… and not much else

"It was my first time using the Camera Module 3 and I was pleasantly surprised by the image quality and the speed of the autofocus. I wasn't expecting it to work so effectively as a point-and-shoot with minimal tweaks," he says.

Keen-eyed readers will notice that there's something missing compared with most camera builds: a screen. There's a viewfinder, and a little OLED that tells you how many shots you've taken, but most cameras show you what the photo is going to look like before you take it. Not this one: instead, you capture the moment as the muse directs you, turn the camera off, then turn it back on when you get home. Its Raspberry Pi Zero 2 W then connects to your home Wi-Fi, enabling you to SSH in to retrieve your images.

The electronics of this project comprise the aforementioned Raspberry Pi Zero 2 W, Camera Module 3 NoIR, 720nm IR lens filter, a rocker switch for turning the camera on and off, a momentary push-button, LiPo battery with 5V buck converter, and an I2C OLED screen. Assembly takes some doing: Malcolm says that anyone willing to try this project needs to be comfortable with disassembling a film camera, cutting metal, soldering, writing and editing Python code, and 3D printing – there's a 3D-printed mount for the Raspberry Pi Zero 2 W within the camera because, as Malcolm says, "It took me awhile to get the right orientation to fit all of the components in the camera body."

There's enough battery life to handle a long walk in the woods, and plenty of storage – "I've been able to get 2–3 hours of use; I haven't done a formal test," says Malcolm. "I didn't put any restrictions in the code for the max number of images. The only limit is the size of the internal memory card."

So, make your own (or something similar) and get out into your nearest forest to take pictures of the autumn leaves – they look amazing in IR.

Kilopixel

By Ben Holmen

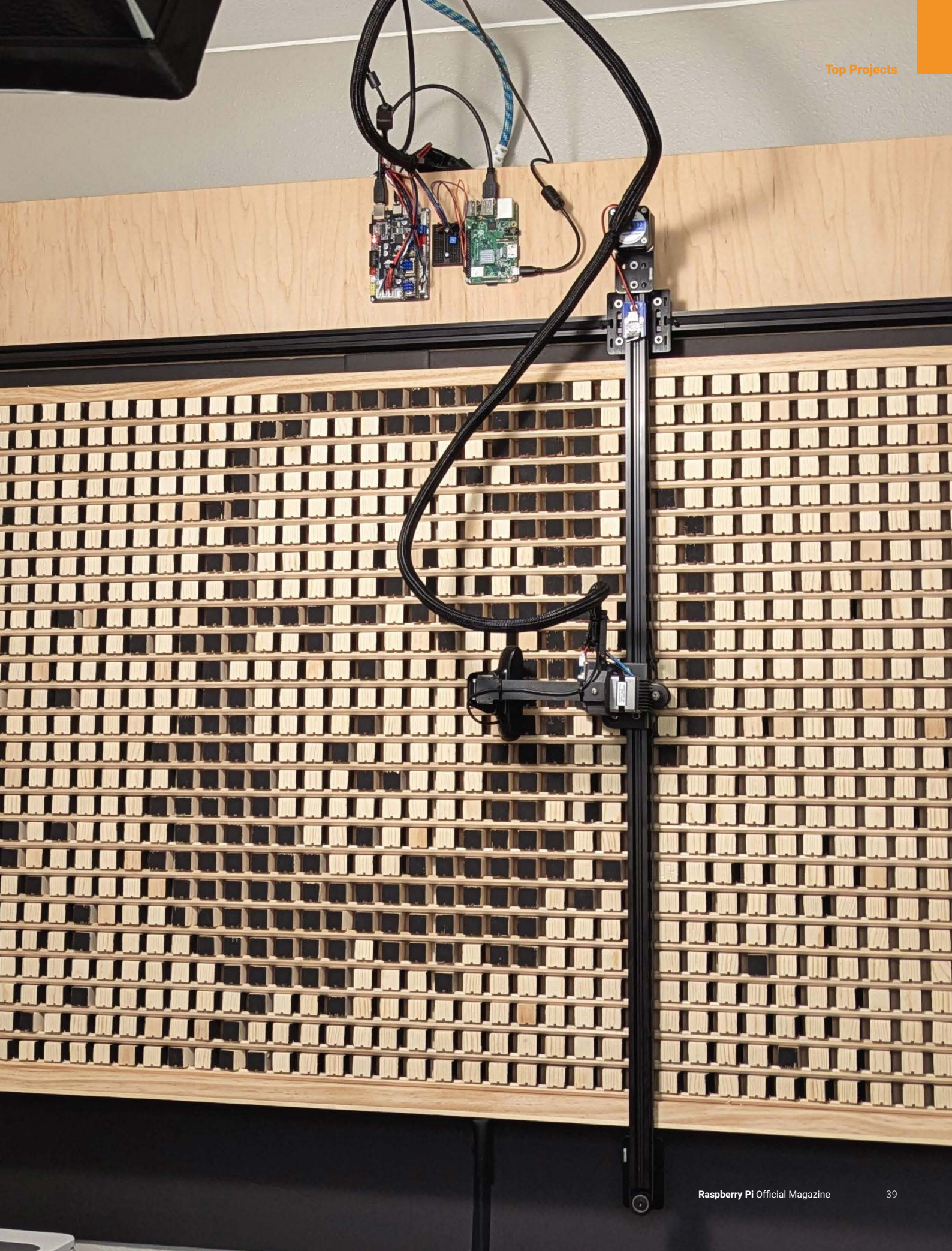
rpimag.co/Kilopixel

Here's a project that has it all: loads of physical building, G-code, sensor data flowing into a Raspberry Pi via GPIO, and a Python script running on Raspberry Pi that queries an API to determine which pixels to move. Plus, it's enormous, and the aforementioned API takes input from the internet, so that humans on the other side of the planet can control what image is displayed. No wonder it took Ben Holmen six years to build.

This 1,000-pixel wooden display is, as Ben says, "inefficient". But that's exactly what he wanted to make, inspired by similar projects including a movie player that updated at 24 frames per hour. This display comprises a grid of 40 × 25 pixels, updating each pixel ten times a minute; compare that with a modern display with millions of pixels updating 60 times a second and you'll appreciate that the aim here is not crisp accuracy, but something far more interesting.

Ben describes the Kilopixel as "essentially a 2-axis machine that uses the third axis for the pixel poking mechanism... I connected a Raspberry Pi to the CNC controller and use it to query my API to get the next pixel, writing the appropriate G-code to get there, activating the pixel poker, reading a light sensor to determine the physical state of the pixel, and returning the state to the API. It does this in a constant loop. This is run with a Python script and depends on pigpio to read the light sensor over GPIO pins."

▶ If you want to play with the Kilopixel, go to **kilopx.com**



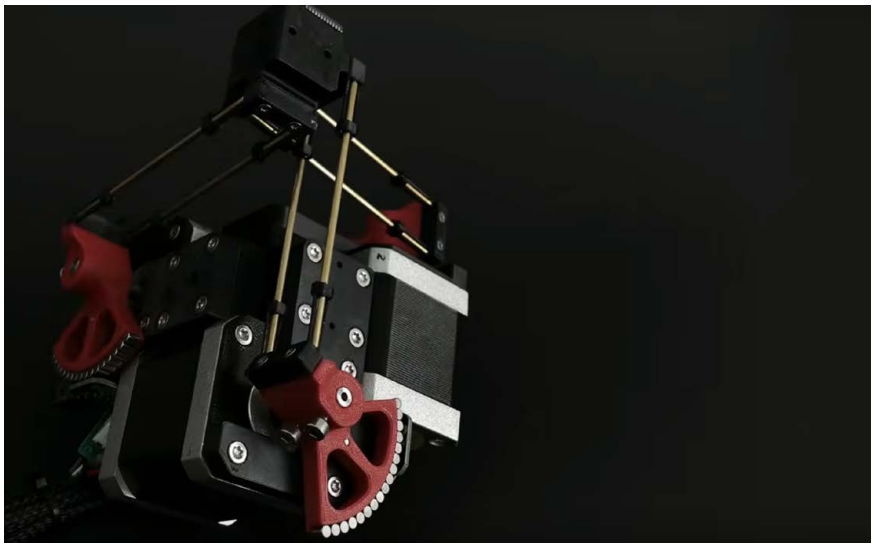
Micro Manipulator

By Diffraction Limited

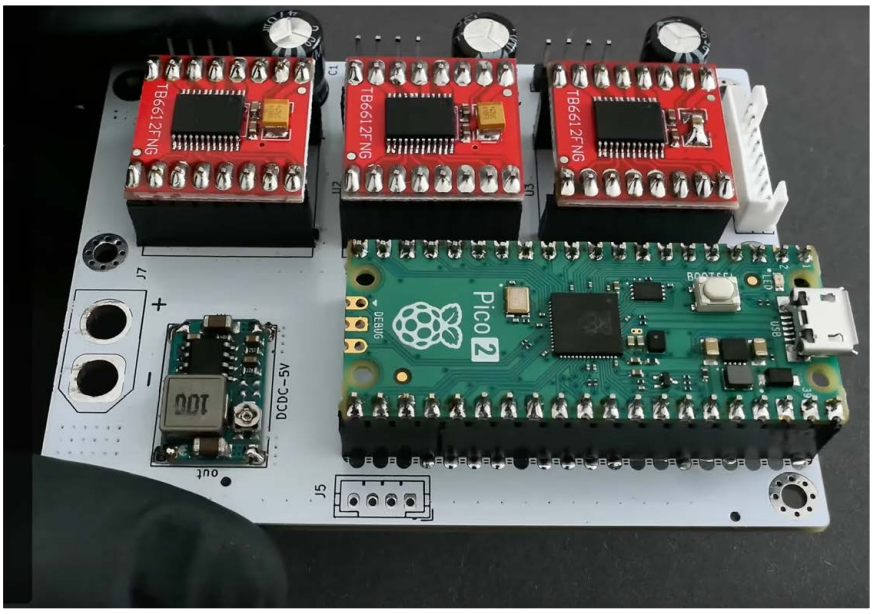
rpimag.co/MicroManipulator

What if the print head of your 3D printer could move with a level of accuracy measured in nanometres? What if it could trace the outline of a benchy so small that it could fit in a 20 × 20 micrometre space? Furthermore, what if this capability was built into an assembly that's open-source hardware, and controlled by a Raspberry Pi Pico 2? That's the mind-blowing prospect offered by this Micro Manipulator by Diffraction Limited.

The Open Source Motorized XYZ Micro-Manipulator, to give it its full title (the maker admits he's looking for alternative names), is built with an ingenious mix of technologies. The teeny tiny ball joints are small to keep friction low, so they have to be pre-tensioned with elastic bands to stop them falling out. The assembly is made from 3D-printed and off-the-shelf parts, and the movement of the manipulator can be controlled via the same standard G-code that will be familiar to anyone who's used to CNC or 3D printing. And despite that, it's unbelievably accurate.



▲ The perfectly acceptable price for accuracy is the small range of motion: just 23 × 23 × 23mm




Lab Rax

MiniLab

By RetroArchangel

rpimag.co/MiniLab

Here's a snazzy solution to the problem of too many Raspberry Pi computers cluttering up your workspace – put them in a custom 3D-printed case. Reddit user and IoT developer RetroArchangel printed this rack with space for six Raspberry Pi machines, plus a HAT for each one, and added OLED status screens for each one. They designed and printed the mounting for the camera mount and the 7-inch LED screen, and all together it's a powerful, neat development lab for development – they're currently using it to work on a "couple of AI vision projects... and a device to encourage social engagement by people in cognitive decline."

The Raspberry Pi computers in the build are five Raspberry Pi 4s, each with a PoE HAT (that way, they can all be powered over Ethernet, rather than through their own separate power supply), and a Raspberry Pi 5 with a Coral Dual Edge TPU HAT for machine learning.

If you're in the market for something similar, check out Gridfinity – it's a system of 3D printing modular storage solutions, and it takes a lot of the headaches out of measuring, tolerances, and other faff.

◀ The small OLED screens associated with each Raspberry Pi computer are I2C devices

PianoPi

By PianoPiPlayer

rpimag.co/PianoPi

Learning to play the piano takes time and effort. Building a machine to play the piano for you also takes time and effort, but it has the advantage that you can add a load of RGB LEDs to light up the keyboard. As the name suggests, the PianoPi is a machine built on a Raspberry Pi 5 that takes the drudgery out of artistic expression, using Raspberry Pi to control 88 solenoids that press the piano's keys.

The user can connect remotely to the Raspberry Pi 5 to choose the song they want to play, or if they're in the same room they can use the touchscreen app, which shows a list of songs, a volume slider, play and pause buttons, and several options to control the lighting. That's right, lighting – each key has an RGB LED pointing at it, so the keyboard lights up as each note is pressed.

Best of all, the PianoPi doesn't require you to take a Dremel to your treasured Steinway – it just drops on top of the keyboard and lifts off when you want to listen to a human play.

▶ Thanks to the Raspberry Pi 5's NVMe SSD HAT, there's plenty of room for MIDI tracks stored locally

Warning!

Electrical safety

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

rpimag.co/electricalsafety

Warning!

Mind your fingers

Do not attempt to play a piano whose keys are covered by 88 solenoid actuators; at the very least, you'll chip your nail varnish.

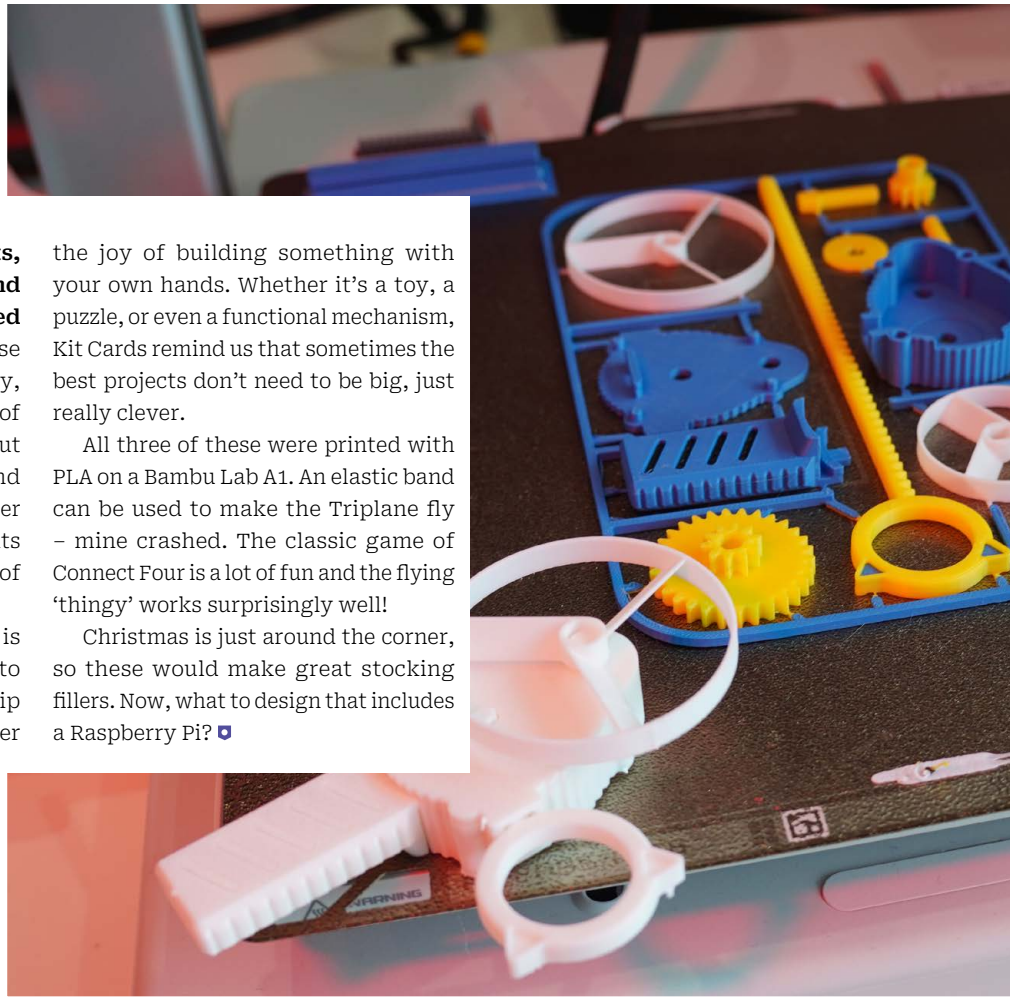
Card prints

Print in two-and-a-bit dimensions for build-your-own kits. By **Toby Roberts**

rpimag.co/Connect4

rpimag.co/Triplane

rpimag.co/PropellerLauncher



If you grew up with Airfix kits, you'll remember the joy (and frustration) of glue-covered fingers and tiny paint pots. Those retro models are still around today, but Kit Cards bring the same sense of excitement without the mess. Pop out the parts, snap them together, and you've got instant fun! If your 3D printer can handle multiple colours, the results look like they've leapt straight out of the box.

They capture what 3D printing is all about, turning imagination into reality. They're small enough to slip in a wallet, yet big enough to deliver the joy of building something with your own hands. Whether it's a toy, a puzzle, or even a functional mechanism, Kit Cards remind us that sometimes the best projects don't need to be big, just really clever.

All three of these were printed with PLA on a Bambu Lab A1. An elastic band can be used to make the Triplane fly – mine crashed. The classic game of Connect Four is a lot of fun and the flying 'thingy' works surprisingly well!

Christmas is just around the corner, so these would make great stocking fillers. Now, what to design that includes a Raspberry Pi?


Mini
Pr-OP
KIT
Unit-01

Raspberry Pi 500+

The **dream machine** is **here**.
Master physical computing with a mechanical keyboard, built-in SSD, 16GB RAM, 2.4GHz Arm processor, and GPIO.
Uncompromising performance, perfectly tactile.

By **Lucy Hattersley**

We are pleased to finally reveal the latest Raspberry Pi computer: 500+. This is the machine we have been waiting to share for over a year. It offers "uncompromising performance, perfectly tactile."

In the words of its engineers: This is the "dream machine". And this is the one that the editorial team want to use to code and write with.

Raspberry Pi 500+ puts the power of Raspberry Pi into an ergonomic and tactile mechanical keyboard with 16GB of RAM and a 256GB NVMe SSD. If you've ever wanted a modern-day Linux machine that you can plug in and start using for product development, coding, circuit building, and engineering, this is the perfect computer.

There are no compromises with Raspberry Pi 500+. The mechanical keyboard features Gateron KS-33 Blue keys with Cherry MX-style switches; the keys are low-profile with the top half of each key backlit. It is a 75% keyboard layout, which is our preferred layout, and offers the perfect balance between size and functionality. It's a clickable joy to type on.

Inside is a quad-core 64-bit Arm processor paired with 16GB RAM and a 256GB NVMe SSD with Raspberry Pi OS pre-installed.

We've been using 500+ at work for a while now, slowly phasing out our laptops and migrating to it for writing, research, and coding. It's a pure powerhouse and is a wonderful computer to use. Raspberry Pi OS is lightning fast and Linux is the best operating system around.

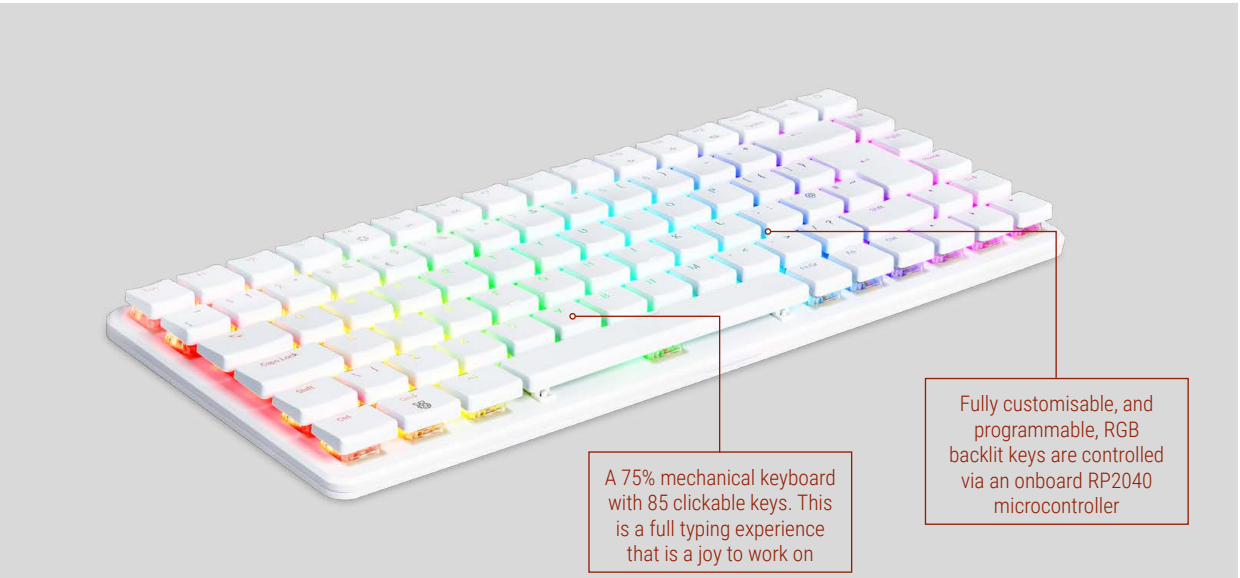
So, if you're looking for a seriously fun computer that's a coding and development control centre with a wonderful keyboard, fully upgradable and repairable, with a host of features that you just don't find on other computers, then Raspberry Pi 500+ is the one for you.

There are no compromises with Raspberry Pi 500+

SPECIFICATIONS

- Quad-core 64-bit Arm Cortex-A76 processor
- 16GB LPDDR4X SDRAM
- Integrated 256GB NVMe SSD with Raspberry Pi OS pre-installed
- Opening tool included for access to SSD (also requires screwdriver, not included)
- 2 × micro HDMI ports (supports up to 4Kp60)
- 2 × USB 3.0 ports
- 1 × USB 2.0 port
- Gigabit Ethernet port
- microSD card slot
- Horizontal 40-pin GPIO header
- 802.11b/g/n/ac Wi-Fi
- Bluetooth 5.0
- 5V DC via USB-C connector
- 85-key mechanical keyboard with clicky switches
- Replaceable keycaps (keycap puller included)
- Per-key programmable RGB LED backlighting

This is the *dream* machine



The clicky keyboard

Undoubtedly the star of Raspberry Pi 500+ is the mechanical keyboard. This provides a much more tactile experience than the chiclet-style keyboard found on a regular Raspberry Pi 500. Giving Raspberry Pi a mechanical keyboard makes it ideal for programmers and office workers who want the best keyboard around. It makes Raspberry Pi 500+ a unique experience that offers powerful desktop computing in a small package with a great keyboard.

A huge amount of time went into developing and determining the perfect keyboard for Raspberry Pi.

The keycaps are a custom in-house design by John Cowan Hughes. "We had a few different designs and took a poll," says Simon Martin, Senior Principal Hardware Engineer at Raspberry Pi. The keyboard has a flat, and low, profile, with a tapered edge at the bottom of each key.

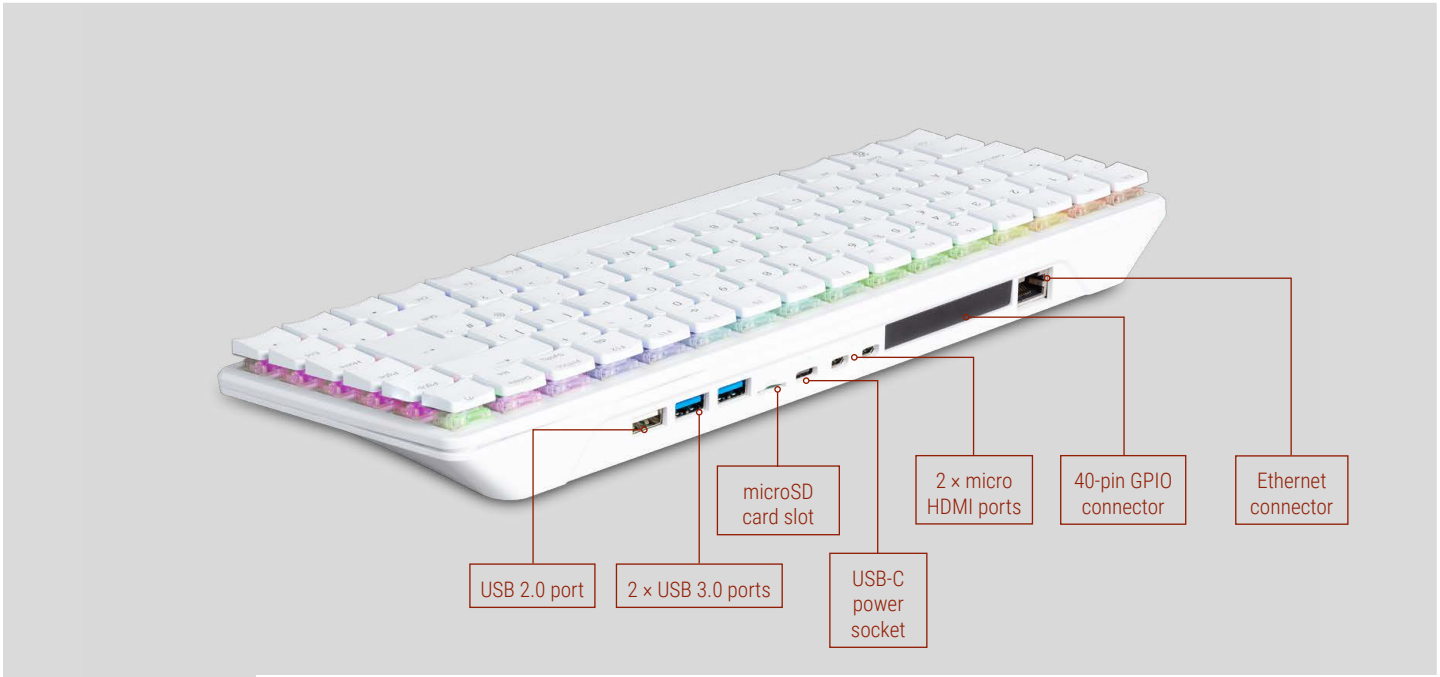
The keycaps can be replaced to a design you prefer (see 'Change the keycaps', overleaf).

The switches below are Gateron Blue. This is a version of the classic Cherry MX Blue switches which are famous for a tactile bump and classic click noise. It has a sound like a typewriter key that is adored by writers, coders, and gamers.

These Blue-style switches are distinct from Brown and Red counterparts. Brown switches are tactile but have a gentle bump halfway down, which removes the click; Red switches have a linear switch with no tactile feedback or click at all – they are quiet but feel squishy.

Mechanical switches are usually a matter of taste, but almost everyone we know prefers Blue keyboards – although Blue switches can be quite conspicuous.

▼ Around the back, we can see a range of useful ports



Light it up

As well as being mechanically minded, the new keyboard has fancy RGB lighting. Underneath each key sits an RGB LED that can light up the keyboard in different patterns. Pressing **FN+F4** cycles through a range of keyboard patterns:

- Pure white
- Pure red
- Rainbow
- Cycle rainbow
- Heat map
- Single key press
- Off

Pressing **FN+F5/F6** adjusts the brightness of the keyboard backlighting.

Get started: raspberrypi.com/500-plus

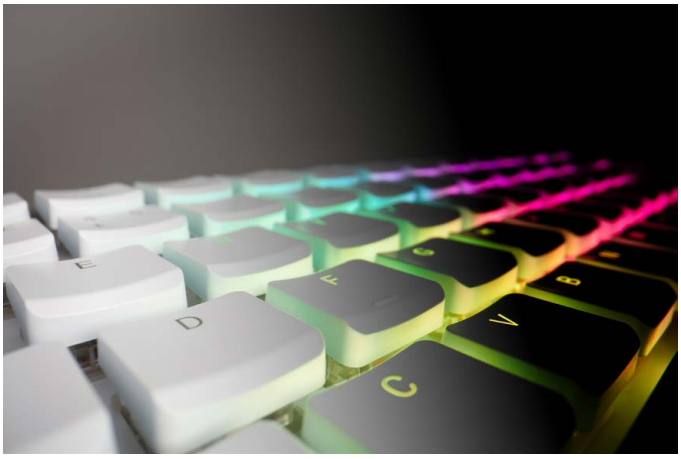
Keyboard variants Raspberry Pi 500+ is available at launch in the following keyboard variants:	These keyboard variants will be available in the following weeks:
• UK • US	• Spanish • Nordic • German • Japanese

Program the keyboard RGB lighting

Inside Raspberry Pi 500+ sits an RP2040 microcontroller, used to control the keyboard and lighting. This makes it possible to reprogram the keyboard and lighting and create your own custom light displays.

As a demonstration, the team has developed a keyboard light version of Flappy Bird that you can play using the Raspberry Pi 500+ keyboard. This program, written in Python, can be uploaded to RP2040 inside Raspberry Pi 500+.

We will have an exploration of programming the keyboard and recreating Flappy Bird on it in a future edition.



▼ Colour us impressed by the RGB lighting

▲ The keycap design used in Raspberry Pi 500+

Change the keycaps

Raspberry Pi 500+ comes with a key remover tool that is used to remove and replace the keycaps.

Raspberry Pi 500+ keycaps are a custom design and don't conform exactly to any of the standard profiles. Keycaps come in different profiles, such as:

- **OEM**. These are the default shape found on most factory-made keyboards.
- **Cherry**. Similar to OEM, but shorter.
- **Spherical All Row (SA)**. These are taller, with a 1970s retro vibe.
- **DSA (Double Shot ABS)**. Short, flat, and round. Most importantly, every row has the same shape.
- **XDA**. Similar to DSA but with a larger, flatter top surface.
- **MT3**. Very tall keycaps with a distinctly retro feel. As found on old terminal keyboards.

Note that most sets won't have a power button, and the volume and RGB keys won't be labelled. So you'll have to remember which function keys to use. The quality of RGB lighting is also dependent on the keycaps.

▼ Using the remover tool to change the keycaps

The keycap remover tool is easy to use. Simply place it under the edges of the keycaps, gently squeeze together to grasp the keycap, and pull it gently up. The keycap will pop off, revealing the mechanical switch underneath. To refit it, simply push the keycap back down and it will clip back into the switch.

NVMe M.2 SSD

Moving deeper down from the keyboard, we find the other stars of the show. The integrated M.2 NVMe SSD is connected to the motherboard via a PCIe interface and is easily upgradable.

This provides the same kind of speed boost that using an M.2 HAT and SSD with Raspberry Pi 5 does, and is vastly faster than using a stock microSD card (although note that a microSD card slot is also provided – which is incredibly useful for backup and quickly prototyping alternative operating systems).

By default, Raspberry Pi OS uses PCIe Gen 2, and you can gain a further speed boost by switching to PCIe Gen 3. To do so, open Terminal and enter:

```
$ sudo raspi-config
```

Then select `6 Advanced Options` > `A9 PCI Speed`.

Choose `Yes` to 'Would you like PCIe Gen 3 to be enabled?'

Choose `Finish` and `Yes` to the option to reboot your Raspberry Pi OS. When the operating system restarts, you will be running Raspberry Pi 500+ with PCIe Gen 3 enabled, which offers roughly twice the SSD speed. You can test the speed before and after using the Raspberry Pi Diagnostics app (found under menu > Accessories). The SD Card Speed Test

option is designed to check if microSD cards are fast enough for Raspberry Pi, but will also speed-test the internal SSD. Click Run Tests.

Here are our results:

PCIe Gen 2

```
seq-write;0;0;420776;102
seq-read;442064;107;0;0
```

PCIe Gen 3

```
seq-write;0;0;799219;195
seq-read;853889;208;0;0
```

	PCIe Gen 2	PCIe Gen 3
Write (MB/s)	411	780
Read (MB/s)	432	834

Raspberry 500+ vs Raspberry 500

Of course, Raspberry Pi 500+ isn't the only option and Raspberry Pi 500 remains available. This previous model has some advantages. While some people might love the clacky nature of the mechanical keyboard, an office full of the chiclet-style keys found on Raspberry Pi 500 will be more amiable to the ears.

Likewise, the lack of backlit keyboards can be seen as a benefit in some low-light environments and a sealed unit may be better for some office environments (although the microSD card is easily removable). The 500 has a Kensington lock, which is useful for physically restraining the movement of the device.

Still. There's no getting away from the fact that Raspberry Pi 500+ is a substantial upgrade and – to our eyes and hands at least – the one we want to use. Here is a handy guide to some of the differences we've spotted:

▲ The SSD inside Raspberry Pi 500+

500	500+
2.4GHz CPU	2.4GHz CPU
8GB RAM	16GB RAM
microSD card slot	256GB M.2 NVMe & microSD card slot
Tactile keyboard	Mechanical keyboard
No backlight	RGB backlight
10 Function keys (11, and 12 as Fn alternatives)	12 Function keys
No media keys	Separate media keys (Volume, brightness)
Power LED light	Power button with LED
Page keys with arrows	Dedicated PgUp, PgDn, Home, End keys
Fn on left	Fn on right
Sealed unit	Screw access with spudger included
Kensington lock	No Kensington lock

▲ Underside of a 500+, along with useful tools

Getting inside

Staying true to the maker ethos, Raspberry Pi 500+ is easy to get into. There are five pan-head Pozi cross-head screws on the back of Raspberry Pi 500+ that can be easily removed.

Then a coin slot located on the plastic case underneath the space bar can be accessed, using the supplied spudger. Move the spudger around the line in the case to separate the keyboard from the lower case.

Gently lift up the keyboard and remove the cable connecting the two. Use the spudger to lift the black plastic retaining clip on the ZIF connector (Zero Insertion Force) on either the underside of the keyboard or the mainboard.

The SSD in our Raspberry Pi 500+ is a 2230. Your model may come with a larger 2280 size SSD. Raspberry Pi 500+ accepts the following sizes:

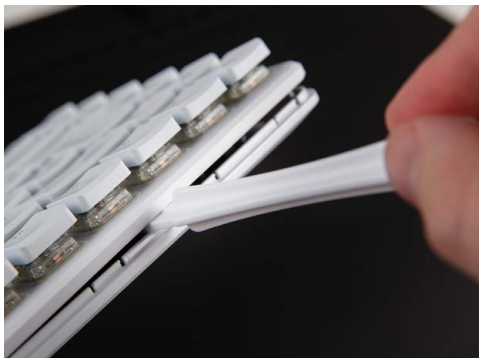
- 2230
- 2242
- 2260
- 2280

The default size is 256GB and you can easily upgrade this to a larger (or smaller) storage size. Remove the screw that holds the M.2 NVMe SSD in place and then gently ease it out. Now fit your replacement M.2 NVMe SSD in place and use the screw to secure it. Make sure the SSD is lined up with the groove in the screw so that it sits level.

CUSTOM SCREW

The SSD screw is a custom design by Raspberry Pi. Normally these screws are found on the underside, but Raspberry Pi wanted the drive to be accessible without going underneath the board.

◀ Raspberry Pi 500+ is easily accessible with five standard screws



◀ A spudger is included to get inside the case

Pricing and options	
	Raspberry Pi 500+ only **$200**
	Raspberry Pi 500+ Kit **$220** • Raspberry Pi 500+ • The Official Raspberry Pi Beginner's Guide • Raspberry Pi Official Mouse 2 • Raspberry Pi 27W USB-C Power Supply • Official micro HDMI to HDMI cable

▶ Flipping the keyboard up to take a look inside

Gently lift up the keyboard and remove the cable

ELECTROSTATIC PROTECTION

You will see a small pad fixed to the metal plate. Its primary function is to help with electrostatic discharge (ESD) protection, ensuring that static electricity is safely discharged and does not damage the keyboard's internal electronics.

Going deeper

Once you have flipped the keyboard off the top of Raspberry Pi 500+, you will see the metal head spreader beneath. This can also be removed to provide access to the mainboard.

You will find inside the plastic light port blocker which, as the name suggests, blocks the light from the RGB keyboard from escaping out of the ports at the rear.

On the mainboard you will find the CPU, 16GB RAM (not upgradeable, sadly), and the connections. It is essentially a Raspberry Pi 5 transformed into a size suitable for the case of Raspberry Pi 500+.

Keen eyes will notice the PoE (Power over Ethernet) circuit that is unpopulated. While those with sharp soldering skills may be able to get PoE installed, this isn't planned as a standard feature and 500+ isn't qualified for PoE.

There is a J5 connector for a power cell battery if you want to add time permanence to your 500+. And there is a UART which can be used to obtain early boot stage information.

Mouse 2

The eagle-eyed might have spotted the new Mouse 2. This looks the same as the standard Raspberry Pi Mouse, but has redesigned innards and a new all-white design (instead of red and white). The sensor and performance remain as excellent as before. There is also going to be a black model of Mouse 2 available.

rpimag.co/mouse

Meet the engineers

Both hardware engineers, Simon Martin and Chris Martin have been beavering away on Raspberry Pi 500+ for years, through a process of iteration that's seen a total of ten factory trips to China, six PCB revisions, and 3000 units that got built with the wrong kind of Return key. What sounds like a simple task – adding a mechanical keyboard, SSD, and more RAM to the already existing Raspberry Pi 500 – turned out to be far more involved than anyone thought, and it's resulted in a device that we think is absolutely brilliant. Thanks, gentlemen!

RPOM: Simon, Chris: what did you do on the 500+?

Simon Martin: I'm a senior principal engineer at Raspberry Pi, and I'm responsible for electronics and electromechanical design. I design a lot of PCBs. I'm responsible for making sure it all clips together. I work on cameras, and I worked on Raspberry Pi Zero 2.

Chris Martin: I'm a hardware engineer. I work directly with Simon most of the time, so I do a lot of different things, but I'm a mechanical engineer. I was fairly heavily involved in the early stages while I was doing a year-long internship with Simon, then came back after I'd finished university. I did quite a few of the revisions of the keyboard PCB, and I've done most of the software for the keyboard as well.

Most of my work from March 2023 to August 2023 was doing very early prototypes of this. We actually did a PCB and 3D print, and originally it was going to be on the 500 base. We decided actually it was worthwhile doing a new base to get the screws together so you could access the SSD.


Raspberry Pi

RPOM: Where do you start when designing something like this: with a Raspberry Pi 5, or a Raspberry Pi 500, or do you start with the keyboard and fit the other components around that?

SM: Yes, I started with a Raspberry Pi 5 schematic and layout. I picked up an early board design back in 2022 and modified it to look roughly like the board in Raspberry Pi 400. I had a sketch of where all the connectors and ports were going to go and laid out the design. It was easy to design a prototype board with the design resources available. In fact, we had a working prototype with 3D-printed plastics before Raspberry Pi 5 was launched.

The next stages were the difficult part. We had to design the keyboard and housing so that they were easy to fit together and easy to manufacture. The company decided to launch the low-cost Raspberry Pi 500 first, back in 2024. This took Raspberry Pi 500+ on a different path for a while. The bare PCB is common between them so much of the work was already done. We just needed to get the new keyboard upper ready in time plus a scramble to get compliance, production test, marketing materials and all the artwork completed in time.

CM: The base kind of followed on from the design of the keyboard. With the size of the 500, you can't get a good keyboard layout with mechanical key switches, because on the 500 you have smaller keys, especially in the arrow keys. Basically, our limitation was the 1.75 width Shift key.

SM: And the other thing is, as Chris said, that you can have an external NVMe just by having USB 3.0. It's just as fast to boot from an external drive as it is to boot from that internal drive. So you could boot from the external drive and have a Hailo [AI accelerator] inside, which would be kind of cool. There's no camera port, though, so there's no image processing – you'd have to have a USB camera plugged into it instead.

RPOM: You'll have seen the people online asking for a mechanical keyboard version of the 400/500. Did this have any influence in the decision to make the 500+?

SM: Yes, this was a strong influence. We took on the feedback from users who wrote in to us. There were a lot of mentions about mechanical keyboards. Other feedback was that it needed an NVMe port so you can plug in an M.2 SSD. Some did not like the USB ports on the left. Some felt it did not have enough RAM. We took all of this feedback and designed it in.

Raspberry Pi 500 and 500+ were originally intended to be released at the same time, and they used the same PCB. So a few people who opened up Raspberry Pi 500 saw that there were features on the board that weren't used in the 500 and correctly deduced that they were intended for a future product – including the NVMe drive. When they were going to be launched at the same time, that didn't matter, but we've had to keep tight-lipped about Raspberry Pi 500+, because nobody knew when it was coming.

> *The key switches are reliable up to 60 million cycles*

RPOM: Mechanical keyboard people are incredibly detail-obsessed, so I guess there's been a lot of thought gone into the choice of key switches?

SM: The key switches are Gateron KS33, which is a good brand – they're reliable up to 60 million cycles, so the spray paint will have worn off the key caps long before the key switch fails. So it's just as well that the keycaps are replaceable!

The easiest way of describing them is that they are Cherry Blue-style, which is clicky. There are all sorts of different variables of clickiness and travel and rebound. It's got a little spring in there which makes the clicks. Within the same line-up, they have a Red, which is a linear, Brown is a tactile, and Blue is clicky.

CM: Some keyboards are really bad for resonance – the click of the keys echoes around inside the keyboard and it makes a rattling sound. This is quite good for resonance. With some of the fancy mechanical keyboards, they have big layers of foam in them to top rattles and echoes. That's what mechanical keyboard people get into now – layers of foam, and switches suspended by rubber and things like that. This, by chance, is naturally quiet without us having to shape the sound profile. I think the fact that we have a metal plate in there and plastic around it helps.

RPOM: Whose baby is Raspberry 500+? I remember talking to Eben about 500+ about a year ago and he seemed ridiculously excited by it.

SM: Eben always wanted to make a computer in a keyboard because it is the way that computers were introduced to beginners in the 1980s and early 1990s. I also started programming when a Commodore 64 arrived under the Christmas tree in 1987.

The first product in the series was Raspberry Pi 400 in 2020. We then evolved it to Raspberry Pi 500 in 2024 and now the 500+. We have all been really excited about this because it is the flagship. I hope others are excited as I was when I switched on my first computer.

We were a bit sad that we could not, at least , leave the port in Raspberry Pi 500 with no drive. The problem was the drop test. Raspberry Pi 500 is a low-cost model, so it needs to clip together tightly without costly screws, but it also needs to withstand being dropped without smashing apart. The tight clips meant that damage was inevitable when users opened it up and there would be complaints and warranty returns. With Raspberry Pi 500+ we have much looser clips and a tool for releasing them, so we include screws to prevent it breaking apart when dropped.

RPOM: Chris, do you remember getting a Commodore 64 under the Christmas tree in 1987?

CM: Shockingly, no! But I do remember getting a Raspberry Pi back in 2012 – maybe it wasn't the first batch, but it was one of the early ones that had the old GPIO headers [original Raspberry Pi models used a 26-pin GPIO header, before switching to the 40-pin headers used today].

I had that for a few years, and then I transitioned to mechanical stuff, and I chose mechanical engineering for my degree, then did a year at Raspberry Pi through the Year in Industry scheme. At the time, I was building an automated chessboard as part of a group project, like in the Harry Potter films. I was doing a lot of the electronics using Raspberry Pi, which is how I got back into electronics.

RPOM: What did you learn while you were making Raspberry Pi 500+?

SM: When we made Raspberry Pi 400, we had the opportunity to make a keyboard product first, so we knew what we were doing when we put the computer inside the keyboard – the main scare was thermals, designing it to stay cool. With Raspberry Pi 500, that was straightforward as well. But the 500+ was a whole new world of screws and circuit boards and FPC cables between them.

> *I hope others are excited as I was when I switched on my first computer*

The connection between the keyboard and the computer took a lot of thinking about what signals are going to pass between them. It's not just a USB, because you've got other things going on like a blinking activity light coming up to the keyboard, and a means of reprogramming and so on need to be figured out.

One of the things we learned while developing Raspberry Pi 500+ was that electrostatic discharge [ESD] certification is quite difficult to pass on this kind of product, and we've ended up having to spin that board to include ESD protection. The keyboard is all exposed pins and switches, so it's very easy to discharge.

There's an air gap within the switch body, and so you have these pins that come down the electrode; the discharges in the ESD test can directly go to those switch pins, which are then directly wired to GPIOs on the RP2040, and it doesn't like that.

The testing requirements are very tough. It has to survive an 8 kilovolt discharge, which is much higher than the RP2040 is rated for.

One example of ESD compliance is this discharge pad; it means any static can go down through the ground pin, dropping through the electronics – the 500 didn't need any of this.

Another thing we didn't need in the 500 was the light blocker that sits around the ports at the back of the unit. So because [the keyboard PCB] is a single-side surface-mount assembly, all the LEDs are soldered down and they point light downwards through the circuit board.

However, they also light on the other side. So this is all shining bright inside the enclosure, and it looks pretty unattractive when all the lights are shining at the back of the ports. So we had to put this special piece of plastic in that blocks the light. And then we did it, and it worked, except that now rattled. So then there was more work to try to stop that rattling.

There are all these little details that we have to figure out which are not in the 500, as a consequence of having extra features. So it turns out that just putting a mechanical keyboard on a Raspberry Pi 500 isn't as simple as it sounds!

Part 01

Make games with Python: Draw shapes and paths

Learn the basics of Pygame by drawing some simple lines and shapes

Maker

Sean M Tracey

Sean calls himself a technologist, which is a fancy way of saying he still hasn't decided what he wants to do with technology – other than everything.

smt.codes

In this tutorial series, we are going to learn to make Python games with Pygame. Pygame is designed to make it easy to create games and interactive software. We'll look at drawing, animation, keyboard and mouse controls, sound, and physics. Each tutorial will add to our knowledge of Python game development, allowing us both to understand the games we play, and to create almost anything our imaginations can come up with. This series isn't for absolute programming beginners, but it's not far from it: we're going to assume that you've written some simple Python (or similar) programs in the past and are able to create files and get around your computer's file system without too much difficulty.

Installing Python and Pygame

Both Python and Pygame are installed on Raspberry Pi OS by default. If you're running macOS or Windows, you'll need to install Python from **python.org/downloads**. Mac users can also use Homebrew (**brew.sh**) to install Python and many other packages. Python3 should be installed by default on most recent Linux distributions.

On macOS, Linux, or Windows, you'll first need to set up a virtual environment (a Python sandbox for installing libraries without affecting your Python installation). If you want to run the most recent version of Pygame rather than the version packaged with Raspberry Pi OS, you'll need a virtual environment there as well. You can use the Linux instructions on a Raspberry Pi. The Linux instructions also apply to the Windows Subsystem for Linux (WSL).

On macOS or Linux, use these commands to create a Pygame virtual environment in the **.virtualenvs** subdirectory under your home directory:

1. Open a Terminal window and run the command `python3 -m venv ~/.virtualenvs/Pygame`
2. Activate the environment by running the command `source ~/.virtualenvs/Pygame/bin/activate`

On Windows, use these commands to create a Pygame virtual environment in the **Envs** subdirectory under your home directory:

1. Open a Command Prompt window and run the command `py -m venv %USERPROFILE%\Envs\Pygame`. If the `py` command is not found, replace it with `python3` or `python` and try again.

Activate the environment by running the following command: `%USERPROFILE%\Envs\Pygame\Scripts\activate`

After you've activated the environment, you'll need to install Pygame (you only need to do this once) with:

```
pip3 install pygame
```

To make sure Pygame is installed correctly, activate the environment as shown in step 2 and then run `python -m pygame.examples.stars`. You should see a moving starfield appear. You can close the window when you're done enjoying the splendours of the cosmos. If it doesn't work, please visit the Pygame wiki at **pygame.org/wiki/GettingStarted** for detailed installation instructions.

QUICK TIP

You must activate the environment each time you open a new Terminal or Command Prompt window in order for it to take effect.

You can configure many code editors, such as Thonny and Visual Studio Code, to be aware of your virtual environment. This can help when you want to run a script from the editor, but may also help if your editor checks the syntax for your code as you type.

If you're using Thonny, click the Python menu (which reads Local Python 3 by default) in the lower right of the window, click Configure Interpreter, and then make sure Local Python 3 is selected in the 'Which kind of interpreter' dropdown. Next, click the ... button to the right of the selected Python executable, and navigate to the Pygame virtual environment folder you created earlier, open the **bin** (macOS or Linux) or **Scripts** (Windows) folder, then double-click on the file named **activate** (macOS or Linux) or **python.exe** (Windows). On macOS or Linux, the **.virtualenvs** folder will be hidden, so you can type **~/.virtualenvs/Pygame/bin** in the file chooser to navigate to the folder.

In Visual Studio Code, make sure you've installed the Python extension from Microsoft (**rpimag.co/vscodepy**), and that you

Make games with Python 2nd Edition out now

This tutorial is part of a series from the latest revision of *Make games with Python*. Grab it today at **rpimag.co/makesgamesbook**

Why not try experimenting with Pygame to produce some interesting pixel art?

Windows Terminal and PowerShell

Windows Terminal runs PowerShell by default, but you can configure it to run Command Prompt instead (click the downward-pointing arrow on the tab bar, choose Settings, go to the Startup section, change the Default profile, and click Save). You can use a virtual environment with PowerShell, but you'd need to modify PowerShell's execution policy, so we suggest using the Command Prompt for simplicity's sake.

Tuple

A tuple such as `(400, 500)` is like a list, but unlike a standard list, a tuple's contents can't be changed (it's immutable). For example, lists support methods such as `append()`, and tuples do not. Lists are delimited with square brackets, for example `[400, 500]`.

have a Python program (a file ending in .py) open. Click the Python menu in the lower-right of the status bar, and you should see a list of all virtual environments. Pick the one named Pygame. The Visual Studio Code Python extension automatically looks for virtual environments in a folder in your home directory: **.virtualenvs** on macOS and Linux and **Envs** on Windows.

Creating shapes & paths

In this instalment, we're going to look at drawing and colouring various shapes in a window. This isn't quite Grand Theft Auto V, admittedly, but drawing shapes is the first step in building just about anything.

To start off, open your preferred code editor, create a new file, insert the following code into it and save it as **hello.py**:

```
import pygame
pygame.init()
clock = pygame.time.Clock()
window = pygame.display.set_mode((500, 400))
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            raise SystemExit

    # Begin drawing statements
    pygame.draw.rect(window, (255, 0, 0), (0,
0, 50, 30))
    # End drawing statements

    pygame.display.update()
    clock.tick(60)
```

In your Terminal or Command Prompt window, run the command `python hello.py`. If all has gone well, a new window will have opened showing you a red square on a black background in the top-left corner of the window. If it doesn't work, make sure you activated your virtual environment. We've just created our first Pygame program; let's walk through it.

Understanding hello.py

The first two lines of our first program are very simple: all we've done is told Python that we want to use Pygame. `import pygame` loads all of the Pygame code into our script, so we don't have to write all of that code ourselves. `pygame.init()` tells Pygame that we're ready to start using it.

The third line defines a clock that we'll use later to maintain a consistent frame rate, expressed in frames per second (fps). Let's look at the fourth line: `window` represents the application window for our Pygame program; each parameter affects the application window's shape and size. Width always precedes height. `window` is also the object that we'll use to tell other lines of code the surface on which they should draw shapes and set colours.

When we create our window, we're calling the `set_mode()` function of Pygame's display module, which is responsible for how the game window behaves. We're passing a tuple to

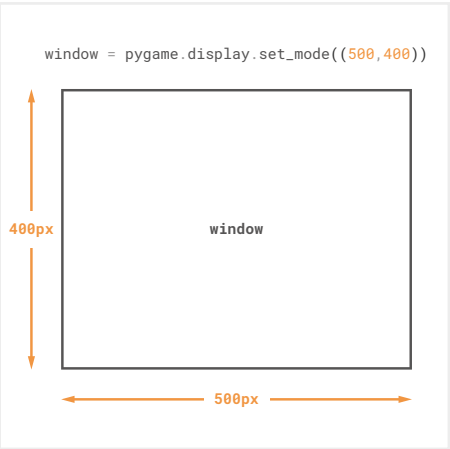


Figure 1: Each argument to `set_mode()` affects the application window's dimensions

`set_mode()` to tell it how big we want our game window to be. In this case, the application window is 500 pixels wide by 400 pixels tall. If we pass numbers that are bigger, the game window will be bigger; if we pass numbers that are smaller, the game window will be smaller, as shown in **Figure 1**.

The next few lines are where we make our program draw shapes on that window. When simple programs run, they execute their code, and when they're finished, they clean up

after themselves. That's fine unless, of course, you want your program to be interactive, or to draw or animate shapes over time, which is exactly what we need from a game. So, to keep our program from exiting, we make a `while` loop and put all our code inside. The `while` loop will never finish because `True` is always `True`, so we can keep running our program and drawing our shapes for as long as we like.

The first thing we do in our while loop is check for any events, such as key presses, joystick motion, or even mouse actions. In this case, we're only checking for a `QUIT` event, which can be triggered by closing the window. If you quit the program in this way, the code will call Pygame's `quit()` function and will raise a `SystemExit` exception to terminate the Python program itself.

```
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            raise SystemExit
```

Next, we draw a rectangle each time through the loop. A rectangle is one of the simplest shapes that we can draw in Pygame:

```
    pygame.draw.rect(window, (255, 0, 0), (0,
0, 50, 30))
```

The three arguments after `rect(` tell Pygame where to draw our rectangle (in the `window`), its colour, and its location and size. The first argument specifies the colour of our rectangle by representing how much red, green, and blue the colour should have in it. We use red, green, and blue because these are the three colours your screen combines to create every shade you can see on it. 0 means that none of that colour should be used; 255 indicates the maximum intensity of colour. We told our rectangle that it should be the colour `(255, 0, 0)`, which is pure red. If we had told it to be `(255, 0, 255)`, it would have been a bright purple, because it's being drawn with the maximum red and the maximum blue. If we had told our rectangle to be coloured `(100, 100, 100)`, it would be a dark grey, because all the colours would be equal.

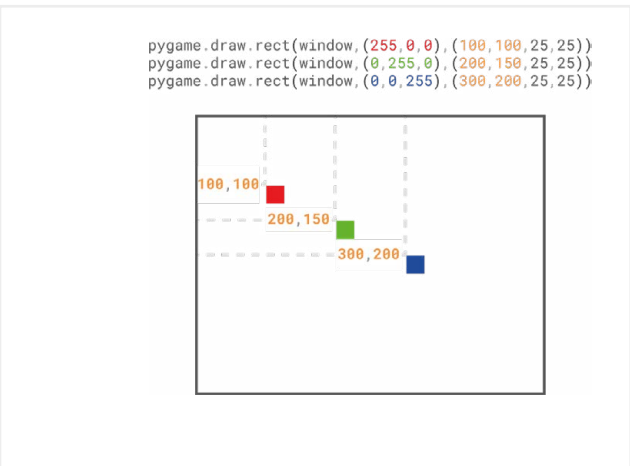


Figure 2: Setting shape attributes

Line width

When drawing a rectangle or ellipse, you have the choice of passing a line width as the fourth argument. If you don't, the shape will be filled solid.

After we've passed in our rectangle's colour, we have to tell it where it should go and how big it should be. We do this by passing a tuple of four numbers. The first number is an X coordinate, which determines how far from the left side of the window to place the rectangle's left edge. The second number is a Y coordinate; this determines how far down from the top of our window to place the rectangle's top edge. The third number gives the width of our rectangle, and the fourth defines its height. So, for example, if we wanted our rectangle to be 50 pixels from the left side of the window, 100 pixels from the top of our window, 20 pixels wide, and 80 pixels tall, we would use `(50, 100, 20, 80)` as the third argument.

The next line in **hello.py** – `pygame.display.update()` – is simple: it tells Pygame that we're done drawing shapes for the moment and that it can now refresh the window. This saves Python having to draw and redraw the screen for every shape that we've created; instead, it can get them all drawn in one go. After that, we call `clock.tick(60)`, which makes sure the game runs at a consistent frame rate (60 fps) on different devices.

Adding more shapes

We've successfully drawn one shape, so let's draw a few more. We'll draw some squares around the screen and mess around with their properties a little bit. There's no need to create a new file, so we'll work with **hello.py** for now. Replace everything between the `# Begin` and `# End` comments so that section of code looks like this:

```
    # Begin drawing statements
    pygame.draw.rect(window, (255, 0, 0), (100,
100, 25, 25))
    pygame.draw.rect(window, (0, 255, 0), (200,
150, 25, 25))
    pygame.draw.rect(window, (0, 0, 255), (300,
200, 25, 25))
    # End drawing statements
```

Now we have three squares: red, blue, and green as shown in **Figure 2**. So far, this is nice and simple, but those squares

have plenty of space between them. What would happen if they were to overlap? Let's find out. Change your code once more to the following:

```
    # Begin drawing statements
    pygame.draw.rect(window, (255, 0, 0), (0,
0, 50, 50))  # 1
    pygame.draw.rect(window, (0, 255, 0), (40,
0, 50, 50)) # 2
    pygame.draw.rect(window, (0, 0, 255), (80,
0, 50, 50)) # 3
    # End drawing statements
```

This time we get two rectangles and a square, which is not what we asked for. So, what has gone wrong? Our code works through what it has to draw and where it has to put it, line-by-line. If one item is drawn and then another is drawn over it, the second shape obscures what is beneath it: some or all pixels of the first shape are lost when covered by another. To see this effect in action, swap the code for the second and third squares:

```
    # Begin drawing statements
    pygame.draw.rect(window, (255, 0, 0), (0,
0, 50, 50))  # 1
    pygame.draw.rect(window, (0, 0, 255), (80,
0, 50, 50)) # 3
    pygame.draw.rect(window, (0, 255, 0), (40,
0, 50, 50)) # 2
    # End drawing statements
```

Now we get rectangle, square, rectangle because the red and blue squares were drawn first and then the green square was drawn over them. The red and blue squares are still there in their entirety, but we can't see all of them, so they look like rectangles.

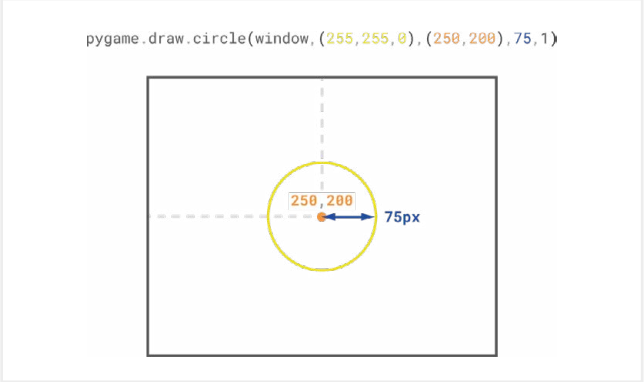
Pygame allows us to do a great deal more than merely draw rectangles: we can make all kinds of other shapes too, including circles, ellipses, and paths (which are made up of many lines between multiple points).

Drawing circles

The process of drawing a circle is much like drawing a square except that, instead of passing a width and a height, we pass a radius and a point around which we draw our circle. For example, to draw a yellow circle with a diameter of 150 pixels, replace the code in the drawing section in **hello.py** with:

```
    # Begin drawing statements
    pygame.draw.circle(window, (255, 255, 0),
                       (250, 200), 75, 1)
    # End drawing statements
```

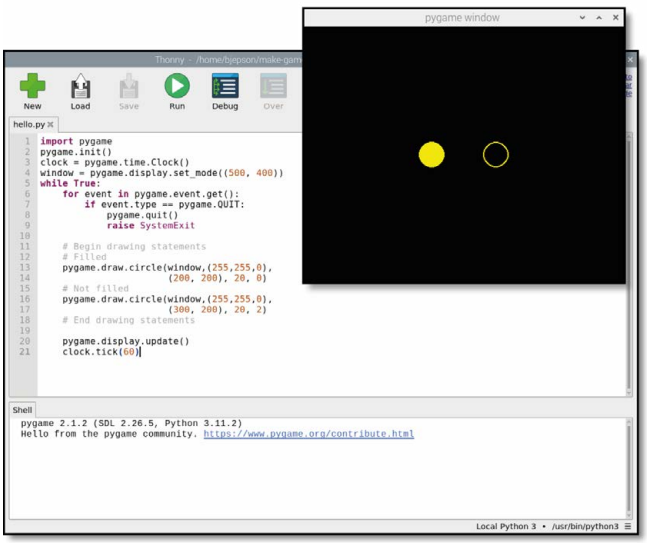
Similar to drawing a rectangle, we tell Pygame on which surface to draw our circle, its colour, where it should go (200, 200), followed by its radius (75), and another argument (as illustrated in **Figure 3**).



▲ **Figure 3:** Drawing a circle

That final argument — the `1` that appeared after our radius — is a value used to determine the width of the line that draws our circle. If we pass `0`, the circle is filled; but if we pass `2`, for instance, we get a 2-pixel-wide line with an empty centre (see **Figure 4**):

```
    # Begin drawing statements
    # Filled
    pygame.draw.circle(window, (255, 255, 0),
                       (200, 200), 20, 0)
    # Not filled
    pygame.draw.circle(window, (255, 255, 0),
                       (300, 200), 20, 2)
    # End drawing statements
```



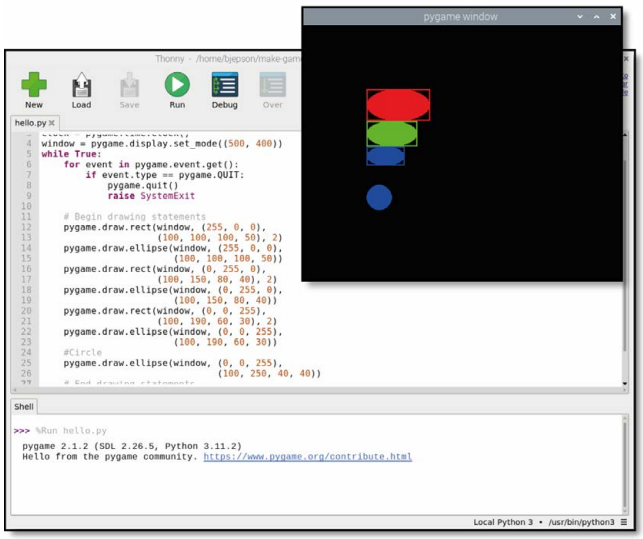
◀ **Figure 4:** When drawing a circle, the last argument determines whether the circle should be filled

What about ellipses? They are a slightly strange cross between drawing rectangles and circles. As we did when we drew a rectangle, we pass an X coordinate, a Y coordinate, a width, and a height, but we end up with an elliptical shape. Let's draw some ellipses.

```
# Begin drawing statements
pygame.draw.ellipse(window, (255, 0, 0),
                    (100, 100, 100, 50))
pygame.draw.ellipse(window, (0, 255, 0),
                    (100, 150, 80, 40))
pygame.draw.ellipse(window, (0, 0, 255),
                    (100, 190, 60, 30))
# End drawing statements
```

Just as before, run your code. You should now see three ellipses: one red, one green, and one blue. Each should be a different size. If you wanted to visualise how these shapes were generated, you could draw rectangles using the same coordinates as you used to draw an ellipse and it would fit perfectly inside that box. As you may have guessed, this means you can also make circles by using `pygame.draw.ellipse()` if the width and height parameters are the same. **Figure 5** shows the result.

```
# Begin drawing statements
pygame.draw.rect(window, (255, 0, 0),
                 (100, 100, 100, 50), 2)
pygame.draw.ellipse(window, (255, 0, 0),
                    (100, 100, 100, 50))
pygame.draw.rect(window, (0, 255, 0),
                 (100, 150, 80, 40), 2)
pygame.draw.ellipse(window, (0, 255, 0),
                    (100, 150, 80, 40))
pygame.draw.rect(window, (0, 0, 255),
                 (100, 190, 60, 30), 2)
pygame.draw.ellipse(window, (0, 0, 255),
                    (100, 190, 60, 30))
#Circle
pygame.draw.ellipse(window, (0, 0, 255),
                     (100, 250, 40, 40))
# End drawing statements
```



▲ **Figure 5:** Ellipses in the rectangles that bound them

A new path

We have covered rectangles, squares and circles, but what if we want to draw a triangle, a pentagon, a hexagon, or an octagon? Unfortunately, there aren't functions for every kind of shape, but we can use paths. Paths allow us to draw irregular shapes by defining points in space, joining them up with lines, and filling in the space we've created. This is a little more complex, so it's time to move on from our original program. Create a new file, call it **paths.py**, and save it with the following text inside:

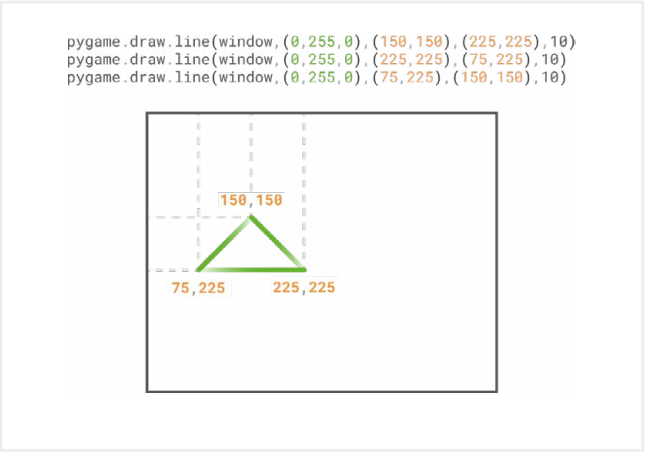
```
import pygame
pygame.init()
clock = pygame.time.Clock()
window = pygame.display.set_mode((500, 400))
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            raise SystemExit

    # Begin drawing statements
    # End drawing statements

    pygame.display.update()
    clock.tick(60)
```

This is simply our bare-bones Pygame app again. If you want to make a copy of this for experimenting without breaking anything, now would be a good time to do so.

Every path is made of connected lines, but, before we start joining things up, let's draw a couple of standalone lines to familiarise ourselves with them. We can do this with `pygame.draw.line()`. Edit **paths.py** so your drawing statement section reads as follows:

```
    # Begin drawing statements
    pygame.draw.line(window, (255, 255, 255),
                     (0, 0), (500, 400), 1)
    # End drawing statements
```

If you run this code now, you'll see a one-pixel-wide white line going from the top left to the bottom right of our Pygame window. The arguments we pass to `pygame.draw.line()` start off the same way rectangles and ellipses do. We first tell Pygame where we want to draw the shape and then we choose a colour. After that, the arguments change a little. The next argument is a tuple with the X and Y coordinates for where we want our line to start, and the third argument is a tuple with the X and Y coordinates for where we want our line to end. These specify the two points between which our line will be drawn. The final argument is the width of the line being drawn in pixels.



▲ **Figure 6:** You can make a triangle from three separate lines

With lines, we can now create shapes by defining points in our window. Let's draw that triangle we talked about earlier (see **Figure 6**):

```
    # Begin drawing statements
    pygame.draw.line(window, (0, 255, 0),
                     (150, 150), (225, 225), True)
    pygame.draw.line(window, (0, 255, 0),
                     (225, 225), (75, 225), True)
    pygame.draw.line(window, (0, 255, 0),
                     (75, 225), (150, 150), True)
    # End drawing statements
```

You should have an image of a green triangle with a 1px edge. However, this code is rather lengthy: so many things, like the colour or the width of the line, are written multiple times. There is, however, a more concise way to achieve the result we want. All we need is `pygame.draw.lines()`. Whereas `pygame.draw.line()` lets us draw a line between two points, `pygame.draw.lines()` enables us to draw a sequence of lines between numerous points. Each XY-coordinate point will be joined up to the next XY-coordinate point, which will be joined up to the next XY-coordinate point, and so on. You can see this in **Figure 7**.

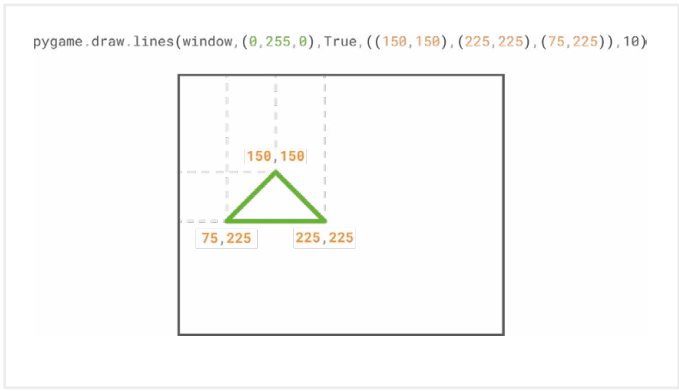
```
    pygame.draw.lines(window, (0, 255, 0),
True,((150, 150), (225, 225), (75, 225)), 10)
```

After running the code in the following listing, you'll see that the resulting triangle is exactly the same, except that we produced it from one line of code instead of three. You might have noticed that we didn't actually close the triangle: Pygame did it for us. Just before we pass the points for our shape to be drawn from, we can pass either a `True` or a `False` value that will let Pygame know that we want it to close our shapes for us. Change it to `False` and we get the first two lines of our shape, but not the third. If we want to make a more complex shape, we simply add more points like so:

```
    # Begin drawing statements
    pygame.draw.lines(window,(255, 255, 255),
True, ((50, 50), (75, 75), (63, 100), (38, 100),
(25, 75)), 1)
    # End drawing statements
```

There you have it: your very own pentagon. If you want to make a hexagon, an octagon, or even a triacontagon, just add more points — it's that easy. Why not try experimenting with Pygame to produce some interesting pixel art?

▼ **Figure 7:** This triangle is made up of one line with multiple points



hello.py

> Language: **Python**

DOWNLOAD THE FULL CODE:

rpimag.co/pygamebookgit

```
001. import pygame
002. pygame.init()
003. clock = pygame.time.Clock()
004. window = pygame.display.set_mode((500, 400))
005. while True:
006.     for event in pygame.event.get():
007.         if event.type == pygame.QUIT:
008.             pygame.quit()
009.             raise SystemExit
010.
011.     # Begin drawing statements
012.     pygame.draw.rect(window, (255, 0, 0), (0,
    0, 50, 30))
013.     # End drawing statements
014.
015.     pygame.display.update()
016.     clock.tick(60)
```

Part 04

Conquer the command line: manipulating text

Connect simple commands with pipes for more powerful text processing

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

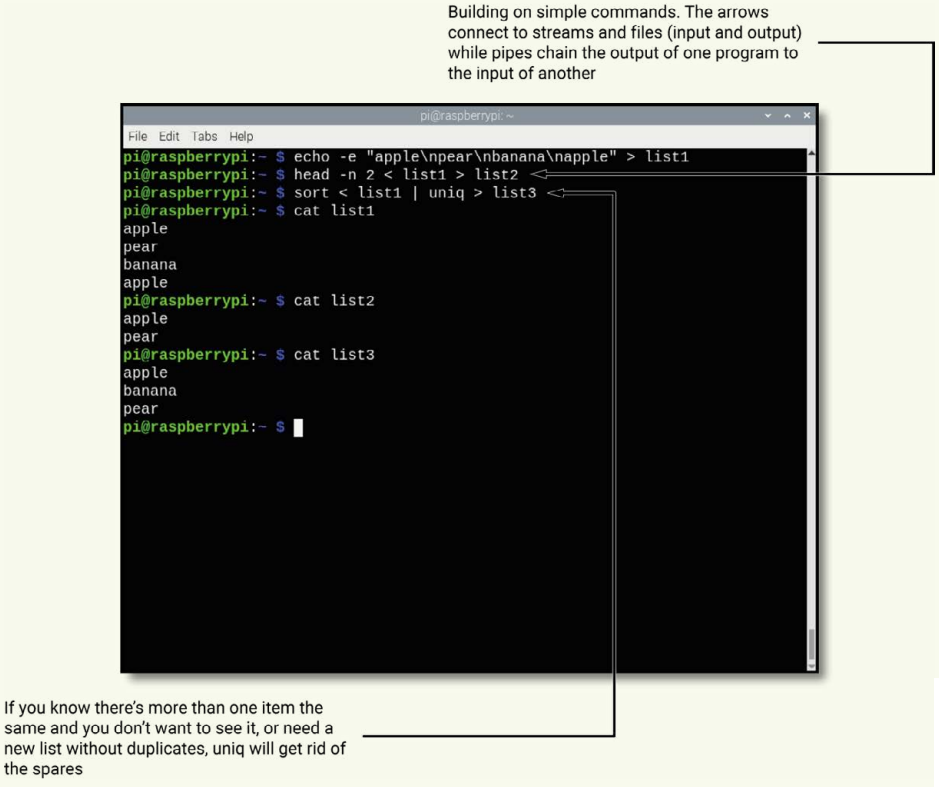
about.me/RichardSmedley

The Unix family of operating systems, which includes the Unix-like GNU/Linux and also Apple's macOS (which is licensed to use the UNIX trademark), deals with data from commands as streams of text. This means that commands can be chained together in countless useful ways. For now, though, we'll focus on giving you a firm foundation to building your own custom commands.

Getting our feet wet

When a command is called at the terminal, it is given three streams, known as standard input (stdin), standard output (stdout), and standard error (stderr). These streams are plain text, and treated as special files. As we noted last issue (**rpimag.co/157**), 'everything is a file': this is what gives Unix-like systems the ability to put together simple commands and programs to build complex but reliable systems.

Normally, stdin is what you enter into the terminal, while stdout (command output) and stderr (any error messages) appear together. The reason the last two have a separate existence is that you may want to redirect one of them – error messages, for example – somewhere away from the regular output your commands produce. We'll look at separate error messages later, but first we need to know how to redirect and connect our output to other commands or files.



◀ Sort and summarize files in all kinds of ways

Connecting commands together are pipes, the **|** symbol found above the backslash on both GB and US keyboards (although the two keyboards for English speakers place the **** respectively to the left of Z, and at the far right of the home row). When you type a command such as `ls -l`, the output is sent by Raspberry Pi OS to the stdout stream, which by default is shown in your terminal. Adding a pipe connects that output to the input (stdin stream) of the next command you type. So...

```
$ ls -l /usr/bin | wc -l
```

...will pass the long listing of the **/usr/bin** directory to the wordcount (`wc`) program which, called with the `-l` (line) option, will tell you how many lines of output ls has. If the `-l` option didn't include a total amount of disk space (measured in one-kilobyte blocks) as its first line, this would be a good way to count how many files and folders are in a particular directory. However, you can get a bare listing with `ls -1` (that's the number one, not a lower-case L), and you can include hidden files, but not the current working directory (**.**) or the parent (**..**) with `ls -A1`.

We'll focus on giving you a firm foundation to building your own custom commands

Search with grep

One of the most useful commands to pass output to is `grep`, which searches for words (or 'regular expressions', which are powerful search patterns understood by a number of commands and languages), like so:

```
$ grep if /usr/share/code-the-classics/soccer/
soccer.py
```

This displays every line in the **soccer.py** file containing the character sequence 'if' (see **Figure 1**) – in other words, not just the word 'if', but words like 'elif' (Python's else if), and words like 'difficulty' if they were present.

You can use regular expressions to just find lines with 'if', or lines beginning with 'if', for example. The `\b` escape character sequence refers to a word boundary, but you need to enclose the whole expression in quotes to prevent the shell from misinterpreting it. This will find only those lines that contain the word 'if':

```
$ grep '\bif\b' /usr/share/code-the-classics/
soccer/soccer.py
```

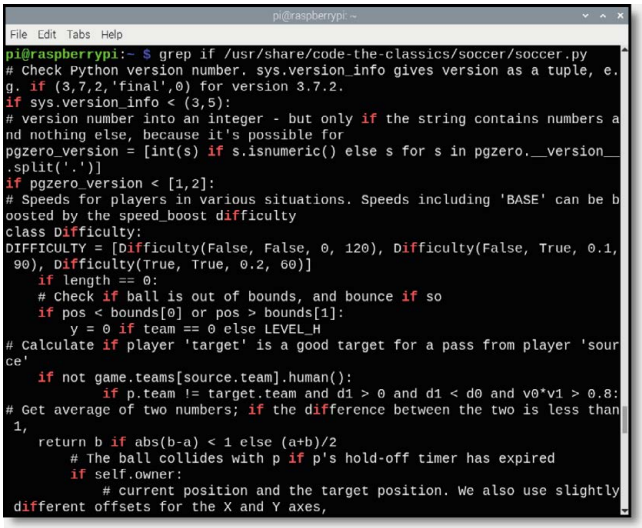
And this uses the `^` symbol to anchor the match to the start of the line, so it will only return lines beginning with 'if':

REGEXP

Regular expressions (regexps) are special characters used in text searches, such as `[a-z]` to match any letter (but not numbers), `^` to match to the beginning of a line, and `$` to match the end of a line.

```
$ grep '^if' /usr/share/code-the-classics/
soccer/soccer.py
```

▼ **Figure 1:** No matter how long the file, `grep` will dig out the lines you need. It's also handy for finding the results you want from a multi-page output



Piping search results and listings to `grep` is the way we find a needle in one of Raspberry Pi's haystacks. Remember `dpkg` from the last chapter, to see what was installed? Try…

```
$ dpkg -l | grep -i game
```

…to remind yourself which games you've installed (or are already installed). The `-i` switch makes the search case insensitive, as the program may be a 'Game' or 'game' in the description column. A simple `dpkg -l | more` lets you see output a page at a time.

`sort` will, as the name suggests, sort a listing into order, with various tweaks available such as `-f` to bring upper and lower case together. One way to collect unsorted data is to combine lists. `sort` will put the combined listing back in alphabetical order (`-f` indicates a case-insensitive sort):

```
$ ls ~ /usr/share/code-the-classics | sort -f
```

Suppose you copied one of the games to your home directory to modify: you know it's there, but you don't want to see the same name twice in the listings. `uniq` will omit the duplicated lines or, with the `-d` switch, show only those duplicates:

```
$ ls ~ /usr/share/code-the-classics | sort -f |
uniq
```

File it away

Pipes are not the only form of redirection. `>` (the 'greater than' symbol) sends the output of a program into a text file, either creating that text file in the process, or writing over the contents of an existing one.

```
$ ls /usr/bin > ~/mylisting1.txt
```

ABSOLUTE PATH

We're using **~/mylisting1.txt**, with **~** short for **/home/pi**. If you `cd` to **~** then you can simply use the file name without the **~/**

Conquer the Command Line 3rd Edition out now

This tutorial is part of a series from the latest revision of *Conquer the Command Line*. Grab it today at **rpimag.co/commandlinebook**

Now look in **/home/pi/mylisting1.txt** and you'll see the output of `ls /usr/bin`. Note that each item is on a separate line (see **Figure 2**). Your terminal displays multiple listings per line for space efficiency; however, for easy compatibility between commands, one listing per line is used. Most commands operate on lines of text; e.g., `grep` showed you in which lines it found 'if'. Note that some commands need a dash as a placeholder for the stdin stream being piped to them:

```
$ echo "zzzz is not a real program here" | cat
~/mylisting1.txt -
```

Appending

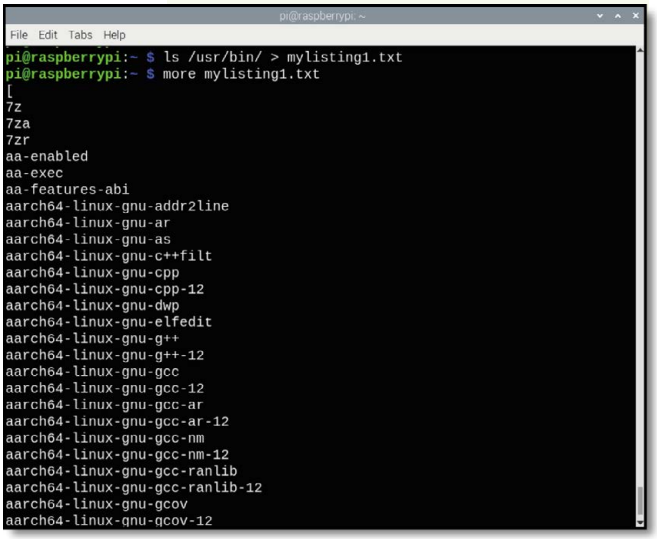
If you want to add something to the end of a file without overwriting the contents, you need `>>`.

```
$ echo "& one more for luck!" >> ~/mylisting1.txt
```

`echo` simply displays whatever is in the quote marks to stdout; the `-e` switch lets you add in special characters, like `\n` for newline (see below). You can look at the last few lines of a file with `tail ~/mylisting1.txt`. The `<` symbol will link a program's input stream to the contents of a file or stream. Make an unsorted list to work on, and sort it:

```
$ echo -e "aardvark\nplatypus\njellyfish\
naardvark" > list1
$ sort < list1
```

▼ **Figure 2:** With redirection, you can get all of the output from a command saved straight into a text file. Save your error messages to ask about them on the forums!



You can also combine `<` and `>`:

```
$ head -n 2 < list1 > list2
```

...will read from **list1**, passing it to `head` to take the first two lines, then putting these in a file called **list2**. Add in a pipe:

```
$ sort < list1 | uniq > list3
```

Lastly, let's separate that stderr stream: it has file descriptor 2 (don't worry too much about this), and `2>` sends the error messages to any file you choose:

```
$ cat list1 list2 list3 list42 2>errors.txt
```

The screen will display the 'list' files you do have, and the 'No such file or directory' message(s) will end up in **errors.txt** – alternatively, `2>>` will append the messages to the file without overwriting previous contents.

FILING HOMEWORK

There are many more commands beyond `grep`, `sort`, and `uniq` that can be chained together. Take a look at `cut` if you're feeling adventurous.

Zuse Z3

From toy construction sets to building a pioneering electromechanical computer

Author

Tim Danton

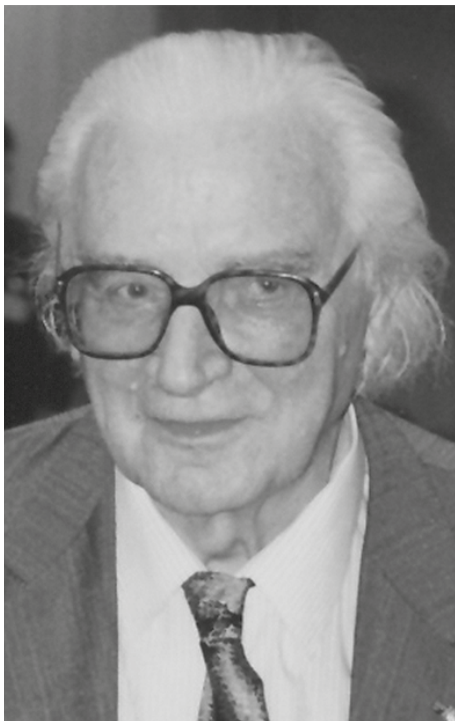
When not writing books about classic computers, Tim is editor-in-chief of the British technology magazine PC Pro. He has also helped to launch several technology websites, most recently TechFinitive.com, where he is a senior editor.

dantonmedia.com

When Konrad Zuse's parents gave their young son a metal toy construction set – much like the Meccano sets popular in Britain and the USA at the time – they had no idea that they were sowing the seed for the world's first binary, programmable computer. For Zuse, with help from family rather than companies, friends rather than universities, created a series of machines that could rival any of their contemporaries.

If he had been based in Boston rather than Berlin, there's every chance that his most admired computer, the Z3, would be as famous as the ENIAC. And that Konrad Zuse, the man, would be as well-known as John Mauchly in the US and Alan Turing in the UK.

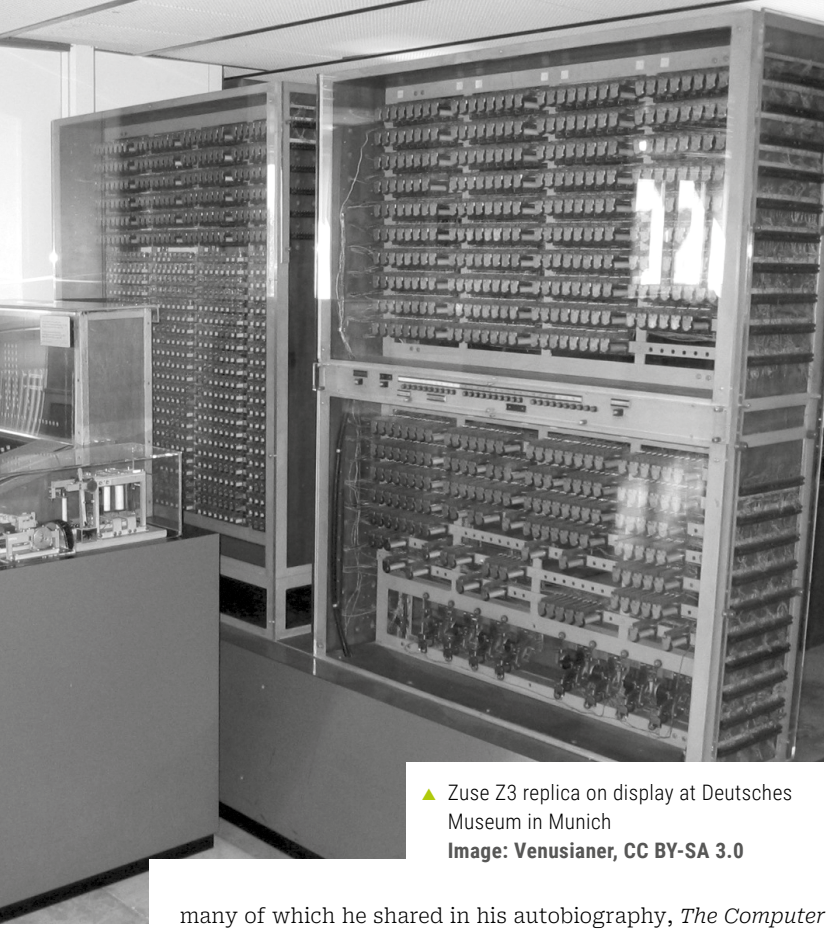
For now, though, let us spare a thought for his long-suffering parents, who not only contributed money to his first computer but would eventually donate their dining room to house it too.



▶ Konrad Zuse
Image: Wolfgang Hunscher, CC BY-SA 3.0

Arguably, this was their own fault. That first construction set led to Konrad spending so much time creating mechanical inventions that he won prizes for them – prizes of extra construction parts that he would turn into ever-more elaborate models. He was particularly proud of a large grab crane that sat on top of the wardrobe in his room and could be controlled using strings from his bed.

So formed a pattern – arguably an obsession – of solving problems using mechanical devices. Zuse devised numerous ways to automate the world,



▲ Zuse Z3 replica on display at Deutsches Museum in Munich
Image: Venusianer, CC BY-SA 3.0

many of which he shared in his autobiography, *The Computer – My Life* (*Der Computer – Mein Lebenswerk* in the original German). He even built an automatic fruit-dispensing machine, much to the delight of friends such as Andreas Grohmann who described this "mandarin machine which took money and gave mandarins, and sometimes indeed returned the money with the goods".[1]

With a mind that refused to run along conventional lines, Zuse did not follow a straightforward academic path. Equally gifted at mathematics and drawing, it wasn't clear to the teenage Zuse what his career would be. Artist? City planner? Photographer, even? But in the end, engineering would win.

This led him to Berlin's Technical College, which meant that a 17-year-old Zuse had to leave his parents' home in Hoyerswerda – roughly 100 miles south of Berlin – and rent a small room on his own. Fortunately, he quickly forged new friendships via the Akademischer Verein Motiv, a student club with a theatrical side. The Motiv, and the Motivers who made up its membership, would prove crucial not only in Zuse's personal life but also for becoming the loyal group of friends who would donate their time and their money (even their parents' money) in support of his computers.

Academically, progress wasn't so smooth. Zuse soon regretted choosing mechanical engineering as his major, complaining that the "creative spirit was left little freedom in the manner of presentation; everything was standardised, everything was decided"[2]. He switched to architecture, but "Doric and Ionic columns left me cold", which led him to his final choice of civil engineering.

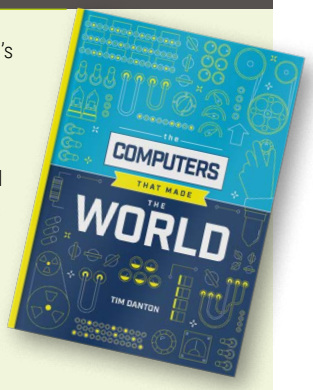
It was here Zuse met his nemesis: statics. This branch of applied physics concerns static bodies such as bridges and roofs, calculating the forces in play upon each of their interconnected parts. Zuse admitted a "pronounced aversion" to the science, adding that he "worshipped the professors in command of this calculus as if they were demigods from another world".[3]

> *During evenings and weekends, Zuse would work tirelessly to create his first machine*

As a student, Zuse faced such problems on a regular basis. Most of the time, according to renowned computer historian Paul Ceruzzi,[4] he would solve them using a slide rule, but he also had experience with mechanical desktop calculators. The challenge wasn't each calculation but "transferring the results

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 12 influential computers built between 1939 and 1950. You can pick up a copy on the Raspberry Pi Press store.



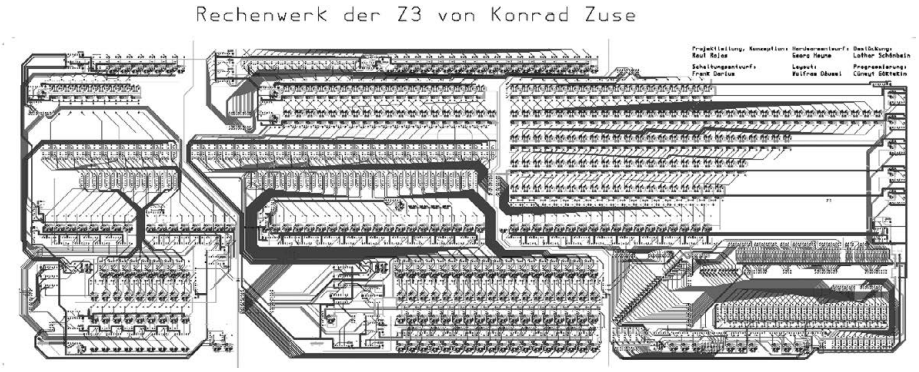
rpimag.co/computersworldbook

1 Konrad Zuse, *The Computer – My Life* (Springer-Verlag, 1991, ISBN 978-0387564531), p35

2 As before, p15

3 As above, p16

4 Paul E Ceruzzi, 'The Early Computers of Konrad Zuse, 1935 to 1945', in IEEE Annals of the History of Computing, Vol 3, No 3, July 1981



▲ Layout diagram of the Z3 processor.
Image: courtesy of Raúl Rojas

of one operation to another part of the problem",[5] wrote Ceruzzi.

The increasingly sophisticated engineering designs of the early 20th century only compounded the challenge. While an individual could handle a series of six equations with six unknowns, real-world projects were far more complex. Ceruzzi cites an example of calculating the stresses on a roof that required a system of 30 equations with 30 unknowns; this took a team of human computers months to work out.[6]

Sketch it out

In typical style, Zuse sketched out mechanical methods to simplify the problem while still a student. One involved a crane, reminiscent of his schoolboy invention, that would automatically "scan the plane of the machine and enter the values into their correct places". He then realised that he could turn this physical design into something more abstract, replacing the crane with the concept of a 'selector' and the numeric values into what we would now call memory. There are clear parallels with the store and reader of Babbage's analytical engine, of which Zuse was (at that point) entirely unaware.

He captured the evolution of his thoughts and designs in his diaries, but we should put aside any notion that Zuse ever followed a straight line. In his autobiography, he describes his "many detours and by-ways"[7] during his university days, which included a year away from studying where he tried to earn a living designing adverts. He finally graduated with a diploma in 1935, eight years after starting at the Technical College. His parents must have breathed a sigh of relief when, soon after, he landed a job as a structural engineer at the Henschel Aircraft Company. Surely their mercurial son would at last settle down to a normal life.

This was not to be. His work called for yet more calculations, fuelling Zuse's desire to delegate such tedious work to a machine. By this time his parents had moved to an apartment in Berlin, and this was where – after quitting his full-time job at Henschel, although he would continue to work for the firm part-time – he "set up an inventor's workshop"[8] to turn his sketches and diagrams into reality.

"One day, shortly after he had received his degree and had been working as a structural engineer but for a few months, he explained to us, a few of his pals, that he was planning to build a universal[9] computing machine," wrote Andreas Grohmann, a fellow Motiver who had just qualified, at the age of 20, to be a mining engineer. "He was looking for helpers... I agreed."

Grohmann describes his "months working all day long with Zuse" to build the machine, across the summer and autumn of 1936 but primarily over the summer of 1937. Zuse's parents' sacrifice yet again comes into clear view through Grohmann's account: "He had set up a small workshop in a small room in his parents' apartment on Methfesselstraße in Berlin, and also used the large living room – now off-limits to his family – as a construction space for his machine."

Over the next two years, during evenings and weekends, Zuse would work tirelessly to create his first machine: what would retroactively be called the Z1. "He was really obsessed with his work," says computer scientist and historian Raúl Rojas.[10] "I think that this obsession is what explains how he could do so many things in 24 months, from 1936 to 1938. He had essentially finished [the Z1] before the war broke out."

Rojas argues that Zuse "founded the first computing startup". Every other computer covered in this series had the backing of a university, corporation, or government – sometimes a mix of all three – whereas Zuse financed his first computer through the generosity of his friends and his family. His father, who had retired from the Post Office, even returned to work for a year to raise funds for materials.

Zuse used his persuasive charm on friends such as Grohmann

5 Paul E Ceruzzi, 'The Early Computers of Konrad Zuse, 1935 to 1945', in IEEE Annals of the History of Computing, Vol 3, No 3, July 1981, p242

6 Paul E Ceruzzi, *Reckoners* (Greenwood Press, 1983, ISBN 978-0313233821), p11

7 Konrad Zuse, *The Computer – My Life*, p21

8 As before, p33

9 Andreas Grohmann is using some poetic licence here as the Z1 would not be considered as a 'universal' computing machine due to its lack of support for conditional branching.

10 Interview with author. Raúl Rojas has written the definitive book on the Z1, Z2, Z3, and Z4 architectures, *Konrad Zuse's Early Computers: The Quest for the Computer in Germany* (Springer, October 2023, ISBN 978-3031398759)

to donate both money and time. Rolf Edgar Pollems, another Motiver, wrote that "we younger students would have gladly helped him realise his dream of building an operational machine… we sawed out metal shapes with the fretsaw according to his instructions, and carried out other similar tasks to help him."[11]

This despite the fact that the theory behind the Z1's operation was well beyond their experience or understanding. "I'm honest enough to admit that I worked blind," wrote Grohmann. "I did not know precisely how this monster, that was taking shape there, was supposed to function. And yet, the machine was completed, worked with an unholy rattle, and supplied precise solutions to complicated tasks."

Indeed, it would have been beyond the ken of anyone who considered themselves an expert at the time, as Zuse took an unconventional approach to his machine. Until then, most calculators operated on the simple principle of repeated addition: if you wanted to multiply seven by six, you would start off with six in the machine's register and then repeat the addition process seven times.[12] By contrast, because Zuse's arithmetic unit worked in binary rather than decimal, multiplication becomes as simple as addition (in binary 1 × 1 = 1, 1 × 0 = 0, 0 × 0 = 0… while 1 + 1 = 10, 0 + 1 = 1, 0 + 0 = 0).

"The most astonishing thing for someone who studies what Zuse did is that he rediscovered so many things," says Rojas. Zuse had no mathematical training, was ignorant of academic developments in that field, so it was his engineer's instinct that pushed him to use binary rather than decimal, to devise his own square root algorithm (introduced in the Z3) and even his own logic.[13]

"Floating-point had already been thought of for calculators and he rediscovered that," says Rojas, "and he also discovered the algorithms for dealing with floating-point numbers. And that's the most amazing thing for me." Using floating-point numbers meant the Z1 and its successors could reduce complex real numbers such as 143.75 into component parts: 143.75 would become 1.4375×10^2.

Going it alone

Where Zuse's computers stand head and shoulders above early contemporaries such as the Harvard Mark I is the theory that underlies them. Remember, Zuse built his computers in near-isolation: he devised his own form of calculus, accidentally replicating the work of mathematicians. The only theory he drew upon, to the best of our knowledge, is Leibniz's work on binary mathematics.

The Z1 was a purely mechanical affair, but if you're imagining something akin to a typewriter then think again. This was a sprawling machine, with sliding metal rods to represent the zero or one of each 'bit' of memory, that consumed roughly the same amount of space as three typical dining tables placed side by side. In certain areas, it also stood three feet high.

If you can't visit the reconstruction of the Z1 at the Deutsches Technikmuseum in Berlin, which Konrad Zuse himself oversaw in the 1980s, then the next best thing is to watch the YouTube video.[14] This also does an excellent job of showing the sophistication of the Z1's design, complete with its separated memory store and arithmetic control units. When working smoothly, this could perform addition or subtraction in around two seconds, with division and multiplication taking a few seconds longer. Operators could enter different functions using a film strip punched with holes.

The big caveat there is 'when working smoothly'. While Zuse's friends did their best to follow the precise measurements for each metal part, they were still cutting by hand. And even the reconstruction, with its precision-cut components, proved problematic. Zuse, in typically self-deprecating style, wrote: "[The Z1] didn't work well at the time and the replica is also very faithful in this respect: it doesn't work well either."[15] Zuse could determine and correct the replica's faults, but since his death in 1995 no-one has been able to fulfil that role.

So, although a work of genius, the Z1 was not a practical device. Improvements were clearly necessary, so Zuse set to work on what has come to be known as the Z2. This moniker is a little misleading, as the Z2's main role was to act as a proof of concept that showed Zuse could replace the currently mechanical

11 Karl-Heinz Czauderna, *Konrad Zuse, der Weg zu seinem Computer Z3*, p85 (translation from German found in Konrad Zuse, *The Computer – My Life*, Springer-Verlag, p36)

12 There were more sophisticated methods around, including one where the sum was split into parts, but at this stage repeated addition (and subtraction) were the dominant and most cost-effective approaches.

13 Zuse called his logic Conditional Combinatoric (Bedingungskombinatorik), which he created to describe the operations his computer would process. This tied in (not that Zuse knew this) with a movement in mathematics to reduce all operations to base logical expressions (work done by David Hilbert and W Ackermann in *Foundation of Logic*, published in 1928, and separately by Alfred North Whitehead and Bertrand Russell in *Principia Mathematica*)

14 See it at **youtu.be/HDxs3-aJSAI**

15 This is a translation from the original German, which is: "Dieses Modell steht als Nachbau im Museum für Verkehr und Technik in Berlin. Damals hat es nicht gut funktioniert und auch der Nachbau ist insofern sehr getreu, auch er arbeitet nicht gut." From *Geschichten der Informatik: Visionen, Paradigmen, Leitmotive* (Springer, 2004, ISBN 978-3540002178), page 36

arithmetic unit with one built using electromagnetic relays. It was an idea he developed with his friend, Helmut Schreyer, who was studying telecoms engineering.

Why not vacuum tubes? These were suggested by Schreyer, but the pair were discouraged from taking this approach when they presented a prototype of how a vacuum tube switch might work to professors at the Technical University in Berlin. When Zuse and Schreyer explained that a complete computer would need 2000 vacuum tubes to work, they were met with shakes of the head. Too expensive, too unreliable.

Schreyer would continue his research into the vacuum-tube relay circuits, earning a PhD in Engineering in 1941 as a result, but it would be many years before Konrad Zuse's computers would incorporate them. In 1942, Schreyer even proposed creating a version of the Z3 with vacuum tubes to the German Army command, but they turned the idea down when Schreyer explained it would take two or three years to build such a computer. By this time, he was told, the war would be won.

Schreyer was a member of the Nazi party, but Zuse never joined. In the early 1930s, he had spent a few weeks of basic army training with the Reichswehr, the Reich Defence, but more as a preventative measure: others who had not shown loyalty to Hitler and the Third Reich were "being forced into line and marched off",[16] so he and eleven other classmates thought it better to volunteer than be commandeered.

Zuse was drafted into the army proper when Britain declared war in September 1939, after Hitler had annexed Czechoslovakia. By this time, he had almost finished the Z2, and though his patron Kurt Pannke – who ran a company that made specialist calculators and had given Zuse 7000 Reichsmarks to help him build the Z2 – wrote a letter to the authorities asking for Zuse to be granted leave so he could continue his work, this met with no success. It didn't help that one of Pannke's arguments was that the Z2 had applications in aircraft construction. On seeing the letter, the battalion commander summoned Zuse and asked him to explain himself: "The German Luftwaffe is top-notch; what needs to be calculated there?"[17]

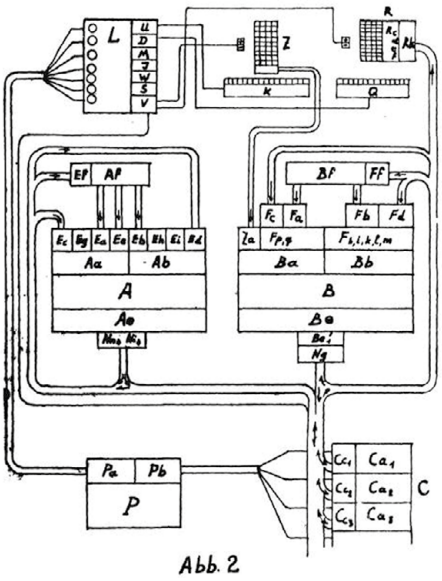
Zuse wisely declined to criticise the Luftwaffe's planes and his permission for leave was refused. Fortunately, he found a more sympathetic audience in his captain, who would sometimes allow the young inventor the chance to work on his ideas in the quiet of the captain's room. Nor did Zuse face any combat action, using his six months in the infantry to work on his ideas and to "formalise the rules of chess using mathematical logic".[18]

His time in the army ended when Professor Herbert Wagner, who was developing remote-control flying bombs at the Henschel Aircraft Company, recruited Zuse as a structural engineer. By all accounts Wagner was a brilliant man, and one of many German scientists who migrated to the USA immediately after the war to help their military efforts.[19]

Now back in Berlin, and with evenings and weekends free, Zuse could continue to develop the Z2 whilst working for Wagner. By September 1940, the computer was ready for its first official demonstration, to Professor Teichmann of the German Aviation Research Institute (the Deutsche Versuchsanstalt für Luftfahrt, shortened to DVL). "Just hours before then I was still frantically trying to get the machine to run, but it was useless," wrote Zuse. "The guests arrived – I was sweating blood – and behold, my machine performed flawlessly."[20]

This demonstration proved enough to persuade Teichmann

16 Konrad Zuse, *The Computer – My Life*, p30



◀ Z3 architecture sketch from the 1941/1950 patent application

17 As before, p57

18 As above, p57

19 Zuse was one of several people who helped Wagner develop the Hs 293 guided missile. Although Zuse downplays the missile's success in his autobiography, saying the Allies learned to jam the radio control systems, this only happened after it had sunk several ships. But as is also clear from the book, Zuse did not appreciate the horrifying extent of the Nazi regime, with rumours of concentration camps being easily dismissed as foreign propaganda. It seems likely that it was only as the war approached its end, when Zuse saw emaciated workers for himself, that he fully appreciated what had been going on.

20 Konrad Zuse, *The Computer – My Life*, p61

and the DVL to sign a contract with Zuse to complete work on the Z3 – from an architecture point of view, the same as the Z1, except now with a square root algorithm. Rojas describes this algorithm as the "jewel in the crown of the Z3",[21] avoiding the traditional approach of guess-and-guess-again until you don't get an error. Instead, the calculation stepped through a precise 18 cycles to determine the square root of any number.

But the chief difference between the Z1 and the Z3 was the use of electromagnetic relays to represent zeroes and ones rather than clunky metal plates. Astonishingly, Zuse completed it before the end of 1941, and at that point the Z3 was arguably the most technologically advanced computer in the world.

Not that you would know this to look at it. Assembled from second-hand materials of various provenances, the relay coils needed much fine-tuning to ensure they activated at the same time. It's one of the reasons why Zuse was the only person who could service the machine when it was complete. Despite this, the Z3 was "relatively reliable", Zuse wrote,[22] although we should bear in mind that the relative in question was the Z1.

He is arguably the father of the idea of 'digital physics', a Matrix-style concept that suggests the universe could simply be a computer program

Go big (or small)

Like the Mark I, the Z3 could calculate with the minuscule and huge numbers necessary for scientific work. It was certainly more than capable of coping with calculations associated with statics calculations in aircraft manufacture, which is what the DVL was interested in.

In his 1997 paper,[23] 'Konrad Zuse's Legacy: The Architecture of the Z1 and Z3', Rojas goes into detail on how this worked. In particular, he explains how "the smallest number representable in the memory of the Z3 is $2^{-63} = 1.08 \times 10^{-19}$, and the largest is $1.999 \times 2^{62} = 9.2 \times 10^{18}$". That's an enormous range.

Still, though, the Z3 had obvious limitations: not due to the size of the numbers it could store but to how many it could store. This was primarily due to the scarcity, price, power demands, and size of electromagnetic relays. Unlike Howard Aiken when building the Harvard Mark I, with the luxury of IBM's workshops and financial backing, Zuse had to pay for everything out of the scant money provided by the DVL and his own backers. Using Zuse's notes from the time, Ceruzzi estimated the cost at around $2 per relay, which was a huge sum in 1940s Germany.

The Z3's storage was split across two cabinets, with each holding 32 columns of 28 relays (896 relays in total). Because six of the relays per column were needed for operation, this left 22 relays per column, each of which could represent a one or a zero. So one cabinet could hold 32 numbers represented by 22 zeros and ones (we would now call these 'bits').

While 64 numbers sounds like a lot, it wasn't enough for the complex calculations required by the DVL. To tackle these, the computer needed to hold hundreds of variables within its storage at any one time, and so while the Z3 worked as an excellent proof of concept it wasn't the real deal. While Zuse could show that it was capable of tackling the type of problems the DVL was interested in, such as calculating the determinant of matrices, they were simplified versions that could already be solved efficiently by humans.

Still, it's worth admiring what Zuse had created and what it looked like. To start with, equations could be fed into the Z3 using punched tape. This was standard celluloid film, as used in movie theatres, but the reader that scanned in the instructions (zeros and ones) was custom-made. This equation was then sent to the arithmetic unit, so that it knew which calculations it needed to perform.

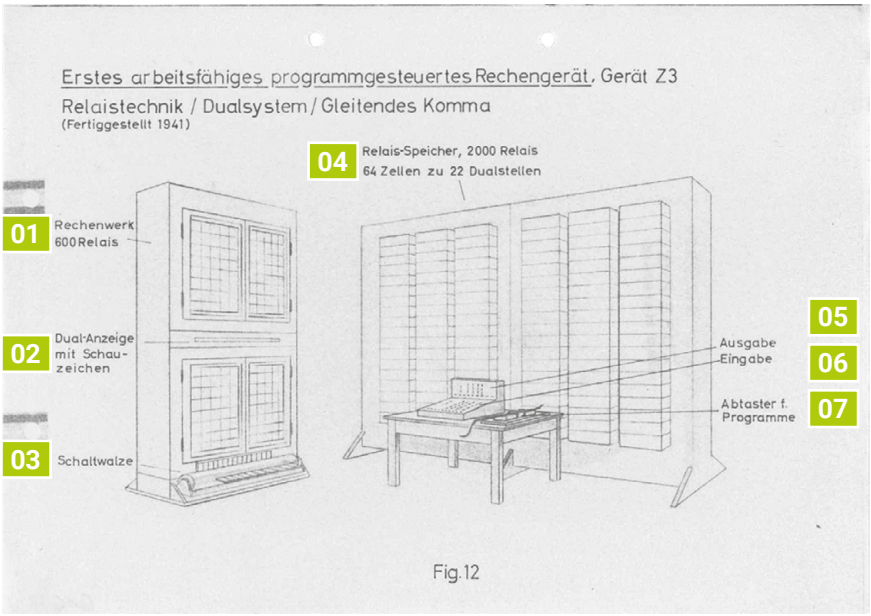
The next step was to load the equation's variables into the memory banks (the 64 columns mentioned above). To do this, operators used a keyboard to enter the numbers as decimals. The Z3 handled conversion from decimal to binary, but they needed to be entered as four digits to the power of ten (or minus ten). They were accurate to four decimal places and ranged from 0.000000001 (a nanometre, or 10^{-9}) to 1,000,000,000 (one billion or 10^{9}). Once the program started running, new numbers would be sent to available, empty storage banks.

Just like modern computers, the Z3 ran on a cycle. A smartphone made in 2025 typically includes a processor that runs at around 2.5GHz, which is 2.5 billion cycles per second. By contrast, the Z3 ran at around five cycles per second. According to Raúl Rojas, addition took three cycles on the Z3, multiplication took 16 cycles, division 18 cycles, and a square root 20 cycles. That

21 Raúl Rojas, 'Konrad Zuse's legacy: the architecture of the Z1 and Z3', in IEEE Annals of the History of Computing, 1997, Vol 19, Issue 2, p13

22 Konrad Zuse, *The Computer – My Life*, p63

23 Raúl Rojas, 'Konrad Zuse's legacy: the architecture of the Z1 and Z3', in IEEE Annals of the History of Computing, 1997, Vol 19, Issue 2, p7



◀ First functional program-controlled calculating device, device Z3 relay technology / dual system / floating point (Completed 1941); sketch from 1941/1950 patent application

01. **Rechenwerk**: 600 Relais – Arithmetic unit, 600 relays

02. **Dual-Anzeige mit Schauzeichen** – Dual display with indicator

03. **Schaltwalze** – Shift drum (impulse generator)

04. **Relais-Speicher, 2000 Relais 64 Zellen zu 22 Dualstellen** – Relay memory, 2000 relays, 64 cells (i.e. words) of 22 dual positions (i.e. bits)

05. **Ausgabe** – Output

06. **Eingabe** – Input

07. **Abtaster f. Programme** – Scanner (punched tape reader) for programs

translates into roughly a second, three seconds, four seconds, and five seconds respectively.

After an appropriate wait – the programs would involve a complex series of such calculations – the answer would appear on the display. Like the input, this would be in decimal, and appear as a series of four digits (via labelled bulbs) times 10^{-12} to 10^{12}. Looking at a photo of the reconstructed Z3, it's striking how much more compact it is than the Mark I or ENIAC.

The replica only includes one storage unit whereas the actual Z3 included two, each filling a cabinet roughly the size of a wardrobe. The other unit contains the arithmetical unit – consisting of around 600 relays – the tape reader, and the control unit, which needed roughly 200 relays to convert punched instructions into code.

It's an incredibly impressive technical feat, and a replica is normally on display as part of the Computers Exhibition at the Deutsches Museum in Munich (the exhibition is closed due to refurbishment at the time of writing, but due to reopen in 2028). You can also find a replica at the Konrad Zuse Museum in Hünfeld.

Detractors of the Z3 will point out that it lacks two key features: the ability to store programs and conditional branching. The former is inarguable – but hardly surprising – while the lack of conditional branching is primarily due to the lack of relays available to Zuse. He was certainly aware of the theoretical value of creating programs that could work on the basis of 'if this happens then do that'.

The Z3 was effectively a proof of concept, something to demonstrate the power of a programmable, electromechanical binary computer, and it served this purpose well. For example, records exist from a visiting group of mathematicians who were suitably impressed when it computed a determinant on demand.

The DVL would also benefit from its development, with Zuse creating a special-purpose version of the Z3 – called the S1 – that was used to calculate the deviations of cheaply produced wings on the Henschel Hs 293 flying bomb. According to Ceruzzi, when completed in 1942 the S1 "replaced a staff of 30 to 35 women who had worked around the clock with electric calculating machines".[24]

By this time Zuse had assembled a team of people to work with him, and they set to work building the Z4 for the German military. It was many times bigger than the Z3 due to its far increased mechanical memory capacity, but the biggest hindrance proved to be the increasingly frequent and more intense Allied air raids on Berlin. Zuse had to switch his workshop's locations several times.

Zuse and his team were close to completing construction of the Z4 by early 1945. However, the authorities decided it should be moved away from its increasingly perilous position in Berlin; Allied bomb attacks had already damaged the workshops where construction was taking place, and the Z3 had been destroyed in an air raid the previous year. They initially sent the Z4, and Zuse, to the safer university city of Göttingen, where he and his team managed to complete construction and run test programs.

"This was the moment for which I had waited for ten years

24 Paul E Ceruzzi, 'The Early Computers of Konrad Zuse, 1935 to 1945', in IEEE Annals of the History of Computing, Vol 3, No 3, July 1981, p256

– when my work finally brought the success I desired," wrote Zuse.[25] "It was now very tragic that precisely in these days the Americans stood ready in [the nearby town] Kassel."

So it was time to move on once more, this time to Hinterstein, a picturesque village tucked in the Bavarian Alps. For the next three years, the Z4 would sit largely forgotten in a cellar. But not entirely forgotten. In 1947, as part of a team of Allied investigators, American mathematician Roger Lyndon interviewed Konrad Zuse to discover more about this rumoured German computer. He also saw the machine, but not in action.

Post war blues

Zuse was not inclined to divulge all his secrets to investigators, having had his computers, workshops, and his apartment – along with his parents' apartment – damaged and even destroyed by Allied bombers. And with severe restrictions on post-war Germany in terms of scientific research and development, it's unlikely he was keen to share his full ideas with American visitors.

Still, it's interesting to read Lyndon's assessment of the computer in his simply titled 1947 article, 'The Zuse Computer'.[26] While he sticks mainly to a prosaic description of the computer and its abilities, he is critical of its "rather rough homemade components" and says an "important limitation upon programming is that the machine must adhere to a prescribed linear course of operation". In other words, no conditional branching.

Lyndon also provides an insight into the conditions Zuse was working under, which he describes as "rather primitive", adding that Zuse is working "with inadequate material". He adds that Zuse only has one assistant to help him.

Even if Zuse's hands were tied from a physical viewpoint, his mind was busy. During the war, he had begun work on his own programming language, and this would evolve into what he called Plankalkül – 'plan calculus' in a direct English translation, but perhaps better understood as a 'calculus for computing plans'. This wasn't about numbers but about logical problems; Zuse was trying to formalise a way to tackle them using what we would now call a programming language. For instance, he set out – in enormous detail – how you would use Plankalkül to create a chess program.

While modern-day programmers would not recognise the Plankalkül's syntax, it set out concepts that are now commonplace – particularly variables and subroutines. Zuse's original intention was to turn this into a thesis for a PhD, but he instead wrote an unpublished book on the subject, which he finished in 1945. Three years later, Plankalkül made its public debut in a paper Zuse presented to the Annual Meeting of the German Society of Applied Mathematics and Mechanics in 1948, and although a paper was subsequently published it failed to garner any attention.

This would irritate Zuse in later life, but better news was around the corner thanks to a visit from Professor Eduard Stiefel of the Swiss Federal Institute of Technology (in German, die Eidgenössische Technische Hochschule Zürich, which we'll shorten to ETH). Stiefel had just returned from a tour of America's computers, in search of technology to replace the desktop calculators the ETH currently relied on. Zuse quotes Stiefel as saying he had seen "many beautiful machines in beautiful cabinets with chrome work,"[27] with the latter no doubt including the Harvard Mark I with its $50,000 cover.

Cellar dweller

Stiefel had heard of a German computer tucked away in a mountain village, and must have been thoroughly shocked when he saw the Z4 in a state of disrepair hiding in a cellar. But it worked. Zuse wrote that Stiefel "dictated a simple differential equation, which I was able to program immediately, feed into the computer, and solve".[28] This, Zuse reports Stiefel as saying, was the first time that anyone had been able to complete the problem so quickly.

According to Ambros Speiser, however, it wasn't Stiefel who had physically seen the American computers but his two assistants, Speiser himself and Heinz Rutishauser. In the book *The First Computers: History and Architectures*, Speiser goes into great detail on how the Z4 came to arrive at the ETH.[29]

The pair received a letter from Stiefel describing the Z4 and asking them to gain Professor Howard Aiken's view of the computer. "Aiken's reply was very critical – the future belongs to electronics and rather than spending time on a relay calculator, we should now concentrate our efforts on building a computer of our own," wrote Speiser. "We reported Aiken's opinion to

25 As before, p57

26 Roger C Lyndon, 'The Zuse Computer, Mathematical Tables and other Aids to Computation', in The National Research Council, Vol 2, No 20, October 1947, pp355-359

27 Konrad Zuse, *The Computer – My Life*, p118

28 As above, p118

29 Ambros P Speiser, 'Konrad Zuse's Z4: Architecture, Programming, and Modifications at the ETH Zurich', in Raúl Rojas and Ulf Hashagen (eds.), *The First Computers: History and Architectures* (The MIT Press, 2000, ISBN 978-0262181976), pp263-276

Stiefel stating, however, that we did not fully agree with him and that, in our opinion, the proposition should certainly not be flatly rejected."

Stiefel, being in the business of solving mathematical problems, agreed. And he literally decided to put parallel computing into action: the ETH would pay Zuse 50,000 Swiss francs up front and effectively rent it for five years. And they would have the option to pay a further 20,000 Swiss francs at the end of that period to buy the computer outright. At the time, these were enormous sums.

The parallel computer was the ETH's own electronic machine, which Stiefel knew would take a few years to build. This would become the ERMETH, Switzerland's first electronic computer, built upon similar lines to the EDVAC and EDSAC (therefore using von Neumann architecture). According to Hans Rudolf Schwarz's account in the IEEE Annals of the History of Computing,[30] it ran from 1956 to 1964.

This means the Z4 was the ETH's main computer for around six years, and it was put to excellent use. Schwarz describes how it checked the security of a dam, the deformation of a Swiss railway bridge, and was used in the design of the first Swiss military plane that reached supersonic speeds.

When Stiefel had seen the Z4 in Zuse's cellar, it was functioning but not complete. Part of the contract between Zuse and the ETH was to both finish the machine and add capabilities that the university would find useful, most notably conditional branching, and this meant it wasn't installed in Zurich until August 1950.

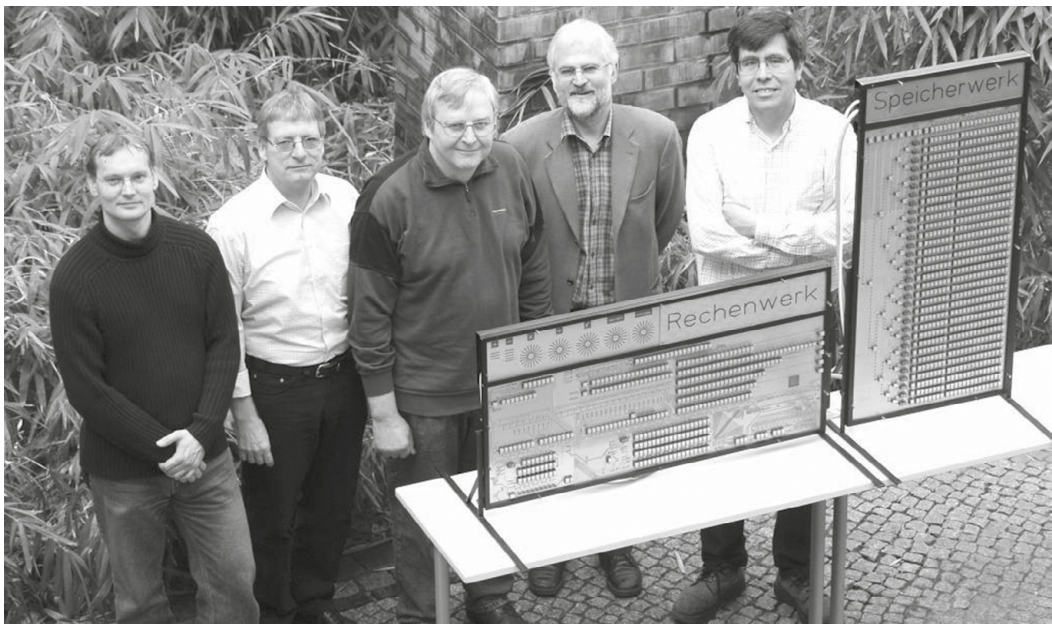
These funds also allowed Zuse to invest in his own computer company, which would go on to build numerous relay computers until the mid-1950s before moving on to electronic computers. Due to cash-flow problems his company was eventually bought by Siemens, ending Zuse's entrepreneurial career but by no means his interest in computing. He is arguably the father of the idea of 'digital physics', a *Matrix*-style concept that suggests the universe could simply be a computer program.

So where does Konrad Zuse, and the Z3 computer in particular, stand in the pantheon of creators? "Intellectually, I wouldn't put him up there with John von Neumann and Alan Turing," says Paul Ceruzzi,[31] who interviewed Zuse in the 1980s and has written almost 20 books on the history of computing. However, Ceruzzi emphasises that Zuse deserves credit for the logical design of the Z3. "In the early days, most people designed [computers] on an ad-hoc basis, and didn't have much theory. He had theory, which I think put him way ahead of other people, although he never got credit for that."

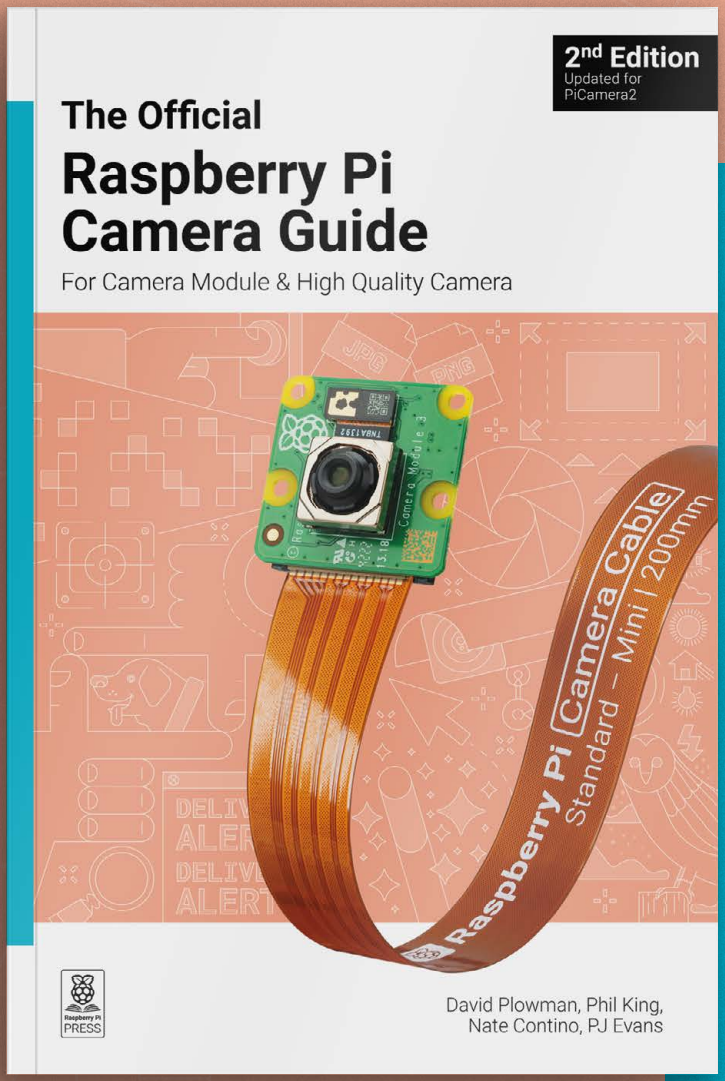
Raúl Rojas agrees, and puts forward a reason why. "The problem with Zuse is that he did astonishing things, but nobody knew what he was doing outside of a small circle in Germany because of the war. He was not communicating with anyone, not even outside of Berlin." Compare this, says Rojas, with his British counterpart. "[Turing] wrote his papers, he had immediate success, he became very well-known and he was also building computers himself. Zuse was rediscovered [outside Germany] in the 1950s and that was too late for him to have a big impact."

30 H R Schwarz, 'The Early Years of Computing in Switzerland', in IEEE Annals of the History of Computing, Vol 3, Issue 2, April 1981, pp121-132

31 Interview with author



▼ The Z3 rebuilt in 2001 for the 60th anniversary of the patent application
Image: courtesy of Georg Heyne



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**

Miniature rooms

Downsize your world because good things do come in small packages

Maker

Nicola King

Nicola is a freelance writer and sub-editor. This month she wishes she was six inches tall, as Lilliput is quite literally down the road from her (it really is… it's a place in Dorset!)

@nicoladj100

In case you weren't aware, miniature rooms and doll's houses in general are having something of a 'moment', with makers young and old alike. Your author was first alerted to the trend a couple of years ago while wandering around our local garden centre. For those not based in the UK, garden centres here often seem to make more revenue from 'bought in' goods – such as jigsaws, and overpriced candles, toiletries, kitchenware, and leather goods – than they do from plants and gardening paraphernalia. Their tearooms are often quite enticing too!

The items that caught our eye on that particular trip were a range of tiny make-it-yourself room 'scenes', which incorporate very detailed features including things like little books, a diminutive writing desk, and even some wee wall art in one library kit that we saw. Basically, the attention to detail is meticulous and impressive and that might be one reason why these scaled-down scenes have become so popular. Making something like this is a way of escaping the real world and transporting yourself to a tiny alternative world, perhaps with a vintage theme that's comforting, cosy, and tranquil. Many people just find joy in such charming and adorable little things, but these kits also leave some room for creativity on the part of the maker, and that's what we want to explore here.

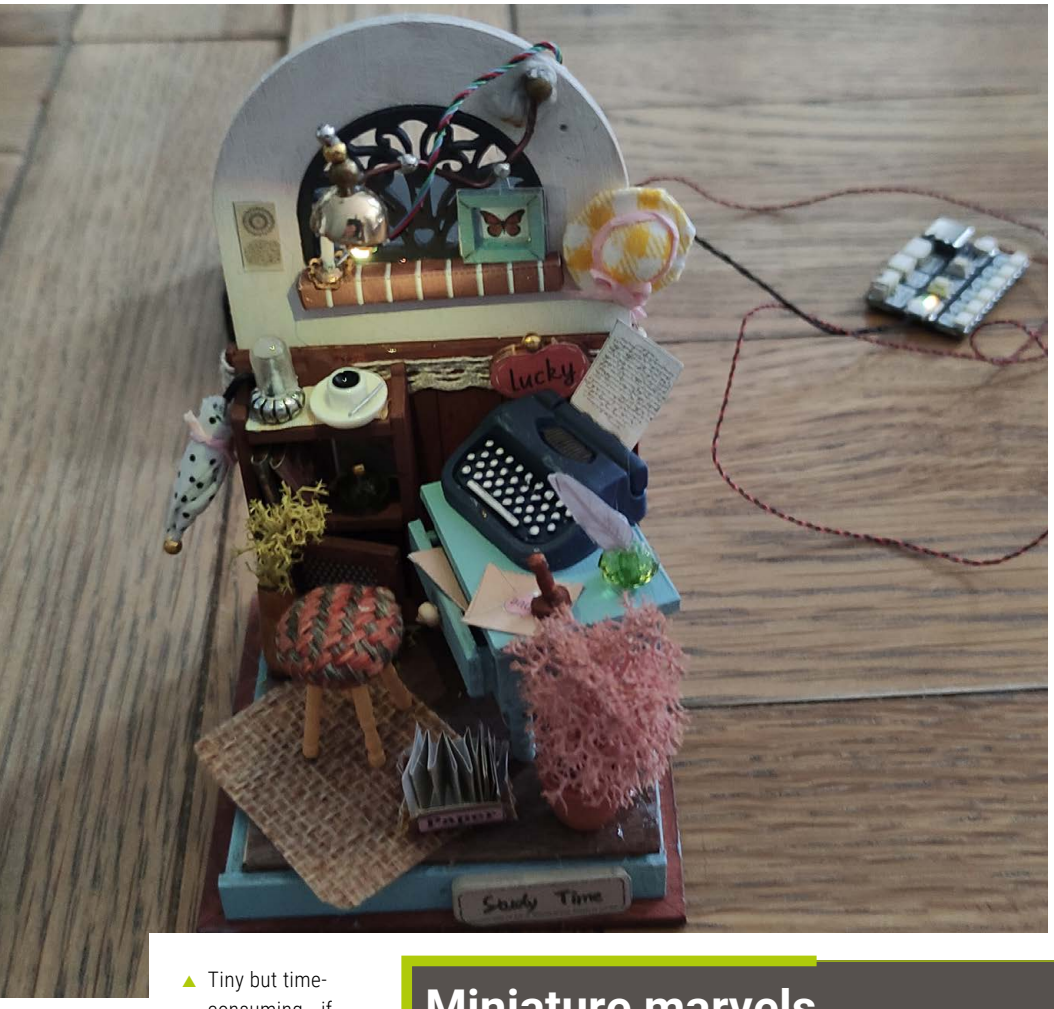
QUICK TIP

We found that superglue may be more effective to use than any glue provided in a kit, so have some to hand.

Scaled-down study

We decided to make up a small shop-bought kit of a cheerful little retro study, complete with desk and typewriter. One thing we would advise from the start is to ensure that you leave enough time to assemble and finesse your miniature. Despite the fact that the box for this kit was tiny, there were lots of elements to making it up that took more time than we had bargained for. We made an assumption that it would be a quick make – you'd think we'd know by now never to make assumptions. For example, the kit came with paint and a paintbrush, and a number of items required delicate painting.

These kits also leave some room for creativity on the part of the maker



▲ Tiny but time-consuming… if you've ever made anything miniature before – a house, plane, or car, for example – you will know that minutes turn into hours, turn into days, and so on

QUICK TIP

Keep any leftover kit pieces for future projects – we had lots of fabric, wire, and other bits left for our next project.

YOU'LL NEED:

- A miniature room kit or doll's house
- Tiny FX, **rpimag.co/tinyfx**
- RGB LED dot, **rpimag.co/tinyfxrgb**
- Battery pack or USB power supply
- Tweezers
- Craft knife
- Small and large scissors
- Ruler/tape measure/pencil
- Magnifying glass (for detailed work)
- Glue selection, e.g. wood, fabric, tacky, super
- Sandpaper

Miniature marvels

Of course, miniature rooms and houses are nothing new and their history is thought to go back at least as far as 400 years. In fact, the considered view is that the original doll's houses were primarily designed for adults and were markers of wealth and status.

Cabinet doll's houses were popular in Germany and the Netherlands, for example, as a way of showing how a household functioned and as status symbols. The film *The Miniaturist*, based on the novel of the same name, is set in 17th century Amsterdam, and is inspired by the doll's house of rich merchant's wife Petronella Oortman, who had a miniature house crafted over a number of years. The house was extremely lavish, and she had it decorated with some very expensive materials.

Continuing the extravagant theme, if you have ever visited Windsor Castle, then you may remember seeing the historically significant Queen Mary's Doll's House (**rpimag.co/qmdollshouse**), generally regarded as one of the most famous miniature houses in the world, and when you see it in person, the craftsmanship is certainly very impressive. In fact, 2024 was the 100th anniversary of Queen Mary's Doll's House. Designed by Sir Edwin Lutyens, and made to one twelfth scale, it is housed in a huge glass cabinet that you can walk around to view every minute Edwardian detail. Conservators regularly check the contents in case repairs are needed, and it's mesmerising. The doll's house even includes electricity and running water!

If you are looking for some inspiration on television, we can highly recommend *The Great Big Tiny Design Challenge*, a Channel 4 series hosted by Sandi Toksvig, where miniature-making skills are tested as crafters work to try and transform a doll's house that just happens to be a mini mansion.

We used this tiny typewriter to write this tutorial… only kidding, but we do wish we could shrink down and live in this lovely space

Clearly, you then need to allow time for the paint to dry, so just factor that in. It's also worth mentioning that, fortunately, this author had her eyes tested a month or so ago, so glasses are up to spec – we're just underlining that working with such small pieces can be problematic if your eyes are struggling. A magnifying glass may help.

The basic kit took a couple of days to complete in total, due to waiting for paint and glue to dry, but it was worth taking the process slowly – you may as well invest some time in order to do it properly. Plus, if you have a modelling toolkit from previous forays into making models, then do dig it out. One point we would emphasise is to read the instructions through before you get started, as this will aid the process immeasurably. Also, we found a video on YouTube of someone making the exact same kit as ours, so that was handy when we had queries. At the end of the day, remember that when you are doing something like this, the whole point is to enjoy it and slow down, not to stress yourself out.

Create then illuminate

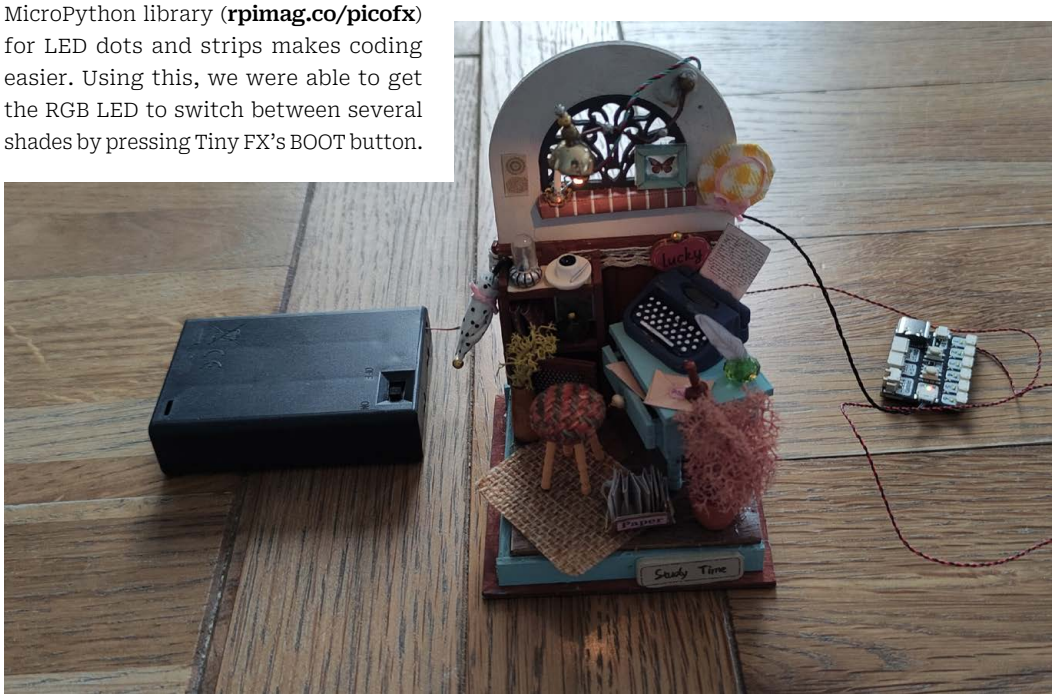
To make your scene really come to life, the optional final step is to light it up. Some kits come with LED lighting included, but ours didn't. You could use a Raspberry Pi Pico to control one or more LEDs, as in our origami tutorial in issue 155, but we wanted a tiny LED dot to fix to the lamp in our scene, so we opted to use Pimoroni's RP2040-based Tiny FX microcontroller board (**rpimag.co/tinyfx**). Its starter kit includes several single-colour white LEDs and RGB ones with thin wires and JST-SUR connectors to plug into the board, along with a 3 × AA battery pack. It's great for LEGO lighting too.

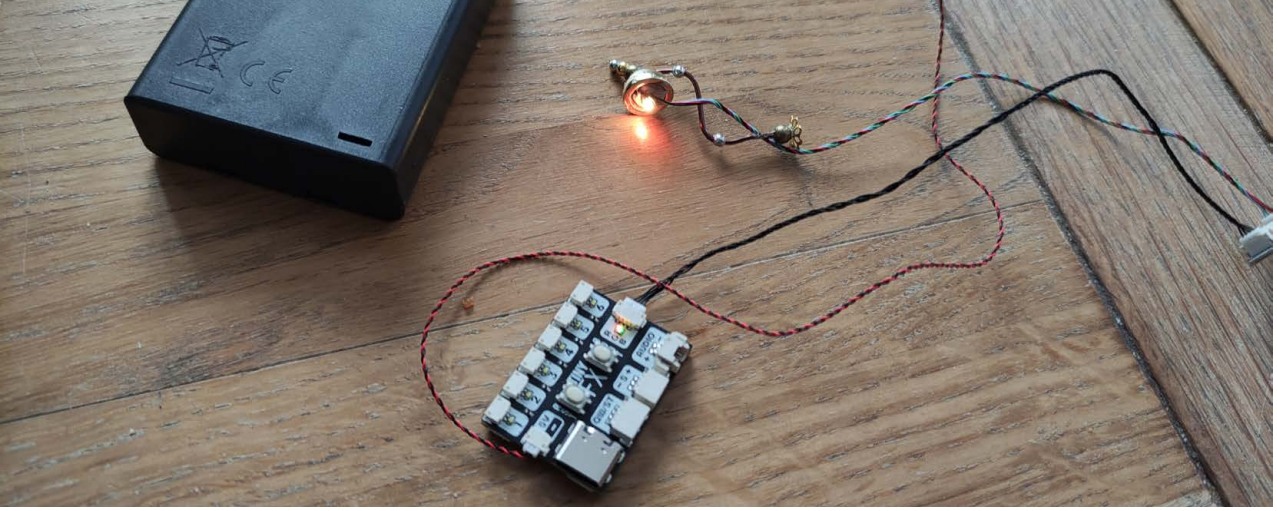
We attached an RGB LED dot to our scene's wall lamp. Pimoroni's PicoFX MicroPython library (**rpimag.co/picofx**) for LED dots and strips makes coding easier. Using this, we were able to get the RGB LED to switch between several shades by pressing Tiny FX's BOOT button.

QUICK TIP

Projects like this are not suitable for very small children due to the tiny pieces involved, so do bear that in mind before involving youngsters in your make.

We used an RP2040-based Tiny FX board and battery pack to light up our scene





▲ We used Blu Tack to stick our LED dot into the tiny lamp; you could use glue instead

Here's our code:

```
from tiny_fx import TinyFX
from picofx.colour import RGBFX
from picofx import ColourPlayer
import time

# Setup
tiny = TinyFX()
player = ColourPlayer(tiny.rgb)

# Define a list of RGB tuples
colors = [
    (128, 64, 0),      # Orange yellow
    (112, 80, 0),      # Yellow
    (104, 64, 24),     # Warm white
    (72, 72, 48),      # Cold white
    (0, 0, 0),         # Off
]

idx = 0
player.effects = RGBFX(*colors[idx])
player.start()

try:
    last_state = False
    while True:
        pressed = tiny.boot_pressed()
        if pressed and not last_state:
            # Button just pressed: cycle color
            idx = (idx + 1) % len(colors)
            player.effects = RGBFX(*colors[idx])
        last_state = pressed
        time.sleep(0.05)

finally:
    player.stop()
    tiny.shutdown()
```

Here, we use a list called `colors` to store a range of RGB tuples for different shades (after experimenting to find ones we like). An `idx` variable, used to select a colour from the list, is incremented every time the BOOT button is pressed during our infinite `while True` loop. We also use a `last_state` variable and short time delay as a debounce mechanism to prevent multiple button presses being registered when pressing once. As usual, save the code as **main.py** so it'll run automatically on bootup. The end result: our miniature room can now be lit up subtly in different shades to suit our mood.

▲ This is a different kit, but it gives you a flavour of what you'll find in the box: everything from paper cutouts to laser-cut and plastic pieces

QUICK TIP

You could use wallpaper samples or scrapbooking paper for wall coverings, and carpet samples for the floors of your room/house.

To sum up, we'd highly recommend going small and just spending some time thinking about nothing else other than painting tiny things and squinting hard to get that dinky coffee cup stuck down in the right place… it's mindful and it's enjoyable. Bringing Raspberry Pi RP2040 or Pico into the mix is easy to do, whether that's with some lighting or even perhaps some sound effects… and on that note, we'll be delving into the realm of adding sound to a crafting project in a future tutorial, so stay tuned!

Lilliputian options

1. Get kitted out

We took the easy option and bought a DIY kit. There are numerous places to find kits online; our only advice would be to opt for one that has some decent instructions so you are not left frustrated and regretful within five minutes of opening the box. Popular brands include Rolife (**robotimeonline.com**) and CuteBee (**cutebee.net**).

2. Be your own architect

If you don't want to buy a miniature kit and you have the time required, how about designing and constructing your very own tiny house, shop, or room? Researching the topic, this author realised that she had written a project showcase on the very subject some seven years ago. Two creative students, Maks Surguy and Yi Fan Yin, created an interactive doll's house for their master's degree course using machine learning. Take a look here at their clever work for some itty-bitty inspiration: **rpimag.co/myhouse**.

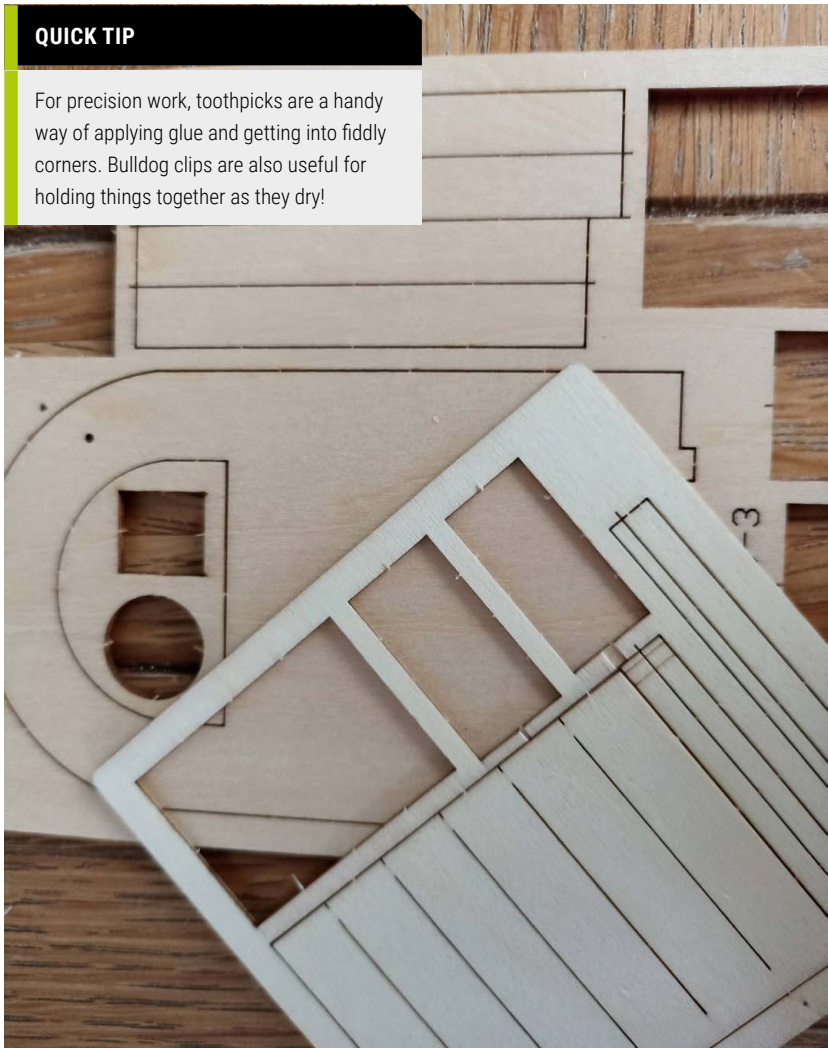
Making your own miniatures means you can use whatever materials you like in your makes, perhaps recycling fabrics, cardboard, plywood, or paint that you have in your stash. The kits mentioned above do tend to include a fair bit of plastic, so the recycling route might be more planet-friendly. Plus, the creative freedom that you have will know no bounds! Use some air-dry clay to make fun-size food items, use fabric scraps to make a mini quilt for a bed or scaled-down sofa… you get the idea. You could even make a version of your childhood bedroom to keep for posterity. So, base it on something you know as a guide.

▶ Kits these days rely heavily on laser-cut pieces, making the process a little easier as things fit together with precision

▼ As you can see, it can be messy, so protect your surfaces with some old newspaper

QUICK TIP

For precision work, toothpicks are a handy way of applying glue and getting into fiddly corners. Bulldog clips are also useful for holding things together as they dry!



Raspberry Pi's GPIO header allows you to connect electronic components and control them with code you've written yourself. Python is the most popular programming language for controlling electronics on a Raspberry Pi, particularly the functions in the GPIO Zero library. With this book, you'll learn how to use GPIO Zero as you build a series of simple electronics projects.

- ***Simple electronics projects including:***
  - *Program some LED lights*
  - *Add a push button to your project*
  - *Build a motion-sensing alarm*
  - *Create your own distance rangefinder*
  - *Make a laser-powered tripwire*
  - *Build a Raspberry Pi robot*

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

Animate a Halloween skeleton

Build your own motion-activated spooky decoration

Maker
Ben Everard
Ben saves money on Halloween treats by scaring the children away.

glowingart.co.uk

YOU'LL NEED

- 4 × 9g servos
- Raspberry Pi Pico (any version)
- Halloween skeleton
- 2 × WS2811 fairy lights
- Distance sensor
- Backing wood (ply or OSB)
- Screws
- Black paint

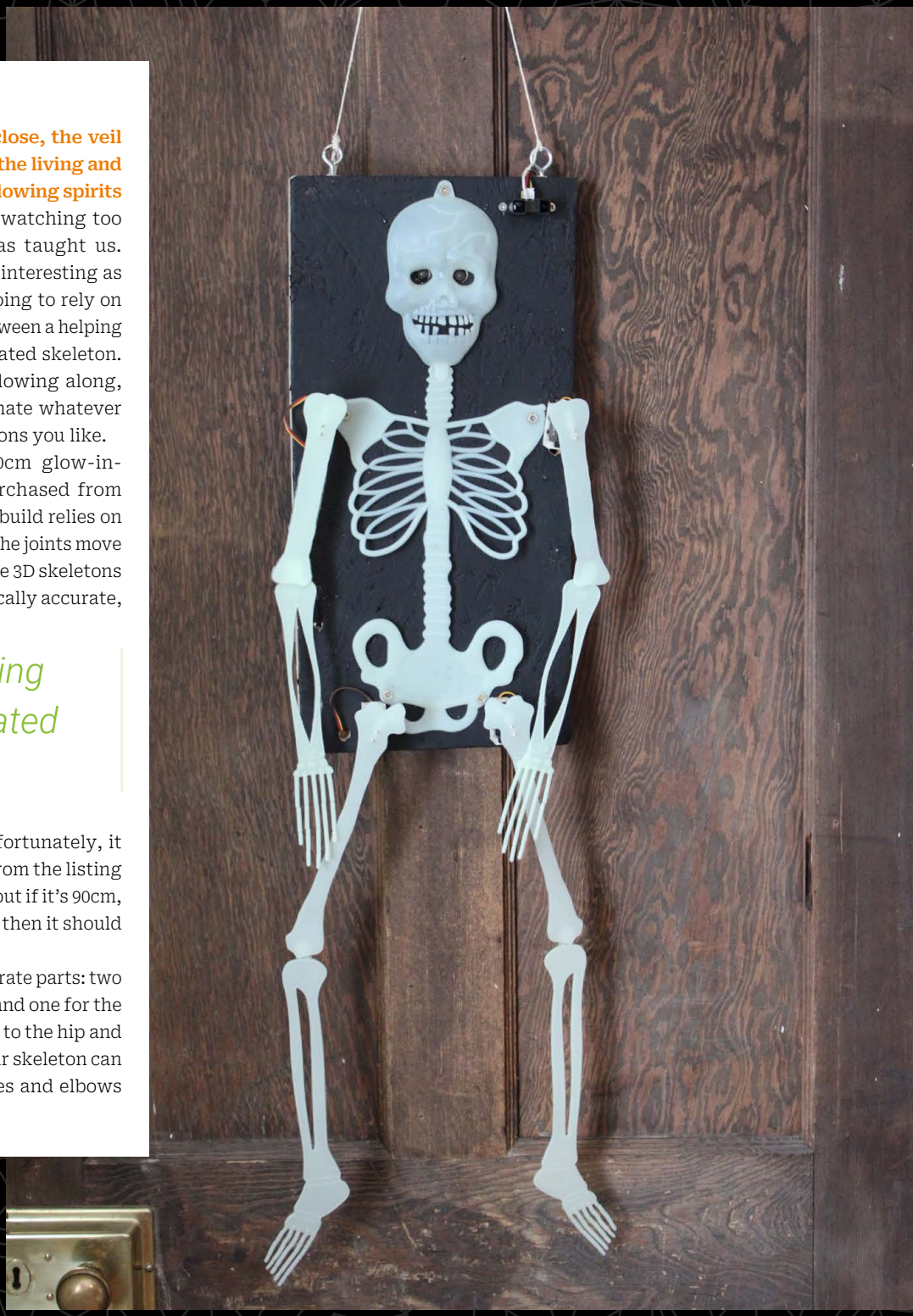
As October draws to a close, the veil between the worlds of the living and the dead grows thin, allowing spirits to cross. At least, that's what watching too many horror/comedy films has taught us. Since the real world is rarely as interesting as Hollywood depicts, we're not going to rely on the dead rising up, and give Halloween a helping hand by creating our own animated skeleton. You can build your own by following along, or adapt the techniques to animate whatever spooky (or not spooky) decorations you like.

Our build starts with a 90cm glow-in-the-dark skeleton that we purchased from AliExpress. The key part of this build relies on it being a '2D' skeleton where all the joints move in the same plane. There are some 3D skeletons available that are more anatomically accurate,

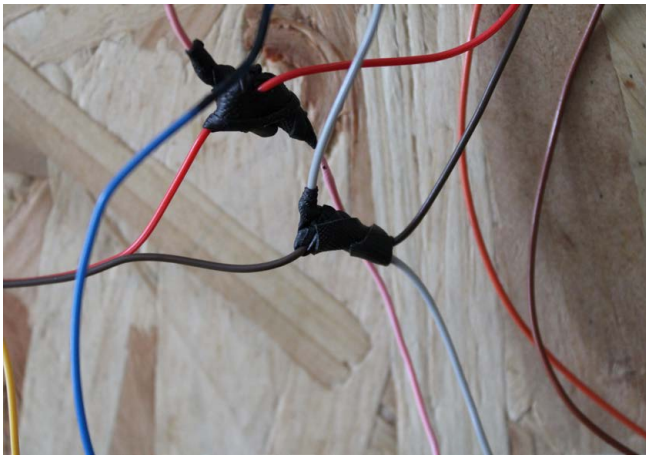
> *We'll be creating our own animated skeleton*

but also harder to animate. Unfortunately, it can be a little hard to work out from the listing whether it's a 2D or 3D skeleton, but if it's 90cm, glow-in-the-dark, and about £3, then it should be the right one.

The skeleton came as ten separate parts: two for each limb, one for the body, and one for the head. We're going to add control to the hip and shoulder joints, meaning that our skeleton can perform a few moves. The knees and elbows we'll leave loose to flail around.



▶ Our assembled spooky skeleton



▲ There are – arguably – better ways for joining four wires together and insulating the joint, but they all require more parts and complexity than a solder joint and self-amalgamating tape.

▲ Always make sure you position your servos before connecting the horns

Before we get to all that though, we need a few additional parts. The skeleton is very flimsy, so we need a firm backing that we can place it on to keep everything in place. We used Oriented Strand Board (more commonly known as OSB) because we happened to have some available, but plywood would also work well. It needs to be at least 45 × 21cm, though if it's bigger, that's fine (and might protect your skeleton a bit more). We painted ours black to blend into the dark night.

For the motion, we used servos. Servos are a little like motors in that you send a signal to make them turn, but unlike motors, they are used for turning a particular amount and usually operate a range of about 180 degrees. You send the servo a signal to let it know the angle you want, and it turns to that angle.

Servos come in different sizes and strengths, and we opted for 9g servos which are just about strong enough for this. If you use anything bigger, you might find that they draw too much current, so we'd recommend sticking with little ones unless you want to add an external power supply.

We wanted our skeleton to react to people around it, so we added a distance sensor to detect when someone is close to it.

Finally, we need a controller to make everything do what we want to. A Raspberry Pi computer will do the job, but for simplicity, we've used a Pico. Ours is a Pico W, but we're not using the wireless interface and any model should do just fine.

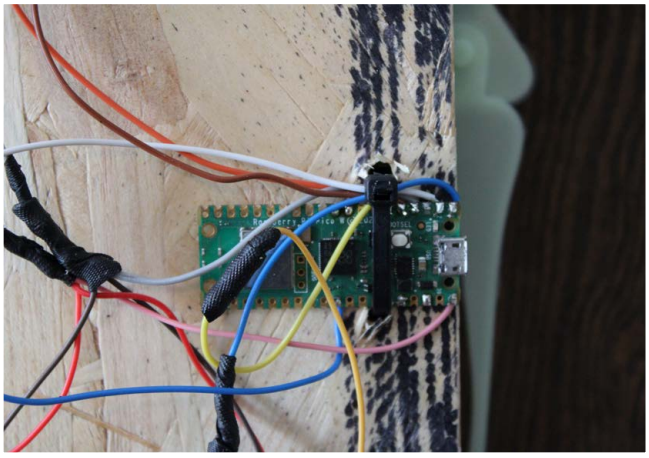
You'll also need a few 3.5mm wide screws that are smaller than your board is thick, some mounting hardware, and a cable tie.

Bare bones

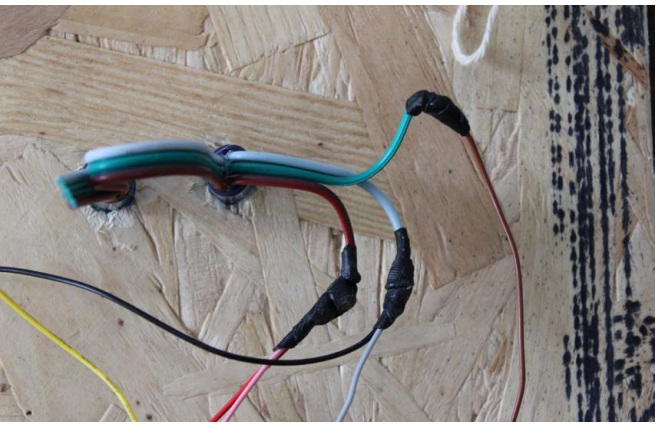
Our skeleton came with moveable joints with a pin-and-socket connector. Our first thought was to keep these as hinges and then hook up an actuator to the limbs to pull them into the position we want (like making a puppet). However, we found that the flimsy plastic of the skeleton had a habit of twisting rather than rotating properly at the joint, so we needed a different plan.

The first step is to disconnect the limbs from the skeleton and they should simply pop out. We're not going to join our skeleton directly, but have the servos mounted directly to the limbs, and held in place near the joint on the main body.

We now need a way of attaching a servo to each limb. Servos have a press-fit drive shaft that connects tightly to 'horns', and you usually



▼ Make sure Pico is mounted in such a way that you can access the USB port



Power, ground, and data – three connections that let you add as many LEDs as you need.

Your servos should come with mounting hardware

get a selection of different-shaped horns with your servo. We used the single horn and screwed it close to the original joint.

Your servos should come with screws that go through the holes on the horn. The only problem with these is that they poke through the thin plastic of the skeleton and could scratch someone who gets too close. You can use a blob of hot glue or attach a small bit of plastic or wood to the end of the screws if you're worried about this.

Once you have the horns attached to each limb, you now need to worry about the other side of the servos. Servos do have mounting screws, but these are usually higher up the servo, designed for a recess, so we need some form of mount to hold them in place.

If you have access to a 3D printer, then the easiest option is to print one. We used this one: **rpimag.co/9gservomount**, but depending on your servo, you might need to use a different option. The important thing is that it holds the servo vertically.

If you don't have access to a 3D printer, it should be possible to create a hole in your backing wood – either by chain-drilling or by drilling a starter hole and using a jigsaw – then mount the servo in there (perhaps with a liberal amount of hot glue).

Once you have your servo mounts, you should have all the key pieces of your motion control hardware.

Eye lights

It's well established in cinema that scary things have red eyes. We're not really sure why this is, since the only thing this author has ever seen with red eyes was an albino rabbit, and that little ball of fluff was the least scary thing in existence. However, since that is the established trope, our skeleton should have red eyes.

There are a couple of options for this: we could use red LEDs, or we could use RGB LEDs and just neglect two of the three colours. Although the latter option might seem a bit wasteful, it simplifies the build a bit and gives us options for the future, so we'll go with it. The specific LEDs we'll use are 12mm WS2811 LEDs. These come in chains (usually of 50) running at either 5V or 12V. We want the 5V version, and we just need two of the LEDs, but we can cut these from a chain.

Dark as the night

The physical setup of the skeleton is based around the back board. This should be a rectangular sheet of engineered wood strong enough to hold everything in place. The skeleton comes with a mounting hole at the top, and you can also use the holes from the shoulder and hip joints to screw it down securely.

Additionally, it'll need 12mm holes in the eyes for the LEDs, and 10mm holes near each joint to allow the wires from the servos to be tidied out the way.

At this point, you might also want to paint the board black so the skeleton stands out.

Screw the horns onto the limbs as close as possible to the original joint

Zero in

We're now ready to attach the servos, but before we do this we need to set up the electronics a bit. Servos don't rotate a full turn. Typically, they can turn about 180 degrees. Because of this, we have to know their position before we attach the arms and legs, otherwise we might not be able to rotate them into the correct position once they are attached. So, before we can continue with the physical build, we need to wire them up and rotate them into a known position.

We've got four servos, but they all need to be connected to the same power pins, so some creative wiring is needed. We wired it up as shown in **Figure 1**. This requires soldering a four-way joint. You might need to solder some extra bits of wire onto the servo wires for extensions. Make sure that the servo wires go through the holes in the board to allow the servos to be mounted properly before they're soldered together.

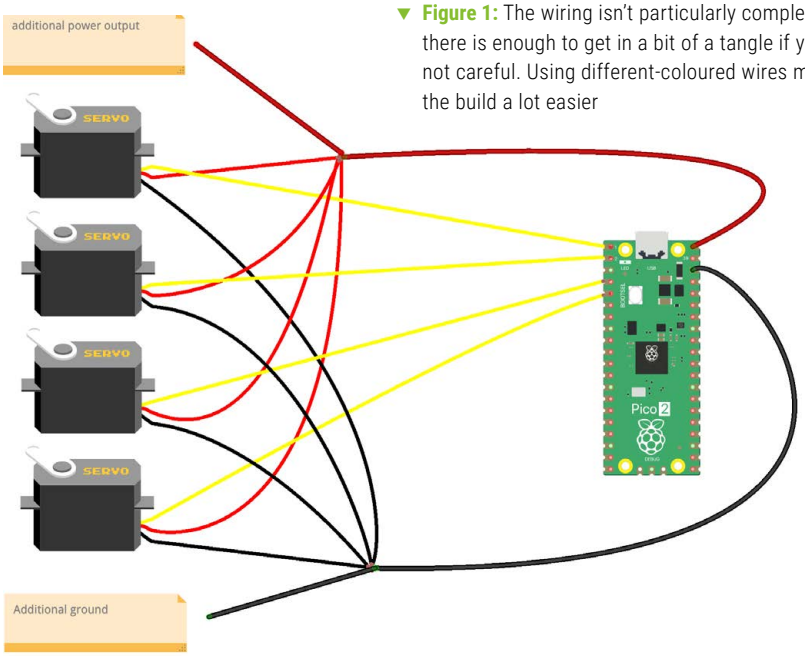
Once you've got the servos soldered up, you can now find out what state they're in and for this we'll need to program Pico.

There are a few options you could use to detect people

INSULATING JOINTS

Conventional wisdom has it that you should use heat-shrink tube over soldered wire connectors to insulate them. This can work for simple wire-to-wire joints but if you're doing anything slightly unusual, then it often doesn't work. This author has now switched to using self-amalgamating silicone tape.

This comes as a tape (usually black) on a backing (usually white). You peel the tape off the backing and wrap it around the joint. The tape will stick to, and amalgamate with, the tape below making a block of silicone. You can wrap it around whatever junction you have and it will insulate it. It's very stretchy, so make sure you pull it tight as you wrap it round to create a good joint. As well as insulation, it can provide strain relief. We find it creates a snugger fit and better protection than heat-shrink (and it helps that you don't have to remember to slide the heat shrink on before soldering which we had a habit of forgetting to do). While it's probably not completely water-tight, we've used it successfully on projects that have survived outside in bad weather for months.



▼ **Figure 1:** The wiring isn't particularly complex, but there is enough to get in a bit of a tangle if you're not careful. Using different-coloured wires makes the build a lot easier

▲ The simple pin-and-hole hinge joints let the elbows and knees move freely as the skeleton dances

▶ The original joint holes make convenient mounting holes

Code configuration

If you've not used MicroPython before, take a look at the getting started guide at **rpimag.co/micropythonsdk**. The short version of this is that you need to install Thonny (from **thonny.org**), and then install the MicroPython firmware on Pico (see **rpimag.co/micropython**). At this point, you can use the Thonny IDE to write code and send it to Pico.

You'll also need the servo library. You can download this from **rpimag.co/micropyservo**. Using Thonny's file manager, create a folder called **servo** on Pico and copy the **__init__.py** and **__main__.py** files into it.

Attach something easily removable to the drive shafts (we used flags made of electrical tape), and run the test code below.

```
import time
from servo import Servo

left_hip = Servo(pin_id=0)
left_shoulder = Servo(pin_id=1)
right_hip = Servo(pin_id=2)
right_shoulder = Servo(pin_id=3)

servos = [left_hip,left_shoulder, right_hip,
right_shoulder]
down_angle = [180,180, 0,0]
up_angle = [90,90,90,90]

while True:
    print("down")
    for i, servo in enumerate(servos):
        servo.write(down_angle[i])

    time.sleep(2)

    print("up")

    for i, servo in enumerate(servos):
        servo.write(up_angle[i])

    time.sleep(2)
```

This alters the state between two different states: `down` and `up`. You should be able to adjust the flags so that `down` points down and `up` points at 90 degrees. This then tells you the angle you want to add the limbs at. In other words: once the flags are adjusted so they point in this direction, you can then add the limbs so they point in the same direction as the flags.

However, before we add the limbs, let's add the final two parts of the electronics: a distance sensor and the LEDs.

There are a few options you could use to detect people. Passive infra-red (PIR) sensors are popular choices for triggering things like this, and one could certainly work in this project. However, they can be triggered very easily and we prefer our skeleton to activate when someone gets closer.

It is possible to do something advanced with a camera and some form of machine-learning person detector, but these don't always work well in the dark (particularly with changing light), so aren't a great choice for night-time installations.

We've used a plain distance sensor. Specifically, a Sharp GP2Y0A02YK. This

▲ Don't tell your woodworking teacher, but wiggling the drill a little as you make the eye holes makes it easier to mount the LEDs

outputs an analogue voltage that corresponds with the closest thing to it (up to about 1.5m). It's easy to work with as we just have to hook it up to an analogue input. The one downside is that there's quite a small field of view, so it doesn't always trigger when you hope it would. Our search for a perfect person detector continues, but for now, this works well enough.

Wiring the sensor requires 5V, ground (both of which are built off the power chain for the servos), and an analogue input (we used A0 on GPIO 26).

The final part of the electronics are the LEDs for the eyes. These should come as a string and you'll need to cut two off. Each LED has six wires attached to it: two for power, two for ground, one for data-in, and one for data-out. This means that the string has three wires for input and three for output, but it's not always obvious which end of the string is input and which is output. If you look very closely at the string, you should see an arrow on the PCB and this points to the data-out line. The data-in line is opposite this. Power should be red and ground should be white.

Power requires 5V, so along with the distance sensor, that's another two devices that need to be connected to the single VBUS connector on Pico, so you'll need to extend the chain of wires built off this pin. We connected the data-in to GPIO 5, but any should work.

It might be tempting to add more than two of the LEDs to the display. You could certainly add more to act as lightning or any other effect. However, we'd recommend that if you do this, you add an additional power supply as we're stretching the USB power supply about as far as we should really push it.

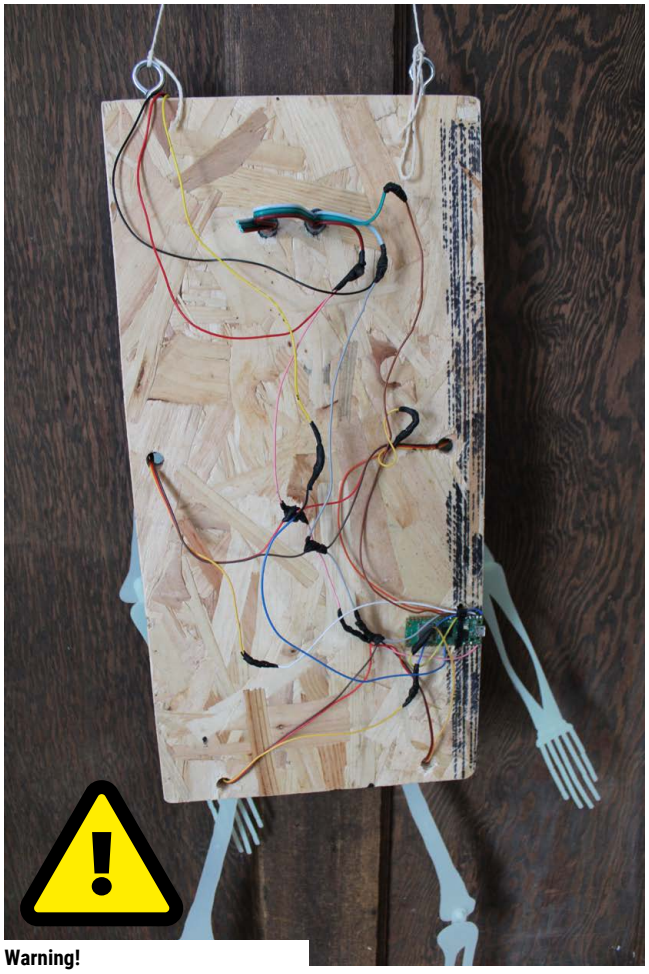
Eye eye

Now let's finish up the assembly. The LEDs push into their holes – just. Although they require 12mm holes, they have barbs to hold them in place. With a bit of force, you should be able to push them in far enough for them to hold in place. If you can't, then the best option is to use a 12mm drill bit and wiggle it in the hole to enlarge it slightly.

Mounting Pico is a bit tricky as it's hard to find wood screws that fit through the M2.5 holes. The most secure option is to use a case with mounting points. However, a quicker and easier method is to drill two 5mm holes, one either side of Pico, and use a cable-tie to hold it in place. It's a bit of a hack-y solution, but good enough for a prop that's going to be out for a few days a year. Just make sure that the USB port is at the edge of the backing otherwise you won't be able to plug in a cable, and also make sure that the cable tie doesn't go over the BOOT button.

Finally, you need a way of mounting your skeleton, and this will depend a little on where you want to display it. We fastened two eye screws into the top of the backing, which will allow us to hang it up. You could also drill through the backing and screw it into place, or just G-clamp where you want it to be.

◀ Eye screws in the end of the board aren't the most secure attachment, but it should hold the weight of the skeleton



Warning!

Power Tools

Be mindful of dangers posed by power tools. Be careful of strain from vibration, noise hazards, and (if necessary) wear protective clothing and goggles. Do your research first!

rpimag.co/powertools

▲ OK, the wiring could be tidier, but if you want to secure it further, you'll need to box in the back of the board

Toughening

Our skeleton is up and running and should be fine for light use, but it could certainly be toughened if you plan to use it a lot. The most obvious thing is boxing in the back. This should be fairly straightforward using a bit of dimensioned wood around the side and another sheet of engineered wood at the back. You'll have to make a hole for the USB cable. This boxing will protect the wires and electronics from any knocks.

Ready for action

That's our skeleton fully set up and working; let's program it. The following MicroPython code will test out all the components. It'll use the distance sensor to trigger the eyes lighting up and the limbs moving.

```
from machine import Pin, ADC
from time import sleep
import neopixel
from servo import Servo

left_hip = Servo(pin_id=0)
left_shoulder = Servo(pin_id=1)
right_hip = Servo(pin_id=2)
right_shoulder = Servo(pin_id=3)

servos = [left_hip,left_shoulder, right_hip,
right_shoulder]
down_angle = [0,0, 180,180]
up_angle = [90,90,90,90]

np = neopixel.NeoPixel(machine.Pin(4), 2)

distance = ADC(Pin(26))

legs_down = True

while True:
    print(distance.read_u16())
    if distance.read_u16() > 10000:
        np.fill((0,255,0))
        np.write()
        dancing = True
    else:
        np.fill((0,0,0))
        np.write()
        print("off")
        dancing = False

    if dancing:
        if legs_down:
            legs_down = False
            for i, servo in enumerate(servos):
                servo.write(down_angle[i])
        else:
            legs_down = True
            for i, servo in enumerate(servos):
```

```
                servo.write(up_angle[i])

    sleep(1)
```

If you save this to Pico as **main.py**, it will launch automatically when you power up Pico, so you just need to connect a USB power supply (capable of at least 2A), and your skeleton should activate.

At this point, we have a fully working person-activated skeleton, but there's one bit that you might need to change.

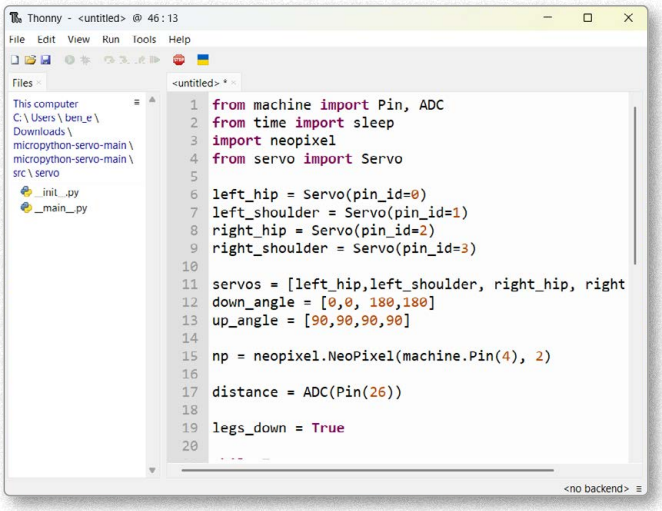
Let's work out the right distance value to trigger the action. This will depend on the place you want it physically set up. If it's in a corridor, alleyway, or other area where there's a wall opposite, then you might need to make the number higher so that it's not triggered by the wall. Other than this, it's really up to you how close you want people to be before it activates. The closer people are to it, potentially, the scarier it might be to them, so you might want to take this into account, especially if there are likely to be young children.

In either case, the number you need to change is the one in this line:

```
if distance.read_u16() > 10000:
```

As the method suggests, it's an unsigned 16-bit number, so the maximum possible value is 65,535.

That's our animated Halloween skeleton set up and ready to scare the local trick-or-treaters.



▲ The code running in Thonny IDE

The closer people are to it, the scarier it might be

skeleton_test.py

DOWNLOAD THE FULL CODE: rpimag.co/github

> Language: **MicroPython**

```
001. import time
002. from servo import Servo
003.
004. left_hip = Servo(pin_id=0)
005. left_shoulder = Servo(pin_id=1)
006. right_hip = Servo(pin_id=2)
007. right_shoulder = Servo(pin_id=3)
008.
009. servos = [left_hip,left_shoulder, right_hip,
     right_shoulder]
010. down_angle = [180,180, 0,0]
011. up_angle = [90,90,90,90]
012.
013. while True:
014.     print("down")
015.     for i, servo in enumerate(servos):
016.         servo.write(down_angle[i])
017.
018.     time.sleep(2)
019.
020.     print("up")
021.
022.     for i, servo in enumerate(servos):
023.         servo.write(up_angle[i])
024.
025.     time.sleep(2)
```

Going further

Perhaps the most obvious way of taking this project further is to add some audio. This could be a creepy laugh or some ominous music. There are some options for playing audio directly from Pico in MicroPython, but the easiest choice would be to add an audio board like DFPlayer.

As well as audio, you could up the lighting effects. More LEDs would be fun, and that's fairly straightforward as you'd only need an additional power supply. Anything that can output 5V at around 4A or 5A should be able to handle quite a few LEDs as well as the current setup.

The same basic technique for using servos to make things move and LEDs to light things up can be added to all manner of Halloween props. It could be a ghost that is hoisted by a servo arm, or a pumpkin whose eyes light up when someone approaches. Your imagination is the only limit here.

We'd love to see what you come up with. Be sure to tag us in your creations on social media.

skeleton.py

DOWNLOAD THE FULL CODE: rpimag.co/github

> Language: **MicroPython**

```
001. from machine import Pin, ADC
002. from time import sleep
003. import neopixel
004. from servo import Servo
005.
006. left_hip = Servo(pin_id=0)
007. left_shoulder = Servo(pin_id=1)
008. right_hip = Servo(pin_id=2)
009. right_shoulder = Servo(pin_id=3)
010.
011. servos = [left_hip,left_shoulder, right_hip,
     right_shoulder]
012. down_angle = [0,0, 180,180]
013. up_angle = [90,90,90,90]
014.
015. np = neopixel.NeoPixel(machine.Pin(4), 2)
016.
017. distance = ADC(Pin(26))
018.
019. legs_down = True
020.
021. while True:
022.     print(distance.read_u16())
023.     if distance.read_u16() > 10000:
024.         np.fill((0,255,0))
025.         np.write()
026.         dancing = True
027.     else:
028.         np.fill((0,0,0))
029.         np.write()
030.         print("off")
031.         dancing = False
032.
033.     if dancing:
034.         if legs_down:
035.             legs_down = False
036.             for i, servo in
     enumerate(servos):
037.                 servo.write(down_angle[i])
038.         else:
039.             legs_down = True
040.             for i, servo in
     enumerate(servos):
041.                 servo.write(up_angle[i])
042.
043.
044.
045.     sleep(1)
```

Powered by Raspberry Pi

Ever since it first launched back in 2012, Raspberry Pi has captured the imagination of hobbyists, educationalists, and pure technology fans. It has also been found in some exciting products, helping monitor Covid and other health issues, remotely sensing ecological events, and capturing celestial movements. Scientific, agricultural, medical, and photographic uses abound. Hundreds of these examples have become important products in their own right – you'll now find Raspberry Pi at touchscreen digital kiosks, embedded in transport signage, and in dozens of smart devices we collectively know as the Internet of Things. Many such products now sport a Powered by Raspberry Pi badge, the customer's guarantee that the item in question has been robustly developed and stringently tested and subject to a slew of technical and operational criteria specified by Raspberry Pi's own engineers and developers. As such, products badged Powered by Raspberry Pi have earned our seal of approval and are ones you can buy with confidence.

rpimag.co/poweredbyraspberrypi

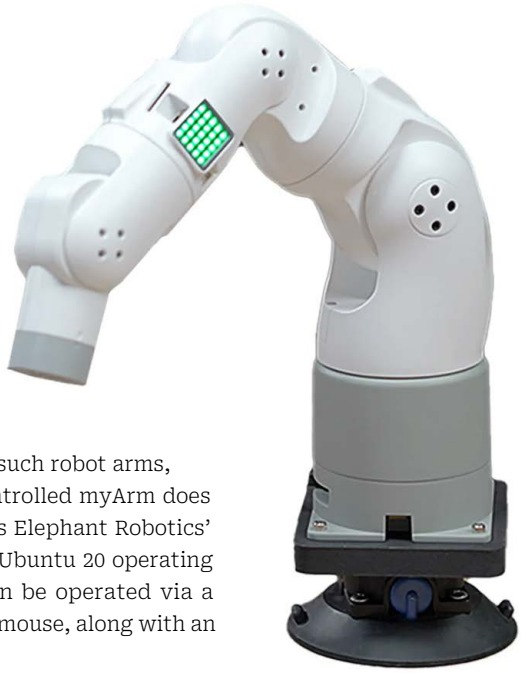
APPLY TO POWERED BY RASPBERRY PI

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

Elephant Robotics myArm

China | rpimag.co/myarm300



With seven degrees of freedom (most robots offer up to six – the extra flex comes from the articulated elbow), Elephant Robotics claims that myArm allows movements "as flexible as a human arm" and is the most portable desktop robot arm available. Indeed, the very purpose of the unusually naturalistic-looking arm is for educational settings, whether robotics, ROS simulation or logic control. Demonstration videos show myArm picking up and gently placing bricks into a small bucket and picking up slim plastic pucks and rotating them through an arc before slotting them into a Connect 4 frame. The arm has clear gaming and industrial uses, especially when paired with items from Elephant Robotics' range of accessories such as fixed bases, grippers, and being used onboard an AGV (automated guided vehicle) chassis.

In contrast to most such robot arms, the Raspberry Pi 4-controlled myArm does not need a PC and runs Elephant Robotics' custom version of the Ubuntu 20 operating system. Instead, it can be operated via a wireless keyboard and mouse, along with an optional monitor.

Flori collection machine

Brazil | flori.tech

Flori is the brainchild of Thaís and Gabriel who entered a 'cleantech cluster' hackathon in Rio challenging entrants to come up with a means of tackling waste. It's a vast issue in Brazil, since only 3% of the country's waste is disposed of properly. As well as the environmental cost, there's a potential R$14 billion that could otherwise be earned from recycling materials. Flori realised that gamifying the sorting of everyday waste such as drinks cans could have an immediate impact and transform the depressingly poor levels of recycling.

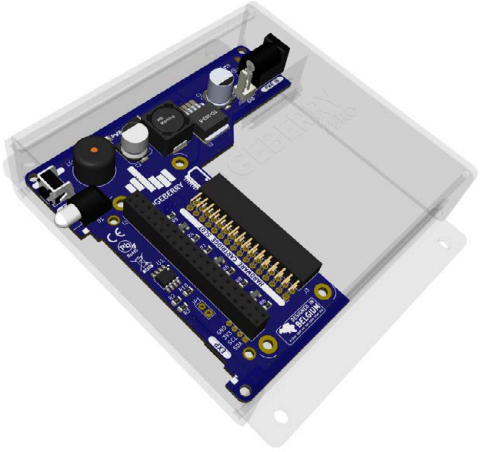
After plenty of research and brainstorming, the duo came up with a 500 litre-capacity smart waste collection machine that accepts paper, plastic, metal, and glass. The user initiates the disposal process by either logging in via the unit's touchscreen or using the QR code to donate items anonymously (Flori requires a 3G internet connection). Recyclable items can be placed in the opening that appears. Their weight is logged and the amount added to the user's account if they are regular recyclers. This element turns the process into a potentially low-key competitive game among colleagues or fellow students, for example.

The whole thing has smarts from Raspberry Pi to sense what's being recycled and sends a message to the owner's app when the container is getting full, helping avoid issues of missed recycling opportunities due to an overflowing bin. Roughly 486 empty 2-litre plastic bottles or 3888 drinks cans can be collected before it needs to be emptied. Medical sharps, bottles, packaging, and textiles can also be optionally collected for sorting and recycling, making it a great setup for specific industries.

Edgeberry Baseboard Zero

Belgium | github.com/Edgeberry

Edgeberry is an open-source ecosystem specifically designed by Belgium-based maker Sanne Santens to make it easier to create Internet of Things networks mixing and matching devices that will work together well. As the GitHub repo explains, "Edgeberry simplifies the process of bridging the physical world with the digital realm". One of these simplifications is the Edgeberry Baseboard which provides a chassis containing a power supply status indicator and a unique identifier so the IoT device you assemble can be found and added to the ecosystem. The Baseboard Zero is the Raspberry Pi Zero version, offering 12 to 24V DC power input, buttons and indicators to show it's working and has been recognised, plus an expansion slot for Edgeberry Hardware should you want to use your IoT self-built device for particular applications. A clear acrylic housing protects both Pi Zero and any components you decide to attach.



Persys AI

USA | persys.ai

Persys is a Raspberry Pi 5-powered 'local AI inference device' on which data is processed locally, ensuring no-one can snoop on what you're saying or scrape the conversation details. We recently tried out a demo of a similar device by Plaud, but Persys has the advantage of powerful Raspberry Pi processing as well as the all-important Powered by Raspberry Pi badge. Persys is designed to be a personal cloud device that acts as a companion to your smartphone or laptop. You can discreetly consult it for definitions and references as well as pop culture details safe in the knowledge your enquiries won't be shared. Better yet, it has onboard storage so you can save photos, music, documents, contacts, chats and more, with 1 and 2TB configurations available. Since the LLM (large language model) processing is done on-device, there's no need for an internet connection either.

Coho 4G smart camera

France | mycoho.eu

Animal lovers are wont to pay close attention to the needs and health of their trusty companions. In the case of horses, they don't spend all day, every day in close proximity to their owners, but the bond between horse and rider is often profound. French company Animalinks came up with a natty way to monitor the day-to-day health of equine friends, using a Raspberry Pi 4-controlled 'intelligent camera' that roughly resembles the shape of a horse's hoof. Coho is designed to be installed in stables and wirelessly send back time-lapse images or live videos along with health data. The setup allows for Raspberry Pi to control up to five cameras per stable and send live updates about each horse 'direct from the stall'. There are also fittings for trailer and nomadic (meadow) use. Updates can be viewed on a personalised MyHorse app. As well as physically observing each horse (and get alerts about potential intruders), the owner can track the stall or stable temperature, steed's recovery from injury or illness, oversee mares foaling, and track each animal's performance. A pilot scheme was launched in 2019 and, with both practical and positive participants' feedback, a commercial product developed in 2020. Since then, Coho has embraced AI in order to discern horse health from its behaviour and movements.

Although Coho is marketed as being for everyday horse lovers as well as breeders and trainers, such a sophisticated setup is likely to be of particular interest to thoroughbred owners. An undoubted fillip to the business came in 2023 when Animalinks was approached by the French national equestrian team who bought multiple Coho smart cameras as part of their investment ahead of the Paris 2024 Olympic Games.

Drimaes DRIM-Simhub

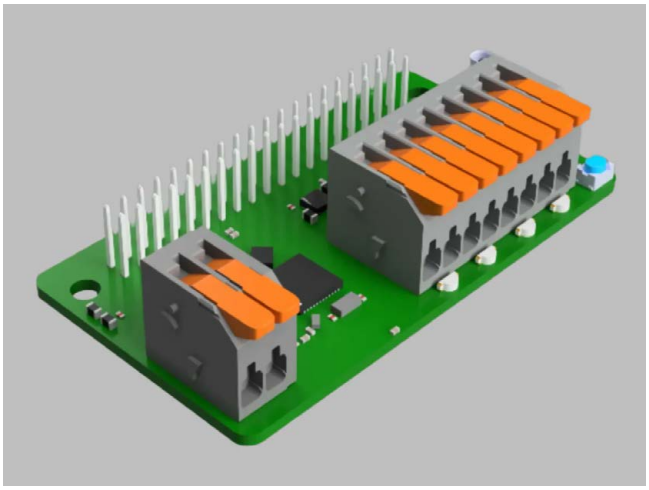
Korea | drimaes.com/solution-sils

A common complaint about modern cars is they aren't as easy to fix as in the pre-computer age. However, SDVs (software-defined vehicles) such as most electric cars can be upgraded and performance tweaked over the air, meaning incremental improvements as well as critical fixes can be applied as firmware updates. Smartphones and connected consumer electronics have been updated in this manner for years, so it makes lots of sense that vehicles can have smart upgrades too. Drimaes's DRIM-Simhub provides a means of simulating ECU (electronic control unit) software so it can be tested, debugged, and validated well before it gets anywhere near a vehicle. Applications designed in automotive industry standard MATLAB/Simulink (AUTOSAR or non-AUTOSAR) are all supported, and the Drimaes hub itself works with Windows, macOS or Linux, including a Drim-Pi Raspberry Pi 5 hardware module simulation platform. The DRIM-Simhub offers testing, monitoring, emulation, and reporting for hardware and software 'in-the-loop' and integrates with third-party software.

Mantis Flex Controller

USA | rpimag.co/flexcontrol

Mantis is a US-based company with a focus on developing STEM products that inspire young minds. The MantisEDU range consists of devices and sensors that can be used both independently, by code clubs and as part of a scheme of work (lesson plans are provided). Sensors can be used for real-world data gathering and logging, thereby acting as live case studies and citizen science projects. Depending on participants' ages and existing knowledge, there are Scratch, Python and MakeCode options, plus options to sync with computers – including Raspberry Pi, of course – and smartphones. The Mantis Flex Controller is ideal for anyone keen to build robots and vehicles. It can be used as a high-current 12-volt motor controller via Raspberry Pi, MantisCode (Scratch), or Xbox controller. It also works with Mantis large-frame robots and rovers.



ONLY THE **BEST**

Tiny boards

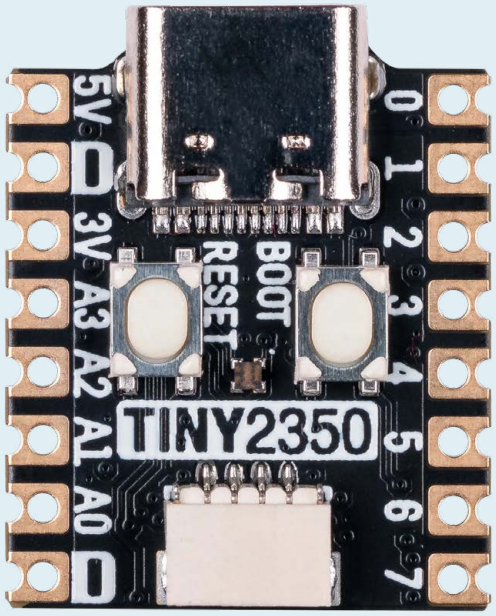
By **Phil King**

For portable projects, space is typically at a premium. Trying to cram in a Raspberry Pi computer isn't always possible. Even a Raspberry Pi Pico, roughly the size of a stick of gum, may be too big for some projects such as wearables. So, for those times when you need something a little smaller, we've rounded up some of the best tiny boards around, all based on a Raspberry Pi RP2040 or RP2350 microcontroller chip, as used in Pico and Pico 2 respectively.

As you'd expect, a smaller footprint necessitates some compromises, such as a reduced number of physical pins available to break out the processor's GPIOs. While this may well be a key factor when choosing a tiny board, most of them offer a good range of I/O functionality, such as analogue inputs and PWM. Some boards even add a few extra connections via solder pads on their underside, or a Qwiic / STEMMA QT port. Let's get our magnifying glass out and take a closer look...

Tiny 2350

Pimoroni | £19 / $21 | pimoroni.com



Measuring a mere 22.9 × 18mm, Pimoroni's tiny microcontroller board packs a Raspberry Pi RP2350 dual-core processor, as used on Pico 2 / 2W. It's almost identical to use, too, featuring a USB-C port to connect to a computer for installing firmware and coding (in MicroPython, CircuitPython, or C/C++). Along with a boot select button, the board crams in a handy reset button, RGB LED, and 4MB of QSPI flash storage.

You can buy the Tiny 2350 with or without pre-soldered pin headers. As you'd expect, with a reduced pin count of 16, fewer GPIO pins are broken out than on Pico 2: twelve, plus another couple on the Qwiic / STEMMA QT port. However, they do include four 12-bit ADC channels (vs Pico 2's three) and encompass two channels each of the I2C, SPI, and UART protocols. Like its Tiny RP2040 sibling, it's a little gem of a board.

Verdict

Perfect for portable projects and easy to use.

▲ It's about the size of a standard UK postage stamp

Adafruit QT Py RP2040

Adafruit / The Pi Hut | £10 / $13 | adafruit.com / thepihut.com

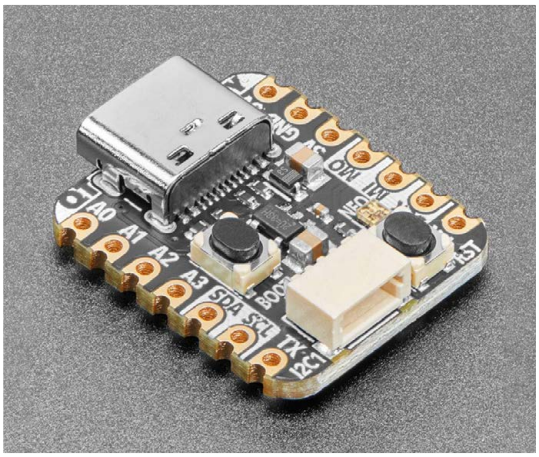
QT Py (pronounced 'cutie pie') is a series of boards by Adafruit, all with the same tiny form factor but based on different processors, including the original SAMD21 model, an ESP32 one, and this RP2040 version.

At 21.8 × 17.8mm, it's even a smidgen smaller than Pimoroni's Tiny 2350 (and 2040) and similarly features boot select and reset buttons, along with an onboard RGB LED. It has a couple fewer I/O pins on the twin headers, with seven on each edge. Eleven of these are GPIO pins, encompassing four analogue inputs, plus SPI and UART protocols. I2C is also covered by an extra pair of GPIOs in the Qwiic / STEMMA QT port.

A big plus point is the 8MB of onboard flash storage (four times that of a Pico). As with other RP2040 boards, you can program it (via the USB-C port) in MicroPython, CircuitPython, or C/C++.

Verdict

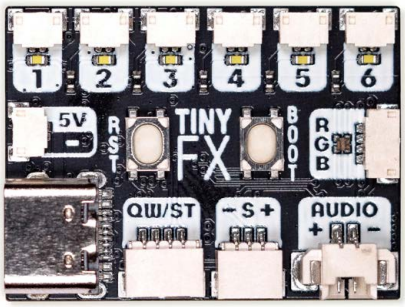
A truly tiny board with all the power of Pico.



▶ One of the smallest microcontroller boards around

Tiny FX

Pimoroni | £15 / $17 | pimoroni.com



Verdict

Ideal for lighting miniature models and LEGO.

▲ Spiffing for tiny lights… and sound

This RP2040-based board is focused on delivering mini LED lighting suitable for LEGO kits, doll's houses, and miniature models (as in a tutorial this issue). You can even add audio thanks to its onboard 3.2W I2S amplifier, by connecting a mini speaker.

At 31.2 × 23.2mm, it's not the smallest board on test, but still very compact. Instead of GPIO pins/holes, it has six two-pin JST-SUR ports along one edge, for connecting single-colour LEDs, plus a four-pin RGB connector – which can be used with a mini multi-way adapter to control multiple RGB LEDs. Pimoroni stocks mini LEDs with JST-SUR connectors, or you could adapt standard ones to fit.

A Qwiic / STEMMA QT port can be connected to one or more sensors to trigger your lights and sounds. Pimoroni's PicoFX MicroPython library makes the board easy to program. There's also a wireless-enabled Tiny FX W model available.

Waveshare RP2040 Tiny

Waveshare / The Pi Hut | £5 / $6 | waveshare.com / thepihut.com

A tiny board at a tiny price, it measures 25.5 × 18mm. Some space is saved by the omission of an onboard USB-C port – instead, it's on a separate adapter board which you can connect using an FPC ribbon cable when you want to program the RP2040 Tiny in MicroPython, CircuitPython, or C/C++. The adapter board also features boot select and reset buttons.

While this may seem a rather odd arrangement, the upside of losing onboard USB and buttons is that there's more room for I/O pins: 23 of them. Located around three edges, they include an impressive 19 GPIO pins, encompassing four analogue inputs and two channels each of SPI, I2C, and UART, plus all 16 PWM channels (as on Pico). You get the standard 2MB of onboard flash storage to play with.

Verdict

Good if you need more GPIO pins for your project.

The USB-C port and buttons are on an adapter board

Seeed XIAO RP2350

Seeed Studio | £4 / $5 | seeedstudio.com

Measuring a mere 21×17.8mm, it shares the accolade for tiniest footprint with the Adafruit QT Py 2040. While this only leaves room for 14 pins, including eleven GPIOs, a further eight GPIOs are broken out via tiny connector pads on the underside. So, if you don't mind a bit of fiddly soldering, you can make use of them all – encompassing three analogue inputs and two channels each for SPI, I2C, and UART. There are a couple of pads to connect a battery, too.

A USB-C port on the top enables easy connection to a computer for firmware flashing and coding (in MicroPython, CircuitPython, or C/C++). Either side of that port are three LEDs, including an RGB one. One slight annoyance is that the boot select and reset buttons are really tiny and fiddly to press. Still, you get the extra processing power of the dual-core RP2350 for very little money.

Verdict

A clever compact design with extra GPIOs on the rear.

There are extra connector pads on the underside

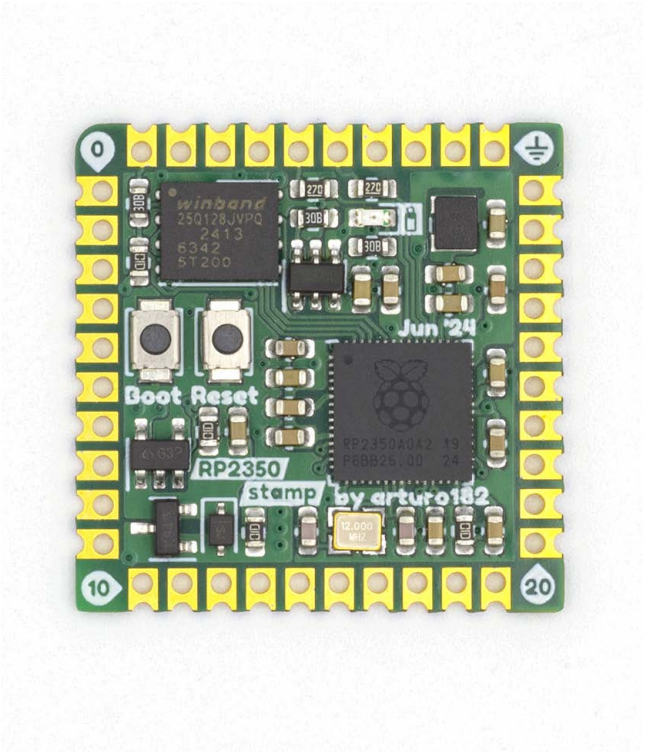
RP2350 Stamp

Solder Party / Pimoroni | £12 / $13 | solder.party / pimoroni.com

For makers who want to go even smaller, the RP2350 Stamp is an interesting option. Measuring just 25.4mm (1-inch) square, it's aimed at making it easier to embed an RP2350 processor into your own custom PCB (by surface-mounting it), aided by Solder Party's design footprints for EAGLE, KiCad, and EasyEDA.

There's no USB port – that's broken out on two of the 40 I/O pins (with 2mm pitch) around all four edges. All 30 of RP2350's GPIOs are there too, along with boot select and reset functions and power/battery connectors. For even more GPIOs (48 of them), a Stamp XL version is available, based on the RP2350B chip.

For easier project prototyping, you can mount the Stamp (via headers or FlexyPins) on one of the optional Carrier boards, which route all its I/Os to standard 2.5mm male pin headers around the outside, and the USB connections to a USB-C port.



◀ Mount the Stamp on your own PCB or a Carrier board

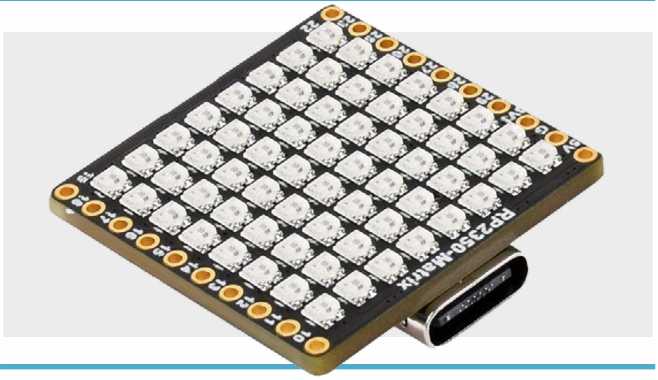
Verdict

Useful for embedding RP2350 into PCB designs.

WAVESHARE RP2350 MATRIX

Waveshare | £7 / $10 | waveshare.com

This brand new RP2350 board features an 8×8 RGB LED matrix, with a data-out pin if you want to add more. In addition, it somehow crams in a six-axis IMU, 16MB of flash, and breaks out 25 GPIOs via 20 pins and eight solder pads on the underside (along with boot select and reboot buttons).



▶ A tiny RP2350-powered LED matrix and more

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

Adafruit Fruit Jam

If you use it like a computer and program it like a computer, is it a computer? By **Ben Everard**

Adafruit **rpimag.co/fruitjam** $39.99

SPECS

DIMENSIONS:
86 × 54 mm

STORAGE:
16MB flash, 8MB PSRAM, SD card slot

I/O:
DVI port, STEMMA QT, STEMMA Classic, EYESPI, 5 × NeoPixels, 3 × buttons, Wi-Fi adapter

Verdict

Adafruit's Fruit Jam is well thought out on both the software and hardware sides, to deliver a machine that's easy and fun to hack on.

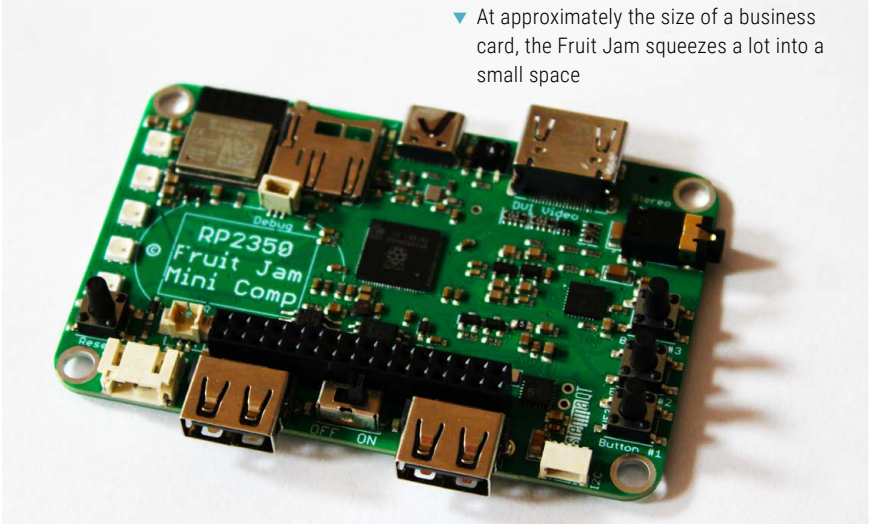
10/10

What is a computer? What is a microcontroller? And where do we draw the line between the two? There are some technicalities that can try to point us in one direction or another, such as whether or not it includes a Memory Management Unit (or MMU). However, this sort of technical distinction misses the point. We're not really interested in dictionary definitions here, but more the question of what the word 'computer' means to you.

A computer – to most people – is something with a screen, and at least one user input device such as a mouse, keyboard or touchpad, that runs a range of interactive software. Take any feature of that away and you have something that we'd probably call a microcontroller.

The Adafruit Fruit Jam, then, is a computer based on an RP2350. However, the RP2350 is usually considered a microcontroller. If this all sounds a bit contradictory, it's because the Adafruit Fruit Jam only just makes it through our definition of computer by cramming in extra hardware and squeezing all the performance it can out of the RP2350.

The two features that make it feel most like a computer are the HDMI output (technically a DVI port that works with an HDMI cable) and USB input, both provided by RP2350 using HSTX and PIO respectively. This is coupled with extra RAM and an SD card slot for storage to allow the software to behave something a bit like a late 1980s or early 1990s computer where you can use a mouse to launch different bits of software from a menu.



▼ At approximately the size of a business card, the Fruit Jam squeezes a lot into a small space

▲ The Adafruit Fruit Jam uses the larger RP2350B package which has 48 GPIOs (as opposed to 30 on the RP2350A package)

Expansion options

There are also a few features that you're more likely to find on a microcontroller: STEMMA QT and STEMMA Classic ports are included, both of which let you connect sensors and other hardware. There's an EYESPI-compatible GPIO header that lets you add a TFT screen if you don't want to use the DVI port. In other words, it's easy to expand with more hardware.

Why, in a world where there are 2025's computers, would you want a microcontroller dressed up akin to a computer from 30 years ago? There are three things that we think the Adafruit Fruit Jam really excels at...

> *Given the hardware, we'd expect to see some emulators of classic computers popping up*

Nostalgia – keeping retro computers running takes work and they can take up a lot of space as they're often specific about which keyboards and monitors they accept. The Adafruit Fruit Jam gives a similar-ish experience, but lets you use modern hardware. For many people, they'll already have all the peripherals and just need the board.

Understanding – This is arguably the most understandable computer on the market today. It's completely conceivable for a reasonably technical person to fully understand how this works, and that's probably not possible with any other machine that has HDMI output and USB input. It's not simple – a huge amount of work has gone into making this possible – but it is limited in scope and open source.

Fun – This is an easy computer to mess around with. There's not much to go wrong, it's all Python, and you can very quickly re-flash it if it all goes wrong. There's a low bar to tinkering with it, yet still plenty of power in there to do interesting things.

These are perhaps all slightly different ways at looking at the same reason. It's a hackable little computer that's easy to mess about with.

Programming and OS features

You can program the Fruit Jam in just about any way that you can program any other RP2350 board, though Adafruit has put a lot of work into getting CircuitPython well supported. As well as a module to make it easy to access the underlying hardware, there's Fruit Jam OS which adds a launcher and a way of managing apps.

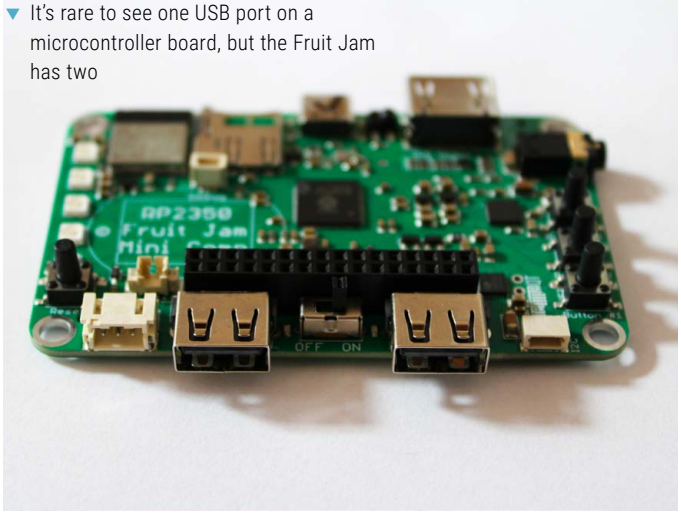
The default apps include a text editor, IRC client, and some classic games (including versions of Snake, Minesweeper, and Breakout). There's PyDOS that gives you a DOS-like interface for managing files. As well as fun little apps, we can see this being useful for controlling hardware in situations where you want a monitor and mouse or keyboard, but not the complexities of a full OS.

Given the hardware, we'd expect to see some emulators of classic computers popping up – Lady Ada has shown a Macintosh 128K emulator running on a Fruit Jam on YouTube. You can download the code from **rpimag.co/picomac**. The RP2350 at the heart is certainly capable of emulating more modern machines, provided someone is willing to invest the time to port emulators to this hardware.

Documentation was limited at the time of review, but updated resources will likely be available at **learn.adafruit.com** by the time you read this.

There's nothing unique about the features on the Fruit Jam. You can add DVI and USB ports to Pico 2 using various add-ons. However, the Fruit Jam does bring together a very useful set of hardware at a very reasonable price.

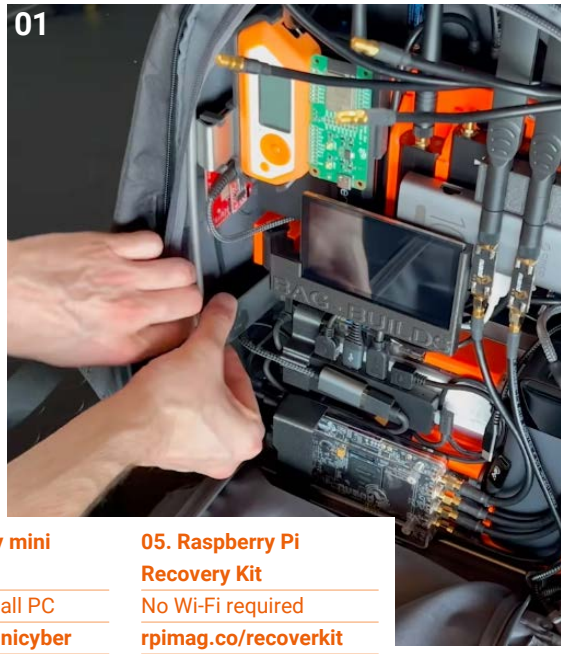
▼ It's rare to see one USB port on a microcontroller board, but the Fruit Jam has two



10 amazing:

Raspberry Pi cyberdecks

Preparing for the cyberpunk future with these rad custom PCs, choom



Raspberry Pi is small and compact, making it much smaller than most accessories you need to plug in to make a computer. Utilising this size and portability makes it perfect for your own custom cyberdeck, built to your exact liking, and there are plenty of folks who have done so. Let's go off the grid with just ten of them...

01. Backpack Cyberdeck

Lab with straps

rpimag.co/backpackcyber

A mobile workstation for wireless communications analysis and pentesting, this backpack has a special hand-crafted frame so that everything fits. It also has a ton of antennas.

02. Cyberdeck portable entertainment system

Plug and play

rpimag.co/cyberes

Not every cyberdeck needs to be able to hack into the mainframe, sometimes you just need to take a break and play some Street Fighter II.

03. Clamshell BlackBerry cyberdeck

Cyberdeck for ants

rpimag.co/berrycyber

BlackBerry was once the platform of choice for corpos. And now with a Raspberry Pi installed, it can be your netrunner terminal! Think we're using these cyberpunk terms correctly...

04. Extremely mini cyberdeck

Extremely small PC

rpimag.co/minicyber

Friend of the magazine Michael Horne has been working on this a while – it's the smallest cyberdeck you'll probably see, and it has a whole suite of sensors too.

05. Raspberry Pi Recovery Kit

No Wi-Fi required

rpimag.co/recoverkit

Need a hardened box to last the post-apocalypse? You might be able to use this specific Raspberry Pi build. It can still be used pre-apocalypse, though.

06. CyberPlug

Handheld prototyping

rpimag.co/cyberplug

Many cyberdecks are handheld and aren't always in a typical laptop computer form factor, much like this CyberPlug, which its creator loves to use to experiment with different components.

07. Pilet

Commercial cyberdeck

rpimag.co/pilet

This modular cyberdeck can be modified to be used how you wish, with a retro computer style straight from the late 1970s and early 1980s.

08. PiDex

Waterproof deck for Raspberry Pi 5

rpimag.co/pidex

Using a big, red switch, you can turn on this 'rugged and waterproof' cyberdeck. Although the box is probably the thing that's waterproof.

09. Vertical Runner

Working sci-fi prop

rpimag.co/verticalrunner

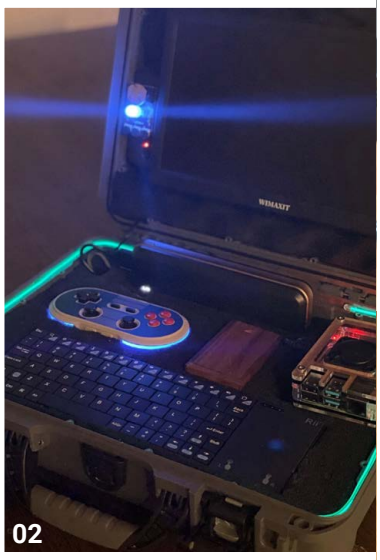
You can read about the full project earlier in the magazine, but this cool-looking movie prop will actually work if you need it to.

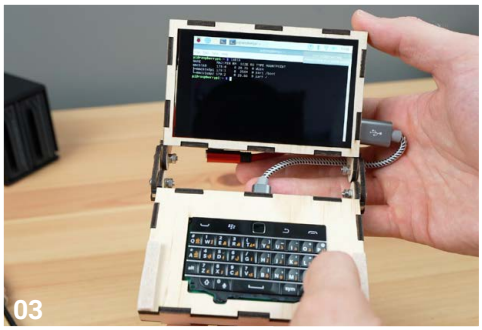
10. Mega six-screen cyberdeck

I'm seeing triple

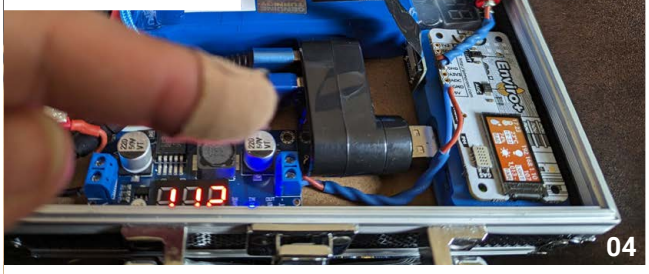
rpimag.co/sixcyber

This six-screen Raspberry Pi-powered cyberdeck easily folds down into a special case – perfect for making a run to the next safe house.


02


03


06


04


05


07


09


08


10

OpenFlexure

Microscopy for everyone

Maker

Jo Hinchliffe

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.

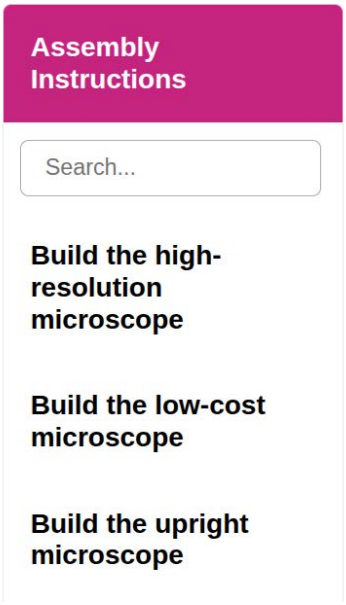
concretedog.blog-spot.com

▲ A fleet of OpenFlexure Microscopes

It's amazing the things people have done with Raspberry Pi. OpenFlexure is an open-source project aiming to make high-quality microscopy available to all. Whilst our hardware hacker forebears used lathes and milling machines to cut and form and grind out manual microscopes, OpenFlexure makes use of affordable electronics and access to 3D printing.

The main project of OpenFlexure is the OpenFlexure Microscope, and at first glance, if your knowledge of microscopes is limited to things you directly look through, you might wonder what on earth it is. Consisting of a big pile of 3D-printed parts, there's no obvious eyepiece. That's because this microscope captures images and delivers them to your connected monitor, or indeed another computer networked to the scope.

Before you download and start printing, there are a couple of fundamental choices you need to make. There are three different variants of the OpenFlexure Microscope: a high-resolution option, a low-cost option, and an upright option. The last of these is a newer and less tested design, whereas first two are both well established, tried and tested. The high-resolution option means the design incorporates some commercial microscope parts. You'll need to find an 'objective' (the lens that often is turned into place to change the magnification of a classic 'look through' microscope) and a 12.7mm achromatic lens, as well as a Raspberry Pi and a Raspberry Pi Camera Module 2. The low-cost version offers a reduced resolution and just uses the Raspberry Pi Camera Module V2 in a slightly modified form. Both the high-resolution and the low-cost version place the camera and lens under the sample so you are looking from the underside of a slide, whereas the upright option mounts the optics above the sample. Both these approaches have some advantages and disadvantages in some use cases.



▶ There are three different variants of the OpenFlexure Microscope for you to choose to build

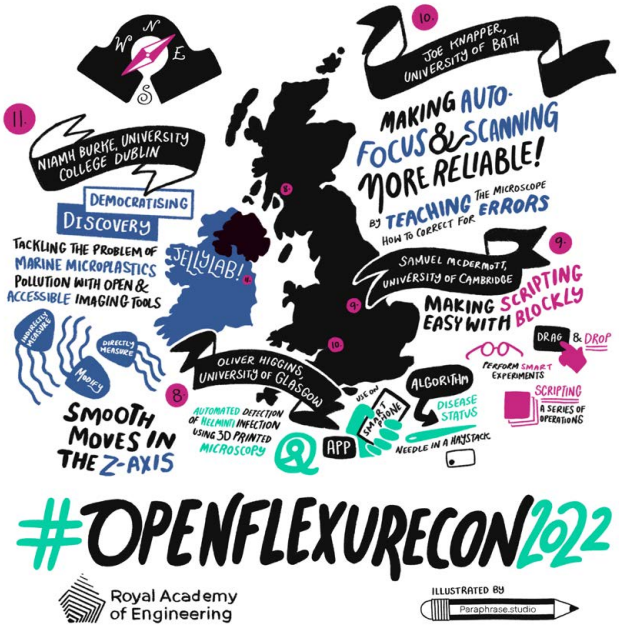


Image credit: Rae Goddard, paraphrase.studio

OpenFlexureCon2022

It's interesting to see different use cases, and indeed different customisations, of the OpenFlexure microscope project. One which caught our eye was presented at the 2022 OpenFlexureCon and was around the use of OpenFlexure in assessing levels of microplastics in the environment. The study was focused on microplastics in the sea, as this is where they tend to end up. One of the problems around microplastics research is that there is very little baseline data. The common approach to assess the concentration of microplastics is Fourier-Transform Infrared (FTIR) spectroscopy, which uses very expensive equipment that isn't suitable for field use.

Another approach is that you can, under a microscope, detect microplastics that accumulate inside the guts of comb jellyfish. This is where the OpenFlexure designs can be useful. Niamh Burke took the optical stack from the OpenFlexure design and attached it to a modified 3D printer. This adapted design created a large platform for viewing and data collection to significantly increase the opportunity to collect data in this vital research area.

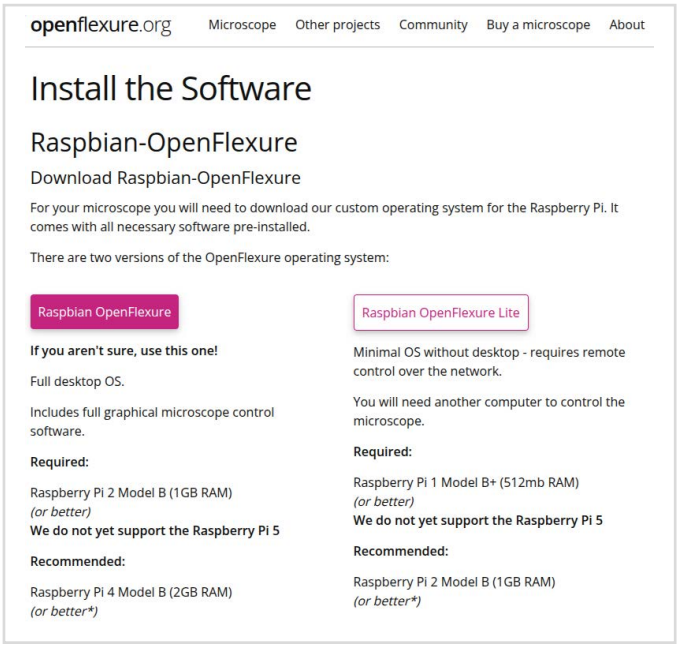
It's an incredibly clever and accurate solution

Figure 1: The OpenFlexure Microscope is designed to flex to create movement needed for scanning microscopy

Why is it called OpenFlexure?

It's not immediately apparent why the project is called 'OpenFlexure' when it could be called 'Open Microscopy', for example. The reason refers to a technical solution that makes the microscope incredibly accurate to use. Often when creating tiled images of samples, you are moving the stage and sample in tiny fractions of a millimetre and your entire collection of images might only be created in a 2mm by 2mm area. This requires accuracy and the OpenFlexure delivers this to achieve a sub-100-nanometre resolution. If the project was built in metal, a design approach would be to use incredibly accurately machined beds in a dovetail slide configuration. These accurate sliding surfaces are hard to create with any kind of precise tolerance and fit on a filament 3D printer. To overcome this, OpenFlexure, well, flexes! The printed design has sections where a budget stepper motor adds torque to the system, accurately flexing the design. It's an incredibly clever and accurate solution (**Figure 1**).

Whilst you could build this for manual hand control, the more common build uses three very affordable 28BYJ-48 stepper motors (**Figure 2**). Two are for creating the flexure motion for the X and Y axes; the third is used for a Z axis which moves the optical components, setting a focal distance. If you have ever tinkered with the 28BYJ-48 stepper motors, you'll know that, whilst excellent for the price, they have a lot of backlash (slop in the mechanism). Cleverly, this is controlled by both the flexure design holding tension in the system and also by adding compensation values in the software. Again, really excellent engineering that keeps the build cost low but the quality high.



▶ **Figure 3:** The software side of things is largely taken care of via the custom OS image for Raspberry Pi

For motor control, there are a couple of different options. There is a custom-designed board called 'Sangaboard', which is a neat design that sits on top of the Raspberry Pi as a HAT and has drivers for the motors as well as drivers for the LED for the illumination stage. It is essentially an Arduino-based board and can be flashed with a provided sketch using the Arduino IDE. It's a really neat solution, but a complex board to get a single piece manufactured; however, looking at the OpenFlexure community and forum, it's periodically available in batches. The second option is that you can create an equivalent driver board combining an Arduino (most examples use a Nano) and the ULN2003 motor driver board the 28BYJ-48 motors ship with. It's not the most complex thing to wire up, but of course the Sangaboard is potentially a neater solution that fits directly in the designed enclosure.

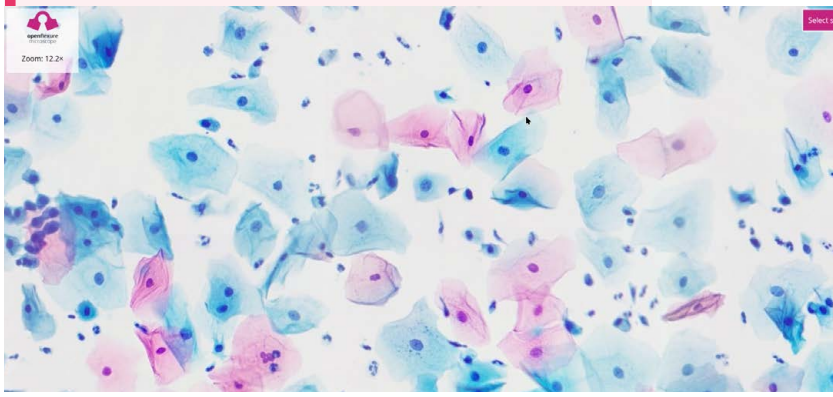
▼ **Figure 2:** The 28BYJ-48 stepper motors are affordable and highly available

Moving to the software/firmware side of things, there is a custom OS image for Raspberry Pi called 'Raspbian Open Flexure'. It comes in two variant flavours: Raspbian OpenFlexure and Raspbian OpenFlexure Lite (**Figure 3**). The former is the default option and includes a full desktop and the graphical microscope control environment, so you can hook up a monitor mouse and keyboard directly to the Raspberry Pi in your OpenFlexure microscope and run everything from there. The Lite version has everything the OpenFlexure Microscope needs to run, but does not include the desktop environment; this, then, is intended for OpenFlexure microscopes that will be controlled via a network connection and another computer, or you can control the microscope from a web browser. This is a great approach as, in many environments, having access to your regular suite of computing tools and being able to image directly to your machine can be really valuable.

On the full-fat version of the Raspbian Open Flexure OS image, you'll find the OpenFlexure Connect software, which is a very nice web interface for operating the microscope. If you are using another machine to control the microscope, you can find downloads for Windows and Linux (with Arm and x86-64 app images, as well as Ubuntu snap packages) on the OpenFlexure website and you can find guides to build the software for macOS.

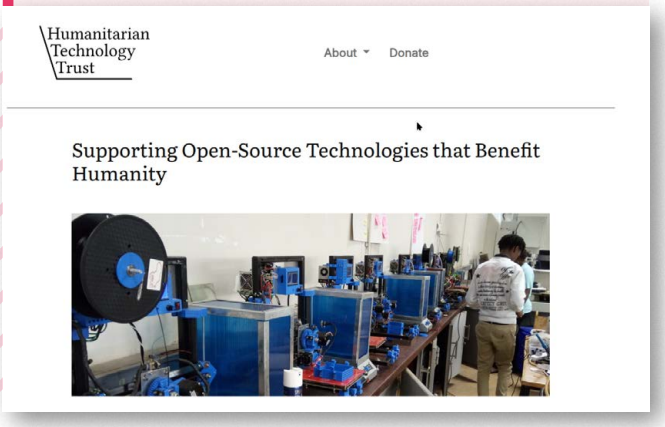
Online viewer

If you are curious about what can be achieved with this fantastic project, there's heaps of information about use cases online. However, to get a tangible feel of the amazing quality of imaging you can achieve with the OpenFlexure Microscope, a really nice set of examples are available in the online viewer (**images.openflexure.org/viewer**). Click the 'Select sample' button in the top right to switch between them. You can zoom in and out – with finger gestures on a touchscreen, or keyboard zoom controls in a desktop browser – to marvel at the depths that can be achieved.



The Humanitarian Technology Trust

It's clear to see the potential impact the OpenFlexure microscopy projects could have in the world and if you want to support the project, there are numerous ways to do so. Firstly, get involved and spread the word, show people the project, perhaps build one, chat to people in the community. If you want to support the project financially, then it's possible to make a donation via the Humanitarian Technology Trust. This is a new registered charity focused on advancing humanitarian technology through local manufacturing. It's a great area to focus on as it not only helps place new technologies in communities, it also helps build resilience and self-reliance in those communities. The OpenFlexure Microscope project is the first major project supported by the Humanitarian Technology Trust.



Once the OpenFlexure Connect software is up and running, it makes easy work of setting up, calibrating, and then operating your OpenFlexure Microscope (**Figure 4**). You can connect locally via an Ethernet cable or via a network. Connect is essentially a web browser and so updating your OpenFlexure Microscope for new and exciting features is primarily achieved by updating the image on the Raspberry Pi. Once you are connected, the interface makes it pretty easy to calibrate, drive and use the microscope. There's a navigation tab for jogging the machine into desired positions, options for autofocus, options for setting up scans by setting the number of images and distances between them and more. In the current stable version, the gallery will return images and will collate scanned image collections which will be saved in folders with sequential names. This means that you can use external image processing applications, such as ImageJ (**imagej.net/ij**) to stitch your images together. For those wanting the cutting edge, the current development version software for the OpenFlexure Microscope incorporates some very snazzy features such as automatic stitching of images, as well as 'smart decisions' – where, if an image is detected as having no value (i.e. a high percentage of background image), it will choose to skip that scan. This means that the microscope can automatically detect and focus on the actual target more efficiently. During the OpenFlexure team's recent FOSDEM talk, you can see some of these features running live (**rpimag.co/openflexuretalk**).

You can control it from a web browser

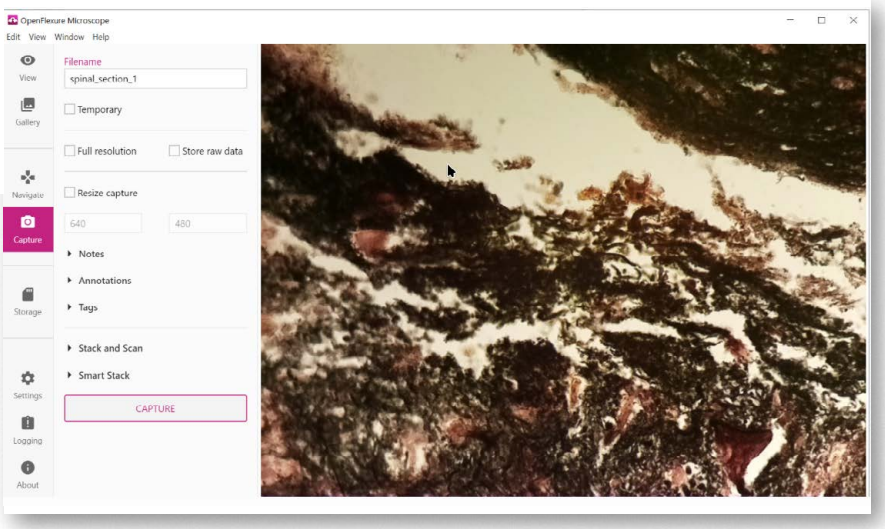


Figure 4: The control interface for the OpenFlexure microscope is straightforward and easy to use

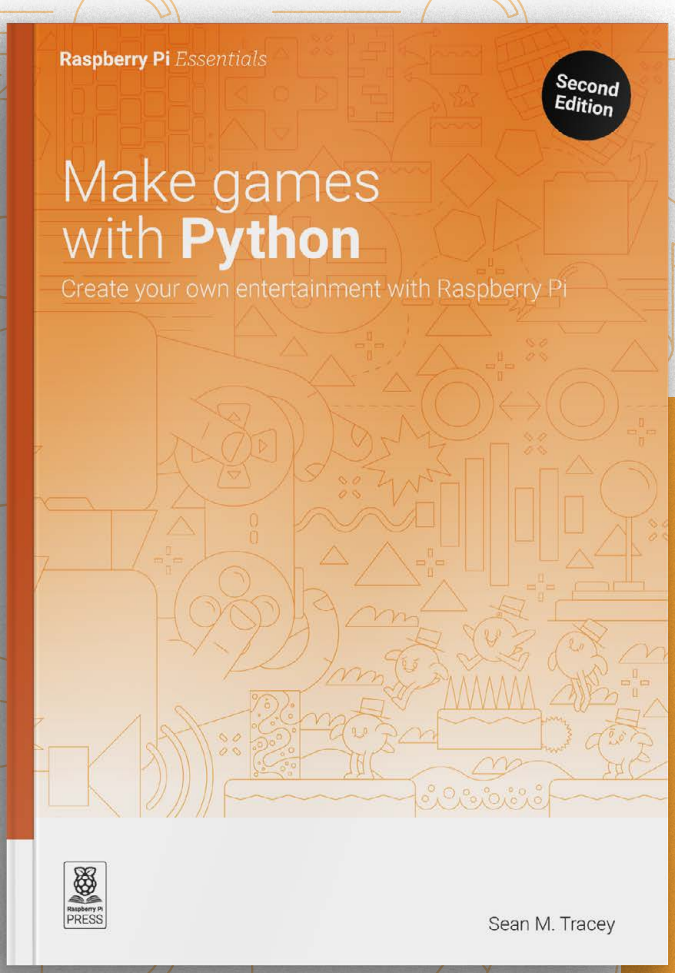
Figure 5: The OpenFlexure microscope has been used in all manner of testing environments; here it is seen in Antarctica, imaging sea ice algae in the field

One of the many wonderful aspects about the OpenFlexure microscope is how it can be made and repaired locally yet still achieve laboratory-grade imaging for a low price. The main 3D-printed parts are optimised for printing in common PLA, although there are great stories within the community of groups using recycled PET water bottles, a DIY 'pullstruder', and homebrew 3D printers to print their microscopes. The 3D-printed parts are well within the capabilities of most 3D printers, even ones with smaller beds around 150mm square. The 3D design work for OpenFlexure is done in the free and open-source OpenSCAD package. OpenSCAD is a scripted CAD environment, which means the project source files are easy to share. One feature that's really cool is that the GitHub repository hash for each individual revision of the design is physically printed into the side of the OpenFlexure print. This means that it's super-easy to see which version of OpenFlexure Microscope it is – important if you end up with a few, but more importantly trackable and auditable if being placed into a workplace.

Figure 6: An OpenFlexure Microscope in portable form being operated in a rainforest in Panama!
Copyright: Saad Chinoy, licence CC-BY

Looking around the OpenFlexure Microscope community is amazing. There's a large cross-section of people using and adapting the microscope for all types of needs. There's a few camps: those that fall into the building a microscope for fun and learning, those building for more structured microscopy education, and those using it for serious research or for medical diagnosis or pathology. It's also great to see it used in such a wide variety of physical environments. Whilst it looks very much at home on a desk in a laboratory, the fact it's essentially a Raspberry Pi means it can be deployed literally in the field, powered by a portable power bank. There's a wonderful picture of it being used in Antarctica, out on the ice, where it was imaging sea ice algae for research and investigation (**Figure 5**). At the other end of the scale weather-wise, there is a great set of images of Julian Stirling using the device in the rainforests of Panama (**Figure 6**). Julian was specifically looking at increasing the portability of the device and wanted data on how it performed in high-temperature and high-humidity environments.

There are too many parts of the world where a commercial lab-grade microscope might be too expensive. Further compounded by a lack of service engineers in all continents, this means an area may never be able to afford to receive a spare part or a service. In many of these places, the OpenFlexure Microscope makes a huge amount of sense. In fact, it's clear that this could have a huge impact on things like earlier diagnosis of malaria and also cancer diagnosis in under-resourced areas. As such, it was listed in the World Health Organisation's *Compendium of Innovative Health Technologies for Low-Resource Settings 2024*. It's a fabulous project that makes the world a better place. We look forward to seeing it thrive.



While millions of us enjoy nothing more than spending hours racking up high scores on our favourite video games, too few are exposed to an even more gratifying way to spend time — making them.
This book teaches Python and Pygame development, helping you to understand the games you play and create almost anything your imagination can come up with.

- ***As you work your way up to creating your own shoot-'em-up game, you'll learn how to:***
  - *Create shapes and paths*
  - *Move sprites and detect collisions*
  - *Handle keyboard, mouse, and gamepad input*
  - *Add sound and music*
  - *Simulate physics and forces*

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

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 absolutely need one of these ourselves

02. A classic Raspberry Pi Zero project – we should try a Pico one

03. We've not done any 'vibe coding' ourselves, but maybe after this video we will

04. This looks extremely rugged

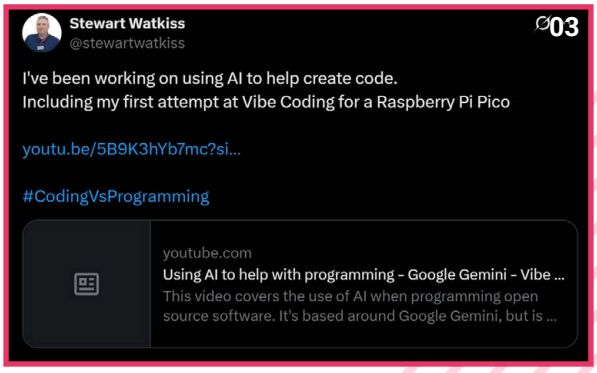
05. We do keep looking at making something like for a car dashboard…

06. It may be cross-shaped, but it's not a propeller. Yet

07. It's project enough for us! A great Pico experiment

08. It's been a while since we've seen a LEGO robot; this one is very advanced

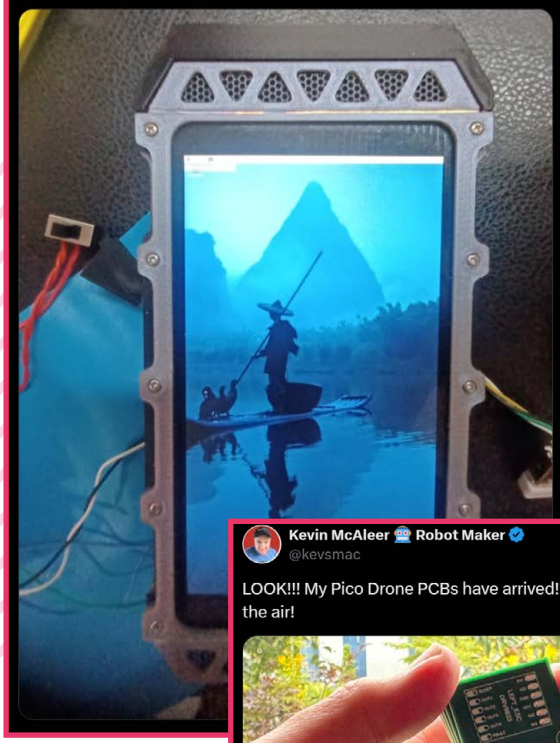
09. Live tracking sounds very fun



bootdsc
@bootdsc

04

Just building a pi5 powered "rugged" 4g lte cellphone.



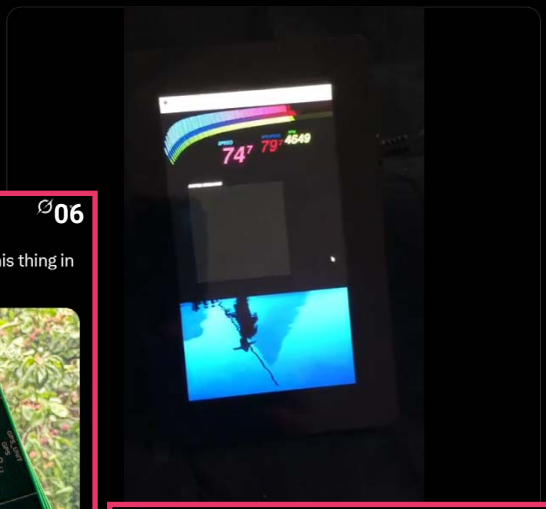
TGD
@TGD_Consulting

05

Happy #MakerMonday!

I've been experimenting with an old Rebol2/View script from @ameridroidSBC on the new @Raspberry_Pi Touch Display 2. It looks really great.

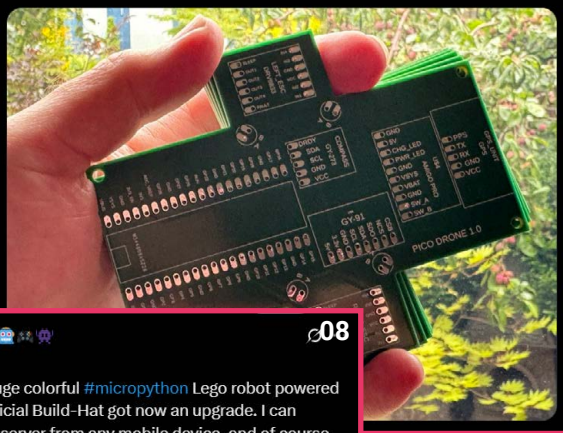
Next, I'm going to code some dashboard stuff with Rebol2/View for my projects. #rebol



Kevin McAleer Robot Maker
@kevsmac

06

LOOK!!! My Pico Drone PCBs have arrived!!! Can't wait to get this thing in the air!



Ohad Gertel
@thegertel

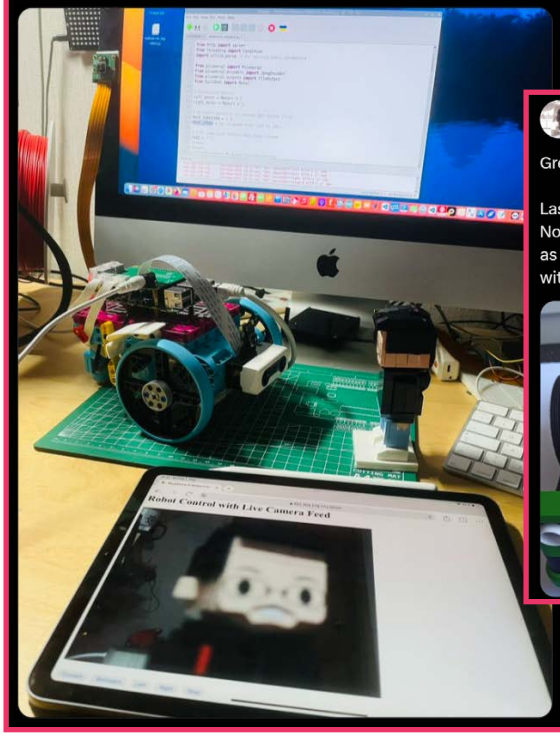
07

Hardly a project but using PIO as a class D amplifier to directly drive a speaker at 12bit / 18kHz

Roland Schulz
@r_schulz_maker

08

Hi #MakerMonday, my huge colorful #micropython Lego robot powered by @Raspberry_Pi and official Build-Hat got now an upgrade. I can control the robot via web server from any mobile device, and of course, with live #raspberryPi Camera Module 3 feed. #maker #robotics

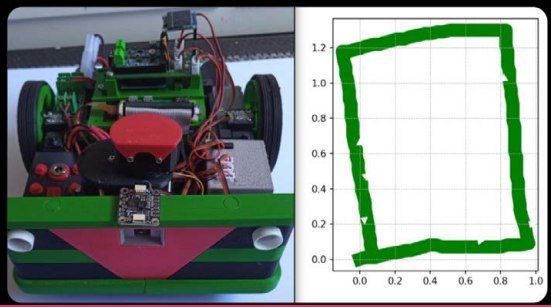


Ulli Sundermann
@ullisun58

09

Great #MakerMonday

Last Monday, I shared my tests with the BNO. x.com/ullisun58/stat...
Now I'm able to record the path of the small test mower. Currently, only as matplotlib png during the next few days, I plan to create a website with live tracking



Raspberry Jam @ Wikimania Nairobi

Wikimedia meets Raspberry Pi

Wikimania is the annual event that gathers together all the various Wikimedia volunteers from around the world and it's full of various workshops and panels and such – and this year, that included a Raspberry Jam for the first time.

Taking place in the Hackathon Room of the event, there were demos, mini-workshops, and general Raspberry Pi discussion.

"The event was a fantastic success, with about 15 enthusiastic people participating," organiser Suyash Dwivedi says. "It was a great opportunity to introduce the Raspberry Pi community to the wider Wikimedia audience. We're already planning our next event and are excited to continue building this new community."

▲ The full crew at Raspberry Jam @ Wikimania Nairobi

▲ Wikimedia contributors discussing Raspberry Pi

▼ Nerd HQ? Yeah, that makes sense

Crowdfund this

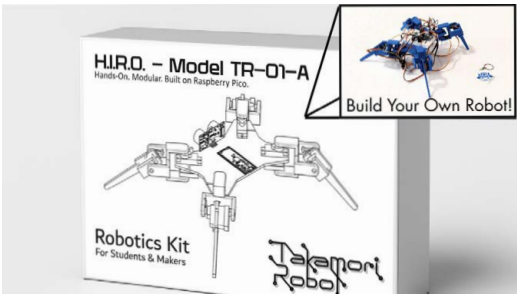
Great crowdfunding projects this month

Ubo Pod

We first saw this coming to Kickstarter a few months ago; however, it looks like the launch was delayed. Things seem to be more ready for this multimodal AI assistant powered by Raspberry Pi that supports vision and voice. For more info, check out the project showcase in this issue!

▶ **rpimag.co/ubopod**

H.I.R.O.



A modular robot kit for students and makers powered by Raspberry Pi Pico, it stands for Highly Interactive Robotic Organism. We're pretty sure that's a backronym, but it does come with motors, various sensors, and can be easily built upon.

▶ **rpimag.co/hiro**

Your Letters

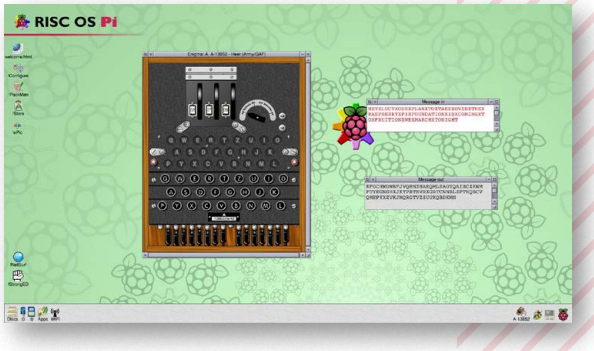
More on the Enigma machine

In the last magazine you are describing the Enigma machine. The article is interesting; however, your editorial comment has historical flaw re who broke the Enigma code. According to historical sources, "the code of the German Enigma cipher machine was first broken by Polish cryptologists, partially by a team of mathematicians from the Cipher Bureau of the Second Department of the General Staff of the Polish Army: Marian Rejewski, Jerzy Różycki, and Henryk Zygalski. They achieved this in the early 1930s, several years before the outbreak of the Second World War. Their work and knowledge were passed to the Allies, including the British."

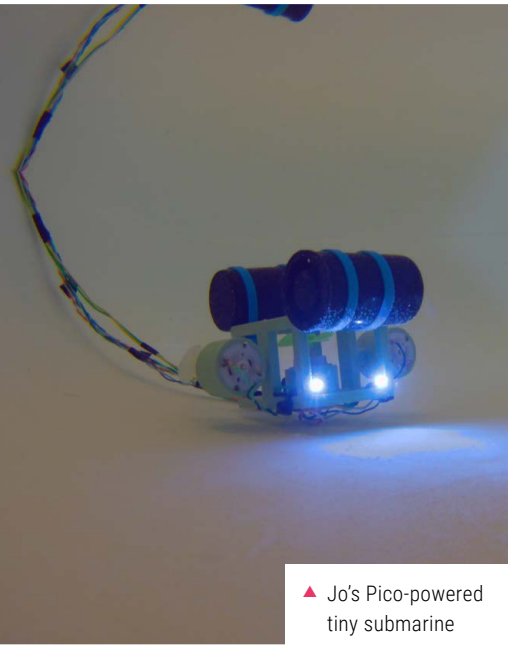
I understand your British roots as a company, but it would be good not to distribute misinformation related to history and to check facts properly.

Jerzy Chlebowski via email

This is absolutely correct, and we should have known better. In fact we did know better, but somehow didn't mention the work of Polish cryptographers who cracked the early versions of Enigma. We're really sorry about that. You can, however, read more details about the 'bombe' devices they used to decrypt the code in our book, *The Computers that Made the World*.



▶ Breaking the Enigma code in RISC OS



▲ Jo's Pico-powered tiny submarine

Tiny Open Source Underwater Vehicle (TOUV)

I've been thinking about a submarine for ages, mainly inspired by the LEGO submarine built by the Brick Experiment Channel a couple of years ago, which used a Raspberry Pi Zero 2 W and a really – really – clever coupling system so that the motor stays dry and the propellor stays wet [visit rpimag.co/beclegosub for the build instructions]. The trouble is that I've been thinking too big, and it's got in the way of me actually doing anything. I'm going to start small, and copy your writer's design; that way I might actually get something finished this decade!

Robert via email

The perfect is the enemy of the good; done is better than perfect. We can overthink until the cows come home, but there's no substitute for sitting back and knowing that you've completed a job. We remember the LEGO submarine. It used syringes and actuators to control the amount of ballast in the submarine, enabling it to control its buoyancy and move up and down in the water. We doubt it was the creator's first attempt – you only get that good by starting small, making mistakes and fixing them in subsequent iterations. With that in mind, Jo Hinchliffe is working on a bigger version of his sub (featured last issue, **rpimag.co/157**), with more sensors, and using a Raspberry Pi 5 rather than a Pico – coming soon to this magazine!

▲ The DOOM cartridge feels like magic

DOOM for the SNES

I've long thought that the best thing about open-source hardware and software isn't that I can play about with things; it's that people far cleverer than me can, and make cool stuff that I wouldn't have the slightest idea how to do. Stuff like the DOOM cartridge that uses an RP2350 chip to play a PC game on a Nintendo SNES: how do you even know that's possible? Where do you start? For someone like me, who feels pretty chuffed if I manage to get some LEDs to change colour, re-engineering a game from one platform to work on another platform feels like magic. I know it's not, but it still feels like it. Now excuse me as I dust off my SNES to kill some zombies.

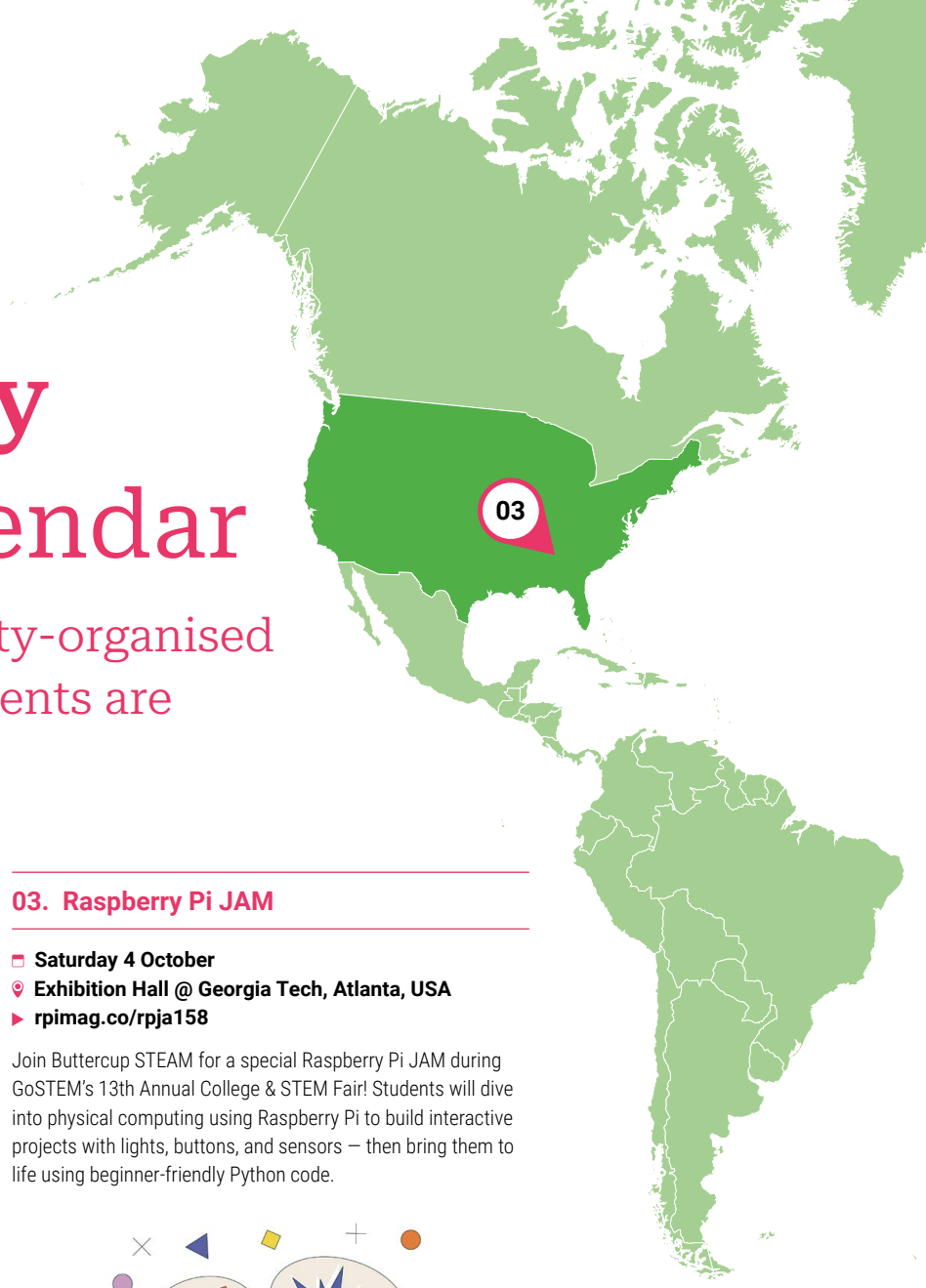
Chris via email

The tribalism that accompanied the great console divide of the 1990s was something to behold. You were either team Nintendo or team Sega. You knew where you stood – nobody played games on a computer, until suddenly everyone did [cough, Amiga – Ed]. And the Xbox, PlayStation, and other platforms really muddied the waters. Suddenly your choice of games platform wasn't part of your identity any more, and we had to find other things to argue about. That said, the existence of a DOOM cartridge for the Super Nintendo Entertainment System brings us joy, even if we sold our SNES many moons ago.

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 Community Meetup Taiwan #47

- **Tuesday 30 September**
- **Tenlong Bookstore, Taipei, Taiwan**
- **rpimag.co/taipei47**

For the 47th Raspberry Pi meetup, the topic is "Raspberry Pi AI Acceleration Solution. Speaker 1: sosorry; topic: When Raspberry Pi meets AI." [Yes, that's its name – Ed])

This is the topic of their COSCUP presentation this year. They will introduce object recognition using Raspberry Pi 4 and 5, acceleration methods and operations using the Google Coral USB Accelerator, Raspberry Pi AI Camera, and Raspberry Pi AI HAT+, along with practical demonstrations.

02. CoderDojo Oktober 2025

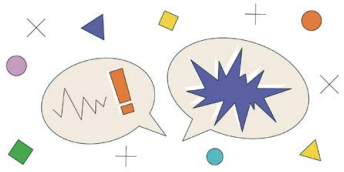
- **Saturday 4 October**
- **Stichting Cultureel Centrum Baarle, Baarle-Nassau, Netherlands**
- **rpimag.co/cdo158**

On 4 October, this CoderDojo will organise the season start for the eighth time. You can join to experiment with programming, robots, laser cutting, and many more things.

03. Raspberry Pi JAM

- **Saturday 4 October**
- **Exhibition Hall @ Georgia Tech, Atlanta, USA**
- **rpimag.co/rpja158**

Join Buttercup STEAM for a special Raspberry Pi JAM during GoSTEM's 13th Annual College & STEM Fair! Students will dive into physical computing using Raspberry Pi to build interactive projects with lights, buttons, and sensors — then bring them to life using beginner-friendly Python code.



04. Coventry Raspberry Jam Show and Tell '25

- **Thursday 9 October**
- **Blue Coat C of E School, Coventry, UK**
- **rpimag.co/crj158**

Join the team for an exciting Raspberry Jam Show & Tell event on Thursday 9 October 2025, from 5:00pm to 6:45pm in the school hall. This event is a fantastic opportunity for students, hobbyists, and tech enthusiasts to showcase their Raspberry Pi-based projects and share ideas with others in the community. The venue offers ample seating and includes a projector connected to a desktop computer for presentations.

FULL CALENDAR
Get a full list of upcoming community events here:
rpimag.co/events

05. Maker Faire Rome

- **Friday 17 October to Sunday 19 October**
- **Gazometro Ostiense, Rome, Italy**
- **rpimag.co/mfrome25**

The Raspberry Pi team is proud to partner with Raspberry Pi Approved Reseller Melopero to be at Maker Faire Rome again in 2025. Come and meet members of the Raspberry Pi team, learn about the latest products, and share what you've made with Raspberry Pi technology.

Official Raspberry Pi Event

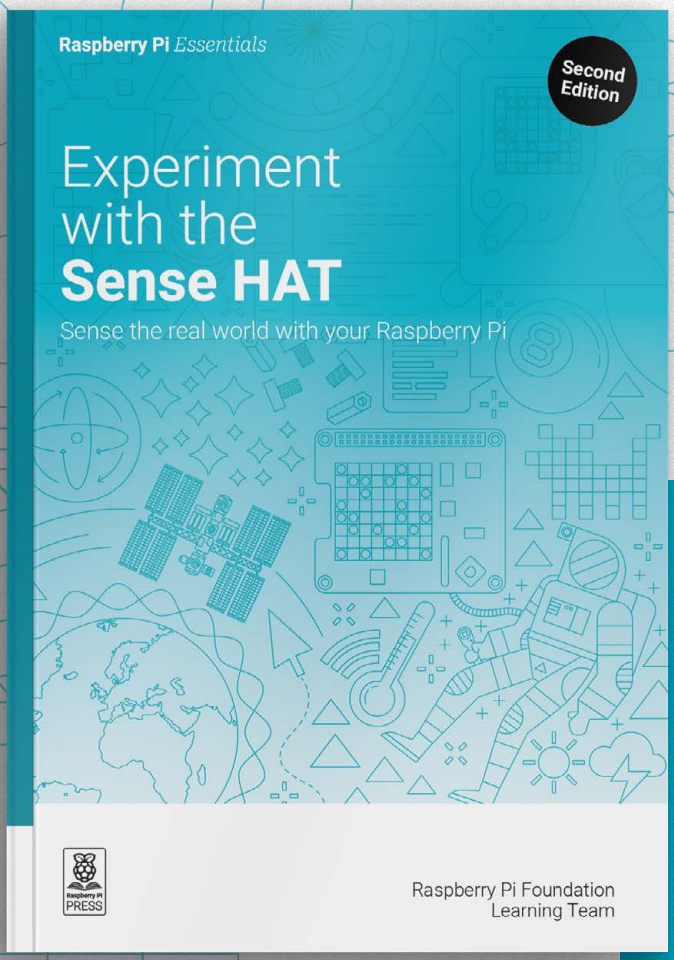
Win 1 of 5 Raspberry Pi SSD 1TB

Want to upgrade the storage in Raspberry Pi 500+ or attach a super-fast NVMe SSD to Raspberry Pi 5 with the M.2 HAT? Raspberry Pi's new high-quality 1TB solid-state drive unlocks outstanding performance for I/O intensive applications, including super-fast startup. It is a reliable, responsive PCIe Gen 3-compliant SSD capable of fast data transfer. 256GB and 512GB versions are also available.

Head here to enter:
rpimag.co/win
Learn more:
rpimag.co/ssd

Terms & Conditions

Competition opens on **24 September 2025** and closes on **30 October 2025**. 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.

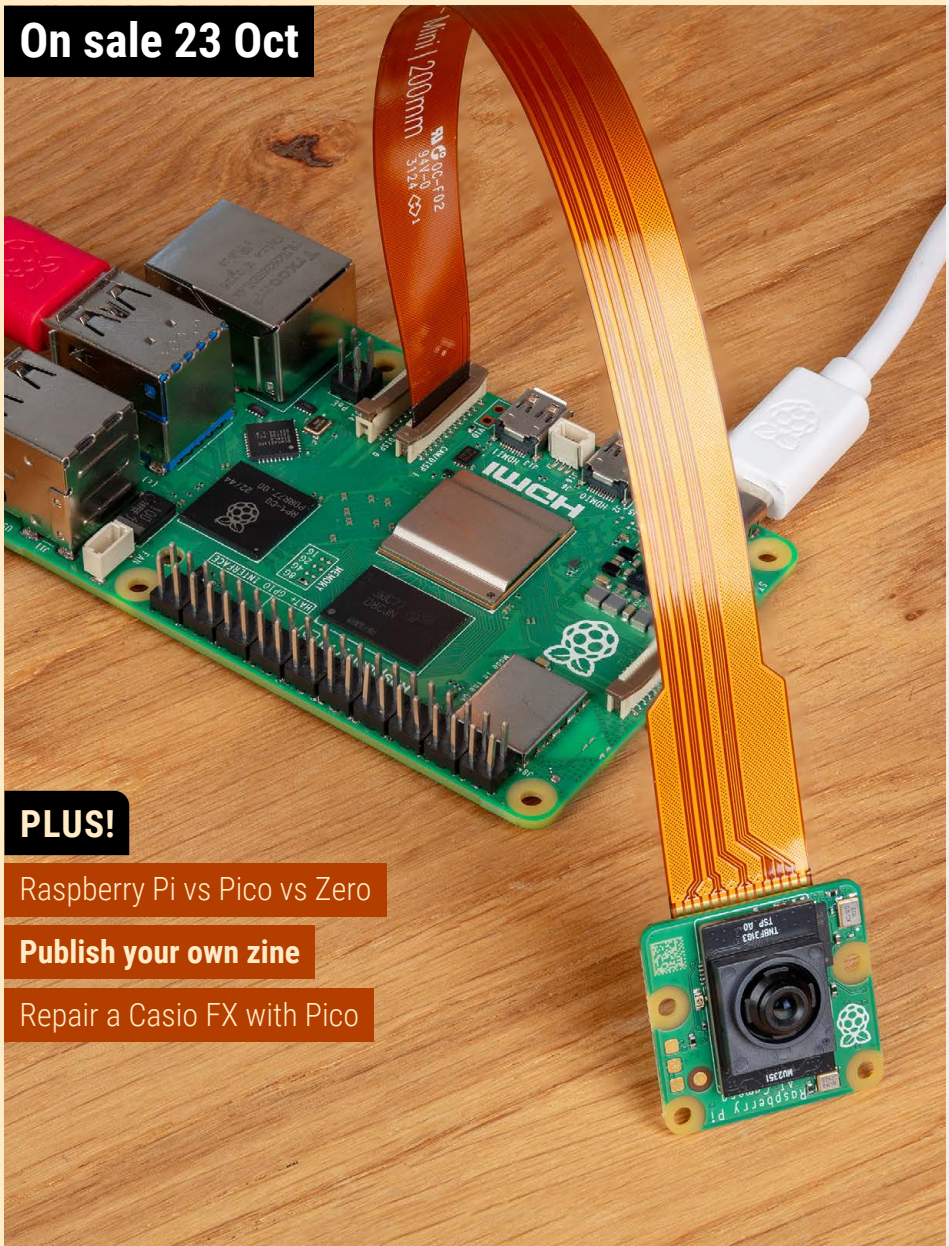
Take the next step –
advertise with us today!

Official Magazine
#159

Next Month

How to train your AI

Train your own AI models using Raspberry Pi and Google Colab



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, Ben Everard, Rosie Hattersley, Jo Hinchliffe, Phil King, Nicola King, Richard Smedley, Sean M Tracey, 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)

Plus good!

Raspberry Pi 500+ is solid technology. By **Lucy Hattersley**

So, Raspberry Pi 500+ has launched and this is the last word in our Raspberry Pi 500+ special edition. It's been a fantastic time to work on a Raspberry Pi magazine.

Over the last few months we've seen Raspberry Pi 500+ come to life: slowly teased out through various test units with niggles smoothed and kinks ironed out. Doubtless we will discover more about this incredible device over the next few months, but for now we're just going to enjoy seeing it glowing on the cover.

I'm not personally done with my Raspberry Pi 500+. I have a 1TB SSD that landed on my desk today, which I'm going to place into my 500+ as a demonstration of how to upgrade the drive. I'm not totally sure what I will do with all that storage: I've been meaning to hack an old iPod with Rockbox and follow KG's tutorials on using a CD/DVD-ROM drive, so maybe it can be a media storage powerhouse.

I've also ordered a set of replaceable keycaps (which are blank for nerd laughs) and I'll be doing a keycap replacement tutorial next month. I've never been that convinced about the blank keycaps as a practical way to use a computer, but they do look ever so stylish in the photos.

Having Raspberry Pi as my main computer has been a sea change. I still need to fire up non-Linux operating systems from time to time, mostly because I need Adobe software to check the magazine pages. But this is increasingly a siloed activity that I do apart from my main job: which is hacking and messing around with technology to see how far you can push it.

> *Raspberry Pi is the ideal playground*

What are you up to?

I'm interested to see what other people are going to do with Raspberry Pi 500+. Do not underestimate how drastic a change it is to have a Raspberry Pi computer that is fully loaded with 16GB RAM, 256GB SSD, running a lightweight and fast Linux operating system, and with an incredible keyboard. All in one space. It's been a much more dramatic change than I expected: even more so than Raspberry Pi 500.

I've been cracking on with Nand2Tetris with my 500+. It's fascinating to learn how chips work at the basic level, while using a computer that you can take apart. I wish I could see all the way down into the Arm CPU. This is my second time around on Nand2Tetris (I bounced at the part where you build the operating system last time). I'm hoping that I've learnt enough now to push on to the end.

Learning exactly how computers work, and how to make real-world interaction with computers (what is known as 'physical computing'), is at the very heart of Raspberry Pi. I'm incredibly proud to be a small part of a much bigger team that produces such incredible things.

It never rains, but it pours. Not that we would have it any other way. Playing around with technology is an absolute joy and Raspberry Pi is the ideal playground for us.

Lucy Hattersley – Editor

Lucy promised herself she wouldn't write about AI this month despite 'AI Caramba' suggested as a great headline. You wouldn't believe the amount of AI tutorials that are in the pipeline.

rpimag.co

HiPi.io

HIGHPI PRO

The new case from the HiPi.io team

- Rapid tool-free assembly and disassembly
- Large internal volume for HATs
- Compatible with Pi 2/3/4
- Multiple lid options
- Passive & Active Cooling options
- Secure microSD card cover
- VESA mount support
- Molding-configurable output ports customizable for volume orders
- Printed logo for your branding

Available at these great Pi stores:

PiShop.us PiShop.ca PiShop.mx 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 · The PiHut · PiShop.us · CloudFree · KUBII · Cytron Marketplace

ROBU.IN Your ideas, Our Parts · ANTOP 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: