

Create
a hologram with
a cube

Affix
a chronowatch to
your wrist

Understand
search algorithms
with Dijkstra



Official Magazine
#168 | August 2026

Raspberry Pi



CYBERDECK

PROJECTS



Survive in style
with Raspberry Pi



£7.99

ISSN
2977-4403
(Print)

Industrial Raspberry Pi **ComfilePi**



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

© copyright COMFILE Technology, Inc. ALL RIGHTS RESERVED

COMFILE
TECHNOLOGY

Welcome to Raspberry Pi Official Magazine

**Editor****Lucy Hattersley**

Lucy's attempt at cyberpunk streetslang on the cover was met with blank stares, and frankly embarrassed shuffling from younger team members. Next time, chooms!



rpimag.co

Cyberdecks are portable, hackable, low-tech computers inspired by the cyberpunk universe – a glorious mash-up of high-tech gadgetry and dystopian living.

The idea is simple: build a computer that's portable, has a small screen, and – ideally – a physical keyboard. Think of it as a laptop reimagined from a retrofuturism universe.

Homespun cyberdecks are hugely popular right now, and Raspberry Pi sits at the heart of most builds. This month we've got a lead feature packed with cyberdeck builds and a brilliant tutorial from Brian Corteil. This is a superb project for levelling up your maker skills – the reward is a computer that looks nothing like anything on a shop shelf.

Lucy Hattersley – Editor





Complete Edge AI Hardware Stack for Raspberry Pi

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

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

START HERE



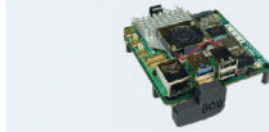
from \$63

SIXFAB AI HAT+ FOR RASPBERRY PI 5

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

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

EXPAND



from \$190

SIXFAB EDGE AI EXPANSION BOARD

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

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

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

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

DEPLOY



from \$650

SIXFAB ALPON X5 AI

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

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

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

SIXFAB × ULTRALYTICS ACCELERATION PATH

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



Raspberry Pi
Design Partner

Intelligented by
DEEPX



ultralytics | Sixfab

Scan to browse
and pre-order



Contact: hello@sixfab.com

© 2026 Sixfab, Inc. 825 Watter's Creek Blvd., Suite 250, Allen, Texas 75013 USA

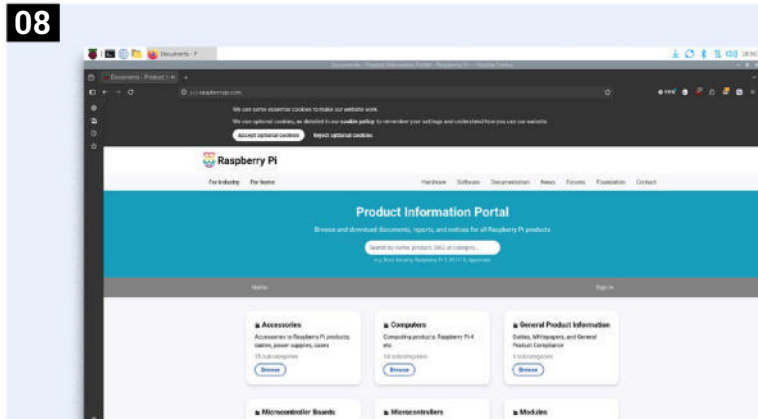
Contents

World of Raspberry Pi

- 008 Cyber resilience and Raspberry Pi
- 020 Subscribe to Raspberry Pi Official Magazine

Project Showcase

- 012 Cubism
- 014 ChronoWatch X2040
- 016 Iris Colour Detection

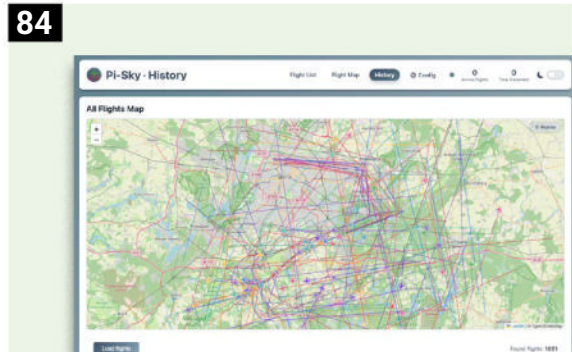


Raspberry Pi (ISSN No: 2977-4403, USPS No: 25-672) is published 12 times per year by Raspberry Pi Ltd, and distributed in the USA by Asendia USA, 701 Ashland Ave, Folcroft PA. POSTMASTER: send address changes to Raspberry Pi, 701 Ashland Ave, Folcroft, PA. 19032

24



84



Top Projects

- 022 Claw stand
- 024 HAL 9000
- 026 Inky Bird Frame
- 028 Smart money box
- 030 3D print showcase

Feature

32



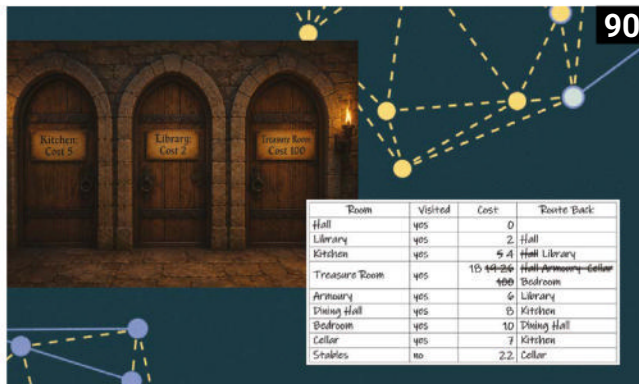
- 032 Cyberdeck projects:
Hacking in style with Raspberry Pi

Tutorials

- 046 Build your own Pi-hole
- 054 Conquer the command line – part 13
- 060 Computers that Made the World: Pilot ACE – part 2
- 070 Make a 4-axis foam cutter – part 2
- 076 Craft wearable beaded tech
- 080 Simple electronics with GPIO Zero – part 9
- 084 Create a Pi-Sky flight tracker

Feature

90



- 090 Coding with Dijkstra's Algorithm:
The secrets behind A-level computing

100



Reviews

- 100 **Only the best:**
touchscreen displays
- 106 Pimoroni Presto
- 110 CrowView Note 15.6
- 112 Powered by Raspberry Pi
- 116 **Ten amazing:**
driving projects

Raspberry Pi Community

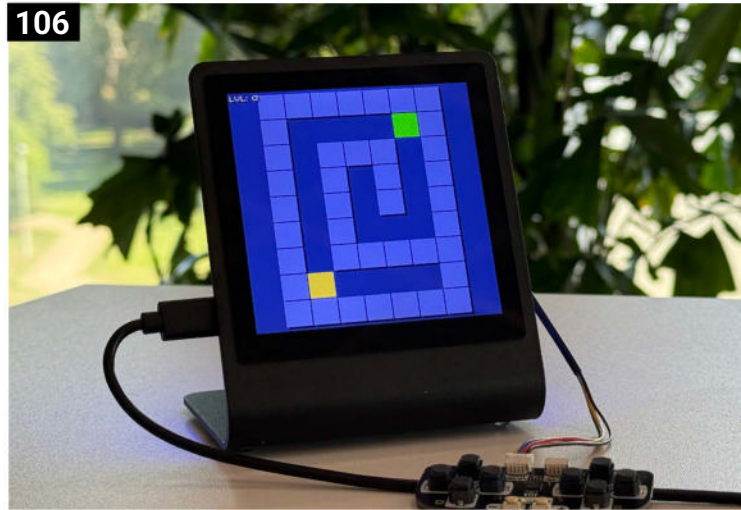
- 118 Interview: Suyash Dwivedi
- 120 This Month in Raspberry Pi
- 124 Your Letters
- 126 Community Events
Calendar

Competition

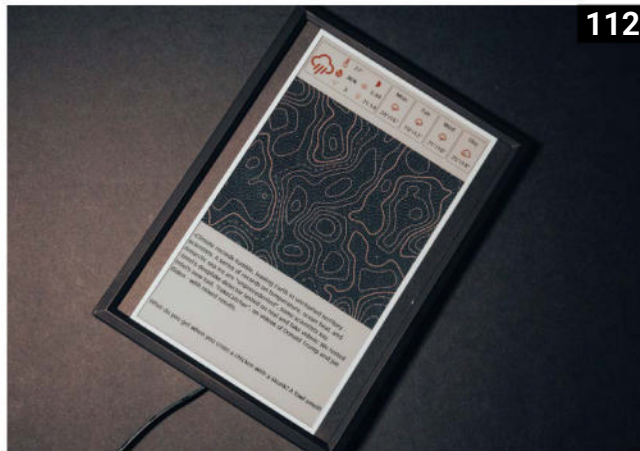
128

Win 1 of 5 Pitower Gen 1 cases

106



112



Disclaimer: Some of the tools and techniques shown in Raspberry Pi Official Magazine are dangerous unless used with skill, experience, and appropriate personal protection equipment. While we attempt to guide the reader, ultimately you are responsible for your own safety and understanding the limits of yourself and your equipment. Children should be supervised. Raspberry Pi Ltd does not accept responsibility for any injuries, damage to equipment, or costs incurred from projects, tutorials or suggestions in Raspberry Pi Official Magazine. Laws and regulations covering many of the topics in Raspberry Pi Official Magazine are different between countries, and are always subject to change. You are responsible for understanding the requirements in your jurisdiction and ensuring that you comply with them. Some manufacturers place limits on the use of their hardware which some projects or suggestions in Raspberry Pi Official Magazine may go beyond. It is your responsibility to understand the manufacturer's limits.

Understanding the EU Cyber Resilience Act

If you manufacture and deploy digital products in the European Union, the EU Cyber Resilience Act (CRA) is something you need to understand. By **Tom Westcott**

At Raspberry Pi, we've been monitoring the development closely and are ready to support customers integrating our computers into their designs.

What is the Cyber Resilience Act?

The CRA is the EU's primary legislation focused on the cybersecurity of digital products. It applies to any hardware or software with digital elements, including Internet of Things (IoT) devices, embedded systems, industrial controllers, smart home devices, and much more; if it's a connected product sold in Europe, it's almost certainly within scope. The CRA will follow the product compliance procedures established in the New Legislative Framework Regulation (EC) No 765/2008, commonly known as CE marking (rpimag.co/cra).

Manufacturers must begin with a cybersecurity risk assessment, evaluating how the digital product operates and where any potential vulnerabilities may lie. Depending on the outcome of the risk assessment, manufacturers may need to implement security features, securely maintain products throughout their life

► The EU Cyber Resilience Act will be fully applicable on 11 December 2027



cycle, provide vulnerability handling processes, and be transparent with customers about security capabilities and limitations.

Manufacturers must also take steps to improve the cyber resilience of their products – from the integrity of the communications they send and receive, through to the confidentiality of their data – by carrying out regular reviews and security tests. Penalties for non-compliance can reach €15 million or 2.5% of global annual turnover.

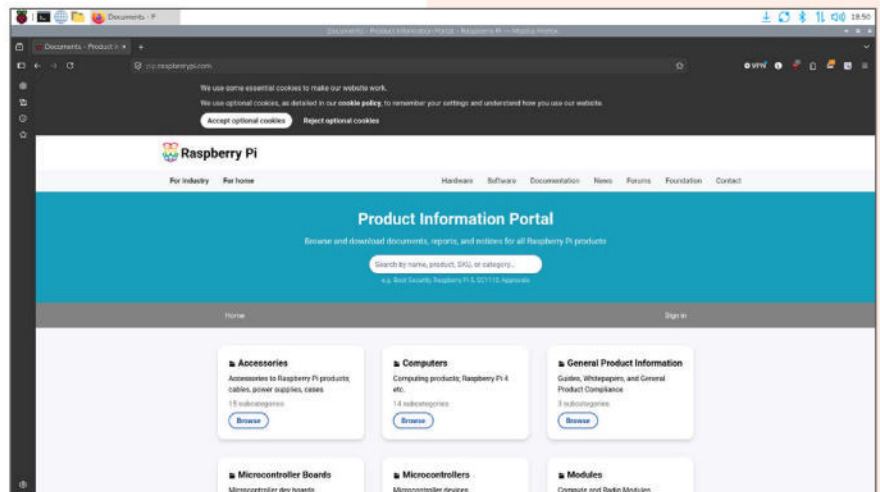
Timelines and requirements

The full CRA will come into force on 11 December 2027, which is the date by which products must meet the requirements outlined in the Annexes. When that time comes, all in-scope products must be CE-marked. However, for teams building products today – choosing silicon, designing firmware architectures, planning certification strategies – the decisions made now will determine whether they arrive at that deadline with confidence.

Vulnerability and incident reporting will become mandatory on 11 September 2026. Manufacturers marketing connected products in the EU must report any actively exploited vulnerabilities or severe security incidents affecting their products. They will have 24 hours to file an early warning and 72 hours to submit a full notification, via a new central reporting platform enabling the secure exchange of data between European Computer Security Incident Response Teams (CSIRTs) and the European Network and Information Security Agency (ENISA).

We're here to help

Raspberry Pi products are at the heart of an enormous variety of connected applications, including industrial automation, smart building infrastructure, edge computing nodes, medical monitoring equipment, and retail systems. Many of these use cases fall within the scope of the CRA. The majority fall within the default, lowest-risk category, but those building more critical infrastructure will face proportionally more demanding obligations.



- ▲ Raspberry Pi's Product Information Portal contains notes and white papers on implementing product security measures
- ◀ The CRA will follow the product compliance procedures behind the regime known as CE marking

building on a platform that has already done much of the heavy lifting. Our security architecture is documented, our vulnerability disclosure process is mature, and our commitment to supporting products with long-term security updates is well established. If you're starting to

The CRA raises the bar significantly for connected-device security

When assessed against the CRA's essential cybersecurity requirements, our hardware and software products provide a strong foundation for compliant system design. Secure boot capabilities, encrypted storage options, robust update mechanisms, and strong cryptographic primitives are all native to the platform, and Raspberry Pi OS, our official operating system, ships with a hardened default configuration.

Compliance is a responsibility shared across the supply chain. When you design a product using Raspberry Pi technology, you're not starting from scratch – you're

develop a new product, integrating Raspberry Pi technology provides a substantial head start, so our customers can focus their engineering efforts on the application itself, rather than on building security infrastructure from the ground up.

Our Product Information Portal (PIP) already contains application notes and white papers explaining how to implement security measures (rpimag.co/pip).

The CRA raises the bar significantly for connected-device security. Encouraging engineers to make products secure by design will help ensure that customers and end users are better protected. 🍓

What the Annexes cover

Annex 1: Essential cybersecurity requirements.

Annex 2: Minimum information that must be given to the customer.

Annex 3: High-security digital products, organised into two classes.

Annex 4: Critical products with digital elements.

Annex 5: Template declaration of conformity.

Annex 6: Simplified declaration of conformity.

Annex 7: Content of the technical documentation (the Technical Construction File).

Annex 8: Conformity assessment procedures, ranging from self-declaration to assessment by a notified body.

Raspberry Pi *Essentials*

AI Projects with Raspberry Pi

High-performance artificial intelligence for robotics,
security, home automation, and vision



Lucy Hattersley,
Brian Jepson

AI Projects with Raspberry Pi

High-performance artificial intelligence for robotics, security, home automation, and vision

With this essential guide, you'll learn how to create all kinds of AI-powered projects on Raspberry Pi!

Combining Raspberry Pi hardware with artificial intelligence gives you new ways to interact with the world: turn your surroundings into a playing field with computer vision models, predict and adjust to changes in environmental conditions, and engage with people in natural ways.

■ ***You'll learn how to:***

- *Identify and categorise objects in pictures or video*
- *Recognise and generate speech*
- *Translate from one language to another*
- *Use generative AI models to create images or text*
- *Train models to make predictions from sensor readings, such as temperature and acceleration*

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

Cubism

Bring images to life with this illusionary device. By **David Crookes**



Maker

Jung Rae Jang

Jung is a funemployed software engineer who loves coding, 3D printing, model building, and crash testing planes in VR flight sims.

rpimag.co/cubismgit

Quick FACTS

- It makes use of an age-old illusion
- Images are created in a web app running in Chromium
- They are displayed in a beam-splitter cube
- It can act as a floating media player
- An AI assistant is also included in the build

In the mid-1800s, an illusion technique called **Pepper's Ghost** captured the imagination of theatre-goers. Popularised by John Henry Pepper, it used controlled lighting along with the reflective properties of a sheet of angled glass to show a hidden object on stage. To the audience, this object (usually

Any displayed images look like they're floating in mid-air

a performer) would look transparent, like a ghost. The technique is still used today and it remains hugely effective.

A number of makers have sought to replicate this amazing trick at home using a beam-splitter cube, a square screen and a Raspberry Pi 5 computer. Among them is Jung Rae Jang who has created a media player and clock device with a built-in AI assistant. He took inspiration from an interactive ASCII Aquarium project he spotted on Instagram (rpimag.co/fishfoodinsta) as well as Xandar Chin's OpenGhost project (rpimag.co/openghostgit).

"I found the STL files for the project and the required parts and decided to build the project using my own code,"

Jung says. As he waited for the hardware to be delivered, he began thinking which tech stack to use for the software and basic system architecture and, as someone proficient in JavaScript, he opted for using Next.js, building in Cursor using its AI-assisted development features.

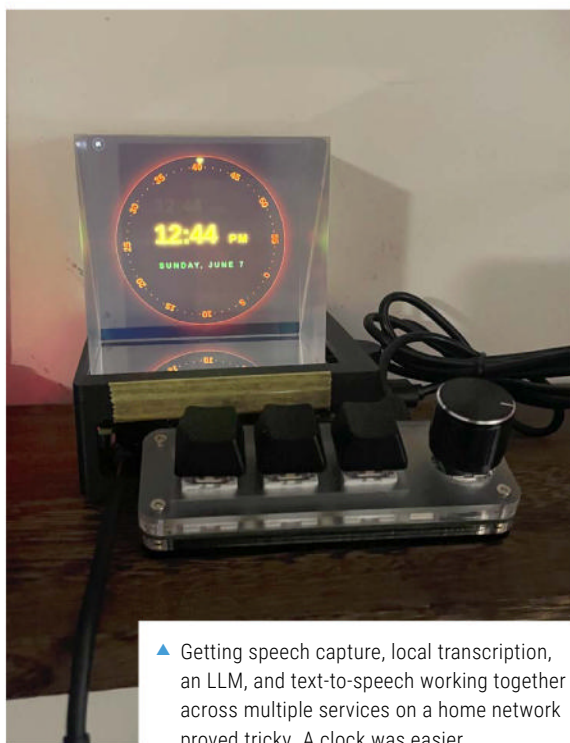
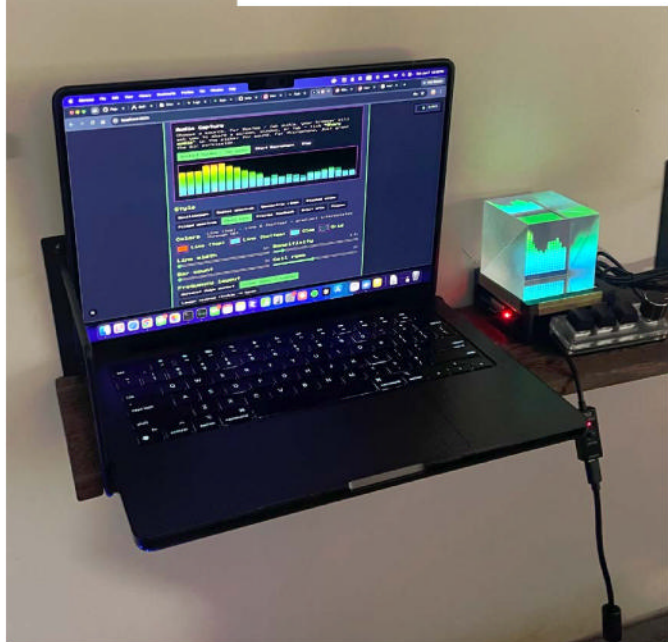
Square deal

He had a list of features he wanted to include. "I wanted features commonly found in other home assistant apps such as a clock, calendar, weather forecasts, stock market tracker and flight tracker," he says. He also wanted to include a Spotify controller and audio visualiser. By playing music, he envisaged wonderful patterns appearing within the cube.

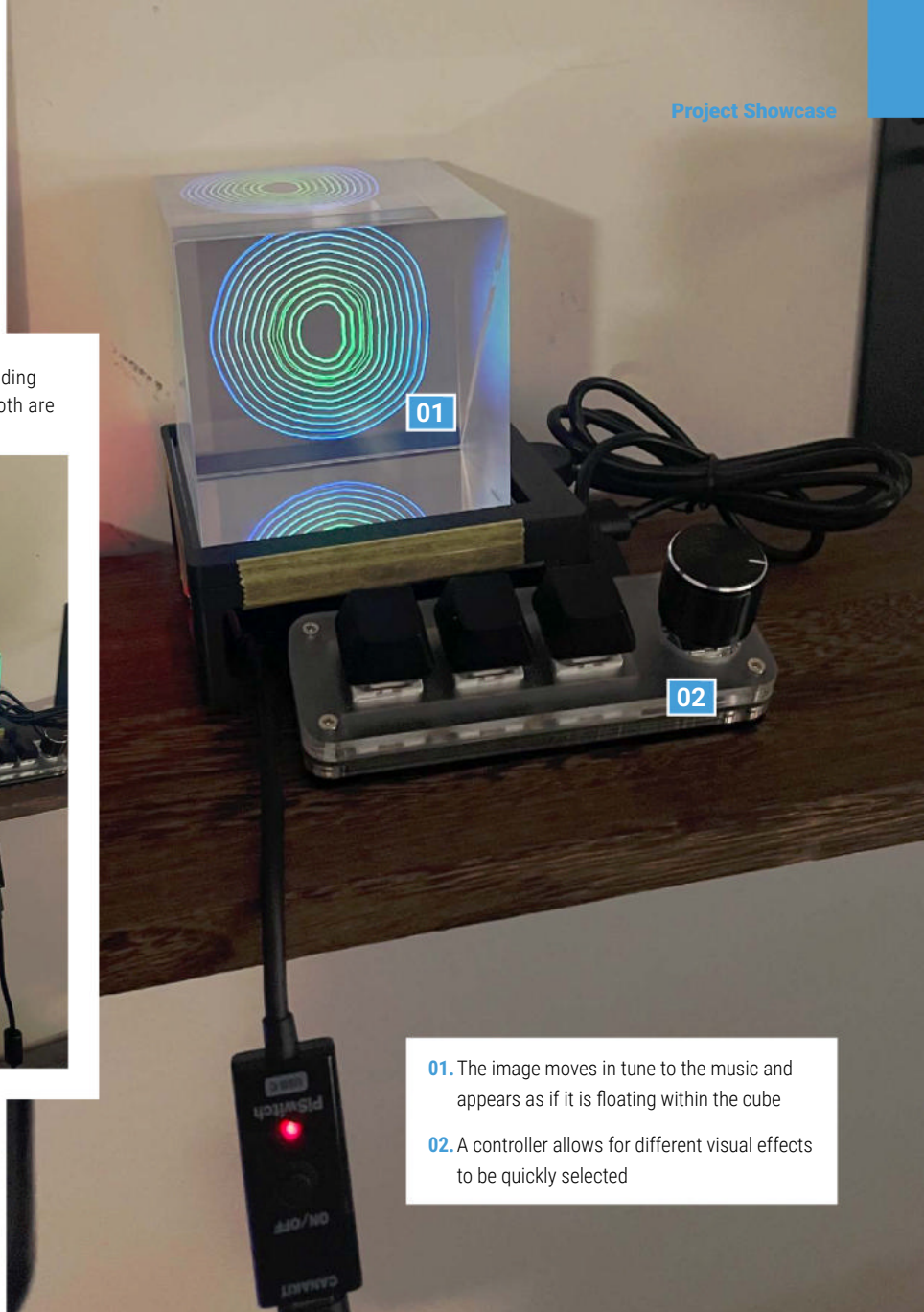
Jung achieved this by developing two apps. One is a control panel on his PC and the other is a full-screen renderer on the Raspberry Pi. In essence he's created a web app running on full-screen Chromium via Raspberry Pi 5. The two computers stay in sync over a wireless LAN through a real-time bridge.

By sitting the beam-splitter cube on the Raspberry Pi display, the Raspberry Pi is able to drive a screen underneath so that any displayed images look like they're floating in mid-air. Optimising the renderer was the greatest challenge – "the biggest headache," Jung adds – but it now works well, proving especially stunning in the dark.

▼ The control panel runs on the PC, feeding data to Raspberry Pi for rendering. Both are separate apps



▲ Getting speech capture, local transcription, an LLM, and text-to-speech working together across multiple services on a home network proved tricky. A clock was easier



01. The image moves in tune to the music and appears as if it is floating within the cube

02. A controller allows for different visual effects to be quickly selected

More power

So how does it work? Jung says he picks a module in the browser – such as clock, weather, Spotify, aquarium, and AI assistant – and Raspberry Pi immediately updates the render. “The system is designed to minimise resource usage on Raspberry Pi, so most of the heavy lifting and API calls are all made on the desktop app. Also, I want to create a system where modules can be added and managed easily for new features.”

For the AI assistant, Jung talks into a USB microphone connected to the project’s Raspberry Pi. “My PC transcribes the audio

locally, runs it through LM Studio, shows the reply on the cube display, and speaks it aloud, all on my home network,” he says. Jung uses Gemma 4 for a local LLM (large language model) with a Brave MCP (Model Context Protocol) connection for the web search.

Jung loves the effect and his head is swimming with more ideas. “I want to add more Winamp-style visuals for my audio visualiser module and I’d love to find a way to run Doom on it,” he reveals. “My friend also suggested having Zordon from *Power Rangers* as an AI assistant face. We shall see.” 🗨️

ChronoWatch X2040

It's a smart-looking watch that tells the time without unnecessary distractions. By **David Crookes**



Maker

Vishal Soni

Vishal is a YouTuber, maker, coder, and 3D designer. He has knowledge of embedded hardware and software.

[rpimag.co/
chronowatch](http://rpimag.co/chronowatch)

Vishal Soni grew up wanting to develop his own smartwatch but, when the time came to put his plan into action, he opted for something a tad simpler. Rather than festoon his device with lots of apps and tracking capabilities, he decided to focus on a watch's primary task: the ability to tell the time. In doing so, he developed an attractive timepiece that offers a range of watch faces. "I wanted to make the watch clean, steering away from distracting notifications and unnecessary step and health tracking," he affirms.

Rather than ape the squared-off look of the Apple Watch, Vishal chose the classic round design of many of its rivals. "This initially led me toward creating a custom PCB," he says, "but that was a big challenge. After doing some research, I found that Waveshare already sells a round display with a Raspberry Pi RP2040 microcontroller integrated and ready to use. I decided to use that module instead so I could focus more on programming and features rather than hardware design."

Saving time

The module in question is a Waveshare Round Touch LCD. It proved perfect for both the project and Vishal's personal development. "I mostly focus on learning and programming and I wanted to learn the Pico SDK," he says. It also meant the actual build was quite straightforward.

Vishal only really needed to connect a 3.7V 950mAh battery using an MX1.25 2P connector and produce a 3D-printed case using Tinkercad. He was then able to add an Apple Watch strap, which has lent both comfort and a modern appearance. It helped that the Waveshare board has a USB Type-C and battery charging support. It also has a built-in accelerometer and gyroscope sensors for motion detection.

Putting in the hours

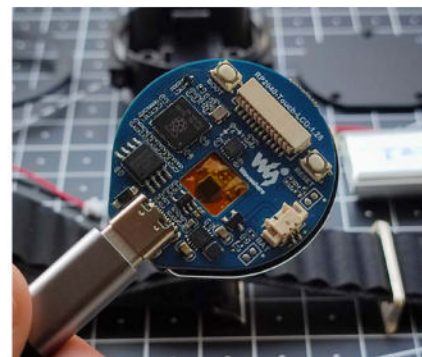
With the physical aspect of the design in hand, Vishal could turn his full attention to programming the device. "The Pico SDK was completely new to me," he says. "After working with Arduino for years, switching to pure C code with



01. Vishal isn't a professional 3D designer, so he kept the look of the watch clean and stylish

02. The multiple watch faces have been designed to match different moods, vibes and environments

▼ Waveshare's touchscreen circular display incorporates the RP2040 microcontroller, making development easier



I mostly focus on learning and programming and I wanted to learn the Pico SDK

minimal library support was challenging. I found a few basic C libraries and modified them to suit my needs and make development easier. But the touch library was particularly challenging, so I created custom functions that can detect whether a specific area of the screen has been touched, making the user interface much easier to implement.”

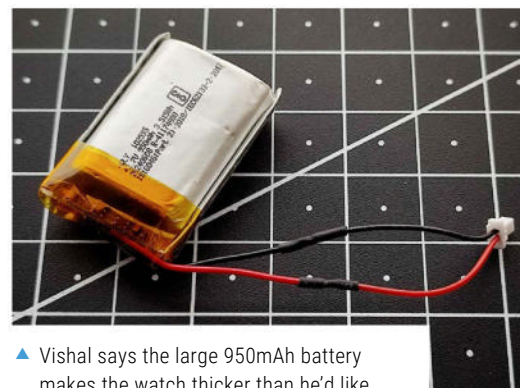
As development progressed, Vishal felt he could add some extra features. Aside from designing different watch faces and allowing a switch between the 12-hour and 24-hour format, he included adjustable display brightness, a stopwatch, the ability to rotate the watch orientation, and a sleep time mode.

The latter proved tricky. “Even with the Pico SDK, I haven’t been able to reduce the power consumption as much as I would like because the watch does not have an RTC module,” he says. “If I put the device into deep sleep, it loses the current time and date.” He’s continuing to work on a solution while also considering adding more features as time goes on.

“I’m thinking about using a real-time clock module to keep accurate time when the device is off and I’d like to add an alarm and step counter,” he says. He’d also like to make the watch thinner and reduce the battery size. “I don’t know how or when I will do that,” he tells us. “But I believe it can be done.”

Quick FACTS

- This watch features a touchscreen display
- It's powered by a Raspberry Pi RP2040 microcontroller (as used in Pico)
- There are many different watch faces
- A stopwatch feature is also included
- It's designed to use Apple Watch straps



▲ Vishal says the large 950mAh battery makes the watch thicker than he'd like

Iris Colour Detection

Take a look into a camera and find out your eye colour. **Rob Zwetsloot** has a gander



Maker

Srikrishna Karthick

A Physics master's student whose research interests and current area of work lie in the development of scientific payloads for space missions.

rpimag.co/icdetect

For the majority of people, just checking in the mirror and recognising your eye colour is fairly simple. Our brains are able to process that information pretty easily. How would a computer do it, though? Chances are if you read this magazine, you already have a half-formed answer in your head based around computer vision. Srikrishna Karthick, a student currently working for the Indian Space Research Organisation (ISRO) on their thesis, already has the answer for you.

“Every year, the Muscle Physiology Laboratory at IISER Tirupati [in India] participates in National Science Day celebrations by setting up an exhibit

the eye colours of attendees could also be recorded and their distribution displayed in real time.”

To avoid a manual tally, and manual data analysis, an electronic way was proposed. With an objective to inspire younger students, and a short deadline, Raspberry Pi ended up being the obvious choice for this.

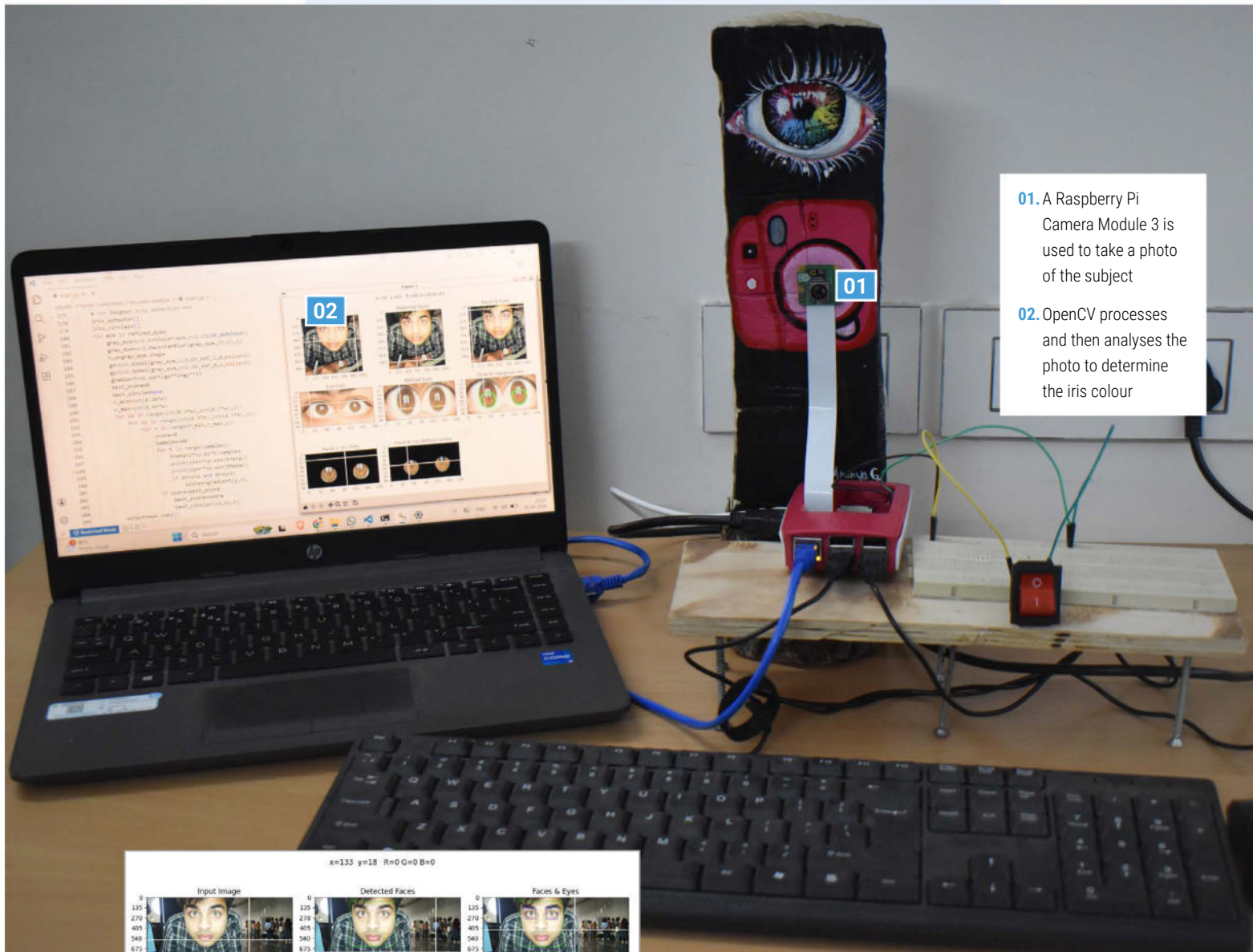
Colour detection

The build mainly consists of a Raspberry Pi 3B+ and a Raspberry Pi Camera Module 3. A button – connected to Raspberry Pi's GPIO header – is pushed, which triggers the taking of a photo that is then handed over to OpenCV for processing.

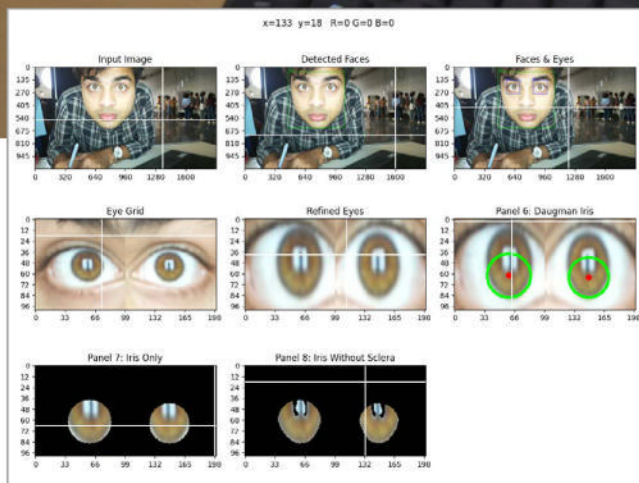
The iris is localised using a Daugman-inspired circular gradient search algorithm

alongside other laboratories and student groups,” Srikrishna tells us. “The Muscle Physiology Lab displays fruit flies (*Drosophila*) with different eye-colour phenotypes, for explaining the genetic mechanisms underlying the inheritance of eye colour. To make the demonstration more engaging and relatable to visitors, it was proposed that

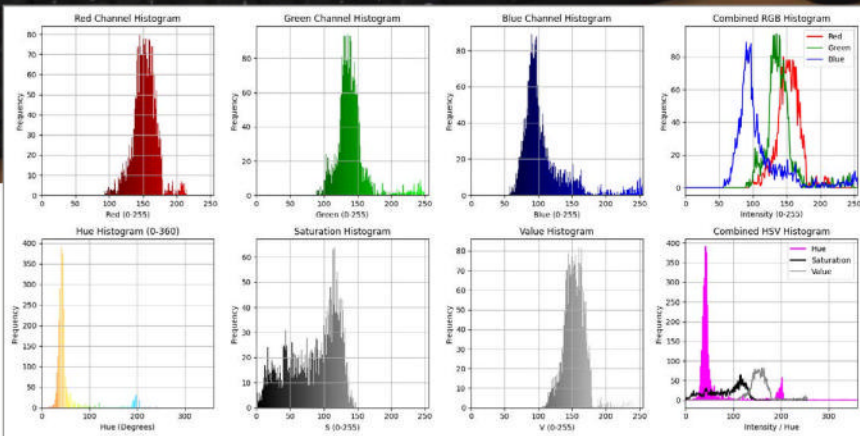
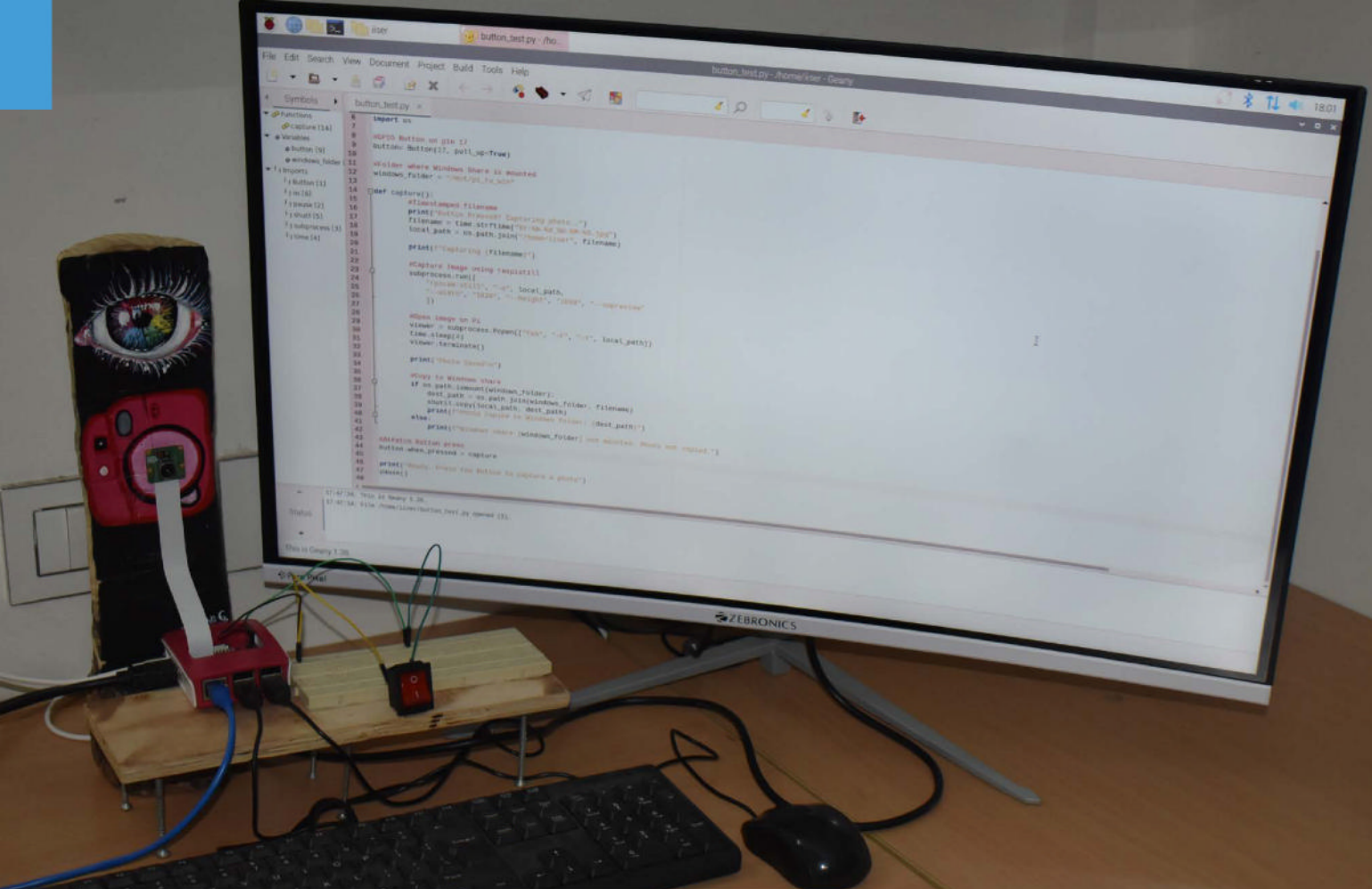
“The captured image is automatically processed to detect the subject's face and eyes using OpenCV Haar cascade classifiers (rpimag.co/haar),” Srikrishna explains. “Once the eye region is identified, the iris is localised using a Daugman-inspired circular gradient search algorithm (rpimag.co/daugman) that estimates the iris boundary.



- 01.** A Raspberry Pi Camera Module 3 is used to take a photo of the subject
- 02.** OpenCV processes and then analyses the photo to determine the iris colour



- ◀ The various processing steps that the system performs to target the iris



▲ The whole thing is written in Python. A monitor hooked up to Raspberry Pi shows live results

“Pixels contained within the detected iris region are extracted, while non-iris areas such as the sclera are removed through brightness and saturation-based filtering. Colour information is subsequently analysed using both RGB and HSV colour histograms. To determine the dominant iris colour, K-means clustering (rpimag.co/kmeans) is performed on the extracted iris pixels in both RGB and HSV feature spaces, and the largest cluster is selected as the representative iris colour. Finally, the dominant colour is classified into categories such as Dark Brown, Brown, Light Brown, Blue, Green, Amber, or Grey using predefined HSV threshold ranges.”

The results showing the eye colour distribution among attendees are displayed in a histogram.

Quick FACTS

- A quote about husbands not knowing their wives' eye colour was included...
- ... and many husbands did not know during the event
- Darker eye colours are more prevalent in the Indian population
- The histogram showed predominantly brown eye colours...
- ... except for one who had a predominant shade of green

▲ Histograms of RGB and HSV values used to determine the colours

Field success

This was Srikrishna's first time using a Raspberry Pi, but he found the abundant online resources more than enough to help get the final project working within the short week-long deadline.

"[The project was] a resounding success; it ran without a glitch and worked as intended for roughly 350 visitors," Srikrishna says. "The ML code was rigorously tested with datasets of human face images. The algorithms that were used are known for their high reliability (such as the Daugman algorithm [that] is used for iris recognition worldwide). The hardware was quite reliable; though I was initially worried, our Raspberry Pi didn't overheat. The push-button connection was fairly simple, so once connected properly, there were no hitches along the way."

Srikrishna notes that clinical applications of computer vision and machine learning have become popular recently, and this project can be modified to benefit that.

"The plan is to upscale the computer vision code for diagnosing cataract and various kinds of fungal infections of the eye," Srikrishna explains. "We have a small team of enthusiastic students working on this since the last three months and would love to be joined by interested folks who're reading this article."

Srikrishna would like to thank Dr Prasanna Katti (Principal Investigator of the Muscle Physiology Lab) and the Science Day Committee at IISER Tirupati for approving the purchase of a Raspberry Pi and accessories for this project. 🍷

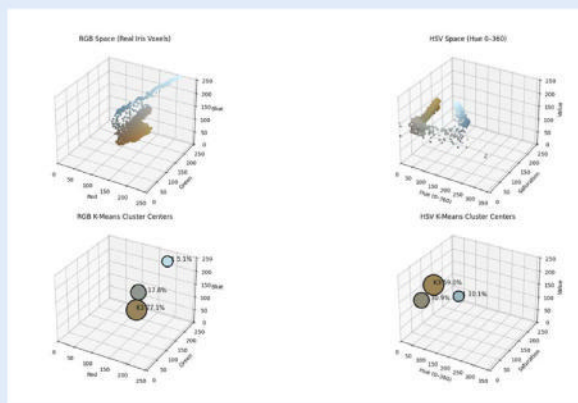
Detecting eye colour



1. The user presses a button to take a photo. OpenCV initially uses an algorithm to perform face and eye detection.



2. A different algorithm is used to localise the iris part of the eye, and RGB and HSV histograms are then used to determine the eye colour, with the largest cluster of a single shade used to determine the predominant iris colour.



3. Secondary and tertiary colours are also determined via OpenCV. These can often be missed by the human eye, but are easier for computer analysis to spot.

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: +44 (0)1778 392472

🌐 Subscribe online: rpimag.co/subscribe

✉ Email: raspberrypi@warnersgroup.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.

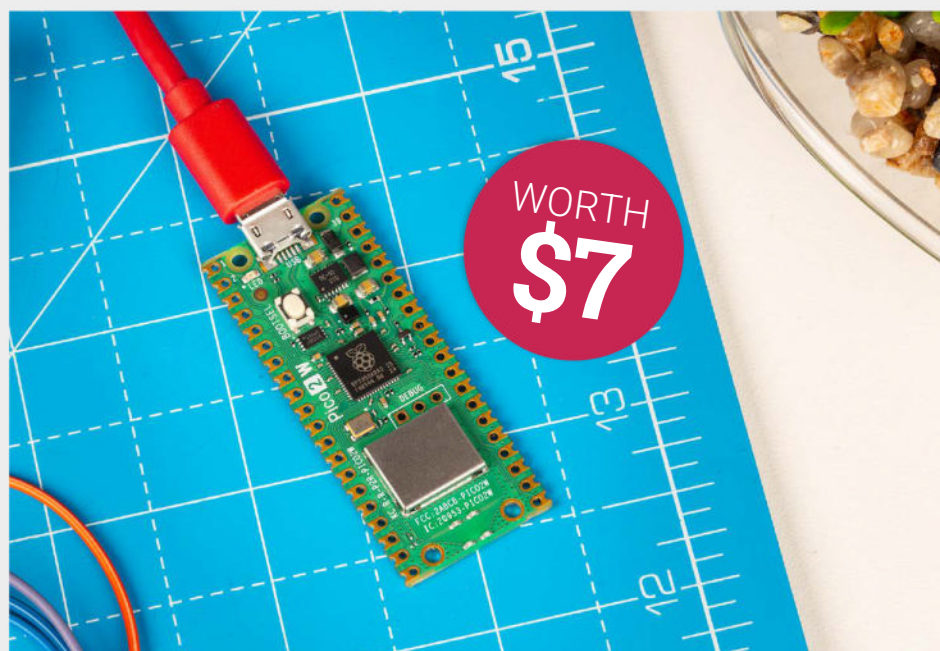
SUBSCRIBE TODAY AND GET A **FREE** Raspberry Pi Pico 2 W (or Pico 2)

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

A brand new RP2350-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



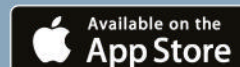
Free Pico 2 or Pico 2 W. Accessories not included. This is a limited offer. Not included with renewals.
Offer subject to change or withdrawal at any time.



 Buy now: rpimag.co/subscribe

SUBSCRIBE
on app stores

From **£2.29**



Claw stand

By Anthony Pyka

rpimag.co/claw-stand

Back in the olden days, this writer briefly studied A-level mathematics. He asked the teacher what the point of calculus was, and got the answer “it’ll be useful if you do a maths degree”. Our hero, eager for knowledge, asked “what else do you get to do as part of a maths degree?”, to which the answer came, “more maths”. Sadly, this exchange persuaded him to drop the seemingly pointless study of mathematics in favour of English literature, which, though equally pointless in the grand scheme of things, was a lot more fun.

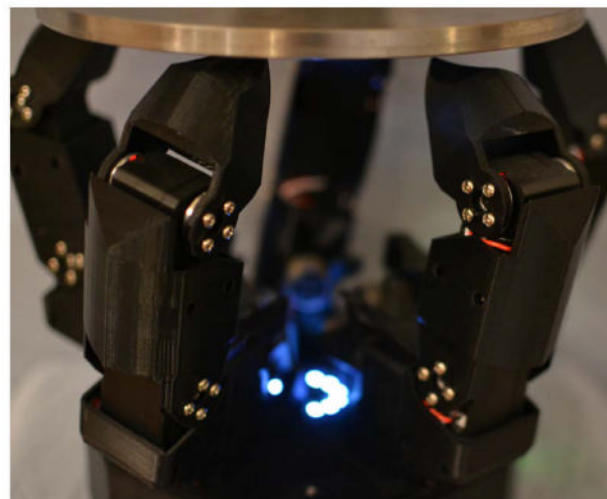
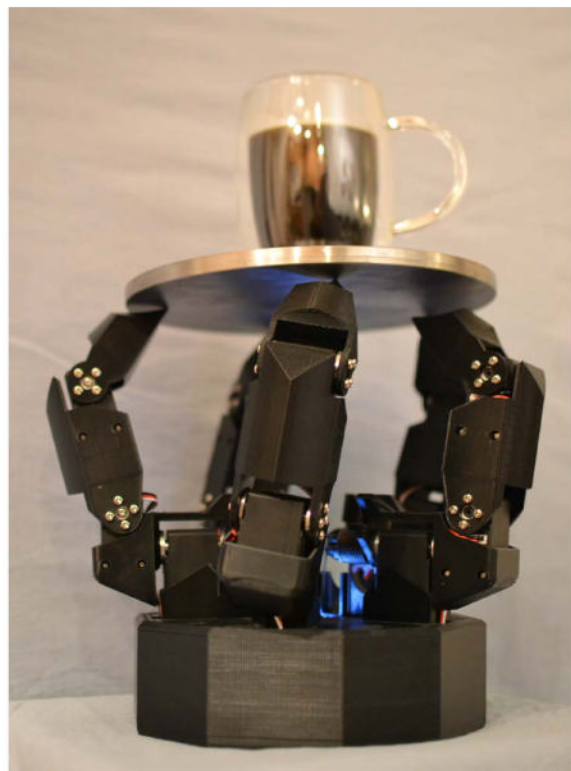
If only this beautiful creation by Anthony Pyka had been around in 1996 to inspire the young student. It’s a motorised claw which holds (and rotates) a glass disc. That sounds pretty simple, but Anthony has gone into a lot of the maths on the project GitHub, and it’s... not simple at all. Spinning a flat object around with your fingers is intuitive to anyone with decent hand-eye coordination, but when you’ve got to explain every movement to a dumb lump of silicon, the instructions do involve a lot of calculus that sadly is beyond us.

The dumb lump of silicon in this case is a Raspberry Pi 5, and it’s joined by a 12V serial bus driver to control the servos (each of the five legs has three servos). A 5V

power supply for Raspberry Pi, a 12V power supply for the motor driver, and a ring of addressable LEDs round off the electronics.

As for the reason why, Anthony told us: “Growing up, my father and I watched *BattleBots* quite often. During the fights where robots were being shredded to pieces, I noticed there was a giant, robotic, claw trophy stand in the background that I thought was more impressive than the bots themselves. I wanted one for myself, but after looking into it some more, I realised the stand is massive, heavy, and expensive. I decided that I could attempt to make one myself, but a smaller, cheaper desktop version to rotate my coffee mug (because why not!).

“So, during many sleepless nights of rocking my daughter back to sleep, while she drifts off, my brain... does not. I started sketching out several proof-of-concept ideas to see what parameters I could manipulate. After enough concept renditions, I modelled my way through simulations to match the parameters to real-world components and geometries. After many hours of developing calculations, simulating models, programming code, and troubleshooting mistakes, I am happy to share I completed my desktop claw stand and I could not be more proud!”





◀ Awoooga!

HAL 9000

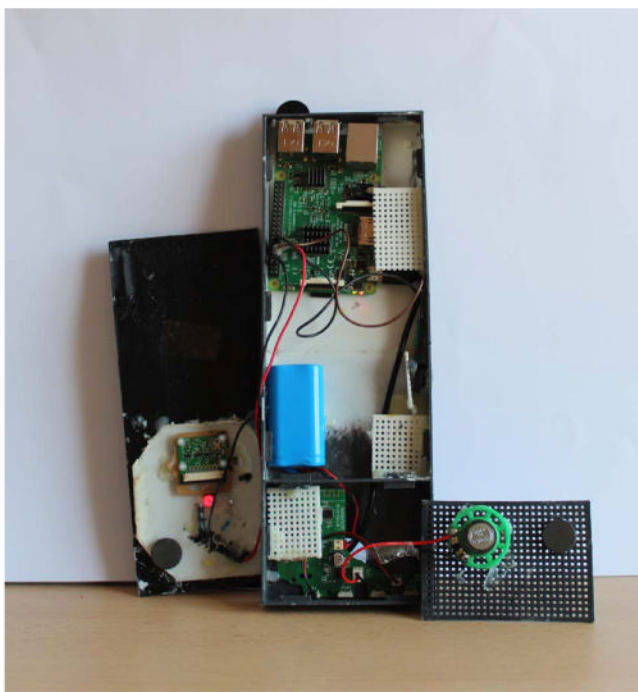
By Jan Aranda Arcas

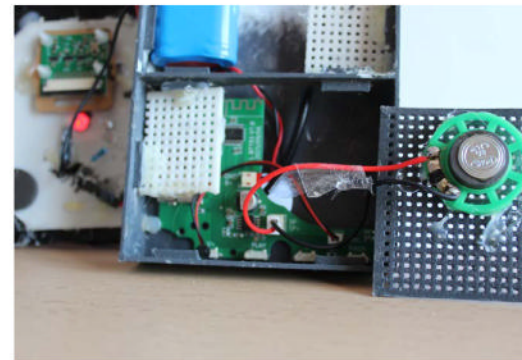
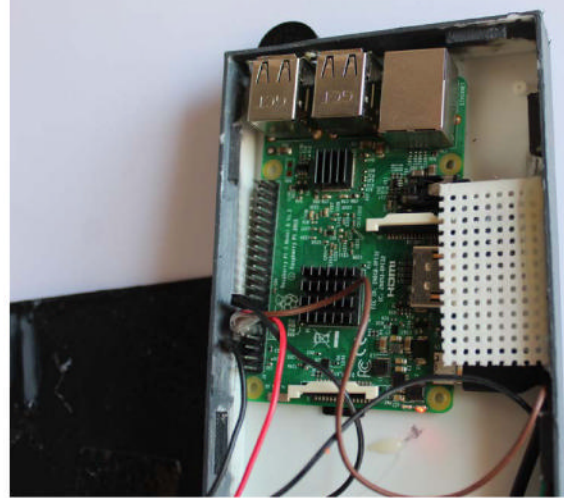
rpimag.co/Home-Made-HAL9000

A **I assistants are a common project.** Raspberry Pi sits in the room with you, listening for commands, translating speech to text, and sending text queries to your AI of choice using that service's API. When the answer comes back from the planet-killing data centres, Raspberry Pi runs it through a text-to-speech engine, and Bob's your uncle: you have a working AI assistant, with all the actual AI outsourced.

What's often confused us is why more people don't take their inspiration from the original AI assistant: HAL 9000, the creepy, disembodied voice of the ship's computer from *2001: A Space Odyssey*. We were therefore delighted to discover this AI assistant by Jan Aranda Arcas, aka Withoutsleep_dev. We asked Jan about his build; here's what he said: "I chose Raspberry Pi because I wanted HAL to be easy for anyone to build while still serving as a useful hardware interface. It handles receiving the AI's responses, playing the audio, and recording the user's voice before sending it to the Python script running on the main computer. The Raspberry Pi offers a great balance of performance, low power consumption, affordability, and excellent community support. Over the years, its community has made it an incredible platform for projects like this.

"I also have a personal reason for choosing it. I received this Raspberry Pi as a gift from a friend of my father's, who encouraged me to experiment and build interesting projects. HAL 9000 ended up being the perfect project for it."





- ◀ The parts are 3D printed and painted by hand, and the 'eye' is glued on; the code is all open source, and uses Python modules including `ear.py`, `translate.py`, `whisper.py`, `voice.py`, and `mouth.py`.

Inky Bird Frame

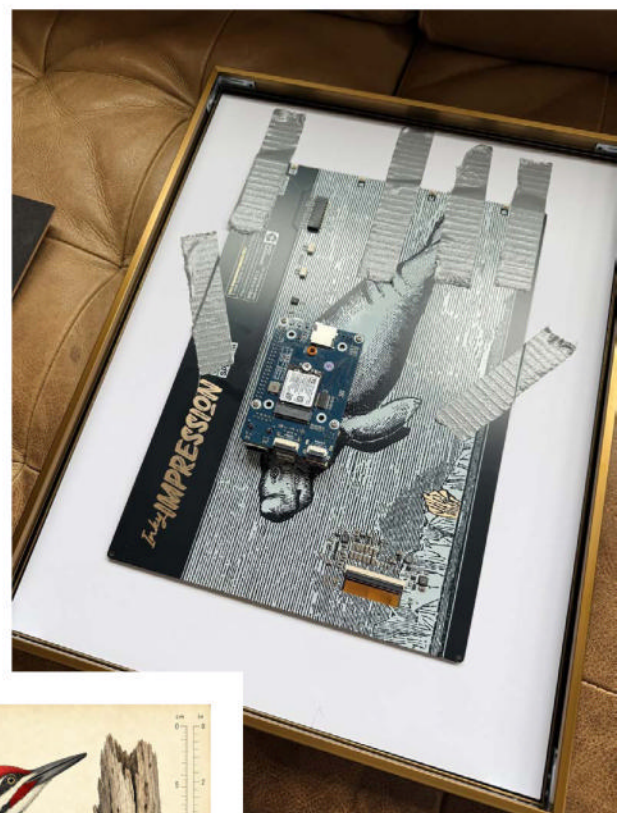
By Henry Sowell

rpimag.co/InkyBirdFrame

There's something extremely relaxing about being out in nature, listening to the sounds, and becoming at one with the animals around you. The dawn chorus in spring is incredible – there's so much life, it's all around you. And it doesn't care about you at all. All you can do is drink it in and forget about everything else. One of our friends is an avid birdwatcher. He's been to the Danube delta looking for storks, Berlin's Tiergarten looking for goshawks, and even wakes up early to go the local landfill site to watch the local seagulls. Yes, we know there's no such thing as a seagull.

Birdwatching doesn't stop when you're in the comfort of your own home; this Inky Bird Frame by Henry Sowell brings the outside in. It comprises a glorious 13.3-inch colour e-paper display, a controller – which in this case is a Raspberry Pi 4 – and a Raspberry Pi Zero 2 W plugged into the back of the screen. The controller checks the iNaturalist.org citizen science website for local wildlife sightings, finds the species plate for the sighting, and sends it to the Raspberry Pi Zero for display.

The great thing about trying a build such as this for yourself is that it doesn't matter how messy the back of the frame is. It doesn't matter if your cuts are ragged or your wiring is tangled, because no-one will ever look there – the important part is the frame, and the display, which at £229.50 from Pimoroni doesn't come cheap, but you can really see where the money goes.



► For a more complicated build, you could get species identifiers from a microphone listening to birdsong



Smart money box

By Vsevolod Bereza

rpimag.co/SmartMoneyBox

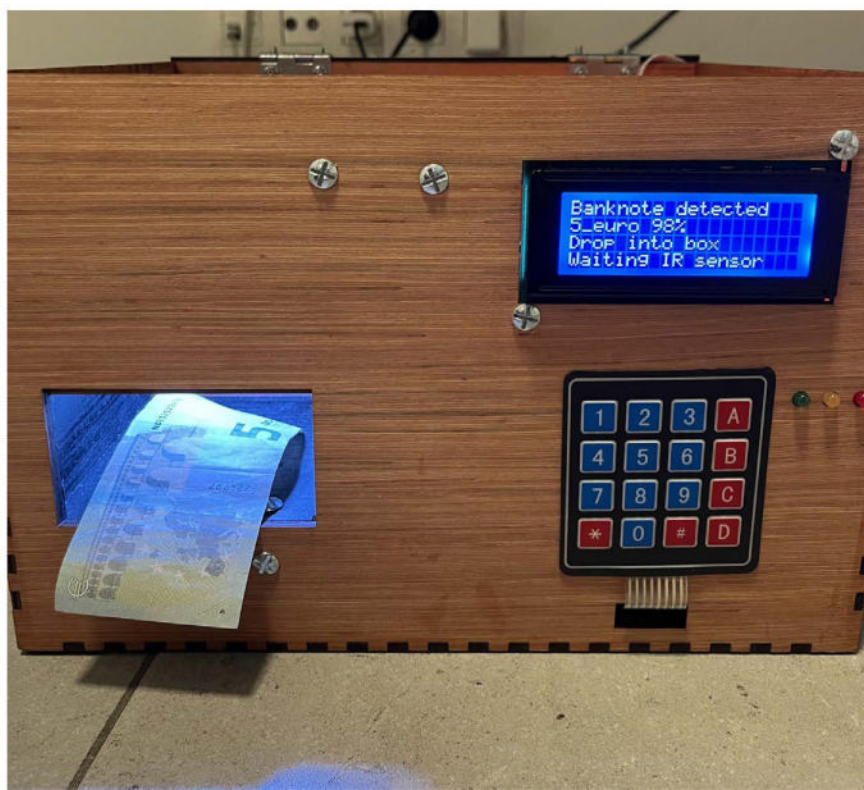
One of the many problems we run into on a regular day is that we have too much money. Piles and piles of the stuff, just lying around, and we've got to somehow count it and store it and add up what we've got. It's a hard life.

We jest, of course. But student Vsevolod Bereza has built a machine to take the drudgery out of counting money, and it has knocked our socks off. It uses Raspberry Pi 5 and a Raspberry Pi Camera Module V2 to take images of banknotes, identify the banknotes, confirm that they've been added to the money box, and keeps a running total of how much the user has in the box.

It also includes a keypad and LCD screen, a locking mechanism controlled by a servo and unlocked with a PIN number, and an ultrasonic sensor to confirm that the money has actually gone in: otherwise you could just take the same banknote and wave it under the camera to trick the machine into thinking that you'd deposited more than you have.

AI image detection depends on good lighting, so the slot that accepts the banknotes is lit by a ring of LEDs diffused by a sheet of white Plexiglas, to ensure even lighting with no shadows or bright spots. The dataset consists of six classes

of banknote, with each class built of 120 separate photos, to give the model something to work on. The maker also gave the model (they used YOLOv26n for the AI) a number of void images, and images with multiple banknotes, so it would learn to only accept one note at a time. And as you've no doubt noticed, it all sits in a laser-cut plywood case. There's so much going on here – electronics, AI, physical making, and more – we're in awe of the creativity and the perseverance it must have taken to get it all working. 🍷





◀ With a little bit nous, AI could even learn to recognise Scottish notes from Clydesdale Bank



3D print

Tell the time with a device that is at once retro and futuristic

rpimag.co/3DPrintedWatch

Aachen, in Germany, is the home of a superb octagonal cathedral, where Charlemagne was anointed and crowned Holy Roman Emperor on Christmas Day in the year 800 CE. It's also where, in 1995, Selective Laser Melting (SLM) was invented, which brings us nicely to this issue's 3D printing showcase. SLM is a form of additive manufacturing in which a computer-controlled laser is directed at a bed of powdered metal to melt precise areas of metal, defined by a CAD file. There's a little more to it than that – the lasering takes place within a sealed chamber filled with inert gas, for example, and there are limits imposed by the type of metal used – but in theory and practice it's very similar to resin printing, in which UV light is directed at a bath of resin to cure areas defined by a CAD file.

Another similarity between SLM and resin printing is the layer height; while standard 3D printers typically deposit melted plastic in layers 0.2mm (200 microns) thick, SLM manufacturing deposits metal in layers 20–100 microns thick, resulting in a much smoother finish. You can see the quality in this 3D printed stainless steel digital watch, by brothers Twan37 and Raceking37. It features a vintage HDSP-2000 LED matrix display, first introduced in the 1970s, as well as a more modern ring of addressable RGB LEDs, and it can be programmed and charged via a custom dock. The programming logic comes via an ATtiny1616 microcontroller. This was the brothers' first foray into SLM manufacturing, and they simply used the same STL file that they used in an earlier FDM prototype – any differences in tolerances could have been accounted for by altering the other, non-metal parts, but even then they didn't have to do much. If you can design for printing in plastic, you can do it in metal too! 📺







CYBERDECK

PROJECTS

Discover the cyberpunk-inspired computers built from Raspberry Pi and parts.

BY ROSIE HATTERSLEY



In a constantly connected world, why stick to a vanilla phone or laptop when unlimited information, entertainment, communication, and gaming can just as easily be yours at the touch of a homemade console with customised buttons and menus designed in whatever imaginative guise you wish?

Forget boring slabs of glass and glue. Cyberdecks are inspired by the computers found in the world of the cyberpunk genre. Clacky keyboards, lines of terminal code, and built to withstand a futuristic dystopia that feels remarkably realistic.

Cyberdecks built as a means of mobile computing and online access are having a moment. Already popular with Redditors, self-builders, and cyberpunk aficionados, they were recently featured on venerable TV news channel CNN after a surge in social media popularity. Many cyberdecks are based around Raspberry Pi; indeed, we've featured many an imaginative build in our magazine pages and online over the years (rpimag.co/cyberdeckceleb).

Here, we salute some excellent examples of cyberdecks showcasing their makers' individuality and creativity, as well as paying homage to the concept's cyberpunk origins. We'll also look at some of the parts you need to get started on your own cyberdeck build. And after this, Brian Corteil will show us how he built his own cyberdeck.

**The computers
found in the
world of the
cyberpunk genre**

Cyberdeck projects

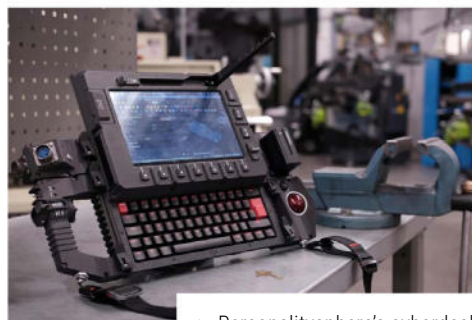
The heavy-duty cases of early laptops and clamshell computers make ideal cyberdeck housings, while others have entirely different origins

A brief history of netrunning

Cyberdecks have their origins in futuristic and speculative fiction from the 1980s and 1990s. William Gibson's 1984 novel *Neuromancer* had an outsize influence on how we think of computers and spawned the concept of cyberpunk. In Gibson's world, cyberspace decks provide a means for console cowboys or hackers to directly connect to a global Matrix via dermatrodes (sensors on their foreheads). As well as cyberpunk aesthetics, these films, novels and games have inspired hundreds of makers to create homemade cyberdecks with which to 'jack into the net', more often than not with Raspberry Pi at their core.

Cyberpunk also spawned Netrunners – cybernetically augmented hackers who uncover systems to hack and who sell the information they uncover to fixers. The name derives from a popular card game of the same name from 1996 (rpimag.co/netrunnerlife). Netrunners' brains connect directly to the Net using more efficient access means than traditional keyboards.

Still, most cyberdecks feature a keyboard as well as a screen, with designs reflecting the early years of portable computers and unusual clamshell handheld devices, and they are generally designed for portability and to withstand potential attack from virtual or real targets.



▲ Personalitysphere's cyberdeck is used for FreeCAD design

Personalitysphere cyberdeck

rpimag.co/personalitysphere

Hacker alias: Personalitysphere

Maker Personalitysphere decided to trade portability for an interesting cyberdeck design. This Raspberry Pi 5 8GB-based cyberdeck features Picatinny rails to hold the camera and trackpad in place using a 'cheap knock-off version' of LEMO 0B connectors used by photographers and videographers to lock in items at particular angles. It runs off a swappable camera battery. Sliding up the screen before beginning to type has been a particularly enjoyable aspect for the maker. Although it doesn't have the best viewing angle, Personalitysphere is happily using his cyberdeck computer for FreeCAD designs. It's sufficiently robust to use in his hot and greasy metal workshop.

Hackberry Pi

rpimag.co/hackberrypi

Hacker alias: Paul Asadoorian

▼ The Hackberry Pi kit is based around Compute Module 5

The portability factor plus usage as a diagnostic and penetration-testing tool provided a clear motivation for maker Paul Asadoorian to create his 'ultimate' cyberdeck based on ZitaoTech's Elecrow Hackberry Pi (rpimag.co/hackberrypicm5), an NVMe SSD, and

Raspberry Pi Compute Module 5 with 16GB RAM. It features a sturdy aluminium chassis, 720 x 720-pixel screen, and a 5000mAh battery for impressive longevity. An assembly of parts rather than a bespoke design, it comes with two USB 3.0 ports to add Bluetooth or a Wi-Fi dongle.





Amstrad PPC 640 cyberdeck

rpimag.co/ppc640

Hacker alias: Mikeydamager

When Mikeydamager needed to display an electronic narrative, he decided to cannibalise an old Amstrad PPC 640 for the purpose. The venerable IBM-compatible PC launched in 1987, priced at more than \$1000, and examples sell on eBay for £250 or more. However, Mikey wasn't in need of a working vintage model: he just wanted an eye-catching device so people would come and read his interactive literature – something that the “moody” LEDs also help with. Raspberry Pi 3 with a new driver board provided the brains, while an old tablet did service for the screen and was a perfect fit in the original Amstrad housing. Having wired up a few switches and dials for the batteries, he disconnected one of the floppy disk drives and wired its copy protection switch to an Arduino to activate a screen when a disk is inserted. This launches the React app that displays his “immersive, interactive narrative”. He admits “there's a little stage trickery going on.” Considering cyberdecks originated as sci-fi props, we think it's more than fair.

◀ This stunning project upcycles an old IBM-compatible PC

There's a little stage trickery going on

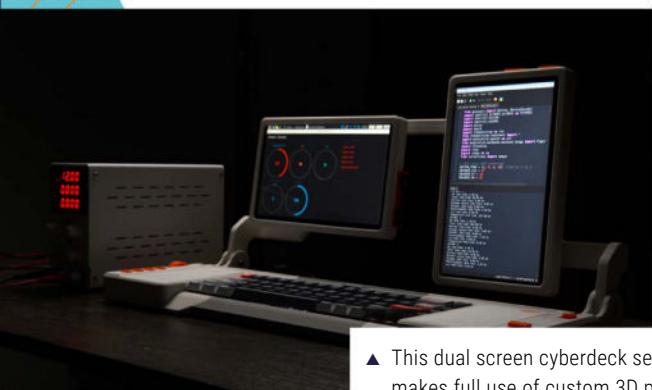
Smith Corona cyberdeck

rpimag.co/smithcorona

Hacker alias: Maz_Baz

This Raspberry Pi 4B-based makeover was inspired by “old-school [1980s] computer terminals” as well as *Star Wars* and cyberdecks. The Mark XL PWP's ‘deteriorating’ 1989 keyboard needed replacing, the disk drive was broken, and the screen was faulty, but Maz_Baz was drawn to the luggable Smith Corona typewriter's ‘old tech, new spec’ potential. An Anker power supply fitted neatly into the original power pack bay and the case was roomy enough for a Raspberry Pi 4B, 40%-size Vortex keyboard, and a generously proportioned 7.9in Waveshare screen. The most challenging aspects were the “frustrating and necessarily inventive” cable wrangling and accurately measuring and 3D-printing the new keyboard chassis and monitor support brackets.

▼ This Smith Corona typewriter makes for an innovative cyberdeck project



▲ This dual screen cyberdeck setup makes full use of custom 3D printing

Dual-screen cyberdeck

rpimag.co/dualcyberdeck

Hacker alias: Sector 07

Maker Austin (aka Sector 07)'s attention to detail when designing and talking about his incredible dual-screen cyberdeck is just wonderful. The painstakingly designed, fully 3D-printed (apart from the Raspberry Pi 4 inside) cyberdeck clips together by hand – no tools needed to keep it assembled – but can also be disassembled with ease. The Raspberry Pi 4 sits neatly in the right-hand portion of the bespoke keyboard, there is a linear volume control slider, programmable rotary controller and other enticing knobs and buttons. Better yet, the GPIO pins are exposed so you can add your own HAT.



Old tech, new spec potential

Clamshell cyberdeck

rpimag.co/clamshellblackberry

Hacker alias: Michael Klements

Rumour has it, the clamshell BlackBerry is back in fashion. This version runs Raspberry Pi OS on Raspberry Pi 4, sports a BlackBerry-style Solder Party keyboard, and is paired with a HyperPixel 4.0 screen, powered from a GPIO pin. Maker Michael found Raspberry Pi "the perfect fit" for his build, as the keyboard footprint is smaller than the display's, while the Solder Party custom carrier board treats the trackpad and keyboard like USB devices, which still allows you to get full mouse and keyboard functionality." Completing the setup, Michael provides step-by-step instructions for creating your own plywood case: rpimag.co/maketinycyberdeck.

- ▼ The unusual plywood case minimises the BlackBerry-esque clamshell's heft



- ▲ Mermaid In The Shell is wearable tech with a 'high femme' finish

Mermaid In The Shell cyberdeck

rpimag.co/mermaidcyberdeck

Hacker alias: Claudia Bimbotech

This very pretty clamshell design offers a really different take on a cyberdeck, using the natural frame of a clamshell

clutch bag to house the maker's fantasy 'deck. Claudia was keen that her cyberdeck be the opposite to a Pelican case "boy build", choosing a pink clutch case and "high femme" finish. Her Mermaid cyberdeck features Raspberry Pi Model A+, a 3.5in touchscreen, white Bluetooth-enabled ZitaTech BlackBerry-style Q10 Bluetooth keyboard and touchpad, connected via neodymium magnets, and is powered by a 3000mAh battery.

Claudia was keen that her cyberdeck be the opposite to a Pelican case boy build

Backpack Cyberdeck

rpimag.co/backpackcyberdeck

Hacker alias: Davide Marchetti

This practical backpack houses an entire portable lab that its keen cyclist maker can easily take from place to place, interacting with all manner of devices as he goes. Davide uses his as a mobile trouble-shooting lab, with a GNU radio to run wireless communication diagnostics and Kali Linux tools for security and penetration testing. The Backpack Cyberdeck runs a "slightly overclocked" Raspberry Pi 4 housed in a protective but lightweight plywood and 3D-printed frame that keeps everything securely in place. The setup can be used remotely and meets flight-safe guidelines for power output, so it can be taken aboard aircraft too.

- ◀ This bag build is a portable diagnostic lab in a rucksack



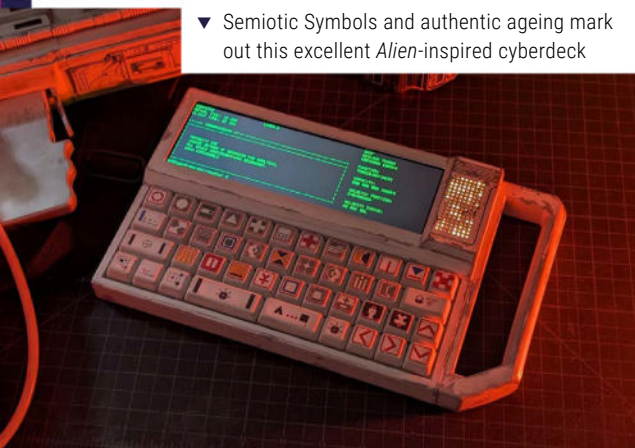
Alien Typeframe PS-85 cyberdeck

rpimag.co/typeframeps85

Hacker alias: Jeff Merrick

Starting with a set of Semiotic Symbols keycaps based on Ron Cobb's futuristic symbols in *Alien*, cosplayer Jeff set about creating a cyberdeck inspired by the typewriter-like 1985 Epson PX-4 computer. For the Raspberry Pi 4-based design Jeff kept things simple and "went with a slate form factor because hinges complicate everything" and a 7.9in Waveshare display. Left with an exposed GPIO header, he attached an Adafruit LED Matrix Bonnet, representing the AI Mother System MU/TH/UR from *Alien*. Reflecting his cosplay prop experience, Jeff added a distinctive handle, then customised the cyberdeck exterior with a paint job including toothpaste to rough up the areas he wanted to appear worn.

▼ Semiotic Symbols and authentic ageing mark out this excellent *Alien*-inspired cyberdeck



The
trademark
battle station
of console
cowboy
anti-heroes
running nets
through
cyberspace

PiDex Doomsday Companion

rpimag.co/pidex

Hacker alias: Raspduino Uno

'Raspduino Uno' Kenneth is somewhat of a prepper, with a classic ruggedised, waterproof box into which he's fitted Raspberry Pi 5 and all manner of tech to survive whatever the future holds. A 7in touchscreen, mini backlit keyboard (for viewing when the lights go out), four-port USB hub, power bank, an accompanying solar panel, copy of Wikipedia on a separate USB flash drive, and several important-looking big red buttons mark this out as a kit worthy of any survivalist.



▼ Preppers gonna prep with this solar-powered cyberdeck designed for off-grid use



Make your own 'deck

Hardware and kit for making your own portable rig

COMPUTER: RASPBERRY PI

You'll need a computer for your cyberdeck build. A Raspberry Pi 5 is ideal, but a Raspberry Pi 4 is a lower-cost alternative that will give you all the power you need.

rpimag.co/raspberrypi5



KEYBOARD: KEEBDECK BASIC

Picking a keyboard is very much a personal choice, but we like the look of this KeebDeck. It's a fully assembled handheld keyboard that's easy to incorporate into your cyberdeck builds.

rpimag.co/keebdeck



SMARTIPI TOUCH PRO 3:

If you like the idea of using your cyberdeck as a central control hub, then this robust 7in touchscreen enclosure is a great option: it has room inside to accommodate not just your Raspberry Pi display, but a Camera Module and any HATs you've attached to Raspberry Pi itself. Add a few lights for a suitably sci-fi effect.

rpimag.co/smartipipro3



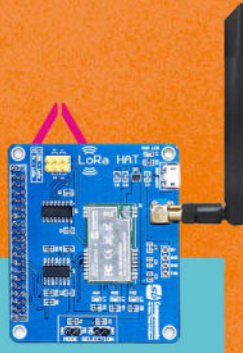
SCREEN: HYPERPIXEL 4.0

Give your cyberdeck a screen. There are lots of options to choose from, but here is Pimoroni's HyperPixel 4.0. This is a high-resolution, 4-inch TFT display for Raspberry Pi with optional capacitive multitouch. It connects via the GPIO header and is ideal for attaching a display to a Raspberry Pi.

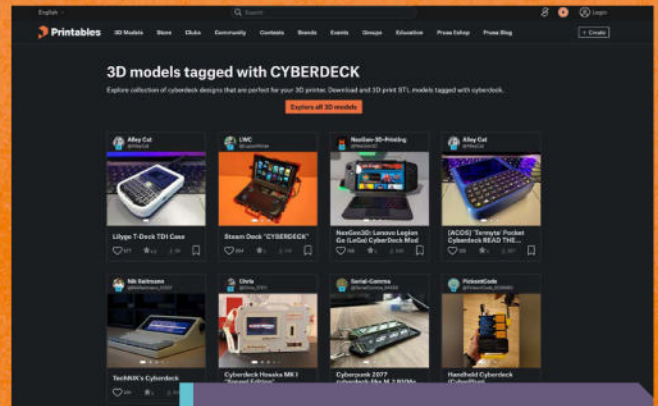
rpimag.co/hyperpixel4



rpimag.co/lipoamigo

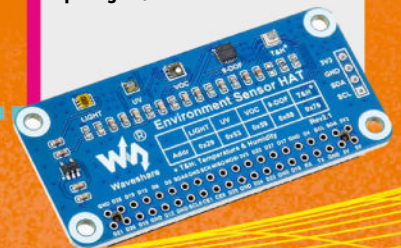


rpimag.co/lorahat

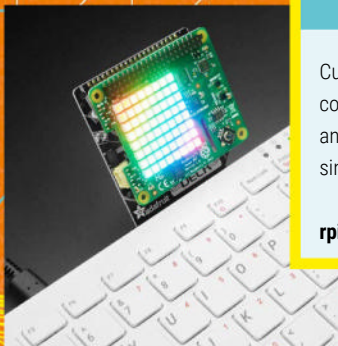


rpimag.co/cyberdeckcases

rpimag.co/envsenhat



rpimag.co/cyberdeckhat



Create your own cyberdeck

A hacker's rite of passage is creating their own cyberdeck. This guide will help you build one from parts



Maker

Brian Corteil

Creator of two digital zoetropes, a Giant Unicorn HAT (made from 64 Unicorn HATs), too many robots to count, plus a cyberdeck or two.

corteil.co.uk

What is a cyberdeck? Is it a wearable computer, pocket computer, launch monitor/centre for rockets or high-altitude balloons, or a knowledge store for the zombie apocalypse? The answer is all of them and anything else you can think of. It is something you create to solve a need that a conventional tablet or computer cannot fulfil, or just for the hell of it to make something personal.

Planning/design thoughts

There are four main parts to a cyberdeck: brains, peripherals, power supply, and case. We are going to look at each in turn.

- **Brains** – The brains of a cyberdeck include its processing unit and long-term storage. Things to think about the central processing unit (CPU) are power consumption, processing power, and size. A single-board computer (SBC) is ideal for this as they use less power than a conventional PC CPU. The Raspberry Pi line-up has some good options for a cyberdeck, and is a popular choice for builds. For storage, the Raspberry Pi has a few options: SD card, USB flash drive, USB hard drive, USB SSD, or M.2 NVMe drive with an M.2 HAT if you are using a Raspberry Pi 5.



- ▲ This unassuming Pelican-style case can be used for the cyberdeck of your dreams
- ▼ The complete cyberdeck is ready for your off-grid life. Hack the planet!
- ▼ The power system, RTC battery, and a plate to mount a Raspberry Pi are added to the bottom first

GET THE PARTS:



rpimag.co/cyberdeck

The Raspberry Pi line-up has good options for a cyberdeck

- **Peripherals** – Peripherals are how you interact with the cyberdeck. They can be standard items like a keyboard, mouse, display, sound, camera, etc. Or be voice-controlled, a pointing device built into the case, a joystick pointer for the mouse, encoder for a scroll wheel, and trigger buttons for mouse buttons. A small USB hub is useful if you are connecting many USB devices. Depending on how you are exposing the SBC's ports, you may need panel-mount port extensions.
- **Power** – The power supply must be large enough to power the SBC, peripherals, and accessories. To calculate the size of the power supply needed, add together the power requirements of each device. You can find this information from the data sheets or the device's manual.

NOTE!

Although Raspberry Pi 5 has a requirement of 25W, it does not draw all of this power unless each USB port is drawing maximum power. If you plan to run it off mains, the official Raspberry Pi 27W USB-C Power Supply will suit this build (rpimag.co/powersupply).

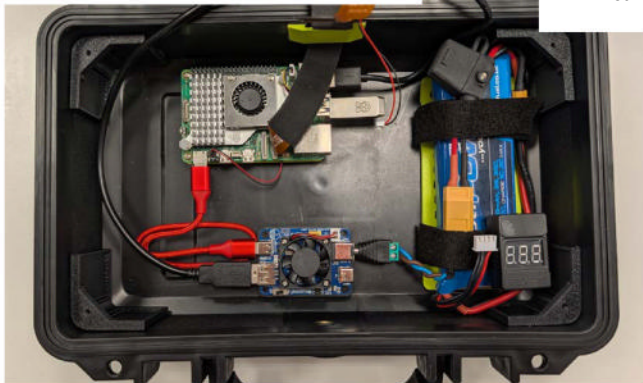


Power components

- Battery: rpimag.co/cdbattery
- XT60 extension: rpimag.co/xt60
- Battery monitor: rpimag.co/battmon
- XT60 adapter: rpimag.co/xt60adapt
- Power module: rpimag.co/powermod

Reduce the fuse on the XT60 extension to 5A or less, and make sure to charge the battery in a charge-safe bag.

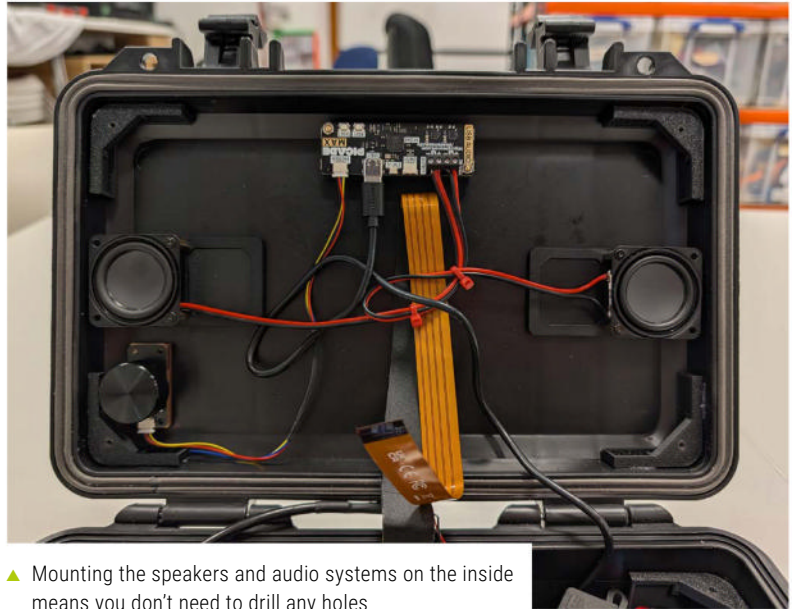
- ▼ Layer up the cyberdeck with Raspberry Pi and the battery too



For example, a Raspberry Pi 5 requires 25W; that's 5V at 5A if you want to use the USB ports at full power (this is the worst-case scenario, with the USB ports fully loaded). The display at full brightness pulls approximately 3W. Therefore, you need a power supply of at least 28W. This translates to approximately 5.6A.

This was calculated with the formula $I = P/V$, where I = current (A), P = power (W), and V = voltage (V), therefore $5.6A = 28W/5V$. Options for the power supplies are a power or UPS HAT, DC-to-DC converter board powered from batteries, an auxiliary power supply, or power bank. A cyberdeck does not need to be powered by batteries.

Some types of batteries, especially LiPos and related types, can make your life interesting for all the wrong reasons: treat with care, make sure they are charged correctly, and are not over-discharged. It's a good idea to fit a fuse as close to the battery as possible and monitor the battery's cell voltages when in use.



- ▲ Mounting the speakers and audio systems on the inside means you don't need to drill any holes

- **Case** – The case defines the cyberdeck. Past builds have used old pocket computers (including using the original keyboard), a briefcase, mint tin, wooden boxes, and prop replicas. A good starting case is a Pelican-style hard case. These are widely available in a variety of sizes and price points. They make a robust case, normally have a carry handle, and are weatherproof.
- **Construction** – A case can be built using fast prototyping methods, 3D printing, laser cutting, plastic foam board or plastic sheet, and even cardboard. The case needs to fit in all your parts and don't forget about the cabling. The construction of your cyberdeck depends on what tools and services you can access. 3D printing and laser cutting are well suited for building your cyberdeck, allowing you to custom-design the case. Mini 10mm extruded aluminium profile is a great strong support for the chassis and also, when used with 3D printed/laser-cut panels, makes a strong versatile case. 3D printer pens, mouldable plastic, plastic sheets, and plywood can be used to form panels, supports, and any other elements. Companies offer 3D printing and laser-cutting services if you can't access the equipment yourself.
- **Extras** – An important part of a cyberdeck is the extras. They finish off and define the build. Switches, LEDs, software-defined radio (SDR), I/O pins accessible for hacking, an I2C port for adding sensors and second displays; a Raspberry Pi Pico to act as a second processor for monitoring switches, sensors, driving a secondary display, controlling LEDs, and anything else you can imagine.

Collecting your parts

For this guide, we have selected the following parts, all of which are available off-the-shelf. Some soldering will be required or you could use Wago blocks instead.

- **Raspberry Pi 5:** this version of Raspberry Pi is perfect for a cyberdeck as it's powerful enough for everyday tasks, controlling other Raspberry Pi projects, and programming. It also has good connectivity options and the GPIO pins for projects. rpimag.co/raspberrypi5
- **Raspberry Pi 128GB flash drive:** These flash drives are faster than a microSD card, more robust, have a larger physical size so are less likely to be lost, and are easy to swap out – especially if you want to use a different operating system such as a media centre or retro gaming OS. rpimag.co/flash
- **Raspberry Pi 7" Touch Display 2:** This display has a resolution of 1280 × 720 (landscape). The touchscreen removes the need for a mouse in most cases. rpimag.co/touchdisplay2
- **Rii RK302 Bluetooth Keyboard:** Selected because it fits in the case used in this tutorial and has a low profile. It is designed to sit on a table when in use and its chassis is flexible and will need to be supported when in the case. It does have the advantage of connecting via Bluetooth. rpimag.co/rk302
- **Pimoroni Picade Max USB Audio:** This is a compact USB-C sound card with a 3W stereo amplifier, a one-cable/PCB solution, with the bonus of keeping the 40-pin GPIO header free. The Picade Max Encoder (rpimag.co/picadeaudioenc) has also been included for volume control duties. rpimag.co/picadeaudio
- **Power:** Using a power bank simplifies the arrangements and is a safer option for powering a cyberdeck than using a LiPo battery. However, this build will be using a LiPo, with an inline fuse with a DC-to-DC converter that outputs 5V 5A via USB-C PD. As noted earlier, a Raspberry Pi 27W USB-C Power Supply (rpimag.co/powersupply) can be used to run the cyberdeck from mains. To keep things simple, this build will be using an external charger for charging the battery, and the battery will be disconnected while charging.
- **Pelican Style Case:** This is a CASOMAN 11.5-inch water-proof case. The Pelican brand is known for its well-made cases and has become the common name for rugged, water-proof boxes, like Hoover is for vacuum cleaners. rpimag.co/casoman
- **Raspberry Pi RTC Battery:** Adding the battery for Raspberry Pi 5 will activate the built-in RTC and will keep time when your cyberdeck is off-grid. You need to add this line to `/boot/firmware/config.txt` for the battery to charge:

- ▼ A plate covers the electronics inside so you can easily fit a keyboard on top



Using a power bank simplifies the arrangements and is a safer option

```
dtoverlay=rtc_bbat_vchg=3000000
```

The battery can be found here: rpimag.co/rtcbattery.

- **USB leads:** For internal routing, we like the flat ribbon-style custom cables available from Adafruit (adafruit.com) and The Pi Hut (thepihut.com). You can mix up the ends; for example, a right-angle micro HDMI to a full-sized HDMI plug. For this build, we're using standard USB cables and a 500mm display ribbon to connect the screen.

QUICK TIP

Cloth tape used for car electrical harness wiring is great for covering the display cable to match the case.



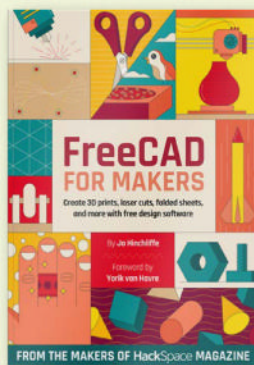
- ▲ The keyboard can be anything you want, as long as it fits the space

Building your cyberdeck

0. Collect together the parts for the build, burn your selected OS to your storage (you can use Raspberry Pi Imager to do this: rpimag.co/imager), then connect the parts together and test that everything works.
1. CAD (Cardboard-Aided Design in this case) helps with laying out your cyberdeck, prototyping the layout, how things go together, and what space is required for cabling.
2. Next, we convert the lessons learned from step 1 into (traditional) CAD models. We used the Maker version of Solidworks (rpimag.co/solidworksmaker), but other popular software includes Fusion360 and the open-source FreeCAD.
3. Print, laser-cut, build your supports as designed, ready for installation. You will find that you have to make adjustments to your designs as you build your cyberdeck.
4. Start with the lid and check that the parts fit. Begin with the speakers and the Picade Max USB Audio; the amp is installed behind the display at the top and the speakers either side of the display behind the grids on the front plate. The volume control sits behind the top plate, central to the left speaker. The volume control knob protrudes through the display plate. Next, fit the display into the top plate using double-sided tape/foam to hold the display in place and fit the back brackets. Glue the supports into place with a hot-glue gun or double-sided foam tape after checking the fit for a second time. Connect the speaker wires to the speaker connections, then connect the 500mm display ribbon and power cables you will need to extend the display's power cable. The display ribbon is covered in a cloth tape used for electrical harness; this supports the ribbon and blends it into the build, matching the USB cables. A USB-A plug cable with bare wires is soldered to the display's power cable; use heat-shrink to cover the joints. Using a USB cable frees the power pins on the 40-pin GPIO header for HATs and projects. Self-tapping screws are good for securing the display panel to the supports; other options are supergluing nuts or using heat-set bosses.

FreeCAD for Makers

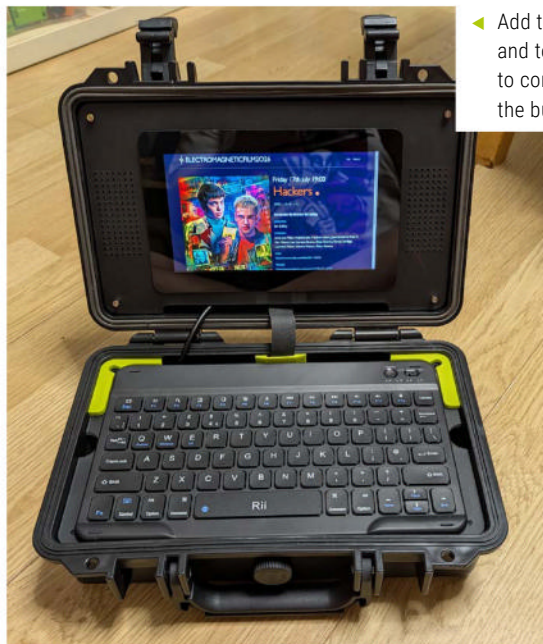
Want to learn about FreeCAD? We have a book all about it here: rpimag.co/freecadbook



QUICK TIP

When using self-tappers in plastic, be careful to not strip or cross-thread the threads. When refitting, screw anticlockwise until you feel a jump: this is the start of the thread. Tighten the screw as normal.

5. Check that your supports and plate fit in the bottom half of the case. Glue the supports into place. Connect Raspberry Pi to the display and make sure it boots and the display is working. You could put a latching switch to turn off power to the display (it needs to be in series with the display's power inputs and the power source powering the display). When you're happy with the position of Raspberry Pi, and sure that both the display and speakers are working, fix your Raspberry Pi into place. It can be glued into place or you can use Velcro tape to hold your parts in place. We designed a mount plate for the Raspberry Pi and the battery holder. Fit the keyboard grips to the keyboard plate. Don't fix down the plate because you will need to be able to access the Raspberry Pi to power it up/down and change the power source.
6. From here you can customise. There are a few ways to customise your build. You could use a Raspberry Pi Pico as a second processor to control a second small display with useful information, or drive some LEDs – because you can't have too many LEDs in a build. You could add sensors to monitor power consumption and keep an eye on your battery level and anything else you can think of. ■



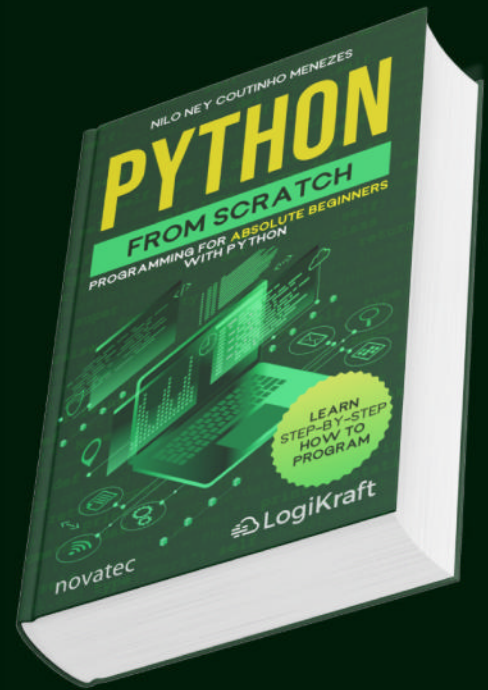
- ◀ Add the screen and top plate to complete the build

YOUR **RASPBERRY PI** RUNS PYTHON. **DO YOU?**

A BOOK FOR **ABSOLUTE BEGINNERS** THAT TEACHES **HOW TO PROGRAM** WITH PYTHON.

No prior coding experience needed

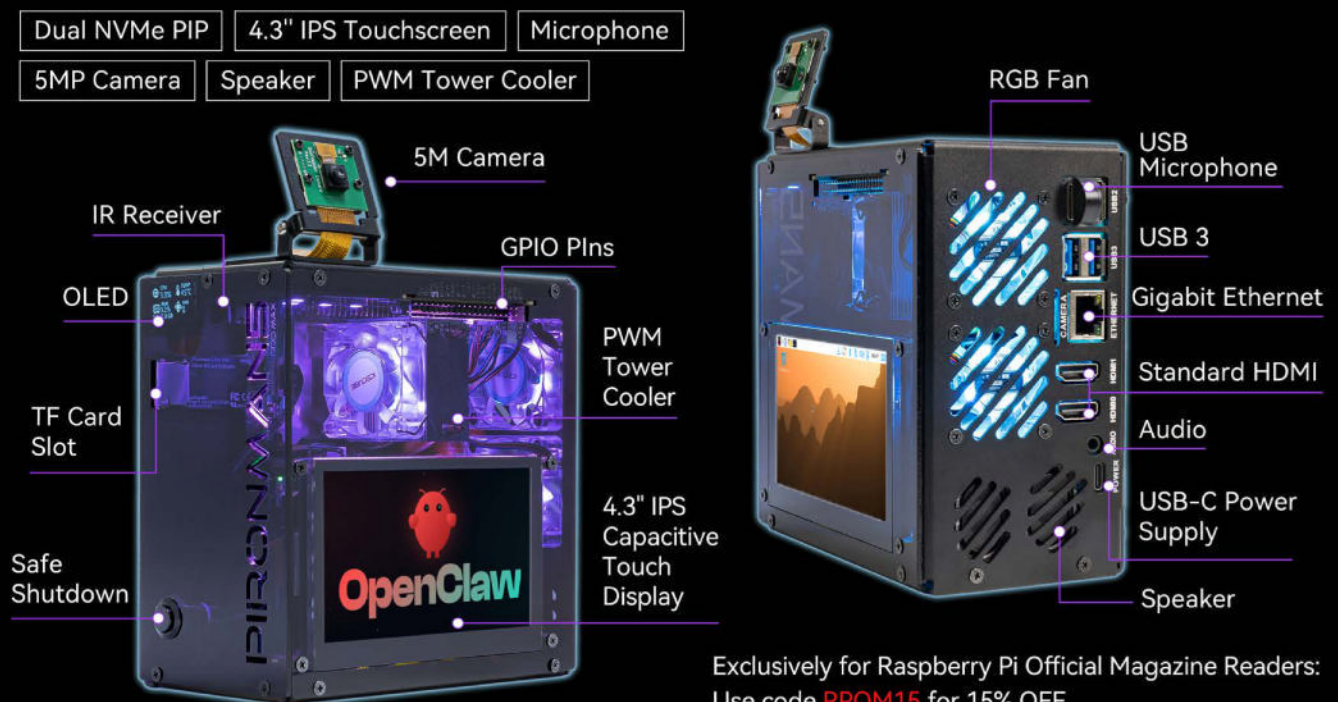
Visit the website for the exercises book, solved exercises, and a free sample chapter download.



PYTHONFROMSCRATCH.COM

Available **Worldwide** - find it in your local Amazon store (US, UK, CA, AU, JP, DE, NL, FR, IN, ES, MX)

Pironman 5 Pro Max: Your All-in-One Raspberry Pi 5 Desktop for AI OpenClaw, Entertainment, Development & NAS



Exclusively for Raspberry Pi Official Magazine Readers:
Use code **RPOM15** for 15% OFF.

Block ads at home with Pi-hole

Pi-hole blocks internet ads at your router, removing bandwidth-sucking and privacy-invading ads from your whole network



Maker

Lucy Hattersley

The editor of *Raspberry Pi Official Magazine*, Lucy is a lifelong tinkerer with a soft spot for privacy-conscious, self-hosted technology.

rpimag.co

Pi-hole blocks ads by acting as a DNS sinkhole

YOU'LL NEED

- A Raspberry Pi computer
- A suitable power supply
rpimag.co/power
- A microSD card
rpimag.co/sdcard
- An adapter to connect your microSD card to your usual computer

Internet advertising can often be a nuisance. It takes up valuable bandwidth. It weighs your browser down with trackers. It reports your data back to untrustworthy third parties.

On some devices, like laptops and phones, you can use ad blockers to protect yourself (and your sanity). But on many smart devices, like TVs, refrigerators, toasters (rpimag.co/revcook-toaster), and toilets (rpimag.co/numi-toilet), you can't install an ad blocker.

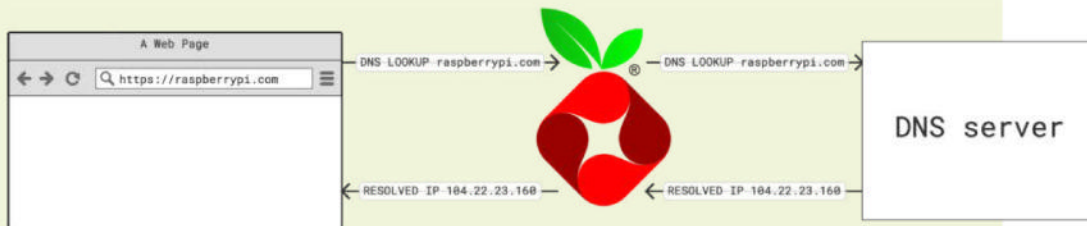
Pi-hole simplifies ad blocking by operating at the *network* level, instead of on individual devices. That way, you aren't restricted by which devices can install an ad blocker; instead, you can protect devices on your network from ever accessing ads in the first place.

How does Pi-hole work?

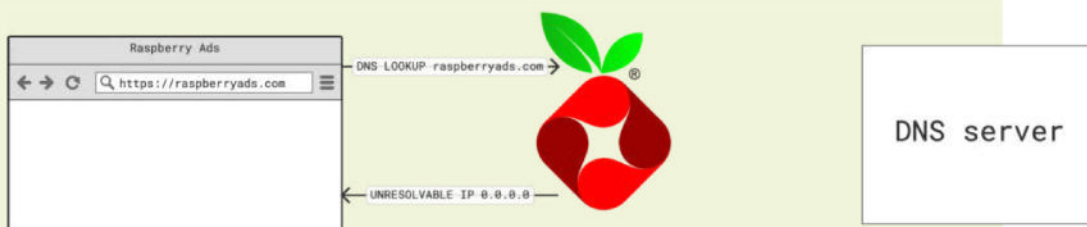
Pi-hole blocks ads by acting as a DNS sinkhole (rpimag.co/dns-sinkhole).

To understand what a DNS sinkhole is, you need to understand what DNS is. Here are a few key terms:

- **Domain Name:** unique alphanumeric address of a web resource (e.g. raspberrypi.com)
- **IP address:** unique numeric address of a web resource (e.g. 104.22.23.160)



▲ **Figure 1:** A diagram demonstrating a successful DNS lookup via a Pi-hole



▲ **Figure 2:** A diagram demonstrating a blocked DNS lookup via a Pi-hole



Warning!

Global settings

This tutorial requires you to change global settings in your wireless network. You might break your internet connection (for a little while). Proceed with caution! Check how to reset your router before starting the tutorial.

- **DNS:** Domain Name System, associates domain names with IP addresses
- **DNS server:** a server that translates domain names into IP addresses

When you type `www.raspberrypi.com` into your browser, your computer asks a DNS server where to find that domain. The DNS server responds with an IP address (e.g. `104.22.23.160`). Then your computer queries that IP address for the resource you're looking for.

We use domain names because IP addresses aren't very human-readable; it's a lot easier to read and write 'raspberrypi.com' than four bytes' worth of numbers.

Usually, your computer queries a DNS server hosted somewhere on the internet. It might be hosted by your internet provider, a website hosting company, or any company that runs a lot of servers. Your machine submits a domain, and the DNS server returns the IP address corresponding to that domain. The DNS server doesn't care if the domain provides something you want (like the article you're trying to read) or an ad. It just resolves domains into IP addresses.

Here's where Pi-hole comes in. Pi-hole stands between your network and a DNS server. Consider a client device, like your smart toilet, performing a DNS lookup for a domain. The Pi-hole in your network acts like a DNS server; DNS lookups from all client devices, whether that's your smart toilet or your phone,

go to the Pi-hole. But Pi-hole doesn't store a perfect up-to-date mapping of all domain names to all IP addresses. Instead, Pi-hole queries a *real* DNS server outside of your network. However, before it queries that real DNS server, Pi-hole checks a blocklist. If the domain passes this filter, Pi-hole requests the IP address from the DNS server, and returns it to the client device on your network. If the domain doesn't pass the filter – if it's included on the blocklist – Pi-hole returns a non-routable address such as `0.0.0.0`.

Figure 1 shows a lookup via a Pi-hole for the *unblocked* domain `raspberrypi.com`.

Figure 2 shows a lookup via a Pi-hole for the *blocked* domain `raspberrypiads.com`.

TL;DR: Pi-hole blocks requests made to ad domains from your network, before the requests ever leave your network. Your client devices *can't* connect to domains that host ads, but *can* connect to domains that host useful content.

Overview

To install a Pi-hole in your network, you'll need to do the following things:

1. Configure a Raspberry Pi running Raspberry Pi OS.
2. Set up Pi-hole (rpimag.co/pi-hole-github) software on your Raspberry Pi.
3. Direct DNS queries on your network to your Raspberry Pi.

For the initial microSD card setup, you will need another computer connected to your network. We'll refer to this as '*your usual computer*' to distinguish it from the Raspberry Pi computer you are setting up.

You can use any Raspberry Pi model for this. We recommend Raspberry Pi Zero 2 W (rpimag.co/zero2w) if you can't decide which model to use.

Configure your Raspberry Pi

To begin, follow the Getting Started documentation to set up your Raspberry Pi (rpimag.co/getting-started). For your operating system, choose **Raspberry Pi OS Lite (64-bit)** to run headless (without a mouse and keyboard). This can be found in 'Raspberry Pi OS (other)' in the 'Choose an operating system' section of Raspberry Pi Imager.

During the OS customisation stage, edit settings as follows:

- Enter a **hostname** of your choice (we suggest **pi-hole** for this tutorial)
- Enter a **username** and **password**; you'll need these later to authenticate
- Enter your network **SSID** (name) and **password**; you can find these in your Wi-Fi settings or on a sticker on your router
- Check the box next to **Enable SSH** so we can connect to the Pi without a mouse and keyboard

QUICK TIP

If the terminal responds 'ssh: Could not resolve hostname pi-hole.local: Name or service not known', try **ssh <username>@pi-hole** instead.

Boot your Raspberry Pi

Once you've written the image to your microSD card, insert the card into your Raspberry Pi and power it on. Wait a few minutes for your Raspberry Pi to boot up and connect to your network.

Connect via SSH

Open a terminal on your computer. If you use a Windows computer, you might need to install an SSH client; we suggest PuTTY (rpimag.co/putty). Enter the following command to connect to your Raspberry Pi,

replacing the **<username>** placeholder with your own username that you chose in Imager:

```
$ ssh <username>@pi-hole.local
```

If **ssh** asks you if you're sure you want to continue connecting, reply **yes**. Enter the password you chose during advanced configuration when prompted.

QUICK TIP

On most terminals, press the **UP** arrow key, then the **ENTER** key to re-run the most recent command.

You'll know you've connected successfully when you see the following prompt with your configured username and hostname:

```
<username>@<hostname>:~ $
```

Now that you've connected to your Raspberry Pi, run two commands to make sure that all of your packages are up to date:

```
$ sudo apt update
$ sudo apt full-upgrade
```

Once the package update commands finish running, reboot your Raspberry Pi to allow all changes to take effect:

```
$ sudo reboot
```

Running this command will disconnect you from the Raspberry Pi SSH session. Wait a few seconds for your Raspberry Pi to reboot, and enter the **ssh** connection command again to reconnect to your device.

Install Pi-hole

Enter the following single-line command to run the Pi-hole setup script:

```
$ curl -sSL https://install.pi-hole.net | bash
```

The setup script is relatively self-explanatory, but follow these tips if you aren't sure how to proceed:

- When warned about needing a static IP address, click **Continue** to proceed; we'll deal with this later
- When prompted to select an interface, press the **DOWN** arrow key and press **SPACE** to select **wlan0** to use your Raspberry Pi's Wi-Fi connection
- When prompted to choose an upstream DNS provider, choose **OpenDNS**
- Include **StevenBlack's Unified Hosts List**
- When prompted to choose a privacy level, use the arrow keys and press **SPACE** to choose **Anonymous mode**


```

rpom@pi-hole: ~
File Edit Tabs Help

Installation Complete!
Configure your devices to use the Pi-hole as their DNS server
using:

IPv4: 10.3.194.76
IPv6: 2a00:1098:3142:c2:8aa2:9eff:fe83:6b1
If you have not done so already, the above IP should be set to
static.
View the web interface at http://pi.hole:80/admin or
http://10.3.194.76:80/admin

Your Admin Webpage login password is L_EGOE1_

To allow your user to use all CLI functions without
authentication,
refer to https://docs.pi-hole.net/main/post-install/

```

▲ The Installation Complete screen of the Pi-hole setup script

When you see ‘Installation complete!’, the setup is complete. This screen shows the IP address of your Pi-hole, a link to the admin interface, and your administrator password.

Save this password somewhere safe, like a password manager – you’ll need it to work with your Pi-hole in the future. Also, save the IP address – you’ll need it to configure a static IP address shortly. Press the **ENTER** key to exit the setup script.

Pi-hole only provides a single administrator account, so there’s no username. Press the **CTRL** key (**COMMAND** on macOS) and click the link to the admin interface that uses an IP address. It’ll look something like `http://192.168.1.228/admin`. Don’t use the `pi.hole` domain link yet; until we configure the Pi-hole as our DNS provider, it won’t work. The link should open in your browser. You can also copy and paste the link into a browser if **CTRL** + click doesn’t work in your terminal. Use the admin password from the setup script output to authenticate. You can now see your Pi-hole admin console! We recommend bookmarking this console for future maintenance.

Configure your network to use your Pi-hole

To complete these tasks, visit the admin interface for your router. You can usually access the admin interface through your router’s IP address. Here are a couple of common ways to find that interface:

- Run the following command on your Raspberry Pi to output your router’s IP address:

```
$ nmcli -f IP4.GATEWAY device show wlan0
```

- Check for a sticker on your router – look for a value called ‘admin URL’ or similar

QUICK TIP

You can change the password with the command `sudo pihole setpassword` and set it to something memorable. If you forget your password, you can reset it with the same command.

Once you’ve found the IP address, log in to your router’s admin interface by typing the address (sometimes with the suffix `/admin`) into your browser. Enter your username and password (if you don’t know these, you may be able to find them on a sticker on your router). Now that you’ve

got your Pi-hole configured, you have three choices to use it to block ads. All of them involve getting Pi-hole between your network and the internet:

1. Configure Pi-hole as the DNS server for your network
2. Configure Pi-hole as the DHCP provider for your network
3. Manually point devices at Pi-hole for DNS

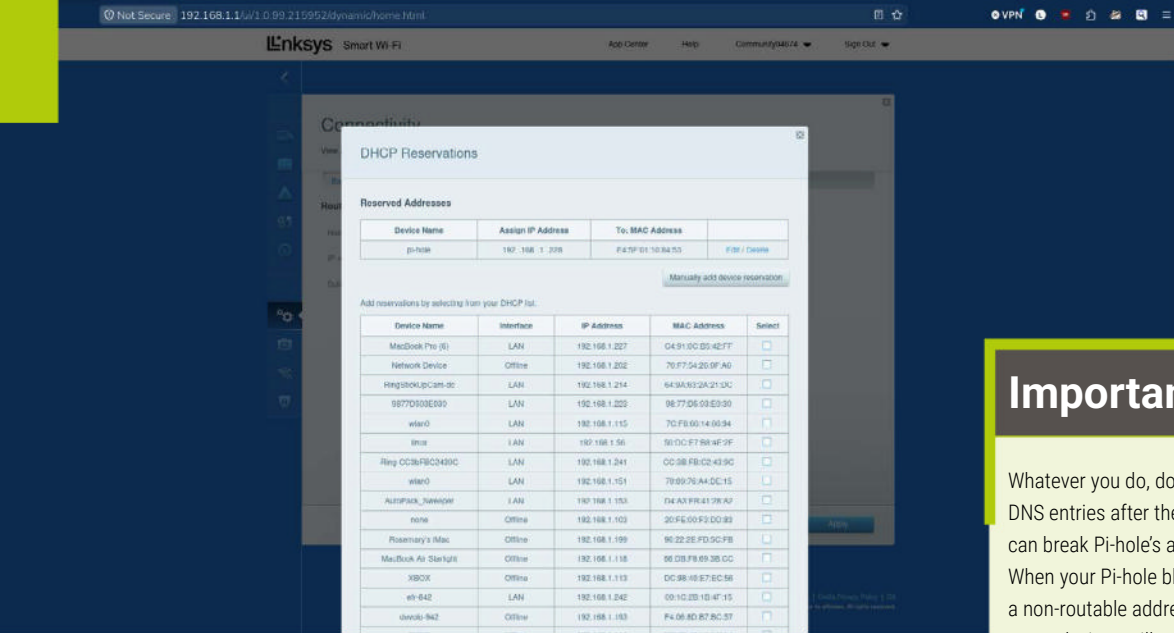
It’s easiest to use your Pi-hole as a DNS server. However, some routers don’t provide a setting to control the default DNS server. If you can’t set a DNS server, try configuring Pi-hole as your DHCP provider. And if you can’t do that either, you can always manually point devices at the Pi-hole for DNS – it’s not as good as full-network ad blocking, but it’s a lot better than nothing.

Configure Pi-hole as your network’s DNS server

This is the most common way of configuring a Pi-hole. For this method, you’ll first assign your Raspberry Pi a static IP address from your router’s interface, then point your router’s DNS server settings to the Pi-hole’s static IP address. With this setup, your router controls IP reservations across your network, but devices on the network send DNS queries to your Pi-hole instead of to a DNS server on the internet.

Assign your Raspberry Pi a static IP address

IP addresses are unique numeric codes that allow you to directly interact with devices on your network. For instance, many routers automatically assign themselves the first address in the IP block they are using, such as `192.168.1.1`. Most networks use Dynamic Host Configuration Protocol (DHCP) to assign IP addresses to devices automatically. These IP addresses are known as ‘dynamic’ IP addresses, because they can change at any time.



Important

Whatever you do, don't add any separate DNS entries after the Pi-hole entries – this can break Pi-hole's ad blocking functionality. When your Pi-hole blocks a domain, it returns a non-routable address such as 0.0.0.0, and some devices will query the secondary DNS server when the first server returns such a non-routable address. If your secondary DNS server isn't a Pi-hole, every single request blocked by Pi-hole will succeed on the secondary server, and ads will load as if you weren't running an ad blocker at all.

To run a Pi-hole on your network, we recommend assigning your Pi-hole a static IP address. A static IP address never changes. This allows devices on your network always to find the Pi-hole at the same address.

To start, run the following command on your Raspberry Pi:

```
$ hostname -I
```

You should see output similar to the following:

```
192.168.1.228
```

This value is the current (dynamic) IP address of your Raspberry Pi on the network.

To assign a static IP address, you also need the MAC address of your Raspberry Pi. A device's MAC address is a hardware identifier that your router uses to uniquely identify it. Run the following command to find the MAC address of your Raspberry Pi:

```
$ nmcli -f GENERAL.HWADDR device show wlan0
```

You should see output similar to the following:

```
GENERAL.HWADDR: A8:42:EA:58:E0:1C
```

The value on the right is the MAC address of your Raspberry Pi. Now that we know your Raspberry Pi's MAC address and IP address, we can configure your router so it always associates the Raspberry Pi's MAC address with its current IP address. Effectively, we're turning the current dynamic address into a static one using the MAC address. In your router's admin interface, configure a static IP address for your Raspberry Pi. There are several ways to accomplish this, depending on your router:

- You might be able to find this setting in the 'Advanced' section of the router admin interface. Look for a list called 'DHCP

- An example of a DHCP reservation for a Raspberry Pi in a router admin interface

Reservations', and enter your Raspberry Pi's IP address and MAC address.

- Look for a list of connected devices and find your Raspberry Pi's IP address or MAC address. Select the option to 'Always use this IP address' to make the IP address reservation static.
- Check the documentation for your router model for specific instructions.

Set Pi-hole as your network's default DNS server

Once you've assigned your Raspberry Pi a static IP address, you can configure individual devices to use Pi-hole as a DNS server in their network settings. But this process is tedious, and some devices don't provide an easily accessible DNS server setting. However, there is an easier way: most routers automatically suggest a DNS server for devices connected to your network. All you have to do is change the suggested server in your router's settings, and your entire network should start using your Pi-hole for DNS.

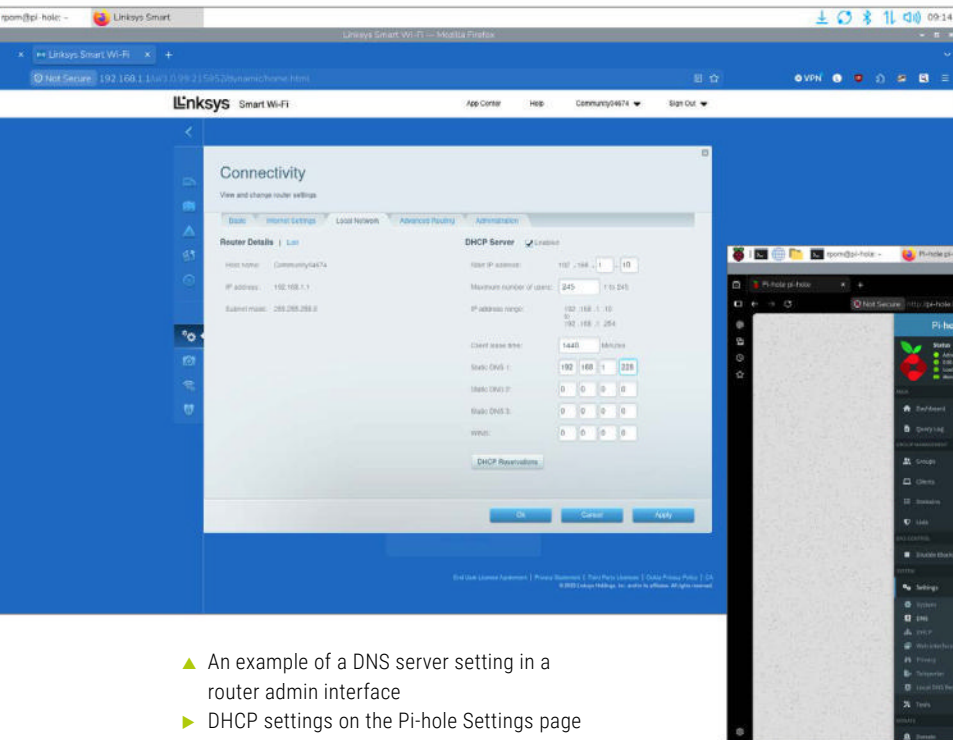
Look for a setting called DNS in your router's admin interface. You may be able to find the setting in a section called 'Internet', 'DHCP', 'Internet Connection', or 'DDNS'.

Enter your Pi-hole's IP address in the DNS (or similarly named) field.

If your router provides multiple custom DNS fields, add your Pi-hole address in each field.

Configure Pi-hole as your network's DHCP provider

If your router doesn't support configuration for static IP addresses or DNS servers, you may still be able to use your Pi-hole automatically across your network. First, check whether you can change the network DHCP server in your router settings.



- ▲ An example of a DNS server setting in a router admin interface
- ▶ DHCP settings on the Pi-hole Settings page

If you can, you can use your Pi-hole *both* as a DNS server *and* as the DHCP server that handles IP address reservations across your network.

First, navigate to the Pi-hole admin console. If you type your Raspberry Pi's IP address into your browser with `/admin`, it should redirect you there. For example:

`http://192.168.1.228/admin`

In the left side menu, select the 'Settings' page.

In the DHCP tab, in the 'DHCP Settings' block, check the 'DHCP server enabled' box.

Pi-hole should prepopulate the IP address range with the IP block that your router currently uses, and the router IP address with the router's current IP address. You can leave these values as they are. Click the **Save** button in the bottom left to start hosting a DHCP server from Pi-hole.

Finally, visit your router's admin interface, and set your Pi-hole's IP address as the DHCP provider for your network. When your Pi-hole is functioning as the DHCP provider, your router delegates all IP-related tasks to it. This includes DNS server configuration, so your Pi-hole can suggest *itself* as the default DNS server for all devices on your network. Check the documentation for your router model for specific instructions.

Manually point devices at Pi-hole for DNS

On many devices, you can configure DNS settings in Wi-Fi preferences. Look in the 'Advanced' section of your Wi-Fi or wired connection preferences for a DNS server setting. Put your Raspberry Pi's IP address in this field. Your device should immediately start issuing DNS queries to the Pi-hole.

Unless you've configured a static IP address for your Raspberry Pi, this IP address can change at any time without warning. Follow the instructions under 'Assign your Raspberry Pi a static IP address' in the DNS section above to configure a static IP address – this will prevent your device from losing its connection to the Pi-hole (and most of the internet) when your Pi-hole's dynamic IP address changes.

```
lucy@raspberrypi500: ~
File Edit Tabs Help
0 packets transmitted, 0 received, 0% packet loss, time 7009ms
rtt min/avg/max/mdev = 8.070/20.501/95.195/28.309 ms
lucy@raspberrypi500:~$ dig flurry.com

;<>> Dig 9.20.23-1-deb13ui-Debian <>> flurry.com
;; global options: +cmd
;; Got answer:
;;->HEADER<<- opcode: QUERY, status: NOERROR, id: 58441
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;;flurry.com.                IN      A
;;
;; ANSWER SECTION:
flurry.com.                2      IN      A      0.0.0.0

;; Query time: 87 msec
;; SERVER: 192.168.1.1#53(192.168.1.1) (UDP)
;; WHEN: Thu Jul 02 13:05:06 UTC 2026
;; MSG SIZE rcvd: 55

lucy@raspberrypi500:~$
```

- ▲ Using dig to test the returned DNS values

Is my Pi-hole working?

With a Pi-hole acting as the DNS server for your network, many pages will load without ads at all. Most privacy-invading trackers won't work either. You'll still see cookie and app install banners, because those can't be blocked at the DNS level.

When Pi-hole is functioning, web domains on the blocklist will be returned as `0.0.0.0`. You can check this is correct using a tool called `dig`, which is part of `dnsutils`. In Raspberry Pi OS,

we need to install dig and then point it towards **flurry.com**. Flurry.com was a web service that delivered adverts to Android and iOS apps and remains on most blocklists.

```
$ sudo apt update
$ sudo apt install dnsutils -y

$ dig flurry.com
```

You should see a response that contains this:

```
...
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;flurry.com.                IN      A

;; ANSWER SECTION:
flurry.com.                2      IN      A
0.0.0.0

...
```

Note the **0.0.0.0** next to **flurry.com** in the ANSWER SECTION.

Check that Pi-hole blocks adverts

To check to see if your Pi-hole is working correctly, open a web browser:

- Check out Adblock Tester (rpimag.co/adblock-tester). Without Pi-hole, many browsers score near 0; with Pi-hole, you should see a score above 60.
- Try visiting <http://pi.hole/admin/>. Pi-hole always routes the **pi.hole** domain to your Raspberry Pi when you use Pi-hole as your DNS server.
- Visit any site where you normally see ads and visually confirm that ads are no longer there.
- Check the proportion of ‘queries blocked’ on the Pi-hole dashboard. Many networks see between 10% and 50% of queries blocked!
- Check the DNS server used by other computers in your network. You should be able to find this information in ‘Details’ or ‘Advanced’ Wi-Fi settings. If you see your Raspberry Pi’s IP address, your configuration worked!

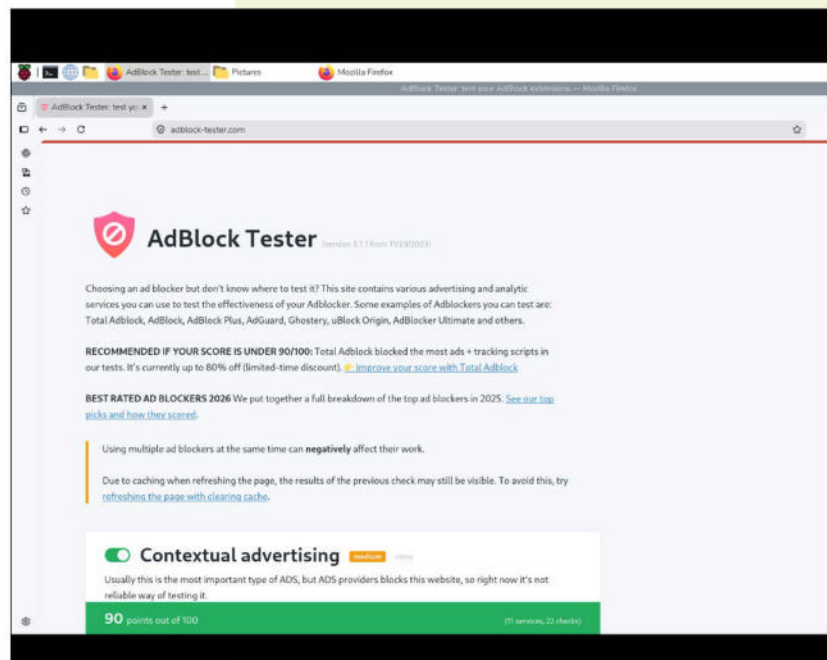
If you’ve completed all of the setup steps but you’re still seeing ads (or, worse, DNS queries don’t resolve), try the trusty solution of turning your router off and on again. This should disconnect all devices from your network and renew all DHCP

leases, putting your new settings into effect for every device. Congratulations! Your home network is now protected from ads. Put a slice of bread in your smart toaster, watch a movie on your smart TV, and flush your smart toilet in the comfort of privacy and security. 🍷



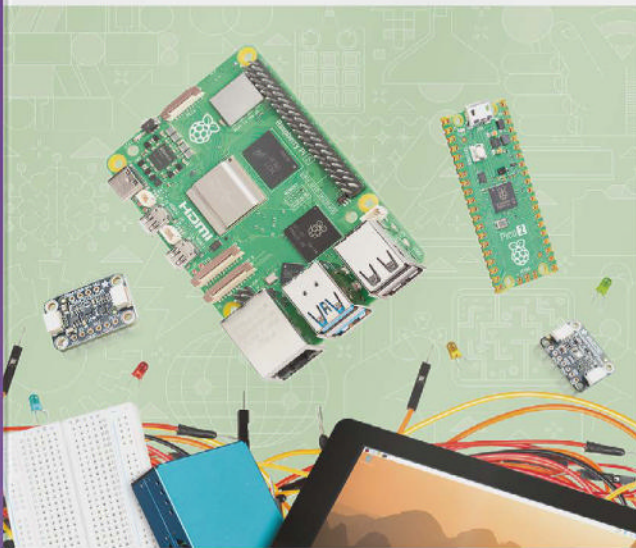
- ▲ Connect a Raspberry Pi running Pi-hole software to your network to block ads

- ▼ Using Adblock Tester to check the level of functionality



Python Projects for Raspberry Pi

Physical computing for work, play, and learning



Ben Everard

Computers are embedded into almost everything we own. Our doorbells, kitchen gadgets, vacuum cleaners, and media players are all powerful computing devices running software that someone has written.

Using the flexible Python programming language, *Python Projects for Raspberry Pi* shows you how to get the most out of the Pico range of microcontrollers and Raspberry Pi computers like Raspberry Pi 5.

■ **Learn how to:**

- *Interact with sensors and motors*
- *Build a weather station*
- *Make your own games console*
- *Add web interfaces to your projects*
- *Recognise gestures with machine learning*
- *Get the most out of your Raspberry Pi Pico with MicroPython*
- *Extend Python with C*

BUY ONLINE: rpimag.co/pythonprojectsbook

Easy compilation

Here we look at how to install software and build it from the source code



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](https://about.me/RichardSmedley)

It's so easy to install most software on Raspberry Pi OS – provided it's mature enough to have been packaged as a **.deb archive**. Often, however, there's some great code available you'd love to try, but it asks you to compile it – sometimes after first cloning it from GitHub.

While not so straightforward as running an **apt install** command, there's little to fear in stepping through the decades-old ritual of compiling software – and when errors do occur, it's often quite easy to get back on track. We'll also look at scripted installs and Python packages, but first let's find out what to do if a package is in the wrong format.

Raspberry Pi OS is just one of many distributions based upon Debian – **distrowatch.com** lists more than a hundred – and the ones you're most likely to have heard of include Ubuntu and Linux Mint. Ubuntu – the name derived from an Nguni Bantu

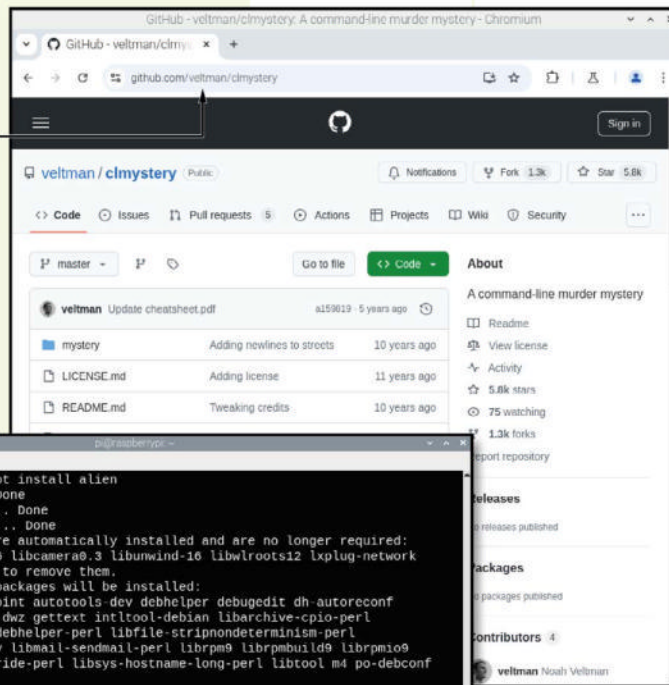
There's little to fear in stepping through the decades-old ritual of compiling software

term meaning 'humanity towards others' – is so popular that many projects, including ones not in the Debian and Raspberry Pi OS repositories, maintain **.deb** packages that you can download and install.

Ubuntu also introduced the idea of Personal Package Archives (PPA) to the Debian family – special software repositories for uploading source packages to be built and published as an APT repository by Ubuntu's Launchpad software like the official Ubuntu repository, but for unofficial software from outside. You won't tend to find these with Raspberry Pi software, but if you've been inspired to try one of the relatives of Raspberry Pi OS on your main PC, you'll find plenty of instructions on them in the Ubuntu community documentation.

Just paste the URL ending `.git` after `git clone` at the Terminal and you'll have the very latest source code

Install `alien` and you'll be able to swap packages between Raspberry Pi OS, Fedora, and other systems' packages



◀ Source code and alien packages

Track your installs

Installing software from outside of Raspberry Pi OS's repository means it must be looked after separately: updates, bug-fixes, security patches, and disaster recovery if something happens to your Raspberry Pi. Keep track of what you install.

is available as an alternative to Raspberry Pi OS should you wish to try it out. What we're concerned with here is situations where you may need to install packages for one distribution, onto a system of the other type. Your friend here is Raspberry Pi OS's `alien` package, which will convert software between packages in `.deb` and `.rpm` (RPM Package Manager, yet another recursive acronym) format (you'll need to install it with `sudo apt install alien`).

To convert from `.rpm` to `.deb`, use `sudo alien some-package.rpm`. You can go the other way with `sudo alien -r mysoftware.deb`.

Compiling code

Before package managers, it was normal to compile your own software, obtaining source code – usually written in the C or C++ languages – and running it through the GCC compiler, before linking libraries, and installing to the correct place on your disk drive.

Many of the headaches involved in the process are long gone as `configure` and `cmake` scripts do all of the hard work, checking dependencies are installed, then running the correct compiler flags for the project and the platform, and even installing the man page in the correct place.

Although most of the software you'll want to run will be available as a `.deb` to install with `apt` or `dpkg`, or come with a shell script which deals with installation, plenty of projects, particularly those you'll find on GitHub or Bitbucket, need unpacking then compiling.

When you run `apt`, or you `apt search` to look for a package, APT quizzes its local record of what packages are available to it. The record of where it gets these packages from – the address of the repositories – is kept in the file `/etc/apt/sources.list` and at files in `/etc/apt/sources.list.d/` to which you can also add repositories by hand should a project you are interested in offer one.

You can also edit `sources.list` (not advised), and meanwhile, plain old `.deb` files can be downloaded, and then installed with the command:

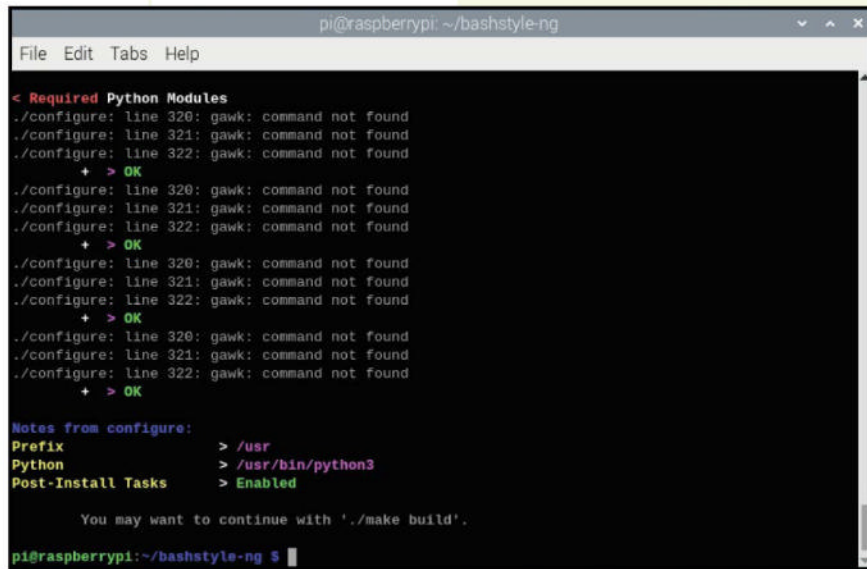
```
$ sudo dpkg -i example.deb
```

And any missing dependencies resolved with:

```
$ sudo apt install -f
```

There is another popular family of Linux distributions, based upon Red Hat, and including Fedora and CentOS. Fedora

- Compilation looks complicated, but most projects provide a well-crafted makefile for the process



After unpacking the archive with `tar xvf latest-software.tgz`, `cd` into the directory created, and look for a file with a name like called **INSTALL** or **README**. It will usually tell you to run a common sequence of commands – but read the instructions as there are variations. Two common patterns you’ll encounter are:

```
$ ./configure
$ make
$ sudo make install
```

and:

```
$ mkdir build
$ cd build
$ cmake ..
$ make
$ sudo make install
```

In most cases, software you build in this way will be installed somewhere in `/usr/local`, with executable files stored in `/usr/local/bin`. This is done to avoid interfering with versions of software that are installed through the package manager.

Dependencies

Commercial software (open-source or proprietary) often comes as a large binary file containing all of the dependencies, statically linked into the application. With most non-commercial software

projects, however, it is more common to just get a list of which version of which library will be needed to compile and run the software. Luckily, for a standard Debian package, there’s a simple command that will install all the dependencies required to build it:

```
$ sudo apt build-dep <package name>
```

However, occasionally you will need to compile and install something else first. Often the reason for compiling is to get the very latest version of the software, for new features, compatibility, or bug fixes. In the last few years, GitHub has become the default place to host free software projects; other repositories are available, but we’ll just look at fetching software from GitHub, to quickly show you what you need to know.

Your first encounter with GitHub’s existence may be seeing a banner on a project page inviting you to ‘fork me on GitHub’, or an invitation to ‘clone’ the software. Yes, we’re in the world of a project big enough to sustain its own jargon. A fork is simply your own development copy to work on, after which you can offer the changes back to the project, or publish them (on GitHub or elsewhere) for others to try or build upon.

You don’t even need to be able to code – many people are now using GitHub for collaborative development of documentation, including scientific research, and even fiction. But to simply download something, we don’t need to worry about other

Git methods – just ‘clone’ the application’s source to your Raspberry Pi:

```
$ git clone https://github.com/veltmann/clmystery.git
```

...which will make a local directory containing all of the source files. `cd` into the subdirectory just created. This is The Command Line Mystery – a text game which teaches you command line use. Entertaining and useful.

Nil desperandum

Sometimes, somewhere along the compilation, something goes wrong, and the script terminates with a complaint of some missing package. On a good day, you’ll `apt search` for the name of the missing dependency and there it will be, easy to install on Raspberry Pi OS.

On a slightly-less-good day, you’re going to have to dig around a bit to find the software, or the latest version that’s being called for. Maybe having to go to GitHub or **SourceForge.net**. This is OK if the developer has a setup not too far removed from your own; the extra install steps may go smoothly. Otherwise, don’t despair: help is available.

Most projects have one or more ways of reporting a problem and seeking help: a mailing list or Discord server; a wiki on the project SourceForge page; an email address for the lead developer; or even a social media account. Try to state your problem with plenty of detail, and remain polite and patient, and usually you’ll find helpful people. Remember, although people want to be helpful, you are asking them to give up time to answer you – if you’re frustrated with the installation process, and we’ve all been there, don’t let that stop you being respectful of anyone going out of their way to help you.

If you don’t have time for chasing up answers, but remain interested in a project, another option is to wait for the project’s

next release. By this time the dependencies may have become more widely available, even making their way into the Raspberry Pi OS repositories, or the problem may simply no longer arise.

A scripted install

You’ll sometimes be asked to grab an installation script and run it directly – as is the case with the excellent EduBlocks, a half-way house between Scratch and Python which is helping young coders get over the large step between the two languages. The EduBlocks install wants you to run:

```
$ curl -sSL get.edublocks.org | bash
```

...which takes the shell script at **get.edublocks.org** (you can look at it there in your browser) and pipes it through to a Bash process to run. The `-s` switch tells curl not to show a progress meter or error messages; `-S` overrides this to show an error message if it fails; while `-L` tells it to follow to wherever the site redirects to for the script.

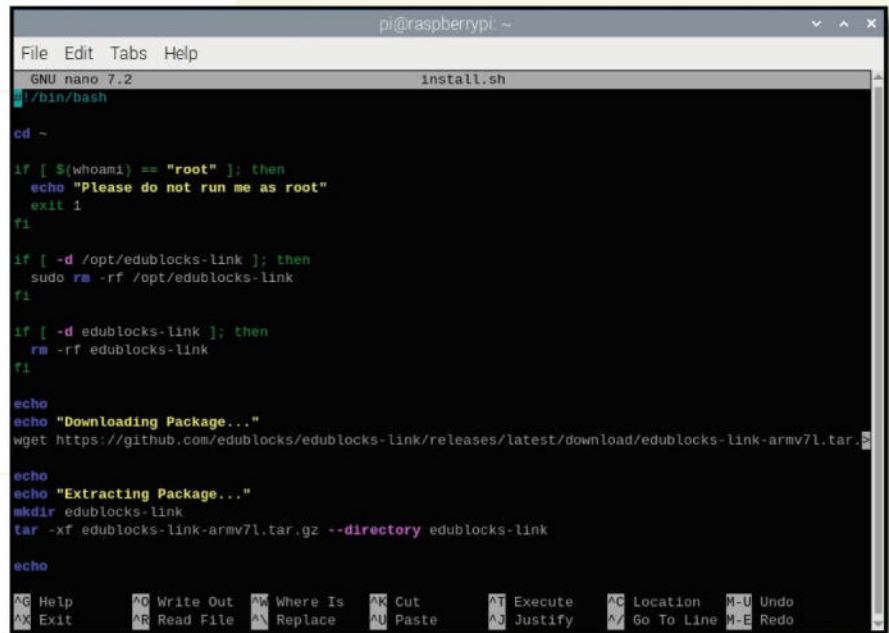
If you don’t like running software without knowing what it is doing to your Raspberry Pi, or simply wish to take a look inside the script and see what the installation does, instead, download and save it as **edu-install.sh** or just **install.sh** if that won’t overwrite anything of the same name in your current working directory:

```
$ curl -o install.sh -L get.edublocks.org
```

```
pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~ $ git clone https://github.com/veltmann/clmystery.git
Cloning into 'clmystery'...
remote: Enumerating objects: 1010, done.
remote: Total 1010 (delta 0), reused 0 (delta 0), pack-reused 1010 (from 1)
Receiving objects: 100% (1010/1010), 6.35 MiB | 5.97 MiB/s, done.
Resolving deltas: 100% (55/55), done.
pi@raspberrypi:~ $
```

◀ Git was developed by Linux creator Linus Torvalds to handle the complexities of multi-million lines of kernel code – yet makes it simple to maintain version control on the simplest of software

- EduBlocks is a good example of a project that makes your life easy by putting every stage of the installation into a shell script, and giving you just one command to run the installation



```
pi@raspberrypi: ~  
File Edit Tabs Help  
GNU nano 7.2 install.sh  
#!/bin/bash  
  
cd ~  
  
if [ $(whoami) == "root" ]; then  
    echo "Please do not run me as root"  
    exit 1  
fi  
  
if [ -d /opt/edublocks-link ]; then  
    sudo rm -rf /opt/edublocks-link  
fi  
  
if [ -d edublocks-link ]; then  
    rm -rf edublocks-link  
fi  
  
echo  
echo "Downloading Package..."  
wget https://github.com/edublocks/edublocks-link/releases/latest/download/edublocks-link-armv7l.tar.gz  
  
echo  
echo "Extracting Package..."  
mkdir edublocks-link  
tar -xzf edublocks-link-armv7l.tar.gz --directory edublocks-link  
  
echo
```

...and you'll see the script downloads and unpacks a tarball with more than one script inside, first running the one that installs dependencies.

You can run the downloaded script with `sh edu-install.sh`. If you don't read the script before running it, you're placing a lot of trust not just in the developer(s) who wrote it, but every step of its journey before it reached your machine.

Python, please

Over the years of installing software from various sources, you learn to recognise signs of whether or not it's likely to be a painless installation – and one cause for hope is something written in Python. Although both good and bad software can be written in any language, Python software and Python libraries just seem to be well-packaged and reliable.

Although Debian-based GNU/Linux distributions like Raspberry Pi OS come with the excellent APT and dpkg (Debian Package Manager), many popular programming languages have evolved their own ecosystems of packing tools and repositories – there are several for the Emacs text editor alone! Two that you are most likely to come across are JavaScript's `npm` and Python's `pip`.

Starting with the version of Raspberry Pi OS based on Debian Bookworm, you need to set up a Python *virtual environment* before installing a Python library with `pip`. This lets you install optional libraries without affecting your main Python environment. The simplest thing to do is set up an environment

under your user account. Use the following command to create a virtual environment in the `env` folder in your home directory:

```
$ python -m venv --system-site-packages ~/env
```

Next, you can run the following command from any directory to start using the virtual environment (you'll need to run this for each new command line session):

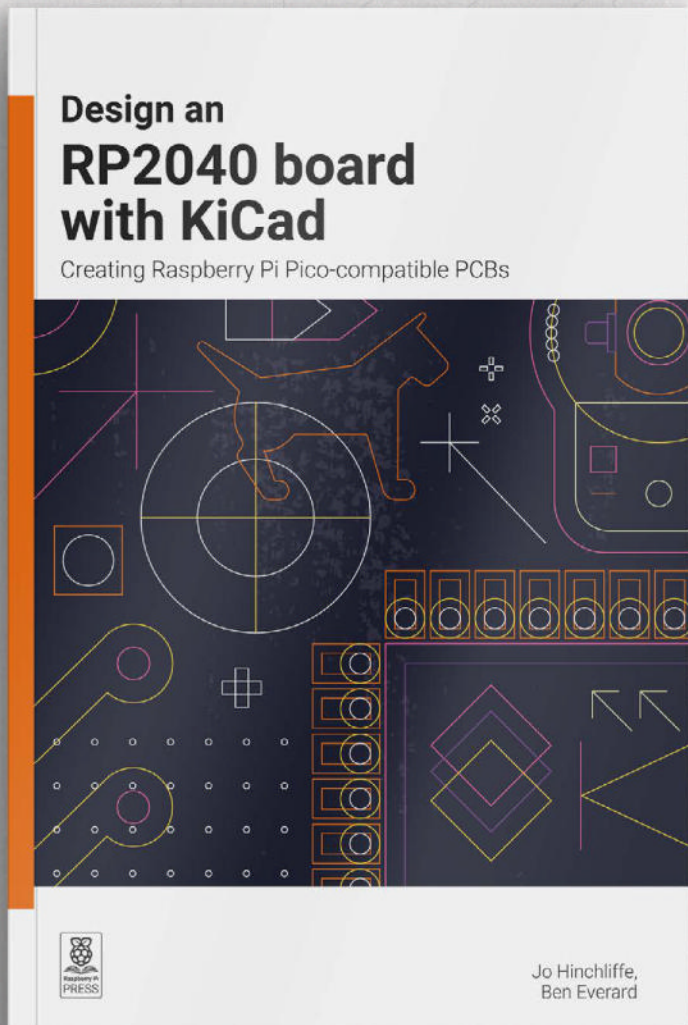
```
$ source ~/env/bin/activate
```

You should then see a prompt like the following:

```
(env) username@hostname:~ $
```

To leave the virtual environment, run the `deactivate` command. Once you have your virtual environment set up, you can install a Python package with a command like `pip install numpy`. If you need a different set of libraries for a different project, you can create another environment with a different name and activate it when needed.

Happy installing, and above all, don't worry – it's quite hard to mess up the Raspberry Pi OS installation, but if you do, at least those backups you made earlier will get tested out. 🍀



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

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

Buy online: rpimag.co/kicad2040

Pilot ACE

Turing's universal machine made real



Maker

Tim Danton

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

TechFinitive.com, where he is a senior editor.

dantonmedia.com

This story of the Pilot ACE continues from issue 167. Commander Ian Fleming devised 'Project Ruthless', and it lived up to its name. The key idea was for a daring crew of five, dressed in Luftwaffe uniform, to crash a German bomber (obtained from the Air Ministry) in the English Channel. When rescued by the Germans, they would "shoot German crew, dump overboard, bring rescue boat to English port".¹ Not the most sporting of approaches, and it was never put into practice.

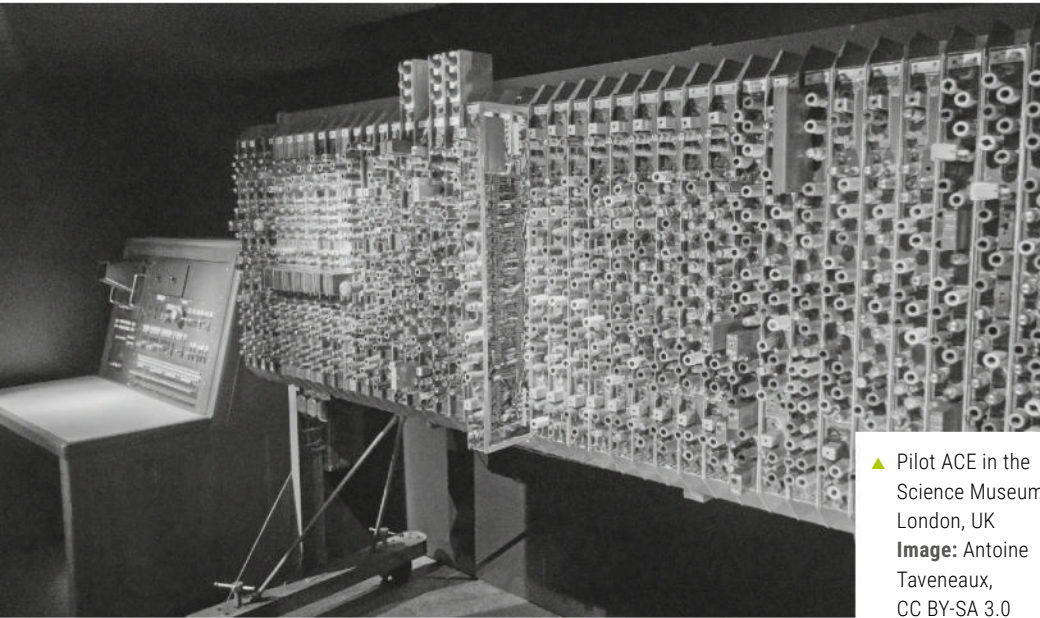
"Turing and Twinn came to me like undertakers cheated of a nice corpse two days ago, all in a stew about the cancellation of Operation Ruthless," wrote Frank Birch, who was in charge of Bletchley's naval interpretation section, in a memo to the Admiralty.² "The burden of their song was the importance of a pinch. Did the authorities realise that ... there was very little hope, if any, of their deciphering current, or even approximately current, Enigma for months and months and months – if ever?"

A long-awaited breakthrough came in March 1941, when a Royal Navy raid on a German trawler off the Norwegian coast yielded the previous month's daily keys. This gave Turing the information he needed to build the bigram tables mentioned above, and now, at last, they could break into Dolphin. Further successful raids meant they could repeat the feat when the Germans issued new bigram tables later that year.

But the course of war never runs smooth, as Hut 8 discovered on 1 February 1942 when the four-wheel version of the Enigma used by the Kriegsmarine went into general use. This meant the bombs had to cope with 26 times more options, and there was simply no way to create 26 times as many bombs. Work began on a four-wheel version of the bombe, among other approaches, but these efforts were more mechanical than theoretical so Turing was no longer central to the work.

¹ 'Operation Ruthless, October 1940', rpimag.co/opruthless

² As above. Peter Twinn worked with Turing in Hut 8.



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

Fleming devised 'Project Ruthless', and it lived up to its name

The four-wheel problem

The four-wheel problem was instead solved – or at least salvaged, as the Germans continued to add new security methods for the rest of the war – by the capture of a U-boat, and the code books it contained, on 30 October 1942. Only a few days later, Turing would sail to the United States to better co-ordinate the two countries' code breaking efforts and help the US build their own bombs at a factory in Dayton, Ohio.

His other task was to work with the Americans on their speech-encipherment system, dubbed SIGSALY. This was a room-filling beast being developed at Bell Labs, where George Stibitz built his relay computers (as we covered in part 03). Turing's time at Bell Labs brought him into contact with Claude Shannon, mainly through lunchtime chats. Shannon wasn't involved with the speech encryption, but he shared Turing's interest in creating computing machines and would go on to found what we now call information theory in a 1948 paper called 'A Mathematical Theory of Communication'.

"Turing and I had an awful lot in common," said Shannon in 1982.³ "And we spent much time discussing the concepts of what's in the human brain... how the brain is built, how it works, and what can be done with machines and whether you can do anything with machines that you can do with the human brain and so on."

When Turing returned to Bletchley Park in March 1943, breaking naval Enigma messages had become almost routine again. Construction of Colossus was under way at Dollis Hill. It was time for a new challenge.

³ 'Claude E Shannon, an oral history' conducted in 1982 by Robert Price, IEEE History Center, Piscataway, NJ, USA. rpiimag.co/shannoninterview

Building Delilah

So Turing turned his attention to the problem of miniaturising a speech encryption machine. He spent the next year devising the system and then, with the help of electrical engineer Don Bayley, manufacturing the machine itself. A device they called Delilah for its deceptive ways.

By the time VE Day finally arrived, on 8 May 1945, Turing and Bayley had produced a working prototype of Delilah. Unlike the American system, Delilah was a portable unit, but it arrived too late and was never put into active service.⁴ Nevertheless, it's another example of Turing's inventive genius and suggests that he was in the right frame of mind to tackle an even bigger project when the opportunity arose.

That ambition was essentially to build an electronic brain. In the universal machine described by his 1936 paper, he had arguably created a logical structure equivalent to everything that takes place in our physical brains.

Then, a month later, came an invitation out of the blue. Through Newman, John Womersley met Turing and offered him a job: to build a computer. This, according to Copeland, was the chance Turing had been waiting for: "Tom Flowers told me that once Turing, and also Newman, saw Colossus, it was just a question of their waiting for an opportunity to build a universal Turing machine in hardware, and along it came in the form of Womersley. So Turing just leapt on it."

Womersley was newly installed at the National Physical Laboratory (NPL) in Teddington, west London, which was pivotal in both inventing radar and developing systems to detect aircraft (although much credit is due to Sir Robert Watson-Watt alone). The NPL also later earned fame for creating the 'bouncing bomb' used in the Dambusters raid. Now that Bletchley Park was disbanding, it made sense for the NPL to be the country's new centre of excellence for computation.

A new home at the NPL

Womersley's new territory at NPL, called the NPL's Mathematics Division, had five pillars. The General Computing Section would tackle computational problems, with support from the Punched Card and Differential Analyser Sections, with a separate section dedicated to statistics. Turing took on the fifth section, dedicated to creating a new computer.

⁴ A replica of Delilah is available to view at Bletchley Park.

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



After a few weeks spent in Germany examining what its scientists had achieved in the war, Turing took up his post.⁵ By this time, he had read von Neumann's draft report on the EDVAC, but unlike Maurice Wilkes – creator of the EDSAC – he did not use the EDVAC as the basis of his design. He wanted to turn his universal machine into a buzzing electronic reality based on his own ideas.

What happened next was Turing at his absolute best. Left alone, with no distractions, he wrote a detailed report simply titled 'Proposed electronic calculator' in the space of three months.⁶ The first section of the report covered the device's composition, including the type and amount of storage required, descriptions of the logical circuits, and example problems it could solve. The second section went into greater detail, including several pages dedicated to the theory of delay line storage. An appendix set out detailed flow diagrams to show how the logical operations worked. It was effectively a blueprint of how to build a universal computing machine.

But it was Womersley who gave the computer its name: the Automatic Computing Engine. This was a nod to Charles Babbage's Analytical Engine, but also translated into a friendly acronym: ACE. The two men combined effectively in a crucial meeting to the Executive Committee of the NPL in March 1946.⁷

Selling the ACE

Womersley's opening pitch was that the ACE had the potential to be greater than the sum of the ENIAC, Harvard Mark 1, and Bell Labs computers. To be superior in both output and versatility. And that in Alan Turing, they had the man for the job. Turing then took over, providing a detailed breakdown of the proposed computer – both its applications and the challenges that stood in their way.

⁵ Tommy Flowers accompanied Turing on this trip, which took place from mid-July to mid-August. Sadly, they did not encounter Konrad Zuse (see part 02) during the investigation.

⁶ You can read the full report on AlanTuring.net at rpimag.co/elecclac

⁷ The minutes of the meeting are published in *Alan Turing's Automatic Computing Engine*, edited by Jack Copeland (Oxford University Press, 2005, ISBN 978-0198565932), pp 47-53

The minutes also make clear that Professor Douglas Hartree, who probably knew of the still-secret Colossus, was an enthusiastic backer of the project. He pointed out that Turing's design was far more efficient than the ENIAC's, needing around 2000 valves rather than 18,000 and yet having a vastly superior storage capacity: 6000 numbers versus 20. "Professor Hartree also pointed out that if the ACE is not developed in this country, the USA will sweep the field," the minutes close.

Darwin, for all his virtues, just didn't know what he'd got

The discussion soon moved on to money. Rather than build the full machine, Womersley proposed that a pilot could be created for "approximately £10,000". This was a small enough sum that it could be found from the NPL's existing budget, although the full machine might cost ten times as much. This "should be regarded as a contribution of the Laboratory to the general good of the country" read the minutes, attributing the comment to Hartree once more.

We can imagine the patriotic stirrings among the attending men at this moment, made all the more vivid by a statement that the "Committee resolved unanimously to support with enthusiasm the proposal".⁸ Over the coming three years, that enthusiasm was to be sorely tested.

⁸ There was even talk of building duplicates, which Womersley estimated would cost around a quarter of the first ACE. But he then poured cold water on the idea, suggesting that "the capacity of the machine would be such that duplicates would not be needed". In other words, one such powerful computer would be enough for the whole country.

This was despite good early progress. Turing continued work on the ACE's design while the NPL delegated the engineering work to a Post Office team led by Tommy Flowers. It was an approach that had worked exceptionally well with the Colossus, after all. But the Colossus was built during wartime, with a sense of existential urgency; in the post-war period, the Dollis Hill team's priority was to tackle a backlog of Post Office work stemming from the war. It couldn't give the ACE project anywhere near the same attention as it had Colossus.

Engineering stalls

Perhaps this should have been obvious to the NPL, but a report dated 7 March 1946 from the Post Office suggested that the team would have a "minimal ACE by August or September".⁹ This despite the fact that Flowers could only spare two men for the project, William Chandler and Allen Coombs, and as summer turned into autumn with no sign of hardware it finally became clear to the NPL management team that they needed to think about alternatives.

Hartree, who was also a professor at the University of Cambridge, suggested that Maurice Wilkes might be able to help. By this time, he and his team were making steady progress on EDSAC, including the manufacture of delay lines, and Hartree reported to the committee that Wilkes was "prepared to give as much help as he could". But while Wilkes was following von Neumann's EDVAC recipe to make a computer, Turing had entirely different ideas. How different is evident from Turing's description of one aspect of Wilkes' proposals as "rank folly".¹⁰ Nothing came of the proposed partnership.

While engineering progress was effectively zero in 1946, the mathematical side raced ahead. In May, James Wilkinson joined forces with Turing – who had worked alone until then – to create a formidable intellectual pairing. While Wilkinson had no prior knowledge of computers, he had graduated from Cambridge as Senior Wrangler. That is, the highest scorer of the entire year's mathematics graduates. Later that summer, the pair were joined by another maths graduate, Mike Woodger.

Programs without a machine

The two newcomers, once brought up to speed on what this new-fangled electronic computer could do, spent most of their time developing 'instruction tables' for the non-existent hardware. By the end of 1946, the team had created instruction tables (programs or even subroutines in more modern parlance) for

division, square root, sine and cosine, logarithms, sets of linear equations and many more.¹¹

During 1947, Gerry Alway and Donald Davies joined the group, plus the sole woman on the team, Betty Curtis. "Woodger described her as the Grace Hopper of British computing," says Copeland. He adds: "One reason why the Pilot ACE was so terrific straight away was because these chaps and Betty Curtis were sitting there writing the programs, while the hardware people were failing to build anything."

Turing, meanwhile, concentrated on the ACE's logical design. By the time Wilkinson and Woodger arrived in the summer of 1946, he had developed his initial design to what appears to be a fifth iteration – Turing dubbed it version V. But Turing was always on the hunt for ways to increase the speed of operation, so by the end of the year he was on version VII, which Wilkinson described as a "very significant advance"¹² in terms of logic over version V.

This moving of the goalposts, as anyone who has worked on similar projects will know, is challenging. It appears to be another reason why the partnership with the Post Office didn't work, with Coombs – one of the two engineers working under Flowers – saying "every time we went [to the NPL] and said 'Right, now! What do you want us to make?', we'd find that the last idea, that they gave us last week, was old hat and they'd got a quite different one".¹³ As the most extreme example, at one point the Post Office engineers were told that the NPL was considering switching to the cathode ray tube storage idea being developed in Manchester, which would have made all their work on mercury storage delay lines pointless.

Parting with the Post Office

Ultimately, and almost certainly to the relief of both partners, the NPL decided to part ways with the General Post Office. This was on the understanding that the GPO could continue development of its own version of the computer separately. Which, as we will discover at the end of this, it did.

Such was the unsatisfactory state of affairs when Harry Huskey joined the NPL on a one-year visit on 4 January 1947. This was arranged by Hartree, who had met Huskey during his tour of American computers in 1946. Unlike any of the others on the Pilot ACE team, Huskey had hands-on experience with computer hardware, having joined the ENIAC project in 1944 and eventually writing the computer's operation and maintenance manuals.

⁹ An extract of the memo is published in *Alan Turing's Automatic Computing Engine*, edited by Jack Copeland, p60

¹⁰ Jack Copeland (ed.), *Alan Turing's Automatic Computing Engine*, p62

¹¹ NPL Report for the Year, 1946, rpimag.co/compmaths

¹² 'The Pilot Ace (1982)', interview featuring Mike Woodger and Jim Wilkinson, located on NPL's YouTube channel: youtu.be/Sf281Jmm-P4

¹³ Jack Copeland (ed.), *Alan Turing's Automatic Computing Engine*, p38

Huskey wasn't willing to sit around writing routines for a non-existent computer. In April, he persuaded Womersley that they could build a 'Test Assembly' version of the hardware themselves rather than wait for outsiders. This would be the first time that they had taken an integrated approach, with one team creating the logical design – a stripped-down version of Turing's Version V – and the physical apparatus.

By now, Turing shared the belief that the theory and physical sides of the project should happen in the same place. A belief cemented by his visit to the United States in January 1947 to see how their various computing projects were progressing. "One point concerning the form of organisation struck me very strongly," he wrote in his report.¹⁴ "The engineering development work was in every case being done in the same building with the more mathematical work. I am convinced this is the right approach."

The version H affair

Even on his return from America, Turing spent time away from the NPL, with Woodger saying that he was often at King's College in Cambridge or visiting the University of Manchester.¹⁵ "[When] he came back from one of these occasions, he didn't know of this work that Huskey was doing, and he looked over my shoulder to see what I was doing," said Woodger. "I had written a preliminary piece of program for this new Huskey machine. And of course, I gave it a name, version H, and he looked over my shoulder and he said, 'What's this version H?' Then it transpired that behind his back, Huskey had the nerve to try to design a computer."

Turing's immediate reaction was predictably furious: quite aside from this happening without his agreement, he likely felt it was a poor use of his team's time. While he never played an active role in the Test Assembly, he did lend his support when requested. For example, in a 2006 interview, Huskey remembered asking Turing about the possibility of reflections from sound waves in the delay lines, thinking they would have to build a line to test this. Turing instead "got out his pencil and paper and worked out the mathematics and said that he didn't think there was a problem".¹⁶ This wasn't unusual behaviour, Huskey added: "[If] somebody had a little problem and they were having a terrible time, he would ... go way out of his way to help them."

Huskey, Wilkinson, and Woodger made slow but steady progress on the Test Assembly, completing circuit designs, commissioning a mercury delay line from the NPL machine shop, and even hunting out valves that met their specification. "Wilkinson and I went around to military dumps where at the end of the war the various military operations had all kinds of parts left over, from bomb sights to you name it," said Huskey.

The project unravels

By October, there was a whispered hope that the Test Assembly would be complete before Huskey departed at the end of the year – although Huskey believed it would have more likely been at some point in 1948 – only for the head of the NPL, Sir Charles Darwin (grandson of his more famous namesake), to halt the Test Assembly project.

The seeds for this decision had been planted at the start of the year, in light of Turing's suggestion that engineering should happen in-house at the NPL as it did in America. The most logical home for this project was within the NPL's Radio Division, Darwin felt, and the opportunity to lead the team was eagerly seized by one of that division's managers, Horace Augustus Thomas. But there were soon signs that Thomas was "behaving as if he is starting up a new Division".¹⁷ Not least when Thomas's team removed the bulk of the Test Assembly's equipment while Huskey and Turing both happened to be on holiday.

It was Thomas who persuaded Darwin to close the Test Assembly project down, leaving the project in disarray. Turing's team had already been reduced, with two members transferred to the Radio Division. Worse still, Thomas was determined to start afresh – despite his team's lack of knowledge about logic circuits – rather than work with Huskey, Wilkinson, and Woodger.

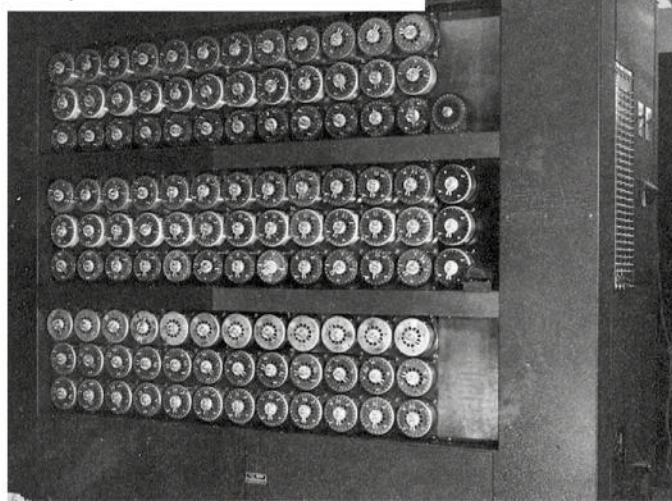
¹⁷ Letter from ES Hiscocks, secretary of the NPL, to Charles Darwin, 12 August 1947, reprinted in *Alan Turing's Electronic Brain* by Jack Copeland and others (Oxford University Press, 2005, ISBN 978-0199609154), p68

¹⁴ Alan Turing, 'Report on visit to U.S.A., January 1st-20th, 1947', rpimag.co/usarep

¹⁵ 'The Pilot Ace (1982)', interview featuring Mike Woodger and Jim Wilkinson, located on NPL's YouTube channel: youtu.be/Sf281Jmm-P4

¹⁶ 'Oral History of Harry Huskey' by William Aspray. Recorded 7 February, 2006, Computer History Museum. See rpimag.co/oralhisthuskey, p22

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



“Darwin, for all his virtues, just didn’t know what he’d got,” says Copeland. “If Thomas had not existed, and Darwin had been a different kind of administrator, I think Version H might well have flourished.” In particular, it could have been the world’s first stored-program computer, a title instead claimed by the Manchester Baby (see part 07). “That was the barest of electronics,” says Copeland. “One Williams tube and a few capacitors and they were away – and they’ve gone down in history.”

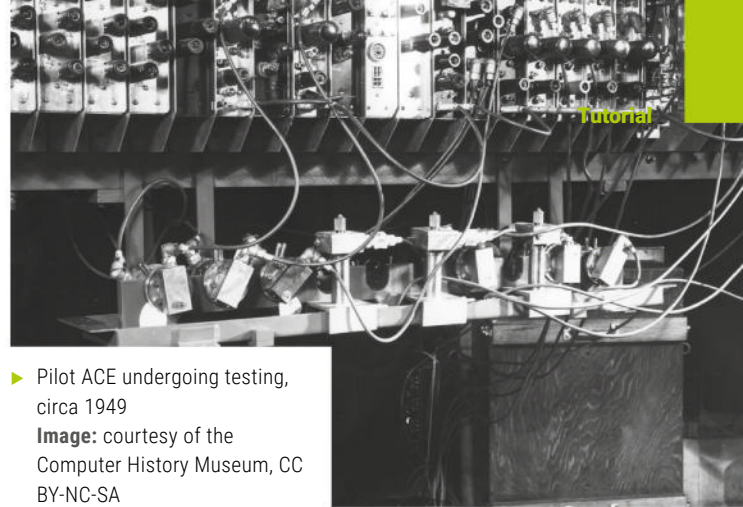
Colebrook steps in

The most dispirited team member of all turned out to be Alan Turing, who took sabbatical leave in late 1947 so that he could dedicate himself to machine intelligence research back at Cambridge. In July that year, Darwin rather optimistically wrote that the ACE “has now got to the stage of ironmongery”¹⁸ and evidently was happy for its inventor to be absent for an extended period.

Turing wasn’t the only one looking for new challenges. In February 1948, Thomas surprised everyone by accepting a job at Unilever, leaving the Pilot ACE without any obvious leader. At this point, the whole project might have fallen away, a giant bullet point heading the list of ‘IT projects that failed’. Fortunately, providence came in the form of Francis Morley Colebrook.

A man in his mid-50s, Colebrook took a mature, collegiate approach, which may be one of the key reasons he was chosen to replace Thomas in March 1948 as head of the newly formed Electronics Section of Radio Division. He soon surprised Wilkinson by suggesting that the two warring groups should join forces and physically work together in the same lab. Wilkinson, once he’d recovered from his shock, agreed.

From this point on, the project progressed as smoothly as could be hoped. Colebrook’s team benefited from the newly imported mathematical brains of Wilkinson, Woodger, Alway, Curtis, and Davies; the mathematicians learned better electronics techniques from experts such as Ted Newman and David Clayden.¹⁹ Finally, the stars were starting to align.



► Pilot ACE undergoing testing, circa 1949

Image: courtesy of the Computer History Museum, CC BY-NC-SA

There were numerous problems to solve, with Newman and Clayden dedicating much time transforming Turing’s logic circuit designs into something that would work efficiently based on the latest techniques. Meanwhile, building reliable mercury delay lines continued to be a challenge.

Building the Pilot ACE

It wasn’t until the start of 1949, wrote Wilkinson, that they “started on the detailed design of the Pilot ACE”.²⁰ It took most of the remainder of the year for the team to finish the plans and for the NPL workshops to build them. Assembly began in the autumn, and slowly the Pilot ACE took shape. “It was finally in May of 1950²¹ that we had sufficient of it assembled for the machine to work, in the sense that it stored the program in its memory and it executed that program,” said Wilkinson.

By this time Darwin had left, replaced by Edward ‘Teddy’ Bullard, who earned his scientific reputation for developing and refining the idea of a geomagnetic dynamo controlling the Earth’s magnetic fields.²² Bullard visited Wilkinson soon after becoming director and was sceptical when told that the Pilot ACE would be working any day.

“He didn’t really believe me,” said Wilkinson.²³ “And I persisted and he said, ‘Well, as soon as it works, you must call me.’ And so when it did work, I tried to get in touch with him, and of course, we couldn’t get him. And so I paced up and down, saying, ‘Where’s the bloody director? You can never find him when you want him.’ And I said this several times, and just as I was about to say it again, he stepped in through the window and said, ‘Well, here’s the bloody director.’”

Wilkinson demonstrated the program, but at this point the Pilot ACE was “in a very primitive state” so all he could show was that it could successfully execute basic arithmetic. “He was obviously very pleased and rather surprised that it was working,” said Wilkinson, “but he turned to me, and he said, ‘The program’s less than epoch-making.’ And that, I had to admit, was true.”

¹⁸ Letter from Sir Charles Darwin to Sir Edward Appleton, 23 July 1947, rpinag.co/darwinletter

¹⁹ Newman and Clayden had recently joined the NPL from Electric and Musical Industries Limited, better known as EMI. By the late 1940s, this had grown from its gramophone roots to become a manufacturer of radar and broadcast equipment. While Horace Thomas is painted as a villain in certain accounts, he deserves credit for hiring both Newman and Clayden. “I thought he’d made a very impressive place to work in,” wrote Newman of Thomas in issue 9 of *Resurrection*, *The Bulletin of the Computer Conservation Society* (ISSN 0958-7403, spring 1994). “I say that because a lot of people have derided him a great deal, and I don’t think it was fair.”

²⁰ Jack Copeland (ed.), *Alan Turing’s Automatic Computing Engine*, Chapter 4, p98

²¹ To be precise, Wednesday 10 May 1950.

²² Equal credit should go to Walter Elasser.

²³ ‘The Pilot Ace (1982)’, interview featuring Mike Woodger and Jim Wilkinson, located on NPL’s YouTube channel: youtu.be/Sf28lJmm-P4

Into the spotlight

From this point onwards, Wilkinson and the rest of the team were under pressure to develop the Pilot ACE to the point where they could demonstrate it to the press and their industry peers. They would need something more impressive than primary school mathematics. This process took another six months, until finally, on 29 November 1950, the computer was ready to make its first public appearance – a little over five years after Turing had started work on the project.

Wilkinson describes the “ordeal” of a three-day demonstration to the national press, technical press and then VIPs, with good humour. “To be perfectly frank, the machine had never worked really well up to that time, and the chances that it would

give a flawless or even an acceptable demonstration over those three days was really rather small. And to everybody’s amazement, and more to ours than anybody else, because we knew its rather shaky condition, it proceeded to give the most impeccable performance.”

It certainly impressed the BBC, which broadcast a two-minute newsreel entitled ‘THE ACE CALCULATOR’ on 1 December 1950.²⁴ “Its details are truly astonishing,” says the well-spoken narrator. “It consists of 800 valves. Its running costs are about a pound a minute, but in that time, it can work out more sums than

a hand-cranked machine could do in a month. For example, to find whether 99,997 is a prime number ... one takes just a set of cards, feeds them into the machine, and within a few seconds, the answer comes up.”

The press takes notice

The Times had published a similarly glowing report the previous day.²⁵ “The Ace itself will be built later,” states the article, “but the model demonstrated here to-day is nonetheless a complete

electronic calculating machine, claimed as one of the fastest and most powerful computing devices in the world.” It adds that this speed “could perhaps best be grasped from the fact that it could provide the correct answer in one minute to a problem that would occupy a mathematician for a month”.

The article ends optimistically but wrongly: “Now it [the Pilot ACE] is ready to ‘do business’ and is expected to more than earn its keep.” In fact, it would take the whole of 1951 – including a redesign of the delay lines and the addition of a multiplier unit – before the machine was ready to be dismantled and rebuilt in its proper home within the NPL’s Mathematics division. This happened in February 1952, at which point the Pilot ACE was indeed put into service.

Profitable service too. It essentially became a computer for hire, there to tackle complex problems that the NPL’s differential analyser and fleets of desk calculators couldn’t easily handle. Famously, it helped to solve the mystery of why a de Havilland Comet plane crashed in January 1954, killing 29 passengers and six crew. Rather than sabotage, as initially suspected, the Pilot ACE team took raw information from the investigating team and then used this to determine what had gone wrong. “Eight million multiplications was a small part of the job,” said Tom Vickers, one of the computer’s first operators.²⁶ “And eventually we found the answer to the Comet disaster.”

By combing through official records,²⁷ Copeland estimated that the Pilot ACE earned the NPL around £100,000 over its working life – £24,000 in 1954 alone – which was an excellent return on a machine that cost less than £50,000 to build.

A computer for hire

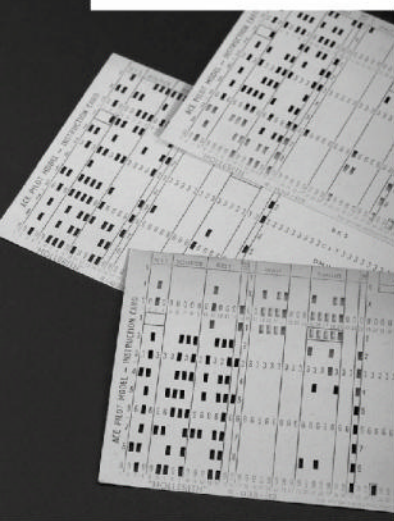
“This was big money as far as the lab was concerned,” said Vickers. “Once ... people got to hear about it, and this was the only machine around, we just filled up.” And clearly the Pilot ACE was worth the high cost, because a multiplication that might have taken 15 seconds on a desk calculator could now be done in a 500th of a second.

It’s worth emphasising just how fast the Pilot ACE was. Turing put speed of operations at the heart of his design, using coding to minimise wait times. This became known as ‘optimum coding’, although Turing never used the term.

“The instructions needed to pop out of the delay line at exactly the right moment,” explains Copeland. Take a program that, at some point, requires values A and B to be added together. “So A pops out of its delay at a particular time, and B pops out

▼ Punch cards for the Pilot ACE

Image: Science Museum
London, CC BY-SA 2.0



²⁴ ‘THE ACE CALCULATOR’, BBC Archive (Facebook), first broadcast 1 December 1950 and republished 1 December 2019, rpimag.co/newsace

²⁵ ‘Month’s work in a minute’, The Times, 30 November 1950

²⁶ ‘Piloting Computing – Alan Turing’s Computing Engine’, documentary located on NPL’s YouTube channel: youtu.be/cEQ6cnwaY_s

²⁷ Jack Copeland (Ed.), *Alan Turing’s Automatic Computing Machine*, p74

of its delay line, hopefully at the same time. With optimum coding, the instruction to add them needs to pop out of its delay line synchronously with A and B.”

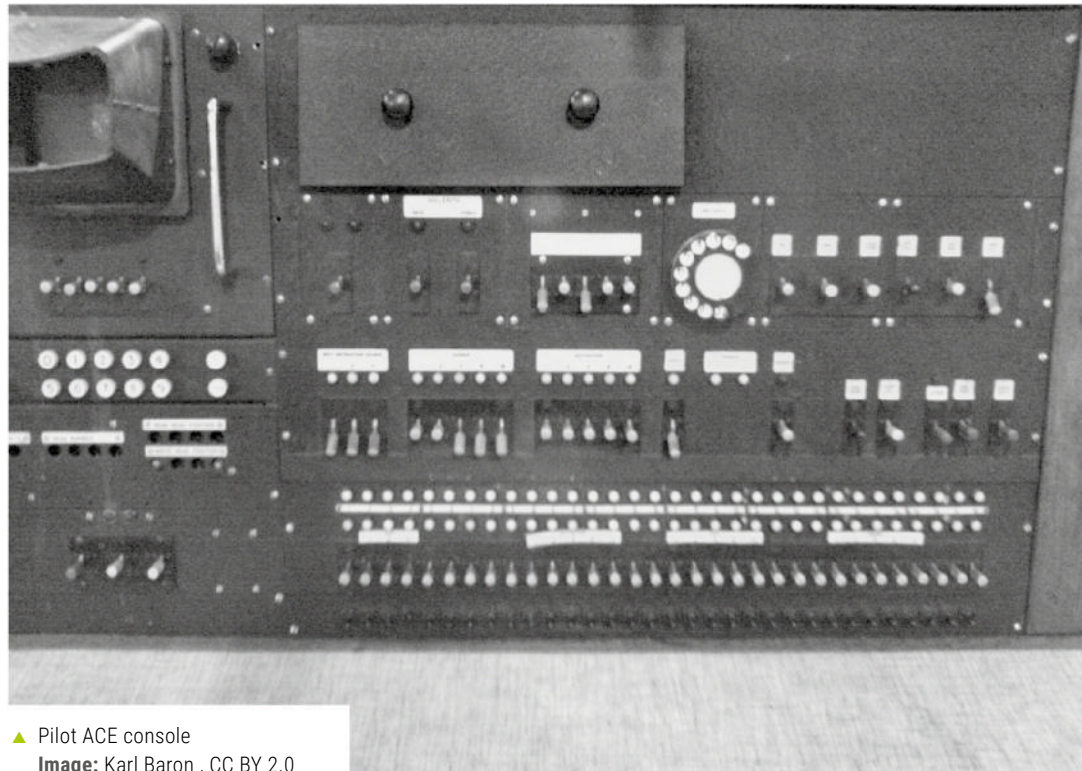
For that to happen, the programmer must place the instructions in the right places in the delay lines, explains Copeland. “Otherwise you’d need some ‘wait’ commands, where if the instruction came out too early then the machine would need to waste a few cycles of time before obeying the instruction.”

While optimum coding was incredibly clever, and in many ways ahead of its time, it also added an extra layer of complication to programming. “It was a total nightmare,” says Copeland. “I’m surprised they kept sane while they were doing it.”

Need for speed

The finished machine ran at 1MHz, twice the speed of the EDSAC, and included two arithmetic units: one supported 32-bit numbers, accurate to roughly nine decimal places, the other for longer 64-bit numbers. While Turing’s initial design for the ACE called for 200 mercury delay lines, the Pilot version included a rather more modest dozen. This limited storage to 768 words, but the team got around this by adding four slower but far more capacious magnetic drums in 1954, bringing an extra 32,768 words into the Pilot ACE’s memory. The downside: it would take up to 7 milliseconds (ms) to read them, roughly 20-30 times slower than a mercury delay line (which refreshed every 200 to 300 microseconds, or 0.2 to 0.3ms)

Turing had always intended to provide even quicker access to vital data, which is where a special cluster of shorter mercury delay lines came into play. “By good programming, the full



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

advantages of the large store can be enjoyed, with a time access not much greater than is given by the short delay lines,” reads an anonymous paper entitled ‘A Simple Guide to ACE’.²⁸ It later noted: “In many cases, the computer will average 15,000 operations per second, each involving extracting two operands,²⁹ storing the result and extracting the next instruction.”

In the meantime, at the General Post Office, a team led by William Chandler and Allen Coombs had carried on developing its version of the ACE after the NPL had decided its manufacturing services were no longer needed. One of Chandler and Coombs’s biggest decisions was to use Version VII of Turing’s design, and they came far closer to meeting his storage demands with the MOSAIC (loosely standing for Ministry of Supply Automatic Computer). This included 64 long delay lines and 7000 valves. MOSAIC was a classified project, but it’s believed to have been in active service from 1955 until the early 1960s.

²⁸ As before, p80

²⁹ An operand is a quantity or value on which an operation is performed. So if you were to add two numbers, x and y, x and y are the operands.

Life after the Pilot ACE

The Pilot ACE kept running until 1955, at which point it was replaced by DEUCE. Which made sense, as the DEUCE was essentially a replica of the Pilot ACE but built to commercial standards by the English Electric Company. We will talk more about this in the next instalment. DEUCE stands for ‘Digital Electronic Universal Computing Engine’ and it proved a commercial success, in the context of its era, selling over 30 units.

Three years later, in late 1958, came a far closer approximation to what Alan Turing had laid down on paper in 1945. ‘The Big ACE’ may not have been an imaginative name, but it was apt: this 6000-valve machine “filled a room the size of an auditorium”³⁰ and ran at 1.5MHz, although surely Turing would have been disappointed to see that it only featured 24 long delay lines. At least the storage drums had been upgraded, now rotating every 5ms rather than 7ms.

Harry Huskey also makes a surprise reappearance in the ACE’s history, having designed a commercial computer that was built by American company Bendix in 1954. The size of a large wardrobe, some have called the Bendix G-15 the first personal computer, but at a price of around \$50,000 that seems optimistic – the advertising suggests it’s “for engineering and scientific computation” and customers included civil engineering firms and universities. It was successful, too, with over 400 computers sold and Bendix even shipping a variety of accessories (such as the PA-3 pen plotter).

But Alan Turing was never to see this or any other successors to the Pilot ACE. He died from cyanide poisoning on 7 June 1954, at the height of the Pilot ACE’s commercial success.

Turing’s final years

While there has been speculation that this was suicide from eating an apple infused with cyanide, based on the presence of half an apple, the apple was never tested for cyanide. What’s more, he was experimenting with the compound in a

We shall never know the true reasons behind Turing’s death

neighbouring room at the time, and toxic fumes from this may have been the real killer.³¹

The suicide theory was given credence by the fact Turing had undergone a year’s worth of hormone therapy, as ordered by the judge in a trial where he was convicted of ‘gross indecency’ in 1952. Gross indecency being code for homosexuality, then illegal in Britain. But Turing had completed the course more than a year previously, and according to friends he had done so with good humour.

We shall never know the true reasons behind Turing’s death.

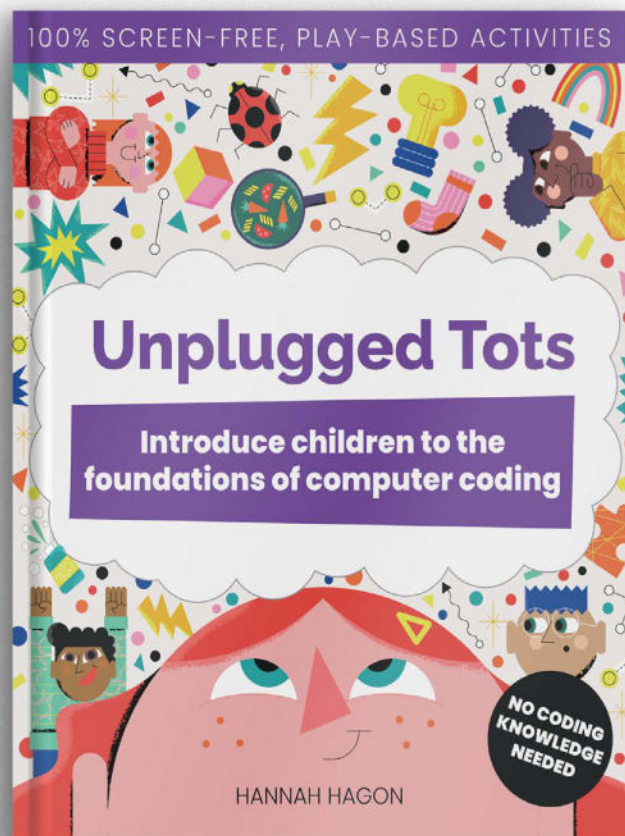
It’s easier to surmise why his elegant architecture, based on optimum coding and minimal hardware, did not survive beyond the computers listed here. As valves were replaced by silicon, hardware’s capacity to complete operations hugely increased and the need for streamlined programming lessened. Some might argue that it was dealt a terminal blow by the success of the Williams-Kilburn tube in the Manchester Baby, which made it possible to instantly access memory rather than wait for it to emerge from mercury storage delay lines.

But ‘terminal’ is too strong a word. In fact, the idea of optimum coding lives on in the power-efficient RISC architecture that our phones run on. Meanwhile, Turing’s contribution to the science of computing – most notably his work on artificial intelligence – has never been more relevant than it is today. ▣

³⁰ Jack Copeland (ed.), *Alan Turing’s Automatic Computing Engine*, p77

³¹ For more information, read Chapter 4 of *The Essential Turing*, by Jack Copeland.

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



rpimag.co/unpluggedtots

Design and build a 4-axis foam cutter

Add X, Y, and Z axes to our RP2040-powered hot foam cutter



Maker

Jo Hinchliffe (AKA Concretedog)

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

concretedog.blogspot.com

In the first part of this two-part series (in issue 166, rpimag.co/166), we began designing and building a custom RP2040 4-axis hot-wire foam cutting machine.

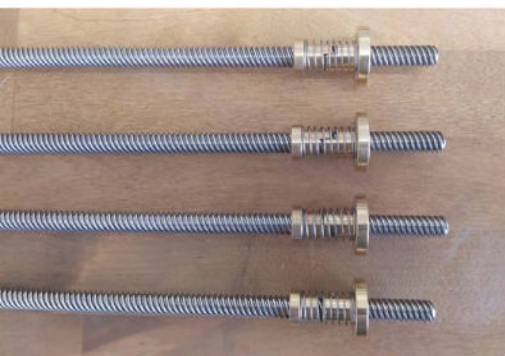
We identified a board – a BigTreeTech SKR Pico V1.0, which is built with RP2040 and a heap of motor drivers. We flashed this with a firmware from the frankly amazing RCKeith's website (rcketh.co.uk) and checked that the board would drive our Nema 17 motors. We drove these tests with the Grbl Hotwire G-code sender application, also available for free from RCKeith's site.

Just make something as a start and then iterate

We also briefly explored setting up FreeCAD on a Raspberry Pi 500 and using an add-on community workbench, 'FoamCut', to explore how we might generate custom G-codes to run our, as yet, unbuilt machine! It's often a good rite of passage to tick off these challenges as then you feel you can invest time in designing out the project further knowing that whilst there may still be pitfalls, you have some of the hard stuff out of the way.

As we have some skills in FreeCAD and 3D printing, we wanted to make a machine in a similar style to the Fila-Winda carbon-fibre filament winder we made in issue 151 (rpimag.co/151). This was built around aluminium extrusions with 3D printed parts and connectors. One nice aspect of hot-wire foam cutters is, although you could class them as a subtractive process machine, they tend not to have many forces acting on them as they cut, so we don't need to build a massively overbuilt and rugged machine.

We made an early decision on some 4020 aluminium extrusions, ordered online: two lengths of 700mm for our horizontal axis and



◀ **Figure 1:**
Trapezoidal lead-screws will give us more accuracy than we probably need



▲ The completed 4-axis hot-wire foam cutter prototype



two of 500mm for vertical. We then went shopping for lead-screws. At the cheaper end of the market, we found numerous sellers for trapezoidal lead-screws in the common T8 size (used on lots of 3D printers). However, many sellers only seemed to have 30cm or 60cm lengths in stock. We knew this would impact on our build, but for this first prototype we bought four 60cm lengths which each came with a brass anti-backlash nut (**Figure 1**).

Sometimes when prototyping it's a good idea to just make something as a start and then iterate from there. We decided to create an attempt at a vertical slide section first. We began by designing a stepper motor mount. The stepper motor will have a flexible coupler attached to the output shaft which then attaches to the trapezoidal lead-screw. The sliding platform, driven by the anti-backlash nut, won't travel over the coupler, so it made sense to extend the stepper motor mount so that the coupler section wasn't over the traversable 4020 extrusion (**Figure 2**). We also decided not to cut the 60cm lead-screw down in case we decided to extend the extrusion section later, so extending this section helps with this also. We find with stepper motor mounts that it can be useful to use slots for the main four motor mount bolts as this then gives some level of positional adjustment; you'll see that we also utilised slot mounts elsewhere in the build, allowing for fine adjustment and alignment.

At this point, we were still considering multiple different ideas about how the vertical slide tower would mount to the horizontal axis, so we designed a bearing block as a placeholder item. We

▼ **Figure 2:** The Nema 17 stepper motor mounts for the vertical axis were the first part we designed



Warning!

Fumes

Cutting foam with a hot wire releases fumes – be sure to only do so in a well-ventilated area, and using a good quality respirator or extraction system.

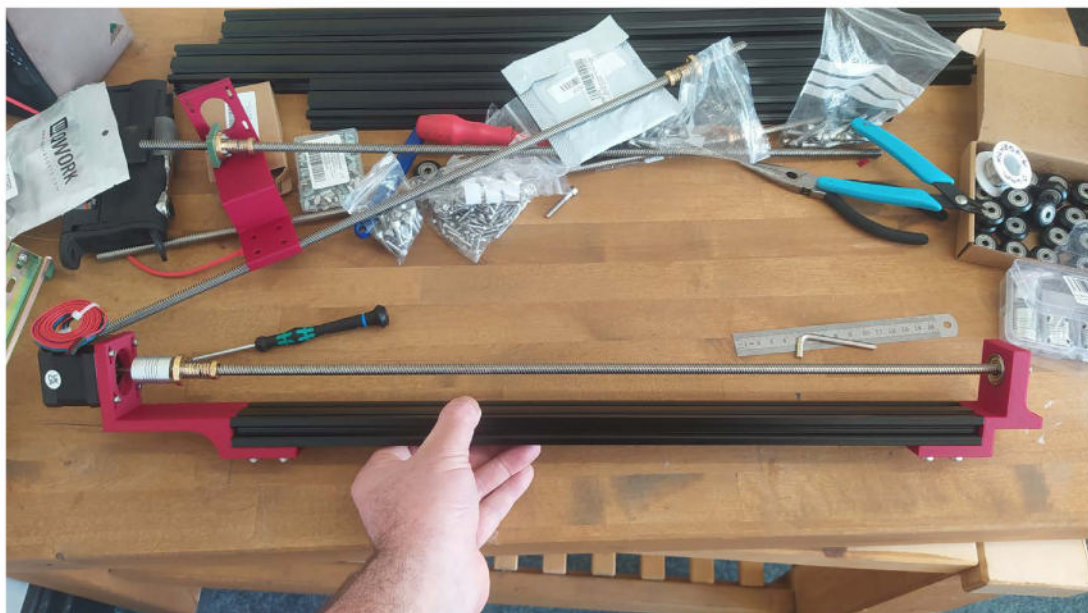
Burn risk

Hot wires don't look any different than cold wires. Switch off, unplug, and be careful!

Electrical safety

This project includes a 24V DC power supply providing 16A of current. Be careful when handling mains electricity in projects.

rpimag.co/electricalsafety



QUICK TIP

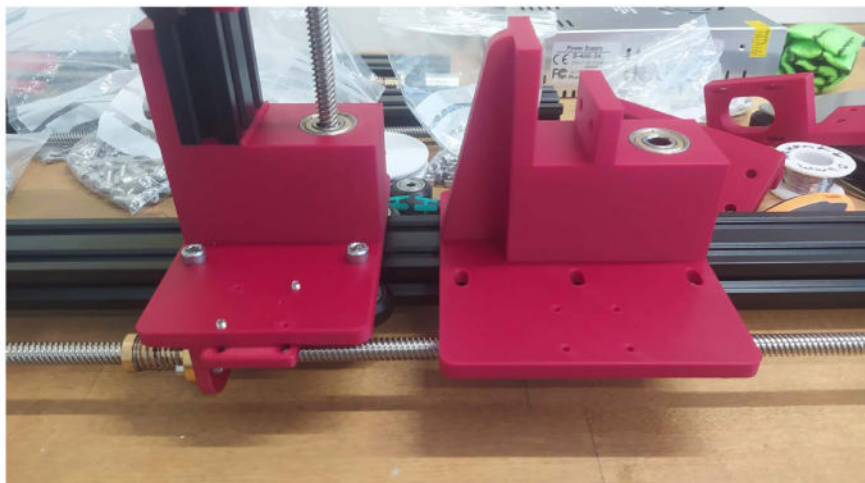
All the 3D printed parts for this project are designed in FreeCAD using simple sketches and padding and pocket operations on the Part Design workbench. Whilst it looks a complex build, each part is a pretty simple FreeCAD design.

▲ **Figure 3:** The first prototype vertical sliding tower. The bearing block end would be completely redesigned into the horizontal slider

► **Figure 4:** Numerous iterations of the horizontal slider block were made. The current design works well, but is a compromise to keep the full length of the purchased lead-screws

made a first iteration of a sliding platform with slots to mount bolts, stand-offs, and V-slot wheels. A simple L-shaped bracket was made to connect between the sliding plate and the anti-backlash nut – again, note that we used slots to connect this bracket, allowing for fine-tuning of its position relative to the lead-screw. At this point, we could now assemble a sliding tower and we could begin to consider the horizontal section of the machine (**Figure 3**).

We designed a stepper mount and a bearing mount which originally bolted only to the underside of the aluminium extrusion, but was prone to flexing, so we iterated to make designs that bolted to the upper and lower sides of the extrusion. The sliding plate for the horizontal axis had numerous iterations. It replaces the earlier bearing block attached to the vertical axis; the bearing block is quite tall so that the excess lead-screw, which travels beyond the end of the vertical extrusion, is accommodated inside the block.



The horizontal slide block went through numerous iterations, beginning with four V-slot wheels; this was increased to six V-slot wheels, and we also went from bolting the vertical slide on just one side of the extrusion to adding bolts on both sides to improve rigidity (**Figure 4**). While the machine we eventually built works really well, a definite improvement would be to upgrade the length of the vertical axis extrusions, which would reduce the need for the oversize bearing block and would allow for more travel on the vertical axis.

With the axis designs somewhat complete, we added a foot design that gave a handy way to screw the two sets of axes to a benchtop. For testing, we have only set up the machine on

Sliding sections

For all our sliding sections, both horizontal and vertical, we at first opted for the same method of mounting the V-slot wheels that we used previously to make parts for the Fila-Winda project. This simply entailed having slots on each side of the part that would receive an M5 bolt to which the wheels and spacers were added. This works OK and is a totally valid approach, but it can be hard, especially if working solo, to be able to tighten the nuts and bolts whilst pushing the bolts in the slots to create a consistent pressure/fit between the V-slot wheel and the V-slot extrusion.

To make life a little easier, we redesigned both the larger horizontal sliding section and

the vertical sliding plate so that it could use eccentric nut mounts. These are often used in 3D printers, and allow for adjustment of the position of a V-slot wheel and shaft bolt after the wheel and bolt and wheel-retaining nut are



already tightened. Our eccentric nuts had an 5mm bore to receive the M5 bolt. The eccentric nut fits into a 7mm hole which we designed and

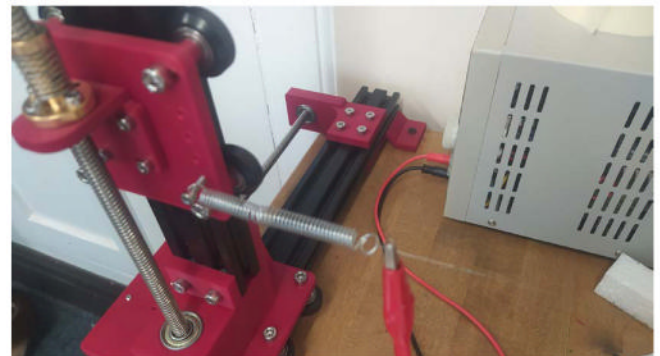
printed as a tight reference fit. The hole in the nut is off-centre, or eccentric, and this means you can first place the eccentric nut in the assembly with the nut rotated to the loosest fit. We then assemble all the sliding plate and V-slot wheels, stand-off and bolts, and use the slots on the opposite side to the eccentric nuts to get the sliding platform nearly a tight fit.

We can then, after everything is tightened, use a spanner to turn the eccentric nuts and adjust the position of the V-slot wheel bolt shafts inwards to take up any slack in the system. The eccentric nuts allow for around 2mm of travel, which is more than enough to tune the tightness to your satisfaction.

opposite sides of a pretty narrow desk. One advantage of this type of machine is that you can essentially extend the machine to accommodate larger jobs by simply spreading the axis further apart by adding a longer length of Nichrome wire.

In our initial setup, we have put a short length of 0.2mm nichrome wire held between two M3 bolts screwed into the vertical sliding plates with a spring attached at one end. The spring currently does a great job of holding the nichrome wire under tension (**Figure 5**), which is important as the wire extends in length when hot. This setup is fine for symmetrical designs where both axis pairs are travelling the exact same path; for example, when cutting a straight wing section. You can of course cut tapered wings where one axis pair will be cutting a larger design and the other end a smaller design, meaning that the wire will need to change length as it creates larger and smaller angles across the machine. We haven't tried this yet, but at some point we may need to create a wire setup that allows more wire to feed into the system. To do this, often people have one end of the wire running through an eyelet with a larger length of wire beyond the eyelet at an angle, again tensioned by a longer spring.

Now it's time for testing. We'd already set up the firmware on the SKR Pico V1.0 board and tested the stepper motors, so we dived straight into checking that all the axes moved in the correct way under the control of Grbl Hotwire. RCKeith has a test G-code file on his excellent website, which is a must read for anyone interested in building these machines. The test file



▲ **Figure 5:** A 0.2mm nichrome wire is fixed between the vertical slider plates with a spring to keep it under tension



▲ **Figure 6:** Using a test G-code from RCKeith's site to cut a small section of a Clark Y aerofoil wing

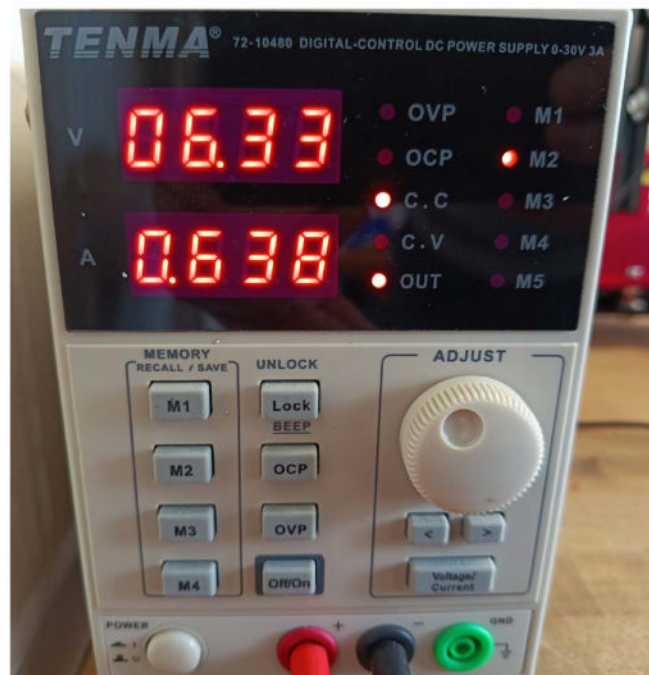
has a Clark Y aerofoil wing profile with some internal cut-outs, so it's a good first test of a machine (**Figure 6**). While we can control the wire directly from the SKR Pico board, we opted to use a constant current supply from our lab power supply simply crocodile-clipped to the ends of the nichrome wire (**Figure 7**). Note that there are lots of options here – RCKeith uses an RC battery charger which actually comes with a hot foam setting to be used for this exact purpose.

Our test cuts went well and we've begun to explore other solutions for creating cut geometries. There are numerous free applications and even websites that can generate cut files for wings, but it gets a little harder when you want to create custom

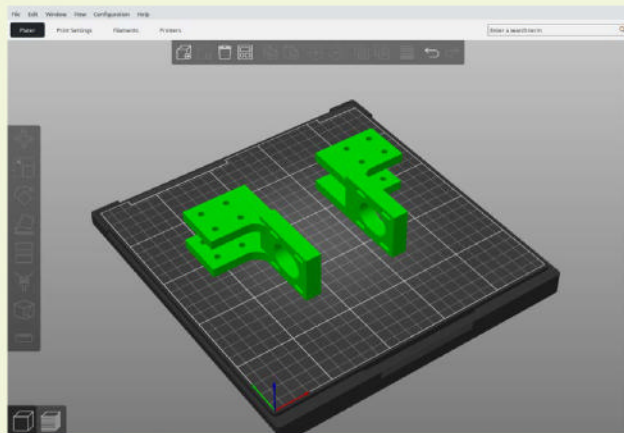
We've got to grips with the FoamCut workbench

jobs that aren't wings. As such, we've got to grips with the FoamCut workbench, which can be installed from the add-on manager in FreeCAD. It takes a little work to learn, but we can now essentially create G-code files from FreeCAD models. This opens up not only additional parts for RC planes, but also general prop making, decorative work, and we imagine this machine would be really useful for those making buildings and terrain for tabletop gaming. 🍷

▼ **Figure 7:** Whilst you can control current to the wire with the SKR Pico board, it can be easier in testing to use a constant current supply to make quick accurate adjustments

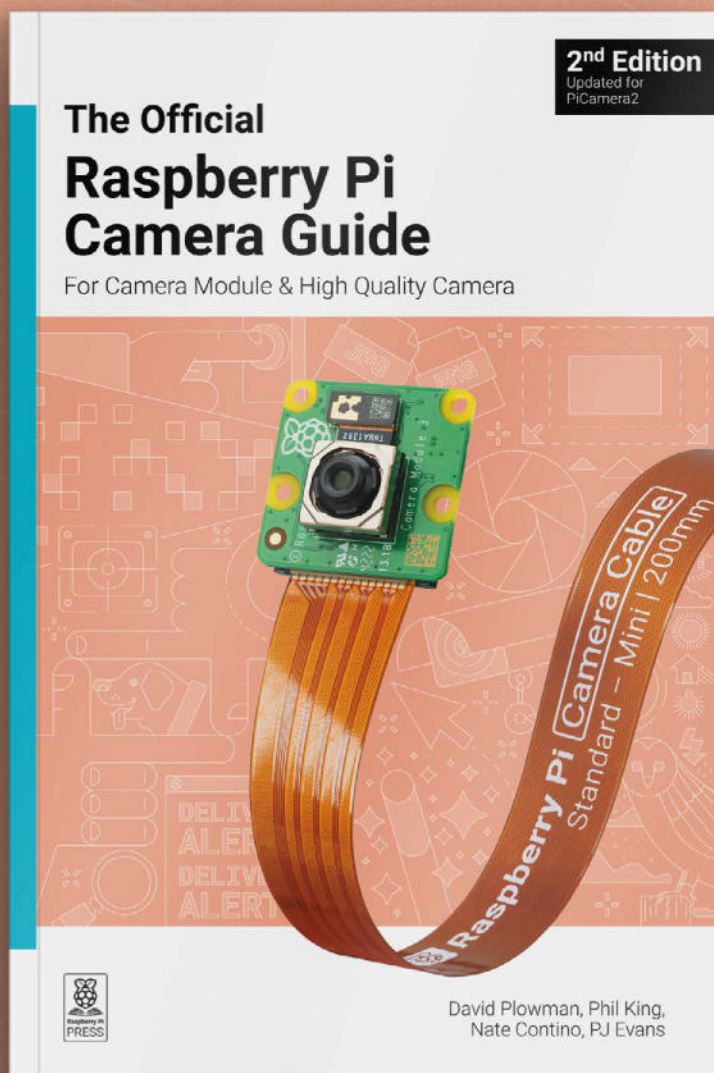


Rotate and mirror parts



If you check out the design files, you'll notice that we only designed eight parts! Many of the parts are simple repetitions. For example, we printed six of the feet parts to fit three per side of our design. Some of the parts can simply be rotated. For example, for the square vertical sliding plates, you can simply print two of them and rotate them such that the vertical array of small holes face each other across the wire section of the machine.

Other parts need to be mirrored; e.g., the large horizontal sliding section that holds the vertical tower and has the recipient bearing for the vertical lead-screw. These can't be simply rotated and need to be mirrored in the Y axis. Instead of remodelling the entire piece in FreeCAD, it's much quicker to use the mirror function in the slicing software. The same applies for parts like the horizontal bearing holder and the horizontal stepper motor mount; using mirroring can create the opposite part with ease.



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 artificial 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

Wearable beaded tech

Stimulate your creativity with beading and make an illuminating accessory



Maker

Nicola King

Nicola is a freelance writer and sub-editor. This month, she's been chasing tiny beads around the kitchen table, and vacuuming up the ones that got away.

@holtonhandmade

QUICK TIP

If you have trouble threading your needle, try using some beeswax on the thread to smooth it down, or you could swap to a big-eye beading needle so you have a larger hole.

► As you can see, not too many supplies are required for this creation. The foundation fabric we're using is proper beading foundation and has a real rigidity to it. Make sure you use something like this, or a very stiff fabric you can get your needle through

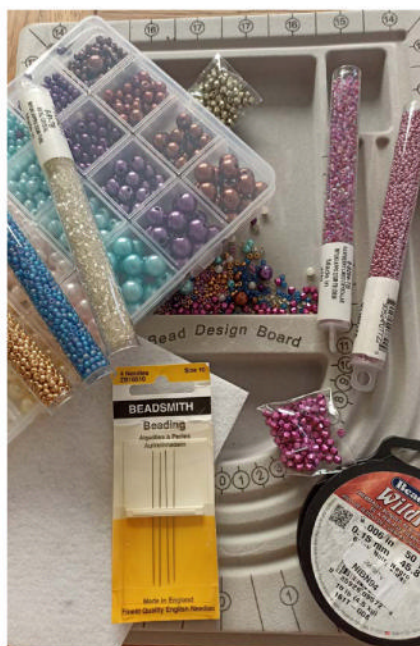
Changing a small, plain piece of fabric to one embroidered with colour is a satisfying pastime, especially when you throw in some Raspberry Pi tech to add a little extra interest and uniqueness. This month, we will be creating a small piece of wearable tech and will also be learning about embellishing and embroidering such items with beads. So, let's bead-a-long and then make it glow.

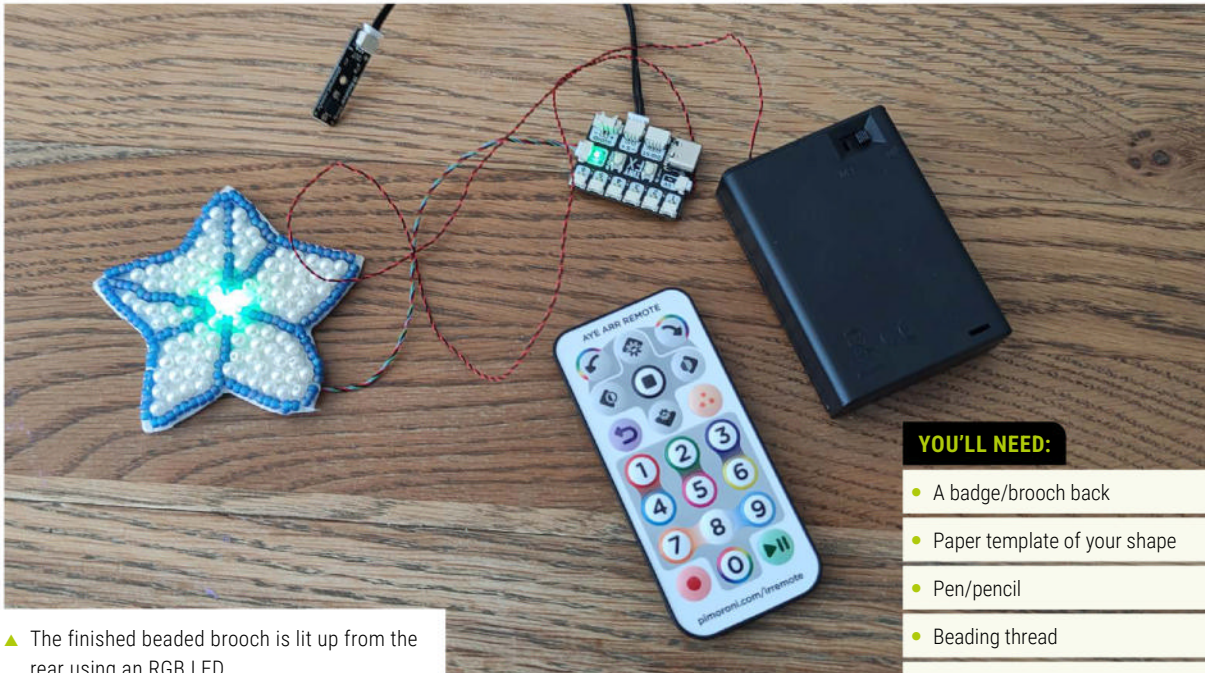
Get beading

We've chosen to make a star-shaped badge/brooch (3-inch size), so we first found a free star shape template online, printed it, cut that out of paper, and drew around it with a suitable pen on our rectangle of beading foundation fabric (available from beading/jewellery supplies outlets online, e.g. rpimag.co/beadfabric). Mark the very middle of the shape with your pen, so you know

the central point. If you want to draw any guide lines on your fabric, and/or if you have a specific pattern idea in mind, now would also be a good time to do that. Be aware that we've not yet cut the star shape out from the foundation fabric, as we'll do that later.

Next, thread your beading needle and secure the thread in the middle of the piece with a small cross. (Note that it's not a good idea to secure thread close to an edge, as it may more easily get damaged when you finally cut the shape out.) We are going to go around the edge of our shape, and over the lines we've drawn internally, with seed beads (size 8). We threaded six beads onto the needle and followed a line from the centre, putting our needle in the fabric, and then came up with our needle three beads back from that point. We then put our needle





▲ The finished beaded brooch is lit up from the rear using an RGB LED

YOU'LL NEED:

- A badge/brooch back
- Paper template of your shape
- Pen/pencil
- Beading thread
- Selection of beads (seed beads, larger beads)
- Beading needles (we used size 10)
- One or two translucent beads
- Bead foundation fabric (or a stiff felt fabric)
- Finishing material for the back of the brooch
- Sticky tape
- Scissors
- RGB LED dot (or alternative)
- Tiny FX (or Raspberry Pi Pico)
- Infrared Receiver Stick + Remote Kit (optional)
- Battery pack

We decided to use a different colour on the edge than on the inside lines

through the three beads we'd just come up under – this action will secure the beads nicely. You'll find that your beading needle is more than able to handle this, as they are very slender and flexible and they find their way easily through the small holes of these beads. We then estimated how many more we needed to reach the edge of the design and followed the same process.

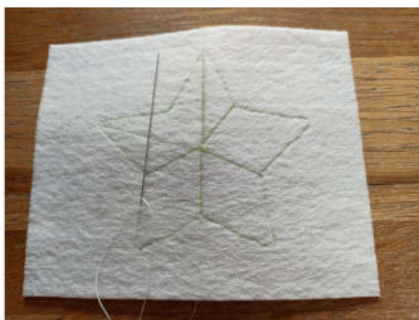
Think about the colours you want to use – we decided to use a different colour on the edge than on the inside lines. The key to remember is to secure your beads by putting your needle through the fabric, coming up three or four beads back, and then threading your needle through those three or four beads and gently pulling the thread taught. Do this all the way around the outside and on any internal lines drawn. We found that six beads at any one time was the optimum; any more than that and you can lose control of the design as it's harder to neatly secure the beads.

▼ **Bead soup!**
There are more expensive brands of bead on the market, and those undoubtedly are of a very high quality. For example, Czech beads are renowned for their superior quality and uniformity. However, you can get away with using cheaper beads for your first projects



QUICK TIP

Beads are slippery little critters, and easily roll around and get lost. Invest in some storage and a small dish/proper beading organiser tray (which usually has a flocked surface).



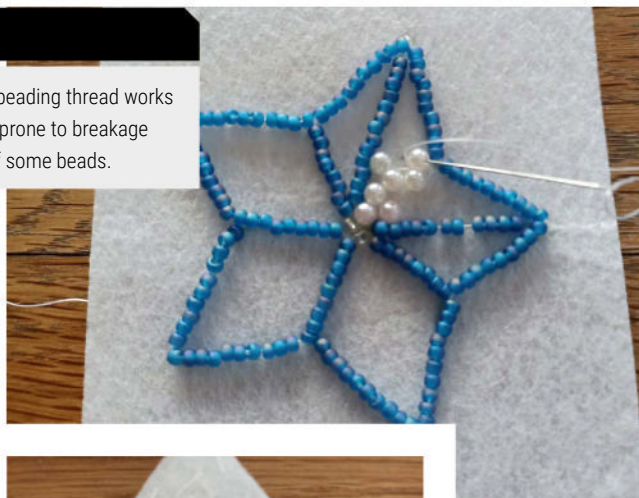
QUICK TIP

Note that *nylon* or *silk* beading thread works best, as cotton can be prone to breakage against sharp edges of some beads.

QUICK TIP

A backing for your project is optional, but it will hide the 'messy' side of your work and make it easier to attach the brooch back, so we recommend adding one.

- ▲ To create a pattern, we used a paper template and drew round it, marking inner pattern lines as required. Here, we've secured our needle and thread
- ▶ Depending on the shape that you're creating, you may find you need more or less beads to cover each section/area. Practise and see what works best for your project. It might be best to avoid threading too many beads at one time, as it's easier to secure a smaller number effectively than a large amount in one go, so six is a good number to begin with



QUICK TIP

There's no need to stick only to small beads either; you can use larger-sized beads for these kind of projects – just make sure you use the appropriately sized needle.

Once you have completed the outside of the shape, and any internal lines that you've drawn, you can start to fill in the 'empty' areas with whatever beads you like. We picked slightly larger beads to do this, for the sake of speed, and sewed in one bead at a time. We also sewed a couple of translucent beads in the very centre, as this is where we will be shining an LED light from the back of the piece, and the translucent beads will give a better effect. Note that you don't have to place these see-through beads in the centre; you can design this piece however you wish.

Having made sure that all thread ends are fully secure, take some fine-tipped scissors and gently cut off the excess foundation fabric. Be very careful not to cut any stitches while doing this as your piece will start to unravel, so watch the back as well as the front while you cut, and cut slowly. Next, cut a piece of felt the same shape as the back of your piece, and sew a badge/brooch back to that piece using the holes on the back of the brooch back.

Finally, stick your LED to the back of the main piece with a piece of sticky tape (the wire will be coming out of one edge), cover that with your brooch back (with the brooch obviously sticking out so it can be attached to something), and then take your needle and thread and place a few stitches to ensure that the back does not fall off. Now, a really professional way of finishing it would be to sew seed beads around the edge (so a straight line of single beads around the edge, sewing from top to bottom) covering all of the raw edges. So, do feel free to do that if you wish.

- ▲ The back of your work will look untidy, so you can understand why you would want to cover that up. This is the side to which you'll attach your LED

Beads of light

For the lighting effect, we opted to use a tiny RGB LED 'dot' (rpimag.co/rgbleddots) with the Raspberry Pi RP2040-powered Pimoroni Tiny FX board (rpimag.co/tinyfx). Alternatively, you could control this or a different RGB LED with a Raspberry Pi Pico.

After downloading a MicroPython code example from the GitHub repo (rpimag.co/tinyfxrainbow) and opening it in the Thonny IDE on a computer connected to the Tiny FX via USB, we got our LED cycling through a rainbow effect.

Since we also had one of Pimoroni's infrared remotes to hand (rpimag.co/irremotekit), we thought it would be fun to make our lighting remote-controlled. We plugged the remote's IR sensor stick into the Tiny FX's Sensor port (marked 'S'), then uploaded the amusingly named Aye-Arr MicroPython library

(rpimag.co/ayearr) to Tiny FX's **libs** folder via Thonny. We grabbed the **control_rainbow.py** code example from GitHub (rpimag.co/tinyfxrcrainbow) and ran it. We could then use the remote's buttons to control aspects of the rainbow effect such as the brightness, saturation, and speed. Note: make sure you have the latest version of the PicoFX library (rpimag.co/picofx) in the **libs** folder – we had to update ours to get this example working.

As per usual, you can make the program run automatically on bootup by saving it as **main.py**. We opted to power the Tiny

FX using the starter kit's 3× AA battery pack, but you could use a USB power bank or suitable LiPo battery.

Attach your finished beaded piece to whatever you want, perhaps a bag for example, hiding the wires and so on in a pocket. This is a straightforward make, but a very mindful and satisfying one too. Hopefully these basic steps, which can be improved upon and developed, have inspired you to try some wearable tech. Why not have a go, and send us pictures of your creations to magazine@raspberrypi.com. 📷

rainbow.py

> Language: Python

DOWNLOAD
THE FULL CODE:



rpimag.co/tinyfxrainbow

```
001. from tiny_fx import TinyFX
002.
003. from picofx import ColourPlayer
004. from picofx.colour import RainbowFX
005.
006. """
007. Play a rainbow effect on TinyFX's RGB output.
008.
009. Press "Boot" to exit the program.
010. """
011.
012. # Variables
013. tiny = TinyFX() #
    Create a new TinyFX object to interact with
    the board
014. player = ColourPlayer(tiny.rgb) #
    Create a new effect player to control
    TinyFX's RGB output
015.
016.
017. # Create and set up a rainbow effect to play
018. player.effects = RainbowFX(speed=0.2, # The
    speed to cycle through colours at, with 1.0
    being 1 second
019.                                sat=1.0, # The
                                saturation/intensity of the colour (from 0.0
                                to 1.0)
                                val=1.0) #
    The value/brightness of the colour (from 0.0
    to 1.0)
020.
021.
022.
023. # Wrap the code in a try block, to catch any
    exceptions (including KeyboardInterrupt)
024. try:
025.     player.start() # Start the effects
    running
026.
027.     # Loop until the effect stops or the
    "Boot" button is pressed
028.     while player.is_running() and not
    tiny.boot_pressed():
029.         pass
030.
031. # Stop any running effects and turn off all
    the outputs
032. finally:
033.     player.stop()
034.     tiny.shutdown()
```

Build an internet radio

Use potentiometers to control an LED and tune in to online radio stations



Maker

Phil King

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

philkingeditor.com

We'll hook up a potentiometer to an MCP3008

Last issue, we used a trick with a capacitor to get a reading from an analogue sensor. The best – and most reliable – way for Raspberry Pi to detect analogue inputs is by using an analogue-to-digital converter (ADC) chip, such as the MCP3008, which offers eight input channels to connect sensors and other analogue inputs. This not only eliminates the need for the capacitor, but it allows you to get more precise and instant readings across the entire range of the sensor's outputs.

In this tutorial, we'll hook up a potentiometer to an MCP3008, to control the brightness of an LED by turning the knob. We'll then add a second potentiometer and create an internet radio, using the two potentiometers to switch the station and adjust the volume.

YOU'LL NEED

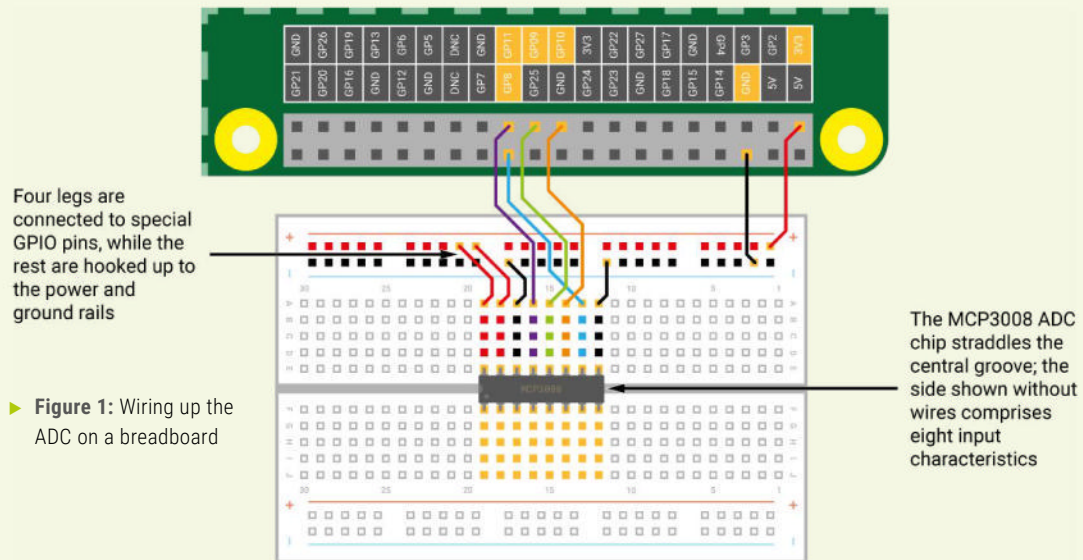
- 1× solderless breadboard
- 1× MCP3008 ADC chip
- 2× potentiometers
- 1× LED
- 1× 330Ω resistor
- 7× pin-to-socket jumper wires
- 10× pin-to-pin jumper wires

Enable SPI

The analogue values from the ADC chip will be communicated to the Raspberry Pi using the SPI protocol. While this will work in GPIO Zero out of the box, you may get better results if you enable full SPI support. First, make sure the Python `spidev` package is installed (it should be by default). Open a terminal window and enter:

```
$ sudo apt install python3-spidev
```

Next, click the Raspberry Pi menu, choose Preferences, and open the Control Centre, then enable SPI in the Interfaces section. Click OK and reboot your Raspberry Pi. You can also use the command-line tool by running `sudo raspi-config` from the command prompt, going to Interface Options, enabling SPI, and rebooting your Raspberry Pi.



► **Figure 1:** Wiring up the ADC on a breadboard

Connect the ADC

As usual, you need to turn off the Raspberry Pi while creating the circuit. As you can see from **Figure 1**, there's quite a lot of wiring required to connect the MCP3008 ADC to Raspberry Pi's GPIO pins.

First, place the MCP3008 in the middle of the breadboard, straddling its central groove. Now connect the jumper wires as in the diagram. Two go to the '+' power rail, connected to a 3V3 pin; two others are connected to a GND pin via the '-' rail. The four middle legs of the ADC are connected to GPIO 8 (CE0), 10 (MOSI), 9 (MISO), and 11 (SCLK).

Read the value

With the ADC connected to Raspberry Pi, you can wire devices to its eight input channels (numbered 0 to 7). Here, we'll connect the first (leftmost) potentiometer, which is a variable resistor: as you turn its rotary knob, Raspberry Pi reads the voltage (from 0V to 3.3V). We can use this for control of other components, such as an LED. As shown in **Figure 1**, connect one outer leg of the potentiometer (bottom-left) to the '+' power rail, the other side to the '-' ground rail, and the middle leg to the first input of the MCP3008: channel 0.

We can now read the potentiometer's value in Python. Create a new file, then save the following code as **test_pot.py**, and run it.

```
from gpiozero import MCP3008
pot = MCP3008(channel=0)
while True:
    print(pot.value)
```

At the top we import the `MCP3008` class from `GPIO Zero`, then set the `pot` variable to the ADC's channel 0. A `while True:` loop then continuously displays the potentiometer's value (from 0 to 1) on the screen; try turning it as the code runs, to see the number change. Press **CTRL+C** to exit.

Light an LED

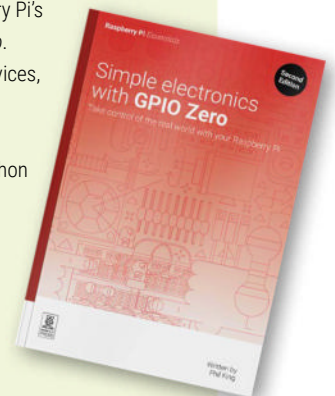
Next, we'll add an LED to the circuit as in **Figure 2** (overleaf), connecting its longer (positive) leg to GPIO 21, and its shorter

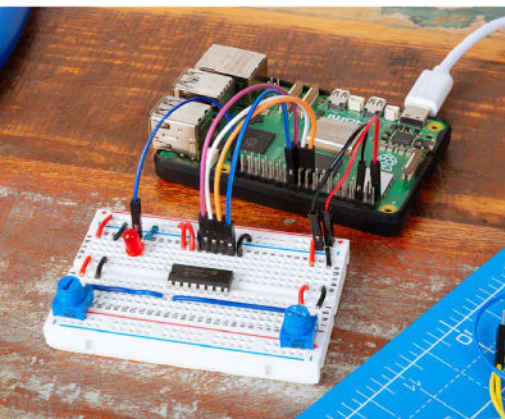
Simple electronics with GPIO Zero

This article is an extract from Raspberry Pi's book, *Simple electronics with GPIO Zero*.

Updated for the latest Raspberry Pi devices, this book has all the info you need to start creating electronic projects using Raspberry Pi's GPIO pins. Coded in Python with the GPIO Zero library, projects include LED lights, a motion-sensing alarm, rangefinder, laser-powered tripwire, and Raspberry Pi robot.

rpimag.co/gpiozerobook

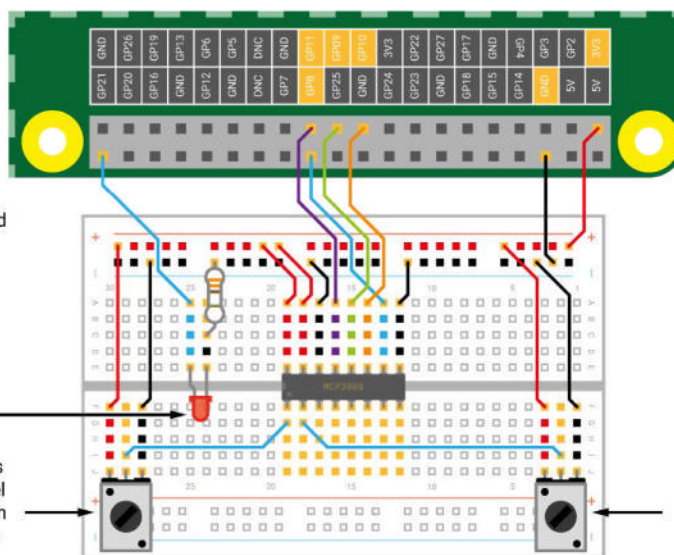




- ▲ The final circuit with two pots
- ▶ **Figure 2:** The LED and pots added to the breadboard

The LED is connected as normal with its long leg wired to GPIO 21 and the other to the ground rail via a resistor

This potentiometer is connected to channel 0 of the MCP3008; on our radio, it's used to switch the station



Connected to channel 1 of the MCP3008, the second potentiometer adjusts the volume of our radio

leg via a resistor to the ‘-’ ground rail. Create a new file, enter the following code, and save the program as **source_values.py**.

```
from gpiozero import MCP3008, PWMLED
from signal import pause

pot = MCP3008(0)
led = PWMLED(21)
led.source = pot.values
pause()
```

The code imports the **MCP3008** and **PWMLED** classes, as well as the **signal** module's **pause** function. The **MCP3008** class enables us to control the brightness of an LED using pulse-width modulation (PWM). We create a **PWMLED** object on GPIO 21, assigning it to the **led** variable. We assign our potentiometer to channel 0, as before. Finally, we use GPIO Zero's clever source and values system to pair the potentiometer with the LED, to continuously set the latter's brightness level to the former's value. Run the code and turn the knob to adjust the LED's brightness. Press **CTRL+C** to quit the program.

Add a second pot

If you haven't already, add a second potentiometer to our circuit as in **Figure 2**, with its middle leg connected to channel 1 of the MCP3008. We'll now use both potentiometers to control our LED's blink rate. In Thonny, create a new file, enter the following code and save it as **two_pots.py**.

```
from gpiozero import MCP3008, PWMLED

pot1 = MCP3008(0)
pot2 = MCP3008(1)
led = PWMLED(21)
while True:
    print(pot1.value, pot2.value)
    led.blink(on_time=pot1.value,
off_time=pot2.value, n=1, background=False)
```

Here, we create two separate **pot1** and **pot2** variables, assigned to the ADC's channels 0 and 1 respectively. In a **while True:** loop, we then print the two values on the screen and make the LED blink, with its on and off times affected by our two potentiometers. Run the code and twist both knobs to see how it changes.

Install VLC

We'll use the same circuit to create a simple internet radio, with one potentiometer used to switch the station and the other to adjust the volume. If it's not installed by default, you'll need to install the VLC media player to be able to play M3U internet radio streams. Open a terminal window and enter:

```
$ sudo apt install vlc
```

radio_new.py

> Language: Python

DOWNLOAD
THE FULL CODE:



rpinmag.co/gpiobookgit

```

001. from gpiozero import MCP3008
002. from subprocess import Popen, run
003. import time
004.
005. station_dial = MCP3008(0)
006. volume_dial = MCP3008(1)
007.
008. url = "http://ice1.somafm.com/{0}"
009. Grooves = url.format("groovesalad-128-mp3")
010. Indie = url.format("indiepop-128-mp3")
011.
012. current_station = None
013. vlc_process = None
014. last_volume = -1
015.
016. def set_volume(vol_percent):
017.     volume = vol_percent / 100.0
018.     run(["wpctl", "set-volume",
019.         "@DEFAULT_AUDIO_SINK@", f"{volume:.2f}"])
020.
021. def change_station(station_url):
022.     global vlc_process, current_station
023.     if station_url != current_station:
024.         print(f"Tuning to new station stream...")
025.         if vlc_process is not None:
026.             vlc_process.terminate()
027.             vlc_process.wait() # Ensure the
028.                 vlc_process = Popen(["cvlc",
029.                 "--no-video", station_url])
030.                 current_station = station_url
031.         try:
032.             while True:
033.                 # Read volume pot (maps 0.0-1.0 to
034.                 # 0%-100%)
035.                 vol = int(volume_dial.value * 100)
036.                 if vol != last_volume:
037.                     set_volume(vol)
038.                     last_volume = vol
039.
040.                 # Read tuning pot
041.                 if station_dial.value >= 0.5:
042.                     target_station = Grooves
043.                 else:
044.                     target_station = Indie
045.                 change_station(target_station)
046.                 time.sleep(0.1)
047.         except KeyboardInterrupt:
048.             print("\nShutting down radio safely...")
049.             if vlc_process is not None:
050.                 vlc_process.terminate()
051.

```

Make the radio

Create a new file, enter the code shown in the **radio_new.py** listing above, and save it under that name.

At the start, we import the **MCP3008** class, along with **Popen**, **run**, and **time**; **Popen** will enable us to start and stop VLC. We create variables for the station and volume dials, on ADC channels 0 and 1 respectively. We then assign variables to two radio stream URLs (we've used SomaFM Groove Salad and Indie Pop in this example).

Next, we create a couple of functions. The first, **set_volume**, uses **wpctl** (the command-line control tool for WirePlumber, the PipeWire session manager) to control the volume. The second, **change_station**, includes an **if** condition so it only triggers when the station set by the first potentiometer position is different

from the currently selected one (**current_station**). If so, it stops the current stream and starts playing the new one, before reassigning the **current_station** variable to it.

Finally, in a **while True:** loop, inside a **try ... except** block, we read each potentiometer in turn and use **if** conditional statements to call the related functions, **set_volume** and **change_station**, when required.

Run the code and try turning both potentiometers to switch the station and adjust the volume. To keep things simple, we've only used two radio stations in this example, but you could easily add more, adjusting the **station_dial.value** thresholds accordingly.

Thanks to the **try ... except** block, pressing **CTRL+C** stops the radio stream before exiting the program. 🍷



▲ Flight GAF866 flyby at ILA Berlin Air Show

Pi-Sky

Build a local flight-tracking dashboard on Raspberry Pi with live maps, history, and replay. See what's happening above



Maker

James Mitchell

Maker based in Berlin. A long-time Raspberry Pi community member and passionate about Raspberry Pi, digital fabrication, and maker culture. He organises Raspberry Jam Berlin and attends Maker Faire events in Germany. Whether you're a hobbyist, educator, or business, he is always happy to discuss projects and new ideas.

bytingidea.com

Ever looked up and wondered where that plane you spotted is going to or where it came from?

Commercial flight trackers show aircraft worldwide, but depending on how much data you want or how much you're willing to pay, having live usable data in hand can be rare. With a Raspberry Pi and an ADS-B 1090MHz receiver, builders can decode those open radio broadcasts from nearby aircraft and can spy into what is going on overhead.

Pi-Sky's superpower is storing all the flight data locally

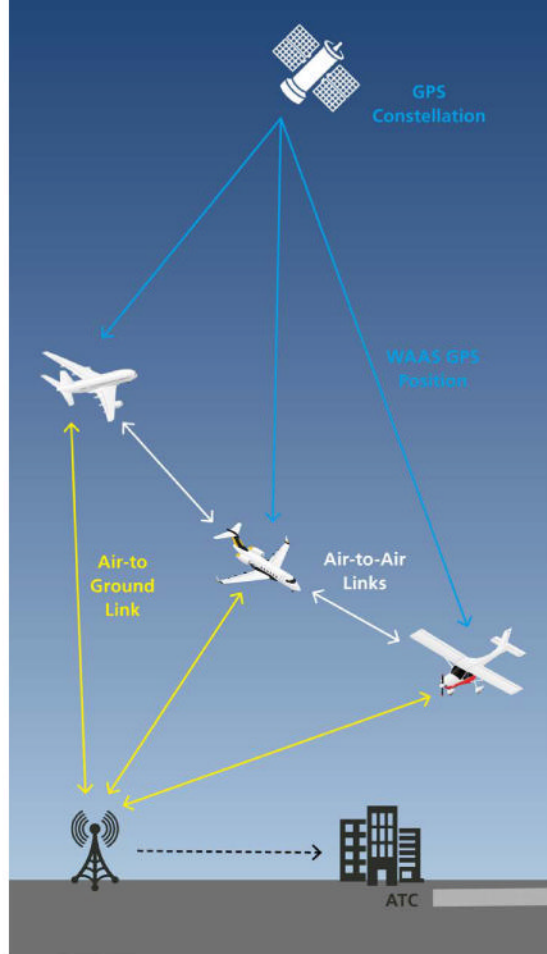
What aircraft transmit and how we receive it

ADS-B stands for Automatic Dependent Surveillance Broadcast. Most modern aircraft broadcast ADS-B on 1090MHz in short bursts, several times per second, carrying an ICAO hex address (an

identification number assigned to a specific aircraft), a callsign (air traffic control identifier for aircraft – for example, RYR3EM), GPS position, altitude, speed, track, vertical rate, and squawk code. However, what is not sent in the broadcast is what type of aircraft it is, where the flight came from, where it's going, when it departed, or will arrive.

Typically when you use an app like FlightAware or Flightradar24, the flights you are seeing are collected from a network of 1090MHz receivers from all over the world as well as the data from flight providers. It's hard to say the percentage,

- Signal chain from aircraft ADS-B SDR receiver, to ATC and Tracking Services
Credit: CC BY-SA 4.0 Atlas Air Service AG



- ▼ The FlightAware Pro Stick Plus ADS-B/Mode S receiver



but receivers can be from enthusiasts, air traffic control, adjacent institutions, or even governments.

Any receiver within roughly 100 to 250km can pick up the 1090MHz broadcast from aircraft; it depends on the altitude of the flight, the antenna type in use (indoor or outdoor), and surrounding terrain.

Build your own receiver

Generally speaking, any modern Raspberry Pi computer with USB and networking can be used as a base to build your own receiver. But you might want to place your receiver outside and that's where power and size become a consideration.

You will need a compatible USB receiver tuned to 1090MHz; e.g, the FlightAware Stick, FlightAware Pro Stick, Airspy Mini, or RTL-SDR Blog V3 or V4. Typically these receivers feed a decoder software: dump1090, which publishes JSON at a local URL such as `/data/aircraft.json`. There are a few generic RTL-SDR dongles out there, but as long as they remain compatible with the dump1090 software, they should work on Raspberry Pi OS.

When you consider which USB stick you want to use, it's worth noting whether they have built-in filters or come with antennas with filters connected. Filters are important to block unwanted radio noise and, in addition to the antenna placement (the higher the better), this can improve the range in which the aircraft can be detected.

It's recommended to use the FlightAware Pro Stick Plus. FlightAware's Raspberry Pi packages are mature, regularly

updated, and widely documented. The Pro Stick Plus in particular combines receiver and filtering in one unit. There is an option to upload your data to FlightAware's network, meaning you become an Enterprise Subscriber and can have an ad-free experience to its FlightAware app.

Introducing Pi-Sky

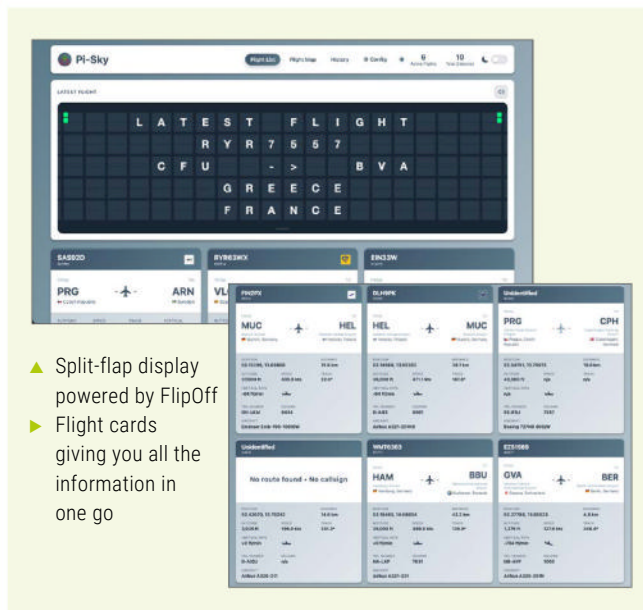
Pi-Sky is a single Python application and web server that reads the dump1090 data, enriches that data through public APIs, and streams live JSON to a connected browser. Everything runs locally, so you have control of the data.



Pi-Sky features

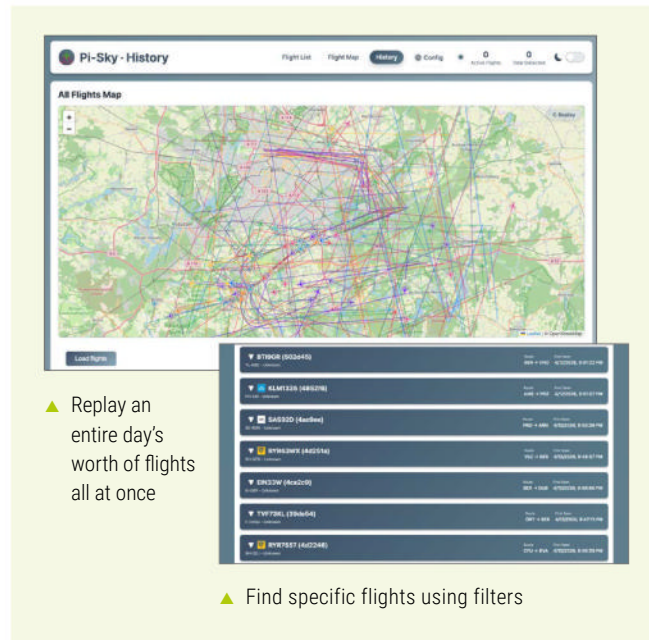
Here are some features to be found in Pi-Sky:

Flight list: Card-style list of flights detected by your receiver, with route details and airline logos. An optional split-flap display shows the latest flight in an airport departure-board style.



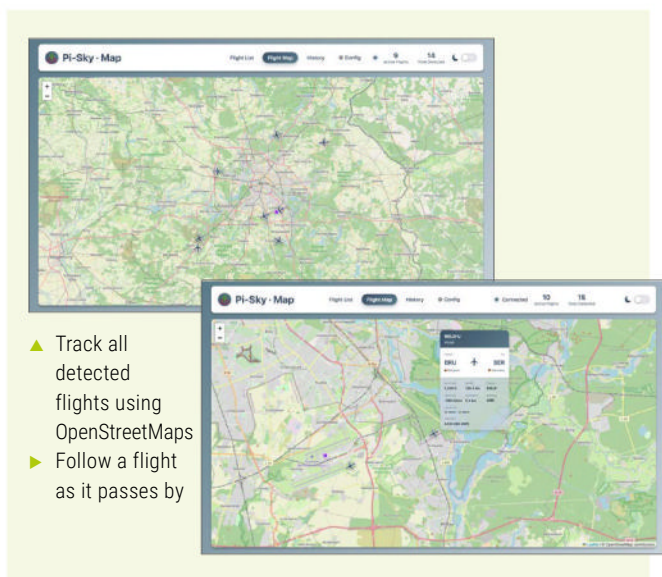
- ▲ Split-flap display powered by FlipOff
- ▶ Flight cards giving you all the information in one go

History: Received flight data is stored in SQLite (for how long is configurable – the default is seven days). On the history page you can replay flights on a map, browse individual flights with mini maps and photos of the aircraft, and filter by date, route, aircraft type, and model.



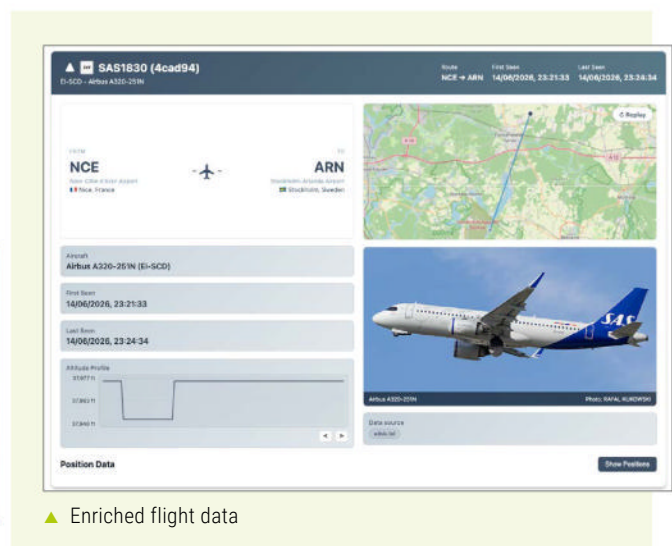
- ▲ Replay an entire day's worth of flights all at once
- ▶ Find specific flights using filters

Flight map: Live local map of detected aircraft as plane icons. Click a flight for details and follow it on the map.



- ▲ Track all detected flights using OpenStreetMaps
- ▶ Follow a flight as it passes by

Enrichment: ADS-B only gives a quick snapshot of each aircraft. Pi-Sky uses free APIs to fill in (enrich) route information, aircraft type, registration, and airport details where available.



- ▲ Enriched flight data

Pimoroni Inky e-ink support: Show a small map and flight card on a connected Inky display.



◀ On display at
BerryBase Berlin

Projects that use OpenSky Network

Recently there have been a number of projects on social media that use OpenSky Network but it's unclear if they are following the terms and conditions or have permission to use the data. If you choose to use OpenSky Network on your own project, you run the risk of having your IP excluded from their service and at worst legal action especially if you are trying to sell a product using their system.

Enrichment data and where it comes from

Pi-Sky uses **adsb.lol** to fill in details that ADS-B broadcasts do not include. Adb.lol is built by aviation enthusiasts and other contributors who share receiver data into one central service, which maintains a database of flight routes. It offers free API access, including historical route data, with open terms of service. If you want to support the project, you can feed your own receiver data to adsb.lol separately, or donate via its website.

When your receiver detects an aircraft, Pi-Sky stores positions in its local SQLite database (if enabled). If the aircraft has a callsign and a GPS position, Pi-Sky asks adsb.lol whether it knows the route information: origin and destination. That works well for regular airline flights that appear often in the historical database. It is less reliable for one-off flights such as private, military, rescue, or police traffic. Any enrichment Pi-Sky receives is saved locally with the flight record.

The adsb.lol service usually returns airport codes such as BER, EDDB, or STN. Pi-Sky then uses the OpenFlights (**openflights.org**) airport database to turn those into readable names: for example, EDDB (or BER) becomes Berlin Brandenburg, while STN becomes London Stansted.

For aircraft details, Pi-Sky looks up the ICAO hex address via adsb.lol to find registration, type, and model when available.

There is also an optional offline Micronics aircraft database you can upload in Pi-Sky's settings; it is indexed by ICAO address and can fill in missing manufacturer, type, model, and registration fields.

In the History page, Pi-Sky can show an aircraft photo when a registration is known. It tries **PlaneSpotters.net** first, then falls back to the Wikimedia Commons API.

It is a patchwork of free public sources, and it is not perfect but that is also why Pi-Sky can exist without paid flight-data subscriptions.

What about OpenSky Network?

With a Google search for flight-tracking APIs you might find OpenSky Network: **opensky-network.org** is at the top of the list. Like **adsb.lol**, OpenSky is built from crowdsourced ADS-B data and offers APIs. In principle, Pi-Sky could use OpenSky for enrichment, but there are important caveats.

The site states: "Free access to full dataset is granted only to research institutions, aviation authorities, governments, and non-profits not for personal use." Digging deeper into OpenSky's terms of use licence, data is again stated primarily for non-profit research and education. More importantly, for Pi-Sky as an always-on flight tracker, it would still require a written licence,

even for non-commercial use. Commercial or for-profit use also requires a separate agreement.

If you do use OpenSky data in published work, its terms require proper citation. API access is token-based (OAuth2), with credit limits on most endpoints meaning it's possible to have a drop in service. For a self-hosted hobby project, adsb.lol is the better fit: Pi-Sky can query it freely when enriching flights.

Installation and configuration

The build starts with a fresh Raspberry Pi OS image written using Raspberry Pi Imager (rpimag.co/install). Once your Raspberry Pi has booted, open a terminal window.

You can install Pi-Sky by cloning this GitHub repository and following the instructions in its **README.md** file:

```
$ git clone https://github.com/monkeymademe/pi-sky
```

Alternatively, Pi-Sky can also be installed with a one-line installer that handles all the requirements: it adds FlightAware's APT repository, installs dump1090-fa, clones Pi-Sky to **/home/pi/pi-sky**, creates the Python virtual environment, writes **config.json** and registers systemd services so both dump1090 and Pi-Sky start automatically at boot.

Before you run the installer, connect the ADS-B USB stick.

```
$ curl -fsSL https://raw.githubusercontent.com/monkeymademe/pi-sky/main/install.sh | sudo bash
```

When the script finishes, it prints the local map URL, typically **http://<pi-ip>:5050/**.

Open that address in a browser on the same network to confirm it's working. If no traffic is overhead yet, check that dump1090 is running with:

```
$ sudo systemctl status dump1090-fa
```

Alternatively, verify the JSON feed with curl:

```
$ curl http://127.0.0.1:8080/data/aircraft.json
```

Fine-tuning happens through the web-based config page at **/config.html**.

It's important to set the location of the receiver in the config (if you have Pi-Sky publicly available, you can hide the location from the map as well as lock the config). If you have an airport close by, you can enter it there (this helps with centring the map).

The next step

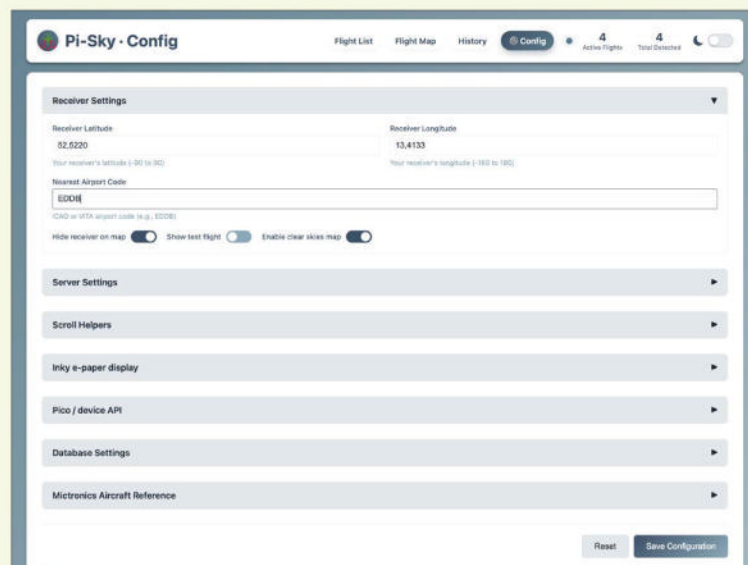
Pi-Sky's superpower is storing all the flight data locally and it's free for you to use to do whatever you want to do with it. Already baked in is the possibility to display some of that data on Pimoroni's Inky e-ink display.

In part two of this tutorial, we will look at how to set up a Raspberry Pi Pico to access Pi-Sky's data and use it in various different displays. 🍷

- ▶ The config can also be locked for publicly accessible sites

PiAware setup

If you already own a FlightAware Pro Stick and wish to contribute coverage to FlightAware's network, you can optionally follow the full PiAware setup first (see rpimag.co/flightawareinstall), but this is not required for Pi-Sky to work.



Raspberry Pi Essentials

Second
Edition

Experiment with the Sense HAT

Sense the real world with your Raspberry Pi



Raspberry Pi Foundation
Learning Team

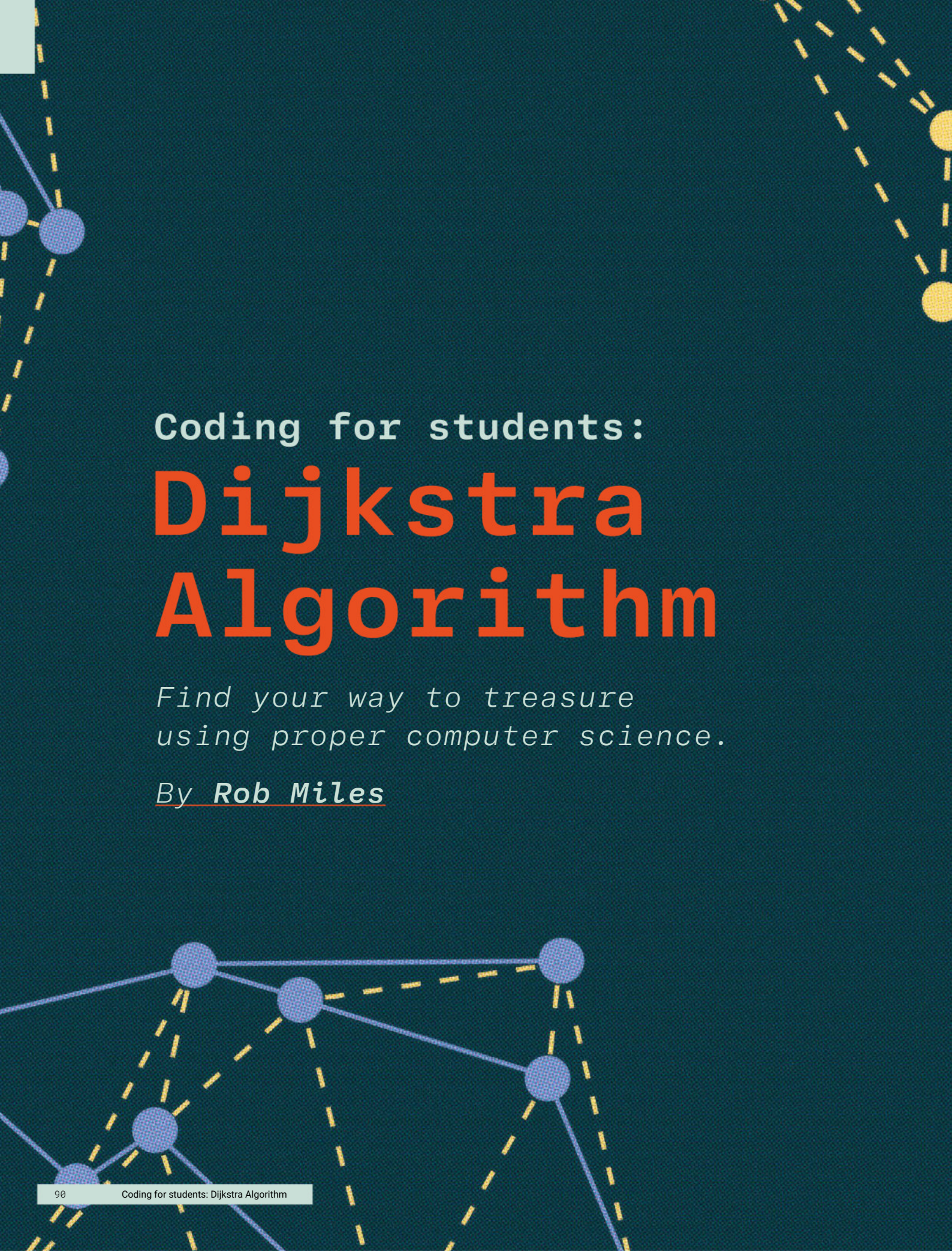
The Sense HAT is an incredibly versatile and flexible bit of kit with plenty of obvious uses, along with a huge number of less obvious ones, that you'll love to make and share. Updated for the latest Raspberry Pi devices and hardware, this book has everything you need to get started.

■ ***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

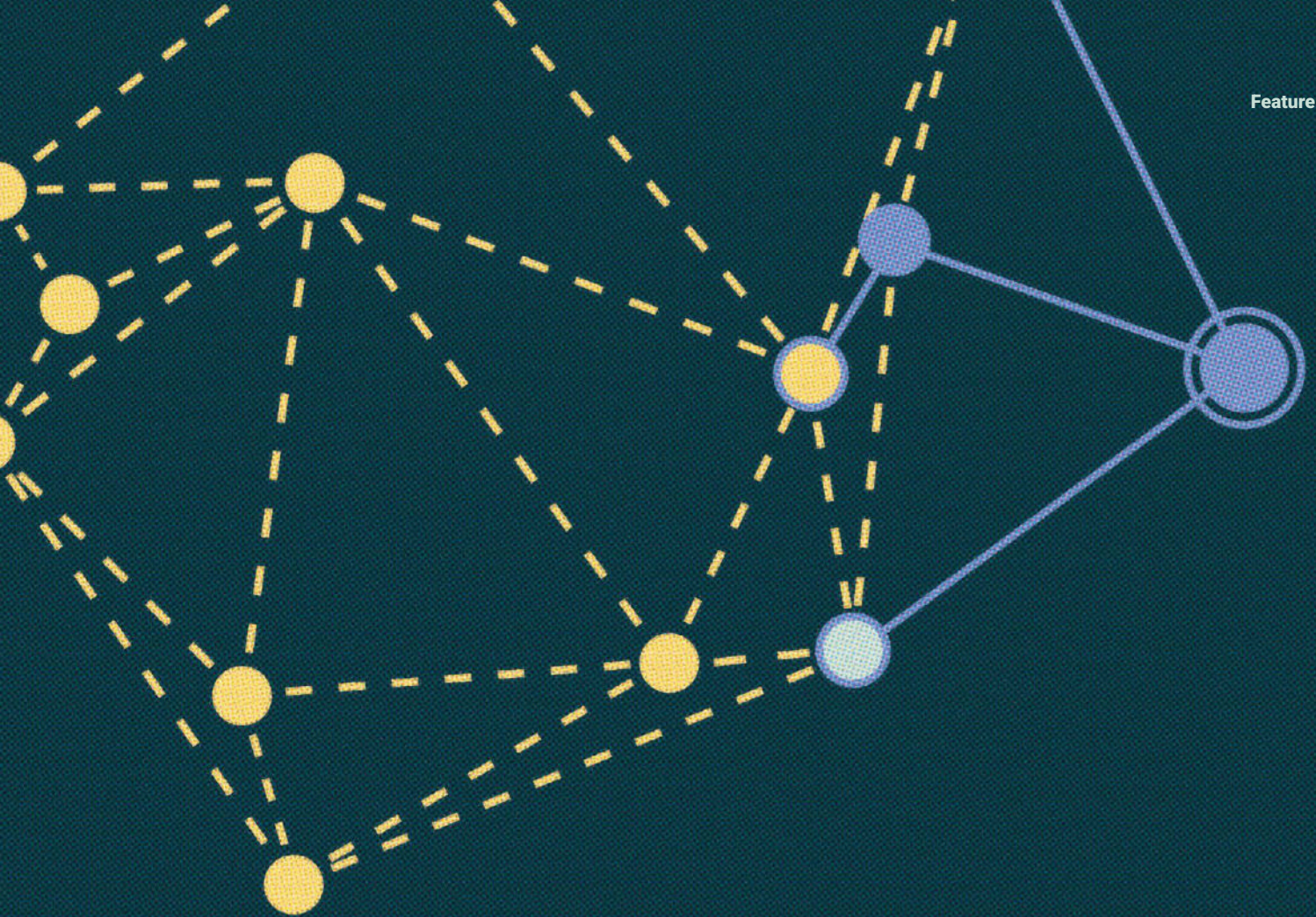


Coding for students:

Dijkstra Algorithm

*Find your way to treasure
using proper computer science.*

By Rob Miles



Using Dijkstra's algorithm to create routing data

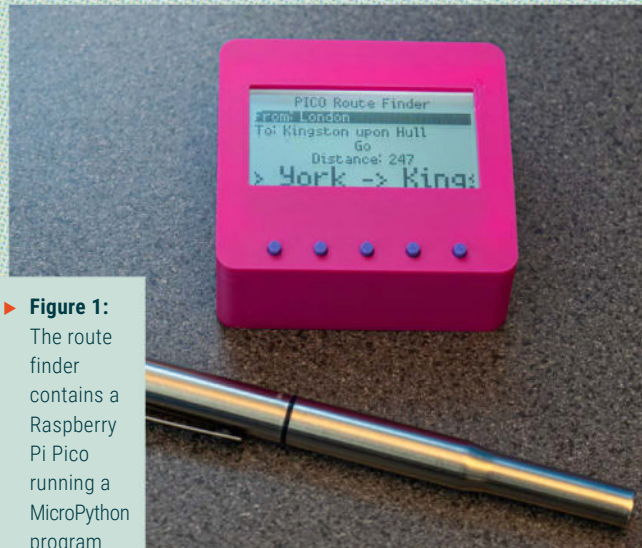
In the 1950s, Dutch computer scientist Edsger Dijkstra developed a route-finding algorithm so important that, 70 years later, it is still part of the computer science curriculum. In this article, we are going to consider what Dijkstra's algorithm does, how it works, and discover where it can be used. Along the way, we'll brush up on some Python programming techniques, find some treasure, and build a pocket route finder.

How does your satnav find your way home? How do networks decide which way to send data packets? How can a robot find its way around an obstacle? The answer to all these questions is by creating a graph of linked data nodes and then using something like Dijkstra's algorithm to create routing data from the graph.

**DOWNLOAD
THE FULL CODE:**



rpimag.co/dijkstracode



► **Figure 1:**
The route
finder
contains a
Raspberry
Pi Pico
running a
MicroPython
program

Dijkstra's algorithm

Figure 1 shows the Pico Route Finder. You can find all the instructions to build it at rpimag.co/piconav. It contains map information for locations in the UK and it will tell you the distance and the routes between them. Let's look at how Dijkstra's algorithm makes it work, starting with how the algorithm came about.

The story is that Dijkstra was having a coffee in Amsterdam and thinking about navigation. In the back of his mind, he was also wondering what he could write to demonstrate a new computer he was working on. After a few minutes of pondering, he came up with his algorithm, which boils down to three rules:

1. Keep track of how much it costs to go to places.
2. Always use the lowest-cost route first when exploring.
3. Keep track of where you have been.

Let's see how we can use these rules to find the cheapest route to some treasure. Suppose we are visiting a castle, and the owner tells us:

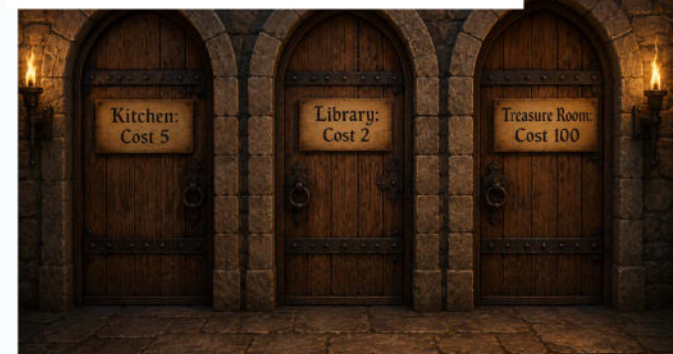
"I want you to chart a path from the Hall to my Treasure Room where I have placed a pile of gold coins. You can have all the coins in the Treasure Room minus the cost of getting there. Each door on your path has a cost. You can roam the castle as much as you like, but when you have finished you can meet me in the Treasure Room, tell me the path you have devised, and I'll give you your reward."

Scores on the doors

We look around the hall and see three labelled doors, as shown in **Figure 2**. The labels say 'Kitchen: Cost 5', 'Library: Cost 2', 'Treasure Room: Cost 100'. We could go straight to the Treasure Room, which would cost us 100 coins. Or we could try our luck finding a cheaper route. Fortunately, we are computer scientists familiar with Dijkstra's algorithm, so we open our notebook and draw a table. Each door tells us about a new room, and the cost of getting there from that room. We enter this information into our table, starting with the Hall, which is where we are now.

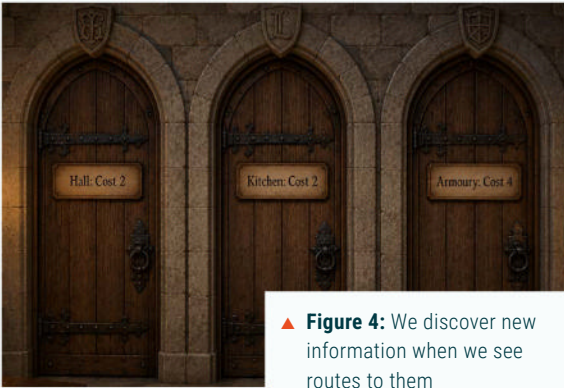
Figure 3 shows our first table. It describes rooms we know about, whether we have visited them, the cost of getting to each room from the Hall, and the route back to the start of our journey. Dijkstra's algorithm says that we should now find the lowest-cost unvisited room, travel to that room and update our table. So, we go to the Library.

▼ **Figure 2:** You would expect the path to the Treasure Room to be expensive



Room	Visited	Cost	Route Back
Hall	yes	0	
Library	no	2	Hall
Kitchen	no	5	Hall
Treasure Room	no	100	Hall

▲ **Figure 3:** We will update the values in the table as we learn more about the paths between rooms



▲ **Figure 4:** We discover new information when we see routes to them

Want gold, will travel

The doors in **Figure 4** tell us it costs 2 to reach the Kitchen from the Library and 4 to reach the Armoury. They also tell us that it costs 2 to reach the Hall, but we already knew that from the door in the Hall. We use these numbers to update the table. Let's start with the Armoury. It costs us 4 to get to the Armoury from the Library and it cost 2 to get to the Library from the Hall. So, the total cost of getting to the Armoury from the Hall is 6. We add the Armoury in our table and note that the route back to the Hall from the Armoury starts by going to the Library and then going on to the Hall.

Now we can update our table to save us some gold. Dijkstra's algorithm tells us to keep track of journey costs. If we find a new cost that is less than one in the table, we need to update the table. It cost us 2 to get to the Library and it would cost 2 more to get from the Library to the Kitchen, making a total of 4. The cost of going directly from the Hall to the Kitchen was initially entered as 5 (the cost on the door in the Hall). But if we travel from the Hall to the Kitchen via the Library, we can save a gold coin. We update the table with this new information.

Figure 5 shows our updated table with out-of-date information crossed out. Our next stop will be the unvisited room with the lowest cost, which is the Kitchen. We go into the Kitchen and repeat this process until we end up in the Treasure room.

▼ **Figure 5:** The cost of getting to the Kitchen has reduced

Room	Visited	Cost	Route Back
Hall	yes	0	
Library	yes	2	Hall
Kitchen	no	5 4	Hall Library
Treasure Room	no	100	Hall
Armoury	no	6	Library

Room	Visited	Cost	Route Back
Hall	yes	0	
Library	yes	2	Hall
Kitchen	yes	4	Hall Library
Treasure Room	yes	18	Hall Armoury Cellar
Armoury	yes	6	Library
Dining Hall	yes	8	Kitchen
Bedroom	yes	10	Dining Hall
Cellar	yes	7	Kitchen
Stables	no	22	Cellar

The final reckoning

Figure 6 shows the table created during our travels. It tells us all we need to know. The cheapest route from the Hall to the Treasure Room costs 18 coins. We use the Route Back column in the table to find the way back from the Treasure Room to the Hall. The first step on the way back to the Hall is to visit the Bedroom. Then we look up the Route Back value for Bedroom to find the next step, and so on. The complete route back is Bedroom > Dining Hall > Kitchen > Library > Hall. You can work through this in **Figure 6**. We reverse this path to create a route from the Hall to the Treasure Room.

▲ **Figure 6:** The path to the Treasure Room has got progressively cheaper as we have explored the castle

▼ **Figure 7:** The numbers on the links do not reflect distances; they are costs on the doors

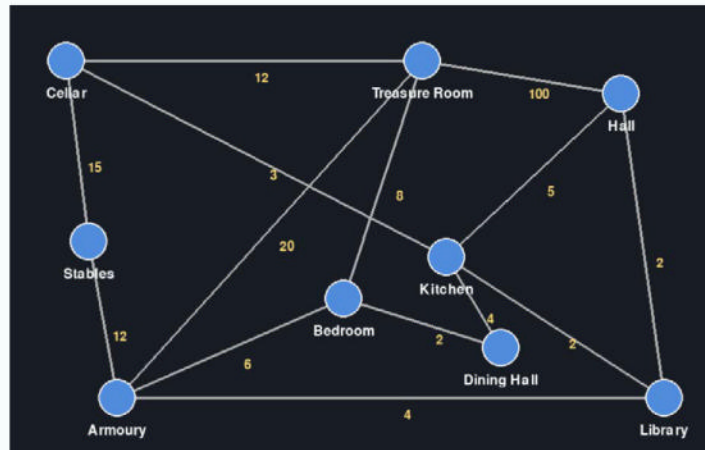
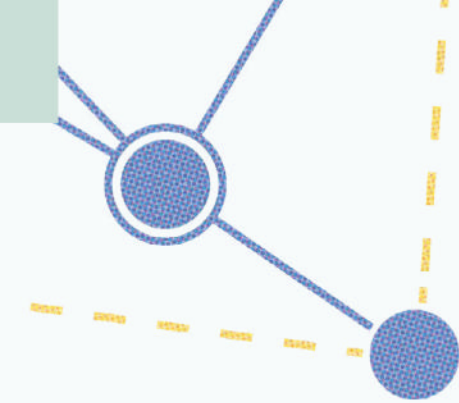


Figure 7 shows the map of the castle that we built on our travels. This is not complete. It only shows the rooms that we know about. If you want to prove that the algorithm works, you can work through the route-finding process to build the table shown in **Figure 6**.

QUICK TIP

If you wanted to build a complete map of the castle and find the cost of travelling to every room from the Hall, you could just continue visiting rooms until there were no unvisited rooms left.



Winning with Dijkstra

The cleverness of the algorithm shines through when you notice how it has ignored places that are not on the best route. We never visited the Stables because routes there were more expensive than cheaper routes that we checked first. At the time Dijkstra came up with his theorem, computers had very limited main memory. His route-finding demonstration was limited to 64 locations because his program used six data bits to hold each location number. A huge part of Dijkstra's initial triumph was fitting all the location information and the code into a very small amount of memory.

Writing Dijkstra

Rather than using a notebook to track costs, we can create our own cost-tracking program in Python. The first thing we need to do is create something to store each room:

```
class NodeRouteData:
    def __init__(self, name, cost, route_back):
        self.name = name
        self.visited = False
        self.cost = cost
        self.route_back = route_back
```

The code above creates a class called `NodeRouteData`. This stores all the information about a location in the castle. We're calling each location a 'node' because we might want to use this program to navigate things other than rooms in a castle. The class contains fields which map directly onto what we put in our notebook. When we create an instance of the class, we set the name, cost, and the route back from this node. Now let's ask the user where they are starting from and where they are going:

```
start_name = input("Start name: ")
end_name = input("End name: ")
```

The statements above read the names of the start and end points and store them in variables called `start_name` and `end_name`. Next, we need to make a node to represent the start location. We don't need to make a node for the end location; we use the value in `end_name` to detect when we have found a route to it.

```
start_node = NodeRouteData(start_name, 0, None)
```

The statement above creates a `NodeRouteData` called `start_node` with the name set to `start_name`. The first argument to the constructor of `NodeRouteData` is the name of the node. The second argument is the cost of going there. For the start node this is 0, because it is where we start. The third argument is the route back (i.e. the place we go to on the way back to the start). Since this is the start, we have no route back, so this value is set to `None`. Now we need to put the start node in the table. We can use a Python dictionary to hold the table, and the name of a node will be used as the key to find that node in the dictionary.

```
nodes = {}
nodes[start_name] = start_node
```

The statements above create an empty dictionary called `nodes` and add the start room to it. You add an item to a dictionary by specifying a key (the thing that will be used to find the item), as shown above. Now we can start checking the doors in `nodes` and updating costs. We use a variable called `current_node` to refer to the node we are working on. At the start of our exploration, the current node will be the start node:

```
current_node = start_node
```

The statement above creates `current_node` and sets it to `start_node`. Now let's kick off our loop:

```
while current_node.name != end_name:
```

The statement above starts a `while` loop which will cause the program to inspect nodes until it reaches one with the same name as the end of our route. Because the algorithm always chooses routes with the lowest cost, we will have found our cheapest route when the two names match. Inside the loop, we will do exactly what we did when we were physically exploring the castle. We look for doors and use the information on them to update our table. There might be more than one door, so we need to create a 'door reading' loop:

```
while True:
    name = input("Room name (empty to end): ")
    if name=="":
        break
```


The statements above create a loop which starts by asking for the name of a room from a door. If the name is an empty string, there are no more doors at this location, so the code uses a **break** to leave the door-reading loop. If the user has entered a name, the program asks for the cost of the route.

```
cost_string = input("Cost :")
cost = int(cost_string)
```

The statements above set the variable **cost_string** to the string that the user types in and then convert this string into an integer stored in the variable **cost**. Now we can use this new cost value to update our table. The first thing we do is work out the cost of getting to the room on the door from the start location. This will be the cost of getting to this room, plus the cost on the door:

```
total_cost = cost + current_node.cost
```

We now have the name of a node and the total cost of getting to that node. If this is a node we haven't seen before, we need to add it to the nodes. Let's do that.

```
if name not in nodes:
    nodes[name]=NodeRouteData(name,
    total_cost,current_node)
```

The code above checks the **nodes** dictionary to see if it contains an entry for the node with the name that was typed in. If the answer is no, the program adds a new node to the dictionary. We set the route back and the cost values for the new node.

If the node is already in the dictionary, we need to check if we have found a cheaper route to it. We can do this by adding an **else** part to the condition we just wrote. This code will run if the dictionary contains a node with the name on the door.

```
else:
    room=nodes[name]
    if total_cost < room.cost:
        room.cost = total_cost
        room.route_back = current_node
```

The code here gets a reference to the room from the dictionary and then compares the cost of using the newly discovered door with the stored cost of getting to that room. If the new route is cheaper, we update the room data. This is the programming equivalent of crossing out the cost and route back entries in our notebook and writing new ones.

Once all the door info from a room is known, the user will enter an empty room name and the room loop will end. Now we need to record that we've visited this room so we won't go there again:

```
current_node.visited=True
```

Now we have updated the route table, we need to decide where to go next. We need to find the next cheapest route in the table and go to where that leads:

```
cheapest_node = None

for node in nodes.values():
    if node.visited:
        continue
    if cheapest_node == None or node.cost <
    cheapest_node.cost:
        cheapest_node=node
```

This code sets the value of **cheapest_node** to the node in the **nodes** which has not been visited and has the lowest cost. Now we can switch to it:

```
print(f"The next node is {cheapest_node.name}")
current_node = cheapest_node
```

If the new route is
cheaper, we update
the room data

► **Figure 8:** A map of the data in the `map_graph.json` file. The example code contains a Python program that draws this map and animates the route-finding process

These two statements tell the user where to go next and then set the value of `current_node` to the cheapest one. Note that the next node might not be through one of the doors which have just been discovered. At some point the loop will stop because the `cheapest_node` becomes the one with the end name and the `while` loop will exit. This is where we tell the user the cost of the route and describe it to them. Displaying the cost of the route is the simpler of the two:

```
print(f"Cost: {current_node.cost}")
```

The search loop stops when the `current_node` refers to the node with the name of the `end_node`. So we just print out the cost of getting to the current node. This will print a cost of 18 coins. Now we build a route by working through all the `route_back` values of nodes in the route:

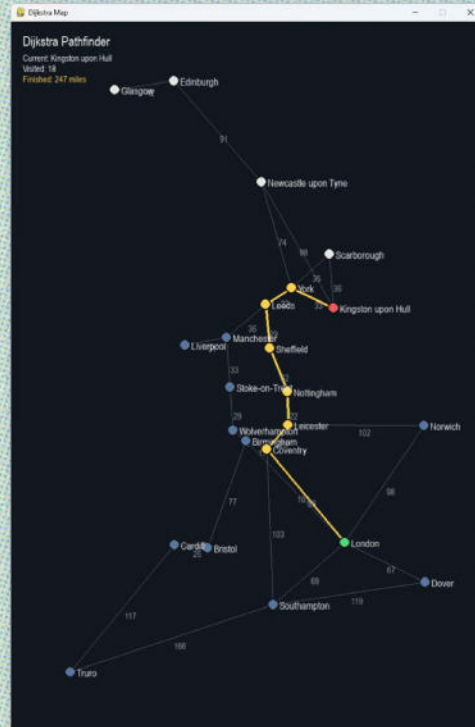
```
route = []

while current_node != None:
    route.append(current_node.name)
    current_node = current_node.route_back
```

The code above uses a loop to build a list called `route`. It starts at the end node and works back to the start. The `start_node` was created with a `route_back` value of `None`, so the loop will stop when it reaches the start. The loop adds the name of each node to the `route` list.

We now have a list of node names, but they are in the wrong order. We need to reverse them before printing them.

```
route.reverse()
print("Route:", " -> ".join(route))
```



The loop will stop when it reaches the start

The two statements here reverse the route and then print out the names in the route, separated by “ -> ” strings. The rather wonderful `join` method works on a string, takes the elements in a collection, and inserts the string between each item in the collection.

```
Hall -> Library -> Kitchen -> Dining Hall ->
Bedroom -> Treasure Room
```

The statement above is the result of a successful run. The sample code for this article contains an implementation of the algorithm. If you’ve got this far, you are allowed to take a deep breath and feel very pleased with yourself. You now know how to implement Dijkstra’s algorithm. You also know more about Python classes, dictionaries, and references.

map_graph.json

DOWNLOAD
THE FULL CODE:



rpimag.co/piconav

> Language: JSON

REFER
TO PAGE
98
FOR THE
FULL CODE

```
001. {
002.   "cities": {
003.     "Birmingham": {
004.       "latitude": 52.4862,
005.       "longitude": -1.8904,
006.       "links": [
007.         {
008.           "destination": "Bristol",
009.           "cost": 77
010.         },
011.         {
012.           "destination": "Coventry",
013.           "cost": 17
014.         },

```

Map navigation

We can modify the castle navigation program to plot routes in the UK. We need a graph that describes routes between locations in the UK. The word graph is used in lots of different contexts. The most familiar one is probably the Cartesian graph of one value against another (for example price against time). For routing we use a node graph. This describes places (nodes) and routes between them. Here is one expressed as a JSON file.

The code in the `map_graph.json` listing shows the first part of the graph. It contains a dictionary of cities indexed by name. Cities have a geographical location and a list of links and costs (distances) to neighbouring cities. A modified version of the castle navigation program uses this node graph to find the shortest route between two cities. The Pico Route Finder uses the same code to create routes and then display them on the LCD panel.

QUICK TIP

If we want to work on a different problem, we just change the node graph.

Dijkstra forever

Dijkstra's algorithm works by finding all the lowest-cost routes in order and stopping when it finds the required destination. When navigating from London to Hull, it found a route to Truro (a long way away from both London and Hull) before it found the route to Hull. For this reason, pure Dijkstra is not used in modern navigation, which tries to reduce the amount of route searching by making a judgement on whether a proposed route is moving towards or away from the destination. But there are many occasions where it is still the best way to create a route and it will always be part of how computer systems find their way around. 🟢

castle_nav.py

DOWNLOAD
THE FULL CODE:



rpimag.co/dijkstracode

> Language: Python

```
001. class NodeRouteData:
002.     def __init__(self, name, cost, route_back):
003.         self.name = name
004.         self.visited = False
005.         self.cost = cost
006.         self.route_back = route_back
007.
008. start_name = input("Start name : ")
009. end_name = input("End name: ")
010.
011. start_node = NodeRouteData(start_name, 0, None)
012. nodes = {}
013. nodes[start_name] = start_node
014.
015. current_node = start_node
016.
017. while current_node.name != end_name:
018.     # Read in route
019.
020.     while True:
021.         name = input(
022.             "Room name (empty to end): ")
023.         if name=="":
024.             break
025.
026.         cost_string = input("Cost :")
027.         cost = int(cost_string)
028.
029.         total_cost = cost + current_node.cost
030.         if name not in nodes:
031.             # we have a new room - create it
032.             nodes[name]=NodeRouteData(
033.                 name, total_cost, current_node)
034.         else:
035.             # Room exists
036.             # Does this door provide a
037.             # cheaper route?
038.
039.             node=nodes[name]
040.             if total_cost<node.cost:
041.                 # Yay! Cheaper route
042.                 node.cost=total_cost
043.                 node.route_back=current_node
044.
045.             current_node.visited=True
046.
047.             # Find the cheapest room to go to next
048.
049.             cheapest_node = None
050.
051.             for node in nodes.values():
052.                 if node.visited:
053.                     continue
054.                 if cheapest_node == None or
055.                     node.cost < cheapest_node.cost:
056.                     cheapest_node=node
057.
058.             print(f"The next node to visit is
059.                 {cheapest_node.name}")
060.
061.             current_node = cheapest_node
062.
063.             # When we get here the route has been
064.             # discovered
065.
066.             print(f"Cost: {current_node.cost}")
067.
068.             route = []
069.
070.             while current_node != None:
071.                 route.append(current_node.name)
072.                 current_node = current_node.route_back
073.
074.             route.reverse()
075.             print("Route:", " -> ".join(route))
```

REACH A GLOBAL COMMUNITY

Advertise in **Raspberry Pi Official Magazine**

Our readers are passionate about technology and the Raspberry Pi ecosystem. From DIY enthusiasts to professional engineers.

Your advertisement can sit alongside our cutting-edge tutorials, features, and reviews. And we have flexible advertising options for a range of different businesses.

*Take the next step –
advertise with us today!*



Email Charlie Milligan at: charlotte.milligan@raspberrypi.com

ONLY THE **BEST**

Touchscreen displays

Add touch control to your projects with this selection of screens

By **Phil King**

Most of us use a touchscreen every day, in the form of a smartphone or tablet, and are very accustomed to tapping and swiping to interact with a device. So why not add touch functionality to Raspberry Pi projects? There's a wide range of options to choose from, featuring different touch technologies, display types, and connection methods.

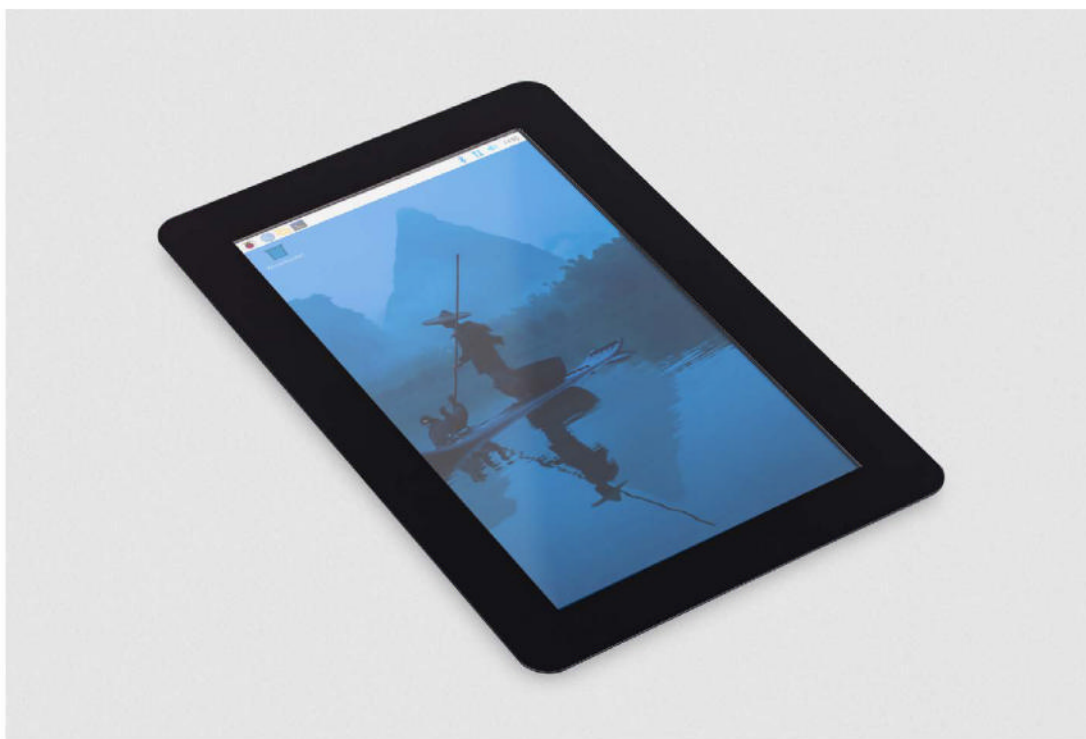
The official Raspberry Pi Touch Display 2 connects to a Raspberry Pi single-board computer's DSI display port and is autodetected by Raspberry Pi OS. Alternatively, you could use a touchscreen HAT mounted on the GPIO header, or a display with a standard HDMI video port. There are also touchscreens that work with Raspberry Pi Pico.

Size-wise, touch displays range from 2.1 inches (diagonal) up to 10.1 inches or more for HDMI screens. Most feature capacitive touch functionality, just like on your phone, while a few use the older-style resistive touch technology.

We take a look at a few different touchscreen options. Swipe right to continue...

Touch Display 2

Raspberry Pi | From £36 / \$40 | raspberrypi.com



◀ Available in two sizes, the Touch Display 2 supports up to five simultaneous touch points

An upgraded version of the original Raspberry Pi Touch Display, it features an improved resolution of 1280 × 720, compared to the original's 800 × 480. This is the case for both versions of the new screen, measuring 5" and 7" across the diagonal, so you do get greater pixel density on the smaller variant.

As before, the display connects to a Raspberry Pi computer – any standard-size SBC-type model from the B+ onwards – via the latter's 'DISPLAY' MIPI port using either of the two supplied (different-sized) FFC ribbon cables to carry video and touch

data. It's powered by a couple of wires connected to the GPIO header's 5V and GND pins. You can even mount Raspberry Pi on the rear of the display.

The screen is autodetected by Raspberry Pi OS and works in portrait mode by default, but can easily be switched to landscape in the Control Centre (or on the command line). An on-screen keyboard appears when you need to type anything.

We couldn't find a commercially available case for the 5" model, but you could always 3D-print your own (e.g. rpimag.co/tdcase3dprint).

Verdict

A high-quality touchscreen that works seamlessly with Raspberry Pi OS.

Meteor 10.1"

Elecrow | £80 / \$90 | elecrow.com

While many displays for Raspberry Pi computers are HATs, others use an HDMI port to connect. The obvious advantage is that this frees up the GPIO pins for other uses. The Meteor 10.1" even boasts a full-size HDMI port, just like on a TV and can be used with other devices if you want.

Measuring 10.1 inches across its diagonal, it features a 1200 × 800 IPS display with a 143ppi pixel density – not the highest, but it looks fine even when playing video. Note that there's no built-in speaker, however, so you'll need to output audio another way.

To enable five-point capacitive touch functionality and power, micro-USB ports on the display are connected via supplied cables to Raspberry Pi's USB ports. As with the Touch Display 2, you can mount Raspberry Pi on the rear of the screen. A plastic stand is also supplied.

One extra feature is RGB marquee lighting, although this can prove distracting (luckily, you can turn it off).

- ▶ With its standard HDMI port, this display can also be used with other devices



Verdict

A large HDMI touchscreen that works well with Raspberry Pi.

HyperPixel Round 2.1

Pimoroni | £54 / \$60 | pimoroni.com

We reviewed this one back in issue 109 and it's still an impressive little circular display with a capacitive touchscreen.

Like its HyperPixel 4.0 sibling, it uses a high-speed DPI interface that enables it to shift five times more pixel data than on a standard SPI-based screen. The colour is vibrant, while the resolution is 480 × 480 minus some pixels where the corners have been rounded off.

A Raspberry Pi Zero fits neatly on the rear and while all the GPIO pins are used by the display, there is a breakout header with five pins for I2C. After installing drivers with terminal commands, rebooting sees the Raspberry Pi desktop appear on the

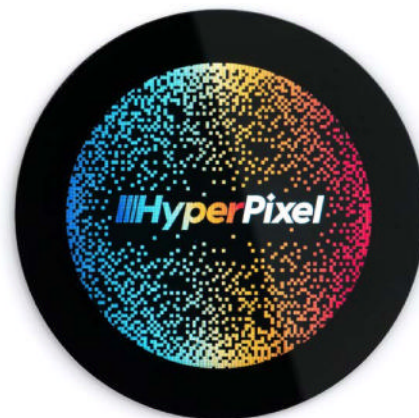
tiny display. It's too small to be practical, so you'll need to rely on a remote SSH connection to issue further commands and install the Python touch library.

Note that due to changes to the Linux kernel drivers, the touch functionality only works with a legacy (pre-Buster) version of Raspberry Pi OS. Pimoroni is working on a solution, however.

Verdict

A stunning circular display with the option of adding touch control.

- ▶ A Raspberry Pi Zero mounts neatly on the rear of this mini circular display



2.13" Touchscreen ePaper Display + Case for Raspberry Pi Zero

Waveshare / The Pi Hut | £22 / \$29 | waveshare.com / thepihut.com

Here, you get all the advantages of e-paper (such as ultra-low power drain and paper-like readability) combined with the option for touch control.

The biggest downside of e-paper is that the screen takes a while to refresh when it changes – about two seconds in this case, although a partial screen refresh can take as little as 0.3 seconds. So it's not suitable for fast-changing graphics/text, but fine for a smart home or weather dashboard.

Protected by toughened glass, the 250 × 122 monochrome e-paper display here

supports up to five-point capacitive multitouch. To use it, just mount your Raspberry Pi Zero onto the display's rear header in the upper part of the black ABS case, then clip it into the lower part. Some thermal tape is supplied to aid heat dissipation. The SD card slot and ports remain accessible.

The documentation includes C/C++ and Python demos, the latter making use of the Pillow library to show images, draw shapes, and print text.



Verdict

This small monochrome display combines e-paper and touch control.

▲ An e-paper display that's also a touchscreen

RP2350 1.43" AMOLED Round Display Dev Board

Waveshare / The Pi Hut | £23 / \$31 | waveshare.com / thepihut.com



▲ Measuring 48 × 48mm, this round touchscreen is driven by a Raspberry Pi RP2350A (as in Pico 2)

This circular device is part of Waveshare's wide range of RP2350-based touch displays which come in various shapes and sizes.

This one has an optional metal case with cut-outs for the boot and reset buttons, USB-C port, microSD card slot (for extra storage), I2C and UART four-pin headers, and GPIO breakouts on the rear.

The vibrant colour AMOLED screen has a 466 × 466 resolution and capacitive touch functionality. In addition, it features a built-in six-axis IMU (inertial

measurement unit) and real-time clock. Portable power is possible via the RTC's battery header or the MX1.25 PicoBlade-compatible lithium battery header with charge circuit. So it could be used for a smartwatch or fitness tracker project.

The MicroPython firmware has only a few code examples, but the C/C++ firmware is far more comprehensive, including the use of LVGL (Light and Versatile Graphics Library) to render text, images, and an example GUI complete with animated graphs and pop-up tiny keyboard.

Verdict

A cool RP2350-based touch display with a built-in IMU and RTC.

3.5" Touchscreen IPS LCD Display for Raspberry Pi Pico

Waveshare / The Pi Hut | £19 / \$25 | waveshare.com / thepihut.com

Yes, you can even get a touchscreen display for your Raspberry Pi Pico, which could prove handy for portable projects. Waveshare's 3.5-inch screen has a 480 × 320 resolution and features twin female headers on the rear to plug in your Pico – a lot easier and neater than wiring up a generic LCD manually to its GPIO pins.

One thing to note is that this is a resistive touchscreen rather than the usual capacitive type. This means that you have to press it a little harder for your touches to register, since they need to push the flexible surface layer into the one just beneath it to make contact. It works best with a plastic stylus (one is supplied) to aid accuracy.

Waveshare's documentation includes some C/C++ code snippets with explanations. The MicroPython library/demo may prove more useful, enabling you to register touches, draw shapes, print text, control the backlight brightness, and rotate the display orientation. ▣



▲ Just plug your Pico into the headers on the rear of this colour touchscreen

Verdict

Good value for money if you don't mind a resistive touchscreen.

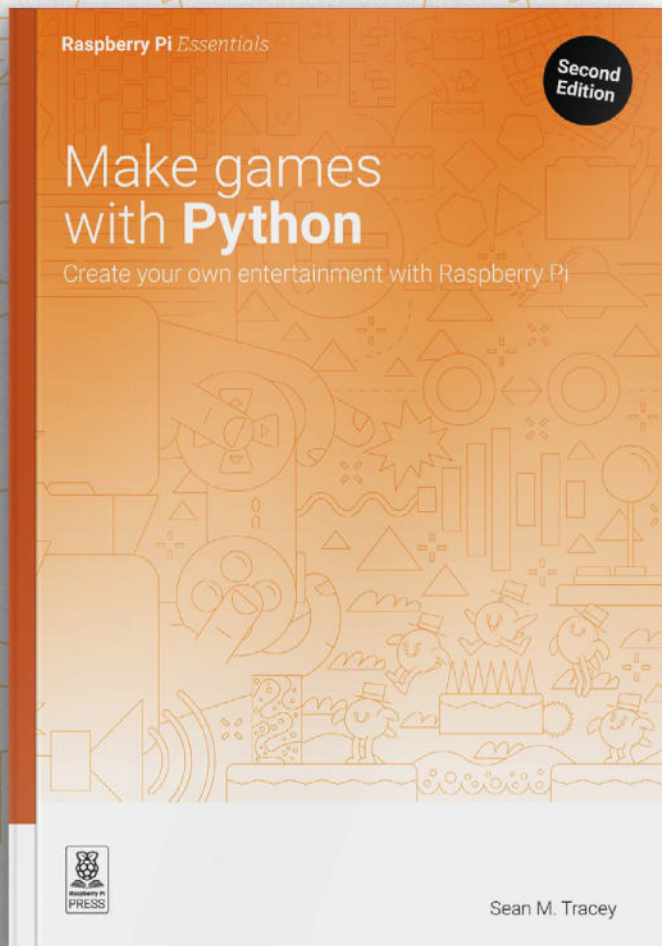
PRESTO

Pimoroni | From £69 / \$76 | rpimag.co/presto

Built into an aluminium stand, this RP2350B-powered 4-inch capacitive touchscreen is billed as a 'desktop companion'. Features include Wi-Fi, Bluetooth, battery connector, and a Qwiic/STEMMA QT port for breakouts. For more details, read the full review in this issue.

- This touchscreen has its own stand and can be programmed via USB-C





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

Pimoroni Presto

Is this the new pinnacle of graphic coding, or just a fancy screen in a metal case? By **Ed Cox, John Lynn, and Matan Naamati**

 Pimoroni  pimoroni.co/presto  £69 / \$76 (Presto only); £99 / \$110 (starter kit)

SPECS

FEATURES:

4-inch (480×480 pixel) IPS LCD screen with capacitive touch overlay, ambient lighting via seven mini SK6812 (NeoPixel-compatible) RGB LEDs, piezo speaker, reset and boot buttons, black anodised aluminium stand with rubber feet; powered by RP2350B (dual Arm Cortex-M33 running at up to 150MHz with 520kB SRAM)

CONNECTIVITY:

Raspberry Pi RM2 module (CYW43439), supporting IEEE 802.11 b/g/n wireless LAN and Bluetooth, USB-C connector for programming and power, Qw/ST (Qwiic/STEMMA QT) connector for attaching breakouts, 3-pin debug connector (JST-SH), 2-pin JST-PH connector for adding a battery (3V–5.5V), microSD card slot

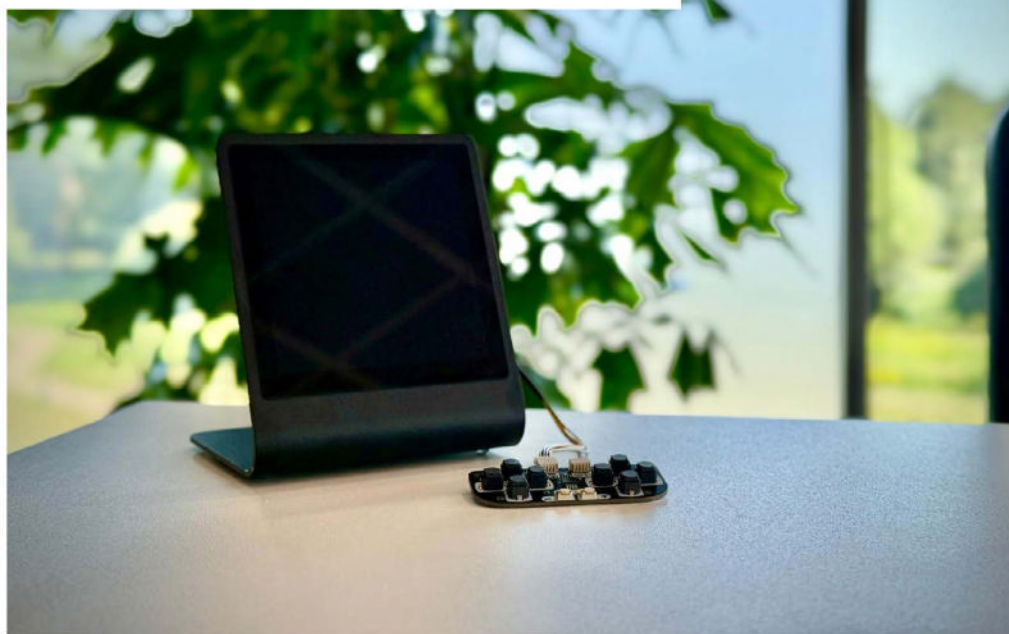
STORAGE:

16MB QSPI flash supporting XiP, 8MB PSRAM

Pimoroni Presto is an RP2350-powered 4-inch touchscreen implemented into a small aluminium stand, with embedded RGB ambient lighting – allowing for a wide variety of programming projects.

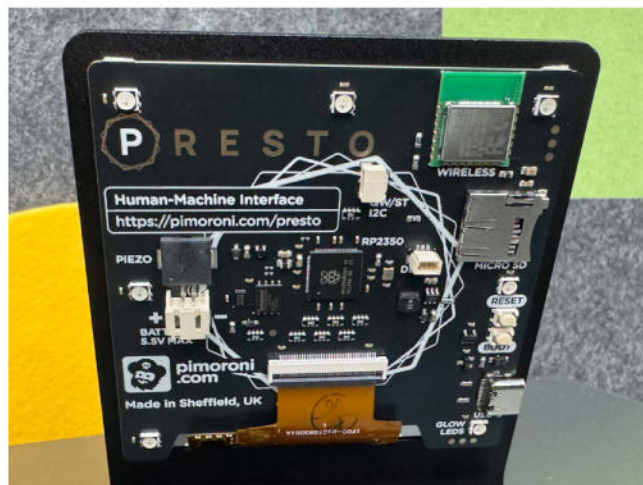
Three Raspberry Pi work experience students took it for a spin, trying to code some retro-style games. Their thoughts are here...

▼ Presto in front of a garden plant





▲ A game we coded onto our Presto



Functions and features

The Presto comes fully assembled, and pre-installed with a wide range of different apps and games – each one showcasing a unique aspect of the device’s software and hardware. However, to use some of the apps, additional hardware, such as controllers or multi-sensor sticks, is required (as included in the starter kit).

Presto features a 4-inch touchscreen with 480 × 480 pixels – allowing the rendering of games and graphics with an enticingly interactive element. You can program Presto with C, C++, or MicroPython to create your own games and apps. As an added bonus, the IPS LCD screen provides bright and dazzling, colours for your programs; this can make coding feel incredibly rewarding due to the vibrant visuals.

The Presto comes with an incredible feature which automatically changes the colour of the rear RGB LEDs, based on what’s showing on the screen – providing ambient lighting and immersing you in the full Presto experience. However, a downside to the vibrant display is that sometimes the colours can be too bright, which one of the testers found to be uncomfortable.

Presto is so magical to use, you just click run, and presto, the screen lights up

- ▲ The rear features buttons and ports
- ▼ The infinite maze game on Presto (requires controller)

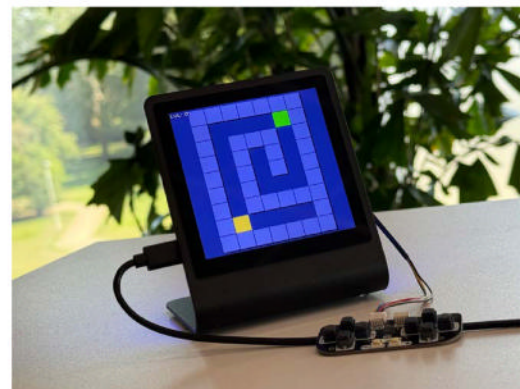
Setup process

The Pimoroni Presto provides a clean, predictable setup that suits a beginner-friendly development environment. Each step of getting code onto the device is simple, and the process feels intentionally streamlined.

Connecting the Presto to a computer via USB-C is immediate, with automatic recognition that removes friction for new users. Presto is so magical to use, you just click run, and presto, the screen lights up.

Thonny remains a practical option for MicroPython work. Switching the interpreter to ‘MicroPython (Raspberry Pi Pico)’ is quick, and the interface keeps everything clear. Stopping the automatically running program on the Presto only requires pressing a button, and it works reliably.

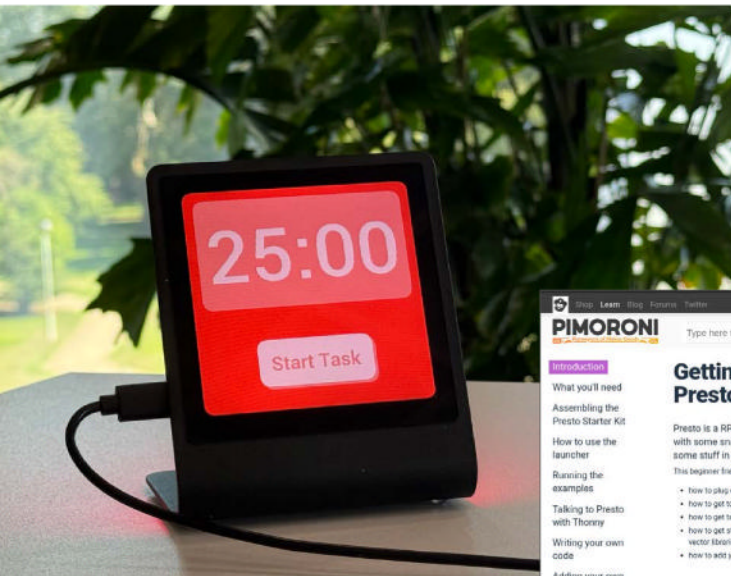
Importing a program into Thonny is straightforward, and running it on the Presto is responsive and consistent. The overall process is efficient, avoids unnecessary steps, and supports quick



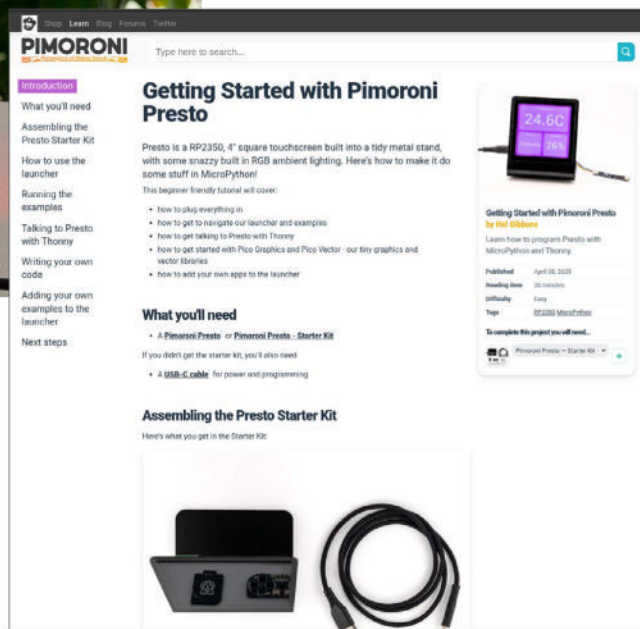
Verdict – Ed

Plenty of useful online documentation: the GitHub repo provides many resources to support learning and running Pimoroni Presto. Official documentation is sufficient, but a trouble-shooting section would be a great addition. As the community continues to grow, so will the support.

7/10



- ◀ The Pomodoro timer app on Presto
- ▶ The official Presto learning portal website



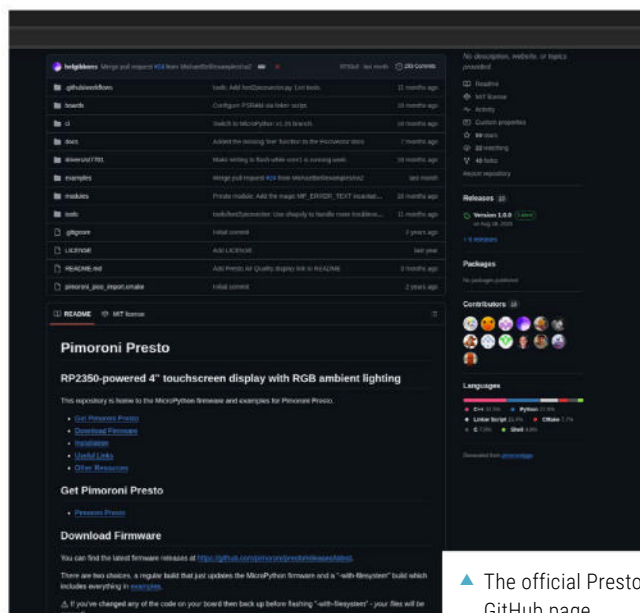
experimentation, making it suitable for beginners and for those who prefer minimal setup.

Documentation and support

The documentation for the Pimoroni Presto is beginner-friendly and accessible. It provides a step-by-step guide on how to set up and run the Presto, a list of the various components and their functions, as well as providing insight into their MicroPython libraries.

The GitHub repo ([rpimag.co/prestogh](https://github.com/pimoroni/prestogh)) is informative, with links to many other documents on the Presto, along with other useful links such as the Pico Graphics and PicoVector library docs. It also contains a small, less detailed but nonetheless effective tutorial on how to run and connect the device to Thonny. Pimoroni also provides releases of the firmware, with many versions to choose from.

Support is community dispensed rather than officially provided. Users share guidance and knowledge of bugs they have encountered, with plenty of examples to help understand errors, but detailed trouble-shooting is somewhat limited, especially for complex or project-specific problems. 🐛



- ▶ The official Presto GitHub page

Verdict – Matan

The Presto is fun to play with and very rewarding to code on, and its touchscreen makes it more user-interactive; however, I found it slightly uncomfortable to use (especially when showing brighter colours).

8/10

Verdict – John

A sleek display featuring a touchscreen and optional Qw/ST game controller integration, perfect for those making their first foray into games development.

9/10



The official Raspberry Pi Handbook 2026

200 PAGES OF RASPBERRY PI

Get started guide

covering every
Raspberry Pi

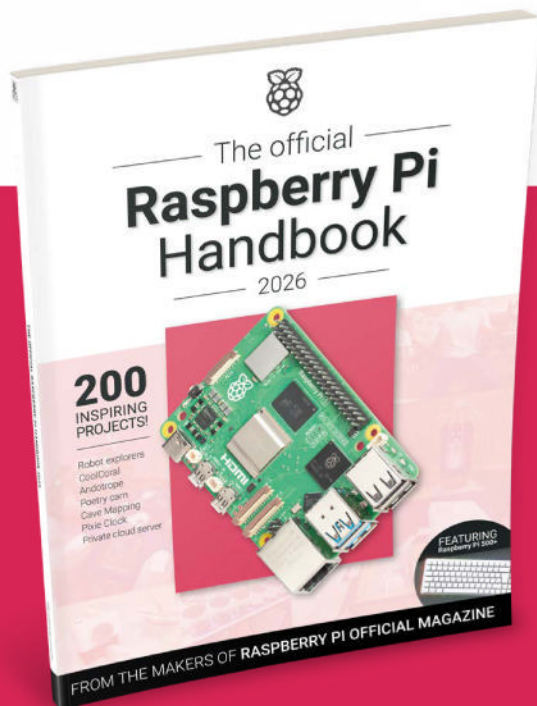
Everything you need to
know about the brand
new Raspberry Pi 500+

Inspiring projects to
give you your next
project idea

Build a Raspberry Pi
5-powered media
player

Explore the world
around you with
roving robots

Play retro horror
games on Raspberry
Pi 5



Buy online: rpimag.co/handbook2026

CrowView Note 15.6

Turn your Raspberry Pi into a large laptop with a 15.6-inch screen. By **Phil King**

Elecrow

rpimag.co/crowviewnote15-6

£160/\$167

SPECS

DISPLAY:

15.6-inch IPS, 1920 × 1080 pixels, 60Hz, 300 nits

FEATURES:

101-key keyboard, touchpad, 2 × mini speakers, microphone, 6000mAh battery, PSU (PD 5V 3A / 9V 3A / 12V 3A / 15V 3A)

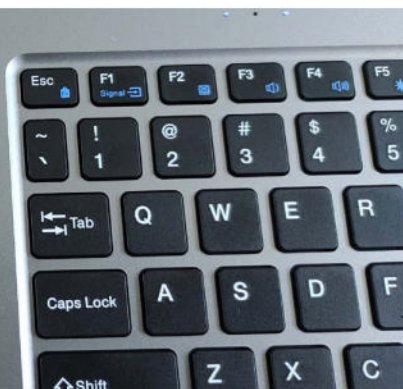
CONNECTIONS:

Raspberry Pi 5 bridge board (optional), USB-C PD power output (5V 5A), mini HDMI, 2 × USB-A, USB-C PD power input, USB-C full-featured, 3.5mm audio

- ▼ The expanded keyboard has 101 keys, including a numerical keypad, although the other keys are shifted to the left



- ▼ The optional bridge board back-powers Raspberry Pi 5 and delivers video, audio, and keyboard/touchpad functionality



We reviewed the original 14-inch CrowView Note a while back, in issue 146 (rpimag.co/146), but there's now a new model available with a larger 15.6-inch screen, extended keyboard, and a few other improvements.

As before, you can attach a Raspberry Pi 5 to the left-hand side of the device via an optional (\$7) bridge board. As well as back-powering Raspberry Pi, this routes its video output to the Note's screen, and audio to mini speakers. An extra USB extender from one of Raspberry Pi's USB 3.0 ports enables the keyboard and touchpad functionality. It's a neat plug-

and-play solution, even if it does stick out quite a bit from the side. There's no room underneath for a Raspberry Pi case, either, although you can still attach an Active Cooler or heatsink.

You could also connect any other Raspberry Pi model, powered separately, to the Note using USB and HDMI cables (the Note has a mini HDMI port).

On the big screen

The most appealing feature of the new model is its large 15.6-inch display. While offering the same 1920 × 1080 resolution as the 14-inch model, which some may find disappointing, we found it to be





◀ The CrowView Note 15.6 showing the Raspberry Pi OS desktop on its large display

sharp and detailed with vivid colours and good brightness (300 nits) on the default settings – you can adjust them using an on-screen menu.

The main chassis lacks a premium feel, being ABS plastic, but there's now room for a full keyboard layout with 101 keys, including a numerical keypad on the right. The downside is that the other keys are shifted to the left, which takes a bit of getting used to. The typing action feels fine, however, and there's a full row of function keys which double as system / media controls.

The touchpad is large and responsive, although we found the desktop cursor speed a little slow until we upped the Mouse Speed setting in Raspberry Pi OS's Control Centre. Only the middle/lower section can be used for left- and right-click. As well as pinch/spread to zoom, two fingers are used for scrolling; three to switch applications in Raspberry Pi OS.

Listen up

For audio output, there are two mini speakers on the underside. As well as lacking bass, they're a little quiet by default (even with the Note's own system volume turned up) and we needed to turn the volume up to 100 in Raspberry Pi OS to achieve a reasonable loudness.

There's now room for a full keyboard layout with 101 keys, including a numerical keypad

You may prefer to use the 3.5mm headphone jack on the right-hand side, which is a nice bonus. There's also a built-in microphone, but no webcam, although you could easily add a USB one (or use a Camera Module attached to Raspberry Pi).

The device feels fairly hefty, weighing just over 1.5kg, part of that being down to the inclusion of a larger 6000mAh battery. According to Elecrow, this will typically last 2–3 hours, depending on the screen brightness and Raspberry Pi workload. You can check the rough battery level using a function button. It can be recharged by connecting the supplied 65W PSU to a USB-C port on the right-hand side.

The CrowView Note 15.6 is a convenient solution for plugging your Raspberry Pi 5 into a laptop keyboard and screen while out and about. There's also a full-featured USB-C port on the right for connecting a smartphone or other device, so you can use it as a standard lapdock. 🍷



▲ Ports on the right-hand side include the USB-C power input, USB-A port, 3.5mm audio jack, and a full-featured USB-C port for connecting a smartphone

Verdict

A handy plug-and-play solution that doubles as a standard lapdock.

8/10



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/poweredbypiaply

From cool tech to keep you hydrated to calming screens that don't strain your eyes, the Powered by Pi badge of quality marks adorns hundreds of excellent items. By **Rosie Hattersley**

Keeping cool in a heatwave just got serious for those of us in usually temperate Europe.

It therefore seems like the perfect time to mention Green Custard's innovative and refreshing smart water dispenser and suggest you wave this page under the nose of whoever's in charge of ensuring staff keep their cool at your office. If a dunk under the waves is more on your mind, check out BlueROV's incredible submersible explorers, or chill out in a shady spot without annoying screen glare with one of InkyCal's e-paper screens. Get creative with a handcrafted waist-level camera, or show off your marketable wares on a custom Compute Module 4-based display.

This month's line-up demonstrates the sheer range and variety of Raspberry Pi-touting products that have achieved Powered by Pi accreditation and mark themselves out as guaranteed to be exceptional quality and beautifully built. Of course, choosing Raspberry Pi as the basis of a product design is a smart move anyway, but the hundreds of products that now sport the Powered by Pi badge have been subjected to additional rigorous testing and standards controls, making them best in class and an obvious first choice when looking for a new piece of technology.

For details of the testing procedures involved in gaining Powered by Pi accreditation, see pip.raspberrypi.com.

Aqua Libra Flavour Tap

UK | rpimag.co/aqualibra

With significant heatwaves across Europe, it seems a good time to highlight this excellent and stylish water dispenser designed by Green Custard on behalf of Aqua Libra and its parent company, Britvic. Raspberry Pi Pico checks the weight and fluid flow to ensure cool still, sparkling, or flavoured water is precisely delivered into the glass or bottle placed on the dispenser plate. Pico also drives the stepper motors and valves used in the dispensing process. Using a

hardware-in-the-loop testing scenario, Green Custard stress-tested the design, checking for bugs and ensuring every element of the Amazon Web Services (AWS)-based drinks delivery mechanism runs flawlessly. This covers selecting the correct valve to deliver still or sparkling water and dispensing from the chosen flavour cartridge, delivering the correct volume and flavour strength, and shutting off the valve once a portion has been poured. The refreshing drinks dispenser launched successfully at the end of 2024.



FELWY Home Unit 1st Edition Plus

Spain | rpimag.co/felwy-home



Madrid-based home security startup FELWY (Friendly Eyes Looking With You) offers round-the-clock audio and video monitoring and WhatsApp notifications. With Raspberry Pi 5 at its core, the hardware provides motion detection alerts and a live camera feed to people in a trusted network of friends, family, and neighbours via WhatsApp. Data is locally stored, so there's no danger of your home security network being hacked into by malign forces either.

The system depends on advanced machine learning processes running on Raspberry Pi 5 to recognise whether there's an intruder or simply a pet making its way across the room. In fact, one of the important marketing aspects is that home owners will not be bothered by domestic animal-based false alerts, while the offline data storage and 'social' approach, rather than calling out the police or the fire brigade, will also appeal.

Blue Robotics BlueROV2

USA | rpimag.co/rov-bluerov2



BlueROV2 autonomous underwater exploration vehicles are remotely operated and are sufficiently robust and watertight that they can be used at depths of 300 metres, with thrusters offering controlled movement with up to six degrees of freedom. A Navigator Flight Controller HAT specifically designed for the ROV sits atop a Raspberry Pi 4 that provides the vehicle's 'brains' and runs bespoke open-source BlueOS ArduSub vehicle control software. Six- and eight-

thruster models are available, depending on customer requirements, while the forward-facing underwater camera can be tilted and provides 30fps video at 1080p. Dimmable lights on the BlueROV's exterior offer up to 6000 lumens, greatly aiding ocean-based exploration. The modular design allows for customisations including sonar, additional thrusters, grippers, and lights. The surface-based 'captain' uses a gamepad-like controller to navigate, aided by viewing live sensor and video data.

WLV-01 Monochrome Waist-Level Camera

Canada | rpimag.co/waistlevel-camera

Holding up a smartphone or SLR camera to shoot comes between you and whatever you'd like to photograph; a waist-level camera offers discretion, so you can snap photos without disturbing your subject. More relaxed, natural photos often result. Malcolm-Jay's Raspberry Pi 5-based waist-level camera offers clean, high ISO performance, even for near-infrared low light conditions and night photography. The Winnipeg, Manitoba-based camera expert created the WLV-01 from a Sony IMX585 sensor and E-mount along with Raspberry Pi 5 running custom camera software that he wrote and maintains himself. Eschewing an eyepiece in favour of a more natural composition technique and a twin lens reflex (TLR) posture, Malcolm provides a waist-height touchscreen viewfinder with a green overlay that shows peak focus. Tapping the touchscreen initiates the 2× to 4× zoom,



and there's also an option to set up 5, 6, or 9-shot Brenizer image stitching. He sells the modular camera ready to shoot, or as a kit, making it eminently attractive to keen photographers who want full control over their photos, and makers keen to try something different using Raspberry Pi and its associated camera modules and lenses.

InkyCal v3

Germany | rpimag.co/inkycalv3



Currently sold via Tindie but with plans afoot for other sales channels, Stuttgart-based Ace Innolab struck a chord with its original e-paper dashboard back in 2018 and has just launched version 3. The InkyCal provides a dynamic calendar with a restful, unobtrusive display showing appointments, anniversaries, weather, and more. Raspberry Pi Zero W provides the wireless connectivity and screen

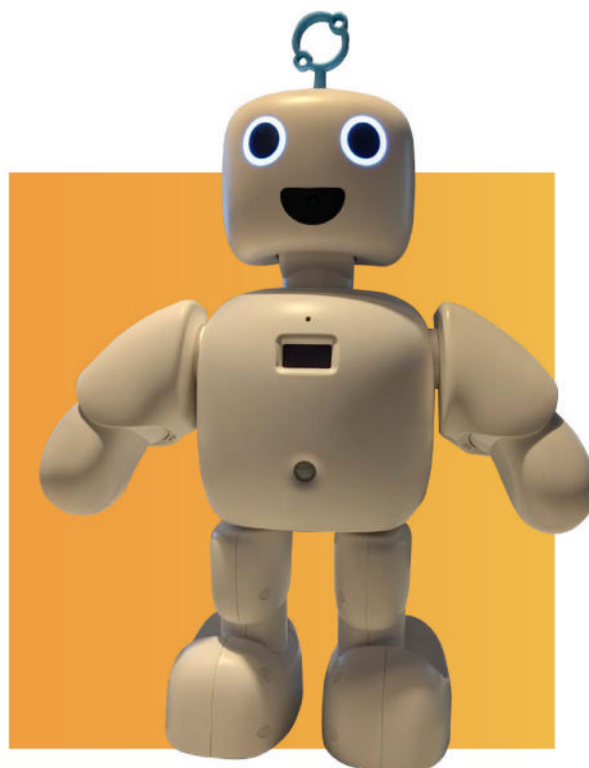
updates. The 7.5in display proffers local and international news headlines, Todoist lists, stock tickers, and other RSS feed-based information while its 480 × 800 pixel resolution lends itself to displaying nuanced greyscale and halftone images and unobtrusive, non-distracting slideshows. InkyCal v3 comes with a pre-flashed microSD card with software you can customise and add information feeds of your choice.

Circul.us Pibo

South Korea | rpimag.co/pibo-biped

Pibo is a 40cm-tall social robot with a touchpad on its forehead for petting and gentle interaction. It comes in educational and AI editions and has Compute Module 4 smarts. Circul.us itself is a platform designed for making and coding, with similarities to Node-RED, but Python can also be used. The LiPo-battery powered Pibo robot features a Raspberry Pi Camera Module, motion sensors, nine AS3103 servo motors to provide movement, a Mini Maestro servo controller, speaker, lights, and wheels and is controlled wirelessly from a laptop or PC dashboard or a smartphone.

The design is intended to appeal to youngsters and encourage them to learn how to interact with their cute companion, changing various features, creating and displaying messages on the LCD embedded in Pibo's chest, and recording and replaying sounds. In our issue 147 (rpimag.co/147) review of the educational version of Pibo, we found the machine learning and voice recognition features impressive and the coding options ideal for moving on from block-based learning to 'proper' programming.



10 amazing:

driving projects

Here with Raspberry Pi, we feel safest of all, especially with these ingenious car customisations

They say that if it has a standard, you can hack it. Or at least, we do, and it turns out the OBD-II standard used by just about every modern car has inspired a lot of people to see how they can modify their motors with a Raspberry Pi. Carefully, mind.



01. Teslonda

Ultimate upcycle

rpimag.co/teslonda

Two makers took a 1981 Honda Accord and decided to power it up with an electric Tesla motor. It also has extra racing modifications, and a fetching colour scheme.

02. Dashcam

DIY safety recordings

rpimag.co/diydashcam

Dashcams are cheap and plentiful; however, sometimes you want to add a little more to yours. This one includes an infrared inside camera for theft prevention too.

03. In-car entertainment display

Raspberry Pi Auto

rpimag.co/incarent

Did you know that Android Auto sends a 'projection' of your phone's software to a screen? You can have a Raspberry Pi accept this signal and then add a touchscreen display to use it in your car.

04. Motorcycle HUD

Two-wheeled display

rpimag.co/bikehud

This very custom project allows for satnav on maker redknotsociety's bike, and it's been made as minimalist as possible. The LED ring acts as a compass if wanted, too.

05. RearPi

Safe reversing

rpimag.co/rearpi

Reversing cameras are one of the best bits of modern vehicle tech. If your car doesn't have one built-in, you can always make one with a Raspberry Pi.

06. Driving Mario Kart

Real-life controller

rpimag.co/carcontrol

Thanks to the power of free will, you too can convert your actual real car into a game controller. Bit more expensive than a Logitech, though.

07. Bumpin' Sticker

Now listening...

rpimag.co/bumpin

Instead of professing your love for just one artist, this electronic bumper sticker shows what you're playing in your car at all times. Do people honk for podcasts too?

08. Back-seat entertainment

DVD who?

rpimag.co/backseat400

Remember in-car DVD players on the back seat? Old hat. A Raspberry Pi 400 powers these screens and shows the same view to both passengers, 1990s airplane style.

09. Experimental HUD

Aerial display

rpimag.co/exphud

A Python-powered HUD projection on this experimental aircraft (can you drive planes?) seems incredible, but do check your local aviation laws before giving it a go. If you own a plane.

10. OVCS Frankencar

It can drive!

rpimag.co/ovcsfrank

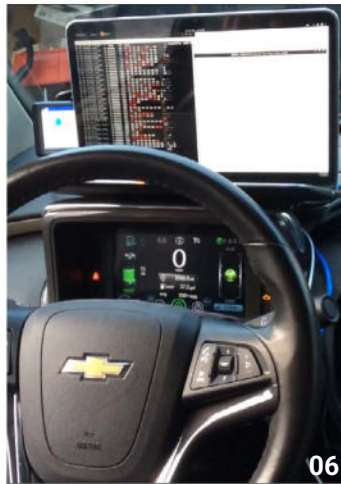
The Open Vehicle Control System, or OVCS, aims to use a Raspberry Pi to allow various electric vehicle parts – each with their own proprietary tech – to communicate with each other.



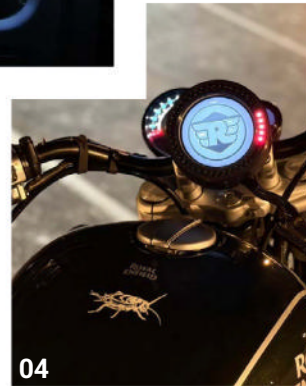
02



03



06



04



05



09



07



08



10

**Warning!****Car safety**

Cars are complicated, expensive, and dangerous vehicles. Make any modifications safely and in a controlled environment. Also, do not obscure your view with a project, or add distractions like video playback. Use some common sense, and check your region's automotive laws.



Suyash Dwivedi

A community organiser at Wikimedia, Suyash helps organise the Wikimania Raspberry Jams

-  **Name** Suyash Dwivedi
-  **Occupation** Electronics engineer
-  **Community role** Wikimedia volunteer
-  **URL** rpimag.co/suyashwiki

Wikipedia is a part of everyday life, at least for people who need to do a lot of research for their jobs. It's just one of the many projects that make up the Wikimedia Foundation, a host of free information and media for all the world.

"I joined Wikimedia after discovering that anyone could contribute knowledge and photographs that benefit millions of people worldwide," Suyash Dwivedi tells us. He's an electronics engineer professionally, having worked in the industry for over 25 years, and also runs Raspberry Jams for 'Wikimedians'. "Photography was my entry point. I began uploading freely licensed images to Wikimedia Commons and soon became involved in other Wikimedia projects."

What is your history with making?

I've always enjoyed building and experimenting with electronics. From designing electronic circuits and embedded systems to creating educational demonstrations, I've been fascinated by

how technology can solve real-world problems. Raspberry Pi became a natural extension of that passion because it makes computing and hardware projects accessible to everyone.



▲ The internet-in-a-box is a concept of having enough free content on something, like a Raspberry Pi, to serve a remote community's internet needs



▲ The first Raspberry Jam at Wikimania was last year; however, Suyash has been organising Jams in general since 2018

When did you learn about Raspberry Pi?

I first learned about Raspberry Pi soon after its early releases. I was immediately impressed by the idea of an affordable computer that could teach programming, electronics, and problem-solving. As someone with an electronics background, I saw Raspberry Pi as an ideal platform for learning, prototyping, and community education. Since then I've used Raspberry Pi for educational workshops.

What are the Wikimania Raspberry Jams like?

The Wikimania Raspberry Jams are informal, hands-on community gatherings held during Wikimania. Participants bring their Raspberry Pi projects, discuss ideas, and explore how open hardware can support Wikimedia and education.

The atmosphere is very collaborative. Some people are experienced makers, while others are seeing Raspberry Pi for the first time. Everyone is encouraged to ask questions, experiment, and share what they're working on.

One of the most rewarding aspects is seeing people from many different countries exchange ideas and learn from one another.

What cool things have people made at the Wikimania Jams?

We've seen a wide variety of interesting projects, including internet-in-a-box servers that provide offline educational content, home automation systems,

*The atmosphere
is very
collaborative*

robotics demonstrations, environmental monitoring projects, retro gaming systems, media servers, PiWikiSpot, some cool hardware that can control camera triggers, AI experiments, and custom electronics using Raspberry Pi GPIO.

Many discussions also focus on how Raspberry Pi can support Wikimedia activities, such as offline Wikipedia

access, digitisation, community outreach, and educational workshops in areas with limited internet connectivity.

One of the things I appreciate most about both Raspberry Pi and Wikimedia is that they are driven by communities that believe knowledge should be open and accessible.

Bringing these two communities together through the Wikimania Raspberry Jam has been incredibly rewarding. People come from different countries, cultures, and technical backgrounds, yet they quickly find common ground by building, sharing, and learning together.

I hope the Raspberry Jam continues to grow at future Wikimania events and inspires more people to explore open hardware, open-source software, and free knowledge. 🍷

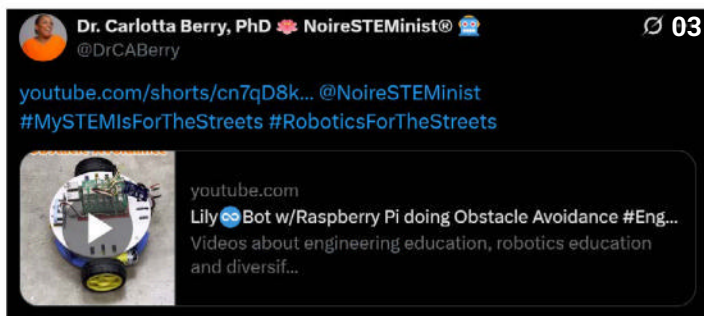
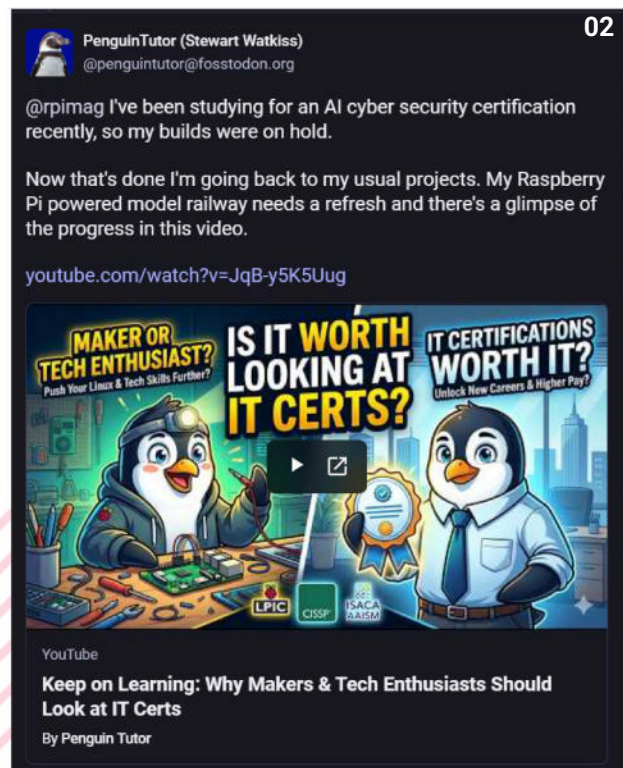
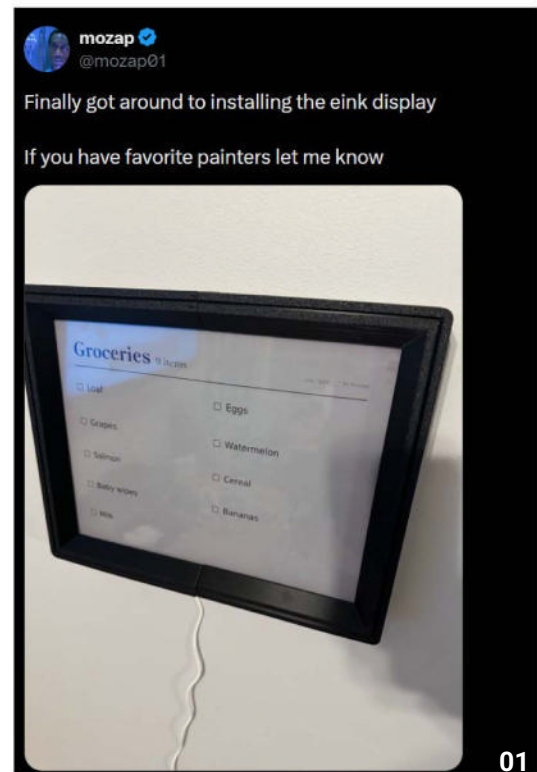
Maker Monday

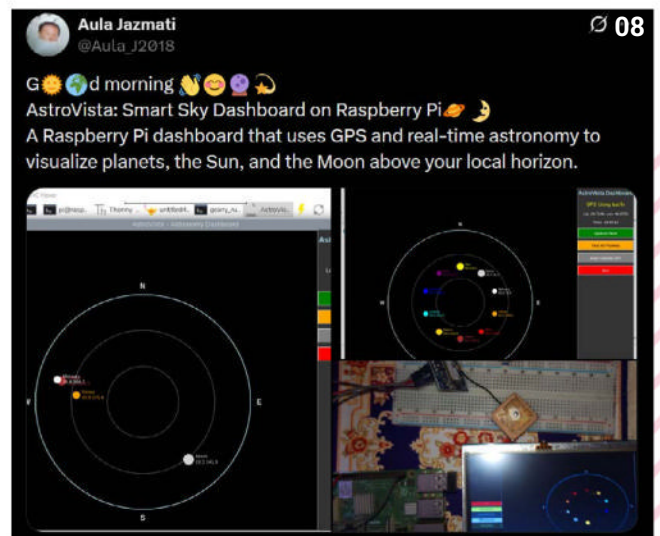
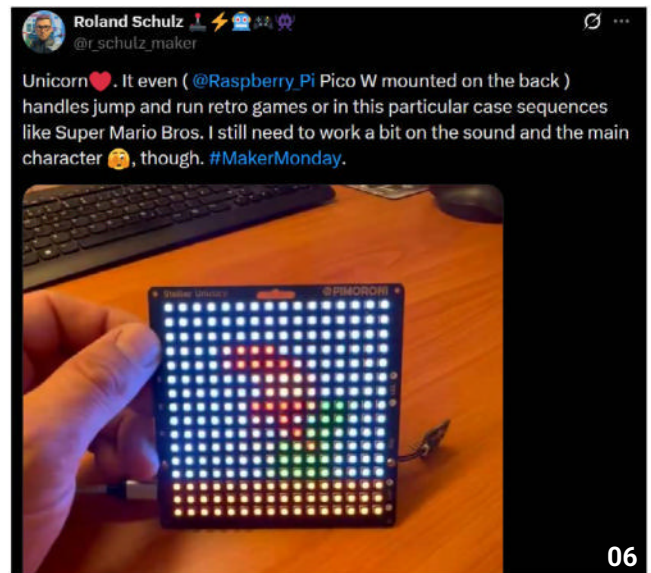
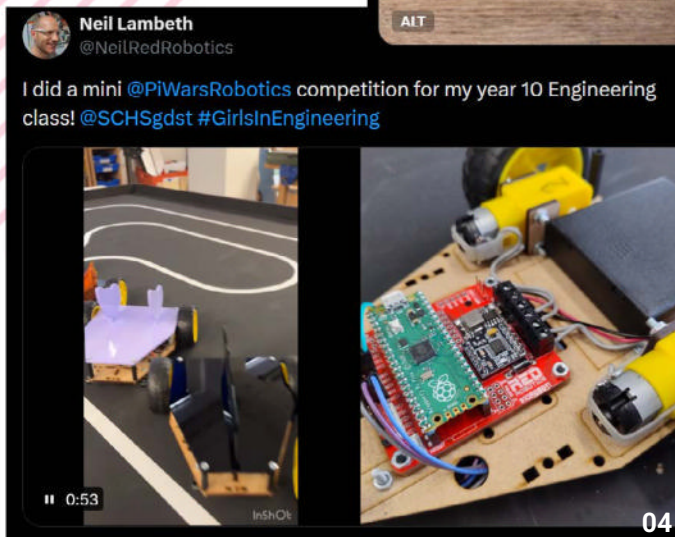
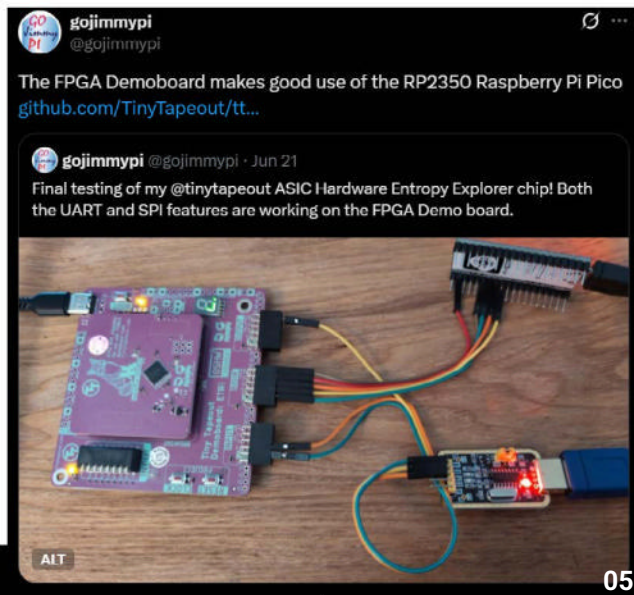
Amazing projects direct from social media!

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. Our favourite painter is Raphael. Turtle or Italian? You decide – rpimag.co/168mm1
02. A life update from one of the long-time makers to appear in the mag – rpimag.co/168mm2
03. This robot is great at avoiding things, but can it avoid consequences? Yeah, probably – rpimag.co/168mm3
04. Pi Wars isn't just for national competitions! It's a great teaching aid too – rpimag.co/168mm4
05. More expansion work with RP2350 – rpimag.co/168mm5
06. This looks a lot better in video, and probably even better in person – rpimag.co/168mm6
07. Moving on from trains, Pater is now onto classic 4x4s – rpimag.co/168mm7
08. We love it when folks simulate the solar system on Raspberry Pi hardware – rpimag.co/168mm8





Robotics Jam in Argentina

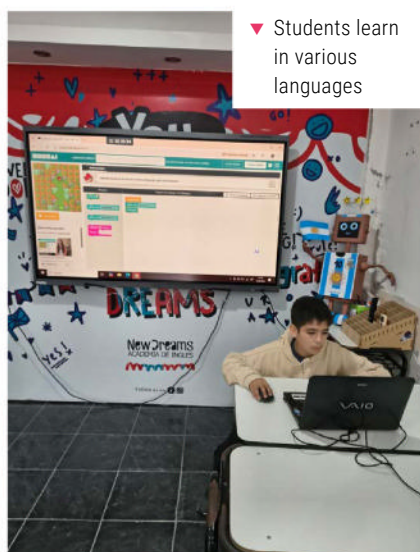
Learning to code through robotics

It's always great to hear from Raspberry Jams around the world, and in different languages too. In Fernandez, Santiago del Estero the regular Raspberry Pi Jams continue. In the most recent workshop, students explored basic programming and robotics.

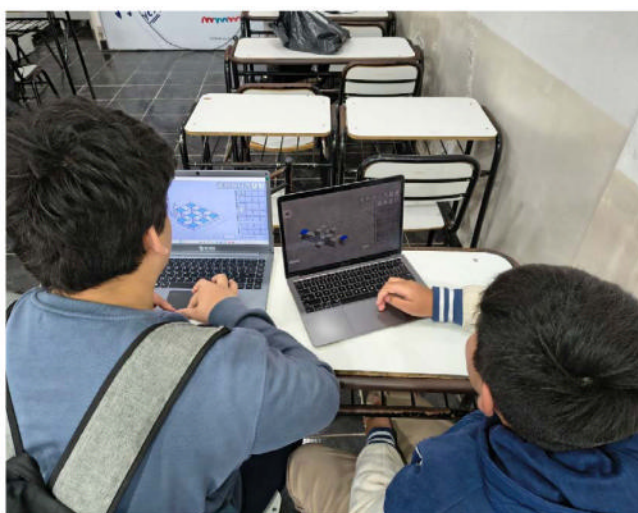
“For many of them, this was their first interaction with a Raspberry Pi,” Eliana

Otrera, one of the organisers, tells us. “The students showed great enthusiasm and curiosity, and seeing them learn, experiment, and work as a team reaffirmed the importance of introducing technology in classrooms at an early age.

“I want to express my gratitude for the support provided and for considering future collaborations. The community is actively requesting the repetition of these workshops to continue motivating more young people to delve into the fascinating world of programming and robotics.”



▼ Students learn in various languages



▲ Cake and coding is an important and useful combo
◀ Multiple teaching methods are employed

Crowdfund this

Crowdfunding campaigns to keep an eye on

Lini



A tiny, 'all-in-one' pocket computer powered by a Raspberry Pi Compute Module 5. It's very small and has a lot of customisability – the regular 40 pins of Raspberry Pi are available for people to use, along with USB 3.0 and C ports. There's a ThinkPad nubbin mouse too, which is always charming.

► kck.st/4fakZvJ

RootBoard



Another tiny computer, this one runs on any kind of Raspberry Pi Zero, and there are cheaper kit versions that allow you to supply your own too. It opens up a lot of ports to the user and the Kickstarter is very proud that it has a full keyboard in the body too.

► kck.st/4gFMxLH

Together we can make a difference

Give young people the opportunity to learn about technology

The Raspberry Pi Foundation enables young people to realise their full potential through the power of digital technologies, but we can't do this work without your help. Your support helps us give young people the opportunities they need in today's world. Together we can offer thousands more young people across the globe the chance to learn to create with digital technologies.

Generous donations from organisations and individuals who share our mission make our work possible.



DONATE NOW



Donate today to make a difference: rpf.io/donate



Raspberry Pi
Foundation

Your Letters



Smoking

I read with interest about the developer who's using Raspberry Pi to control the air inlet of his food smoker, to ensure a consistent burn rate. I don't have the engineering skills to pull that off, but I did once use a Raspberry Pi Pico W to monitor the progress of a smoker that I built out of two kettle barbecues stuck together. One had the fire, which I started out with charcoal to get it going, then heaped a load of apple-wood logs on it to make smoke; the other was connected with a tube from an old washing machine, and had the meat in it, with a temperature probe and a moisture probe sticking into the meat.

It took up far too much space, so I've got rid of it now, but I'm extremely proud of the leg of pork that I turned into almost-but-not-quite Serrano ham.

Javier, via email

Wow. That sounds unbelievable, and makes me regret throwing away my old rusty barbecue. There's so much you can do with food, even if it's something as simple as using a sensor to let you know when something has reached the correct temperature. And if you discover a bug in your code, you're obliged to keep cooking until you figure it out. Yum!



▲ Matti Tahvonen's smart smoker

Radio radio

You keep publishing cyberdeck builds in the magazine. Last month there was a tiny one in a mint tin, and one that used a text-based interface [rather than a graphical user interface]. There have been various cyberdecks with a built-in trackball, made out of wood or 3D printed parts. What I want to know is what's the simplest way to build one?

Brian, via email

The best thing about open hardware is that there are so many options. The worst thing about open hardware is that there are so many options. When you know what you're doing, having a huge array of components and features to choose from is like being a kid in a sweet shop. When you don't know what you're doing, it can feel overwhelming. If we were starting out we'd go for a Raspberry Pi Zero W, a USB battery pack, and a Bluetooth

keyboard (to keep the amount of wiring down). For the screen, we'd look one step ahead and pick one that would work in the project after next (a 7-inch touchscreen that will end up in a jukebox would be our choice, as we want to make one of these by Christmas). After that, the case is completely up to you, and that's where you can start adding things like speakers, trackpads, and breakouts for GPIO pins. We could go on, but it's probably best just to point you to this issue's cover feature on page 32 so you can steal your own ideas.

▼ A 3D printed cyberdeck by hiro24



Radio communication

As the proud holder of an A-level qualification in physics (earned many years ago), I am of course familiar with radio communication. But it never occurred to me that you could buy an add-on for your computer for the cost of a few quid. I know I'm late to the party but LoRa and Raspberry Pi sound like just the thing to send alerts to the bottom of my garden. Every day's a school day, so they say.

Sarah, via email

It gets even better: you can also get boards equipped with 4G, which can relay information from anywhere there's a phone signal. They need a SIM card and a data plan, so you'll need to pay a monthly fee, but it does mean that – for example – you can keep an eye on the temperature in your allotment greenhouse from miles away.



▲ The RP2350-based Perpetuo LoRa board is just one of many available

Contact us!



@rpimagazine



@rpimag



rpimag.co/facebook



magazine@raspberrypi.com



forums.raspberrypi.com

USA SPECIAL!

6 ISSUES FOR \$43

FREE
RASPBERRY
PI PICO 2 W



Subscribe online:

rpimag.co/subscribe

Continuous credit card orders will auto-renew at the same price unless cancelled.
A choice of free Pico 2 W or Pico 2 is included with all subscriptions. This is a limited offer.
Not included with renewals. Offer subject to change or withdrawal at any time.

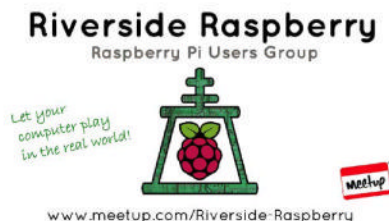
Community Events Calendar

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

01. Riverside Raspberry Pi Meetup

- Monday 10 August
- 3600 Lime Street, Riverside, CA, USA
- rpimag.co/rrpm168

The purpose of Riverside Raspberry is to share knowledge related to Raspberry Pi hardware in particular, and to promote interest in tech development in the Inland Empire in general. While the group is focused on Raspberry Pi specifically, it also covers topics about all kinds of maker technology.



02. Medway Makers: Tinker, Build, Create

- Sunday 16 August
- Mike's House, Gravesend, UK
- rpimag.co/mm168

Whether you're a seasoned expert in electronics, a skilled coder, or someone who's curious about starting their journey into the world of DIY tech projects, Medway Makers offers a welcoming and inspiring environment to exchange ideas, learn new skills, and collaborate on exciting projects. From Arduino and Raspberry Pi to drone flying and digital fabrication, there's something for everyone at the group's meetups.

03. Raspberry Jam SDE

- Saturday 22 August
- Academia New Dreams, Fernández, Santiago del Estero, Argentina
- rpimag.co/rjsde

This event offers an opportunity for enthusiastic children to experiment, learn, and share knowledge about developing technological projects using Raspberry Pi. They have planned activities for children and teenagers, and they will have inspiring talks and interactive workshops, seeking to continue expanding their technology community.



04. Pi Foundations

- Wednesday 29 August
- Odyssey Educational Foundation, Gwarinpa, Nigeria
- rpimag.co/pifound

Pi Foundations introduces participants to Raspberry Pi hardware, Raspberry Pi OS, terminal commands, GPIO, and basic programming, providing the essential skills needed for future projects in robotics, IoT, AI, and automation.



05. Maker Faire Hannover

- 📅 Saturday 15 August to Sunday 16 August
- 📍 Hannover Congress Centrum, Hannover, Germany
- ▶ rpimag.co/mfh2026

The team from Raspberry Pi is excited to be at Maker Faire Hannover for the fourth year in a row. Come and meet members of the Raspberry Pi team, learn about latest products, and share what you've made with Raspberry Pi technology.



Win 1 of 5 Pitower Gen 1 cases

The Elecrow Pitower Gen 1 case is a desktop-style case for Raspberry Pi 5, complete with fans and a little 1.3-inch OLED screen for diagnostics. We reviewed it in issue 162 (rpimag.co/162) and thought it was great and very sturdy – and we now have five to give away!



Head here to enter:

rpimag.co/win

Learn more:

rpimag.co/pitower

Terms & Conditions

Competition opens on **29 July 2026** and closes on **27 August 2026**. Prize is offered to participants worldwide aged 13 or over, except employees of Raspberry Pi Ltd, the prize supplier, their families, or friends. Winners will be notified by email no more than 30 days after the competition closes. By entering the competition, the winner consents to any publicity generated from the competition, in print and online. Participants agree to receive occasional newsletters from Raspberry Pi Official Magazine. We don't like spam: participants' details will remain strictly confidential and won't be shared with third parties. Prizes are non-negotiable and no cash alternative will be offered. Winners will be contacted by email to arrange delivery. Any winners who have not responded 60 days after the initial email is sent will have their prize revoked. This promotion is in no way sponsored, endorsed or administered by, or associated with, Instagram, Facebook, Twitter (X) or any other companies used to promote the service.



Official Magazine
#169

Next Month

RETRO GAMING

Get back to an age of gaming wonder

On sale 27 Aug

PLUS!

Citizen science projects

Second Brain on Raspberry Pi

Digital microscopes



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

Brian Cortell, Ed Cox, David Crookes, Tim Danton, Rosemary Hattersley, Jo Hinchliffe, Nicola King, Phil King, John Lynn, Rob Miles, James Mitchell, Matan Naamati, Richard Smedley, Tom Westcott

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

The Maltings, West Street
Bourne, Lincolnshire, PE10 9PH
+44 (0) 1778 392472
rpmag.co/subscribe
raspberrypi@warnersgroup.co.uk



This magazine is printed on paper sourced from sustainable forests and the printer operates an environmental management system which has been assessed as conforming to ISO 14001.
Raspberry Pi Official Magazine is published by Raspberry Pi Ltd, 194 Cambridge Science Park, Milton Road, Cambridge, England, CB4 0AB. The publisher, editor, and contributors accept no responsibility in respect of any omissions or errors relating to goods, products, or services referred to or advertised in the magazine. Except where otherwise noted, content in this magazine is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported (CC BY-NC-SA 3.0).



ISSN: 2977-4403 (Print)
ISSN: 2977-4411 (Digital)



A helping hand for the programmer

Take the code out of coding – make the machine do it!

You might have missed it, but last month there was a large international football tournament being played in Canada, the USA, and Mexico. At the time of writing, England were on course to win the whole thing, playing with verve, passion, and joie de vivre. But as well as the football itself, the World Cup unleashed something else: a torrent of truly bad AI. Viral videos doctored to change the likenesses of the original protagonists to those of famous footballers. Manga-style cartoons generated depicting footballers as generals, or Samurai, or Vikings. The worst we've seen so far was a pub advertising its coverage of Norway vs England with an AI-generated depiction of the Battle of Stamford Bridge. However, as annoying as this slop might

be for some, it's all a long way from the 'AI will take our jobs' doom-mongering. Yes, the pub in question could have commissioned an artist to design a poster, but they were never going to. AI hasn't stolen anyone's job, because that job was never going to be done by a human – it wouldn't have been done at all if the marginal cost of typing a few commands into an image generator weren't zero or close to zero. And it looked rubbish because very few people had smartphones in 1066, so we don't have many photographs of the battle for the AI model to learn from.

Because it's there

While AI may be rubbish for depicting historical battles with any degree of accuracy, it's excellent at some other things, and it all comes back to the quality and size of the dataset that it infers from. This is where Python comes into its own. Whereas AI has to infer the rules of English from the millions and millions of words that have already been written, Python has a set of rules. It's just maths dressed up to make it more readable for humans, and computers are good at maths.

You didn't get your first Raspberry Pi because you were enthused by the prospect of searching for the right module to use,

or because you love searching the web for error messages. You got one to build something quickly and cheaply, and AI is utterly brilliant at that.

Accept no substitutes

I've been having a play with AI coding recently, and it's great fun. First of all, you can't just type: "code this for me" and expect a sensible answer. You've got to know what to ask, and that's the hard part. When we talk about coding, it's a bit of a misnomer, because sitting at a computer writing code isn't the important part: the important part is problem-solving, and for now at least, human brains are the best in the business at that. The fun part – for me at least – of any project is the physical act of building it, and anything that enables me to spend more time on the fun stuff has got to be worth a try. 🍷

Andrew Gregory – Author

Andrew is features editor of *Raspberry Pi Official Magazine* and has been slowly working through his list of project ideas in an attempt to find one that will work.

rpimag.co

Python is just maths dressed up to make it more readable for humans

HighPi 5S

• ————— The new case from the HiPi team ————— •



Reliable case for complex projects:

- Built for Raspberry Pi 5
- Rapid tool-free assembly and disassembly
- Large internal volume for HATs even with the active cooler installed
- Camera cable slot to connect to external cameras
- Power button and external status LED

- Passive & active cooling options
- Secure microSD cover
- VESA mount support

Customization options for volume orders:

- Output ports molded to your exact specs
- Logo printing for a branded finish

Available at these great Pi stores:



Contact your favorite Pi store if it's not listed here

PiKVM

Remote control **redefined**

Manage your
servers or PCs
remotely!



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



HiPi.io

No reseller in your country?
Check shop.hipi.io (import fees might apply).

List of official
resellers by country:

