

Improve the world
with citizen
science projects

Discover an
upcycled e-paper
typewriter

An introduction to
predictive
AI models



Official Magazine
#169 | September 2026

Raspberry Pi

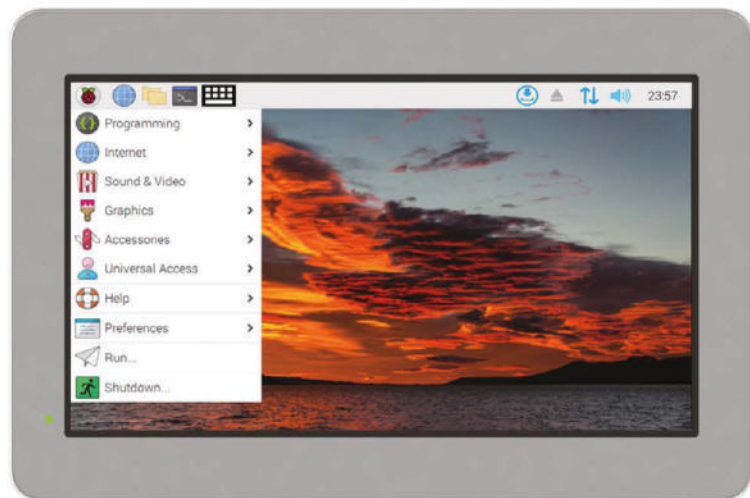
RETRO GAMING

ON RASPBERRY PI

➔ Speed run setting up
a gaming system
on easy mode



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 is editor of *Raspberry Pi Official Magazine* and has been learning to fly in *Flight Simulator 2024*. She first tried out *Psion Flight Simulation* in 1982 and, oh my, things really have changed (rpiimag.co/psion-flightsim).

rpiimag.co

Video gaming has been a key presence in my life, and learning to use computers and playing games often go **hand-in-hand**. Games demonstrate functions, systems, and structures that you first enjoy, and then attempt to recreate.

Of course, anything that arrives when you are young becomes part of your story. And retro video games are part of our collective computing history. And while you can never cross the same river twice, you can dip your toes in repeatedly to re-enjoy the past. That's what retro gaming is about. Rediscovering classic computing software and seeing how it holds up in the modern world.

I may be overthinking things. It's still fun to play Chuckie Egg.

There's a joke online about retro gaming that goes: "Why spend \$80 on a retro system when you can buy a Raspberry Pi and never shut up about it?" Like all jokes, there's a hint of truth. Making your own Raspberry Pi retro games console is so much fun, you will start to evangelise about it to friends, family, and total strangers. To the point of becoming annoying.

But if you love retro games, this is the way: get a Raspberry Pi, some ROMs, and a retro games controller and you'll never look back.

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

Intelligent 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

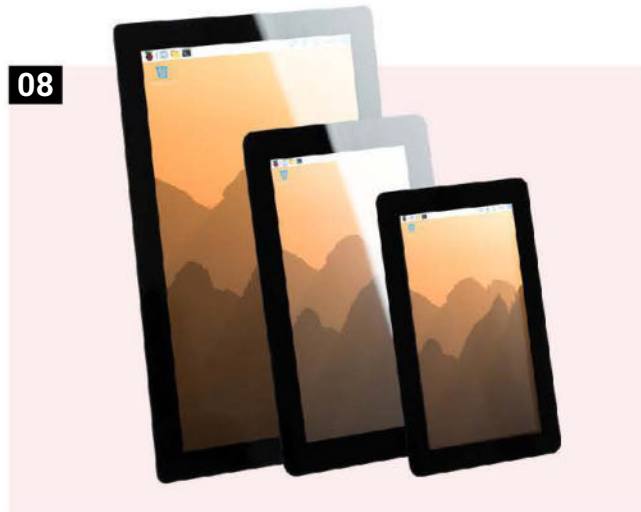
Contents

World of Raspberry Pi

- 008 Raspberry Pi Touch Display 2 - 10-inch
- 010 Listen to Raspberry Pi's podcast: AI on the edge
- 032 Subscribe to Raspberry Pi Official Magazine

Project Showcase

- 012 E-paper typewriter
- 016 In-Vehicle Wrong-Way Driving Detection System
- 020 LesionIQ





24

Top Projects

- 022 Robot arm lamp
- 024 The One Ring
- 026 Pi-Cam-IR-A
- 028 Astrolabe
- 030 3D print showcase

Feature



34

- 034 **Retro gaming on Raspberry Pi:**
A quick and easy guide to get gaming fast



42

Tutorials

- 042 Build an AI picture frame that generates images
- 050 Test AI Camera imaging models with data injection
- 058 Create a private note keeper
- 066 Attach a microscope to Raspberry Pi
- 070 Conquer the command line – part 14
- 074 Simple electronics with GPIO Zero – part 10
- 078 AI projects with Raspberry Pi – part 1
- 090 Add tech to a doll house – part 1

Feature



94

- 094 **Citizen science with Raspberry Pi:**
Data gathering, analysis, and experiments

104



Reviews

- 104 **Only the best:**
storage solutions
- 110 Powered by Raspberry Pi
- 114 **Ten amazing:**
printer projects

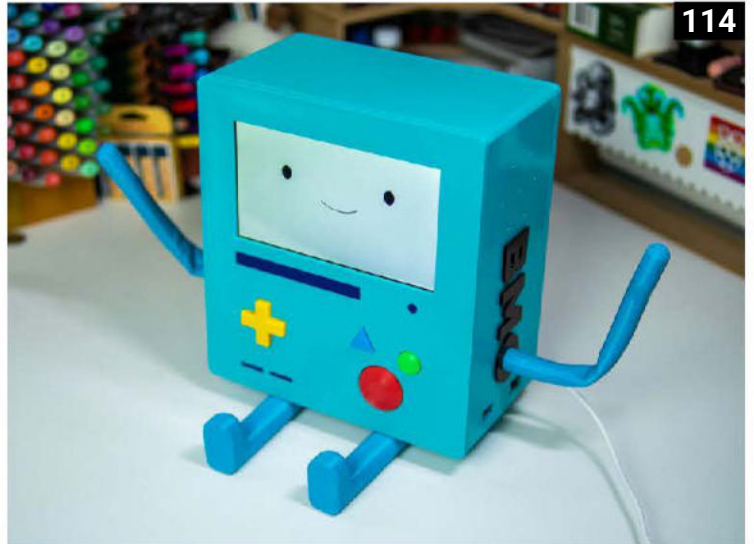
Raspberry Pi Community

- 116 Interview: Sage Cadence
- 118 This Month in Raspberry Pi
- 122 Your Letters
- 124 Community Events
Calendar

Competition

126

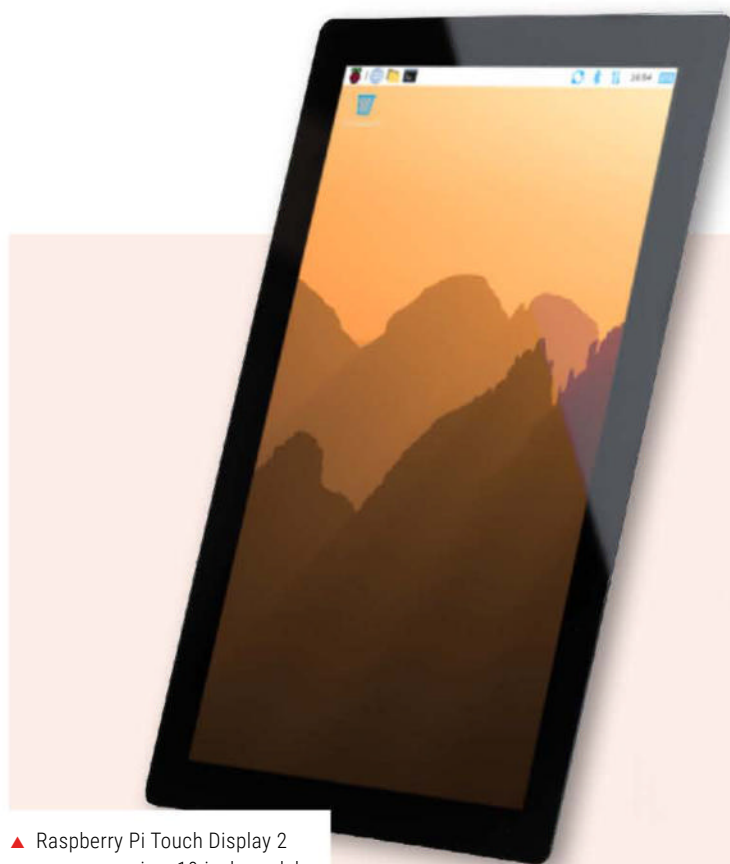
Win 1 of 3
10-inch Touch Display 2



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.

Raspberry Pi Touch Display 2 10-inch

Larger variant offers larger screen size and more resolution for \$80. By **Lucy Hattersley**



▲ Raspberry Pi Touch Display 2 now comes in a 10-inch model with higher resolution

Raspberry Pi has launched a 10-inch variant of its Touch Display 2, offering more screen estate and a higher-resolution display than the current 5-inch and 7-inch models.

Gordon Hollingworth, Raspberry Pi's Chief Technology Officer (CTO) says: "Our new 10-inch display features a higher 1200 × 1920-pixel resolution, a more generous 85-degree viewing angle, and a touchscreen controller with support for ten-finger touch, up from five-finger touch in the 5-inch and 7-inch variants."

Raspberry Pi Touch Display 2 is used for a range of projects that require a display with touch interface. Only two connections are required to connect the display to your Raspberry Pi: power from the GPIO header, and a ribbon cable that connects to the DSI port (not found on Raspberry Pi Zero models).

The Raspberry Pi ecosystem

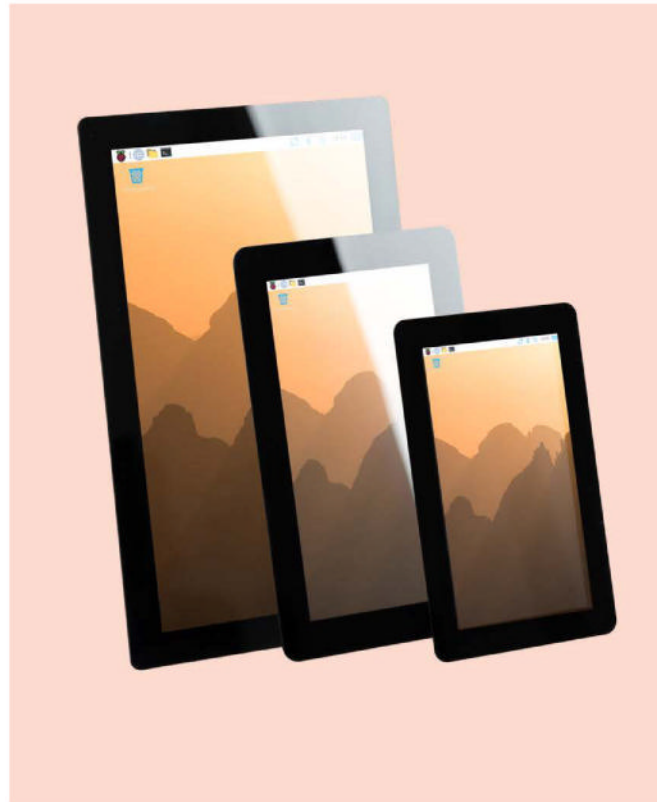
As with all Raspberry Pi display products, the 10-inch Raspberry Pi Touch Display 2 integrates seamlessly with the rest of the Raspberry Pi ecosystem. "It is

powered directly from the host Raspberry Pi, and comes with all the necessary cables, connectors, and mounting hardware,” says Hollingworth. “The capacitive touchscreen benefits from full driver support in Raspberry Pi OS”.

Raspberry Pi Touch Display 2 – 10-inch Portrait is only compatible with Raspberry Pi 5 and Raspberry Pi Compute Module 5, unlike the smaller versions of the screen which also work with earlier models. Hollingworth explains: “While previously released Touch Displays use only two lanes of DSI data to communicate with the host Raspberry Pi – which works well with the older display connector on Raspberry Pi 1, 2, 3, and 4 – the higher resolution of the new 10-inch variant requires four DSI lanes. This in turn requires our narrower, but higher-bandwidth, ribbon cable and the newer display connector found only on Raspberry Pi 5 and the Compute Module IO boards.”

For display options you can use with earlier Raspberry Pi computers, take a look at the 5-inch and 7-inch variants of Raspberry Pi Touch Display 2, or consider the Raspberry Pi Monitor.

The 10-inch Raspberry Pi Touch Display 2 is available to buy now from Raspberry Pi’s worldwide network of Raspberry Pi Approved Resellers: rpimag.co/touchdisplay2. 



- ◀ The 10-inch variant joins the 7-inch and 5-inch models which work with a wider range of Raspberry Pi computers
- ▼ The Touch Display 2 works seamlessly with Raspberry Pi 5 and Compute Module 5

Touch Display 2 documentation

Update the operating system and EEPROM bootloader on your Raspberry Pi 5 or Raspberry Pi Compute Module (any model, with a Raspberry Pi Compute Module IO Board or another suitable motherboard), connect the display to the device, and go. Read the documentation for more details.

rpimag.co/touchdisplay2docs



Raspberry Pi Touch Display 2 integrates seamlessly with the rest of the Raspberry Pi ecosystem

Raspberry Pi podcast: AI at the edge

How our tiny computers learnt to count sheep
(and spot hi-vis jackets). By **Ashley Whittaker**

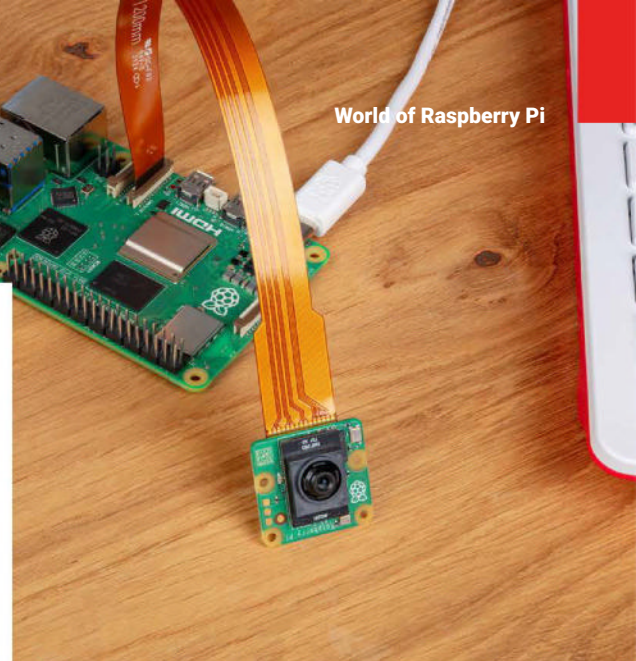


▲ Listen to the full 20-minute chat on YouTube or wherever you get your podcasts

The latest episode of the Raspberry Pi Podcast is here, with Ken Okolo, Commercial Partnerships, Middle East & Africa; and Naushir Patuck, Software Engineering Manager. They explore the rise of artificial intelligence and how Raspberry Pi is making an impact in edge AI.

Naush explains how the growth in AI capability and adoption is being driven by the convergence of big data, deep learning, and increased computational power. He also describes how running AI processing locally, rather than shipping it off to the cloud, can save money, protect privacy, and make sure you keep full control of your data.

Ken and Naush walk through Raspberry Pi's growing AI product line, from AI Camera (rpimag.co/aicamera) with its on-board accelerator, to AI HAT+ for Raspberry Pi 5 (rpimag.co/aihat), right up to the new AI HAT+ 2 with on-board memory for running generative AI models locally (rpimag.co/aihat2). Plus, hear how partnerships with companies like Ultralytics, along with tight software integration through tools like libcamera, are making it easier than ever for developers to build AI-powered projects.



What even is AI, though?

No stupid questions here. And luckily, Naush is adept at conveying refreshingly simple definitions. He explains: “AI is just about letting humans and computers interact naturally, whether that’s a computer understanding what you say or generating something a human can make sense of.”

He also gently reminds everyone that AI has been knocking about since the days of the Turing test; it’s just quietly getting better at fooling us into thinking we’re talking to a person. The real shift, he explains, is that we’ve swapped old statistical tricks for deep learning, which produces far better results but at the cost of serious computational muscle.

Privacy-first option

Nobody really wants their living room camera feed uploaded to a mystery cloud server. Running AI ‘at the edge’ (meaning as close to the camera or microphone as possible) means better privacy, lower costs, and no unexplained subscription fees creeping onto your data bill. Your personal Raspberry Pi can process the video, make its decision, and get on with its day without phoning home.

Naush recalls the early days of OpenCV-based face detection running at roughly half a frame per second: technically

functioning, but practically glacial. Raspberry Pi’s AI product line speeds things up considerably.

Raspberry Pi AI Camera, built with Sony, sneaks an AI accelerator directly onto the imaging sensor itself, meaning even a Raspberry Pi Zero can suddenly perform 30 to 60 frames of inference per second. That’s the difference between “did something just move?” and “yes, that’s definitely your cat, again.”

The AI HAT+ bolts a Hailo AI accelerator onto a Raspberry Pi 5, delivering up to 26 TOPS (tera-operations per second) of processing power, enough to run multiple camera feeds at once.

The AI HAT+ 2 goes a step further with 8GB of on-board memory and support for generative AI applications, meaning you can now run your own local chatbot on a Raspberry Pi with zero internet connection required. Somewhere, a ChatGPT-alike is running in someone’s shed, entirely off-grid.

Ken and Naush walk through Raspberry Pi’s growing AI product line

- ▲ AI Camera packs a Sony IMX500 sensor for fast image inferencing
- ◀ AI HAT+ 2 adds a powerful Hailo-10H AI accelerator and 8GB of on-board RAM to Raspberry Pi 5

What are people using Raspberry Pi AI for?

Alongside more predictable use cases such as counting shop footfall, monitoring car parks, and checking PPE compliance on construction sites, Naush mentions a few delightfully rural ones: farmers counting sheep and cows, and conservationists using object recognition to track endangered wildlife wandering past a camera. It’s a nice reminder that for every corporate security deployment, there’s a Raspberry Pi somewhere quietly keeping tabs on livestock. 🐑

Subscribe to our podcast

Don’t forget to subscribe to the Raspberry Pi Podcast on Spotify (rpimag.co/podcast-spotify), Amazon (rpimag.co/podcast-amazon), or Apple Podcasts (rpimag.co/podcast-apple).

We’ll also be releasing each episode on our YouTube channel (rpimag.co/youtube), where you can watch for free. There’s also a handy RSS feed (rpimag.co/podcast-rss).

E-paper typewriter

This custom vintage typewriter is something to write home about.

By **David Crookes**



Maker

Jamie – Myth Made

Jamie is the creator of Myth Made, where she builds fantasy-inspired props and whimsical inventions using engineering, electronics, and creativity.

rpimag.co/mythmade

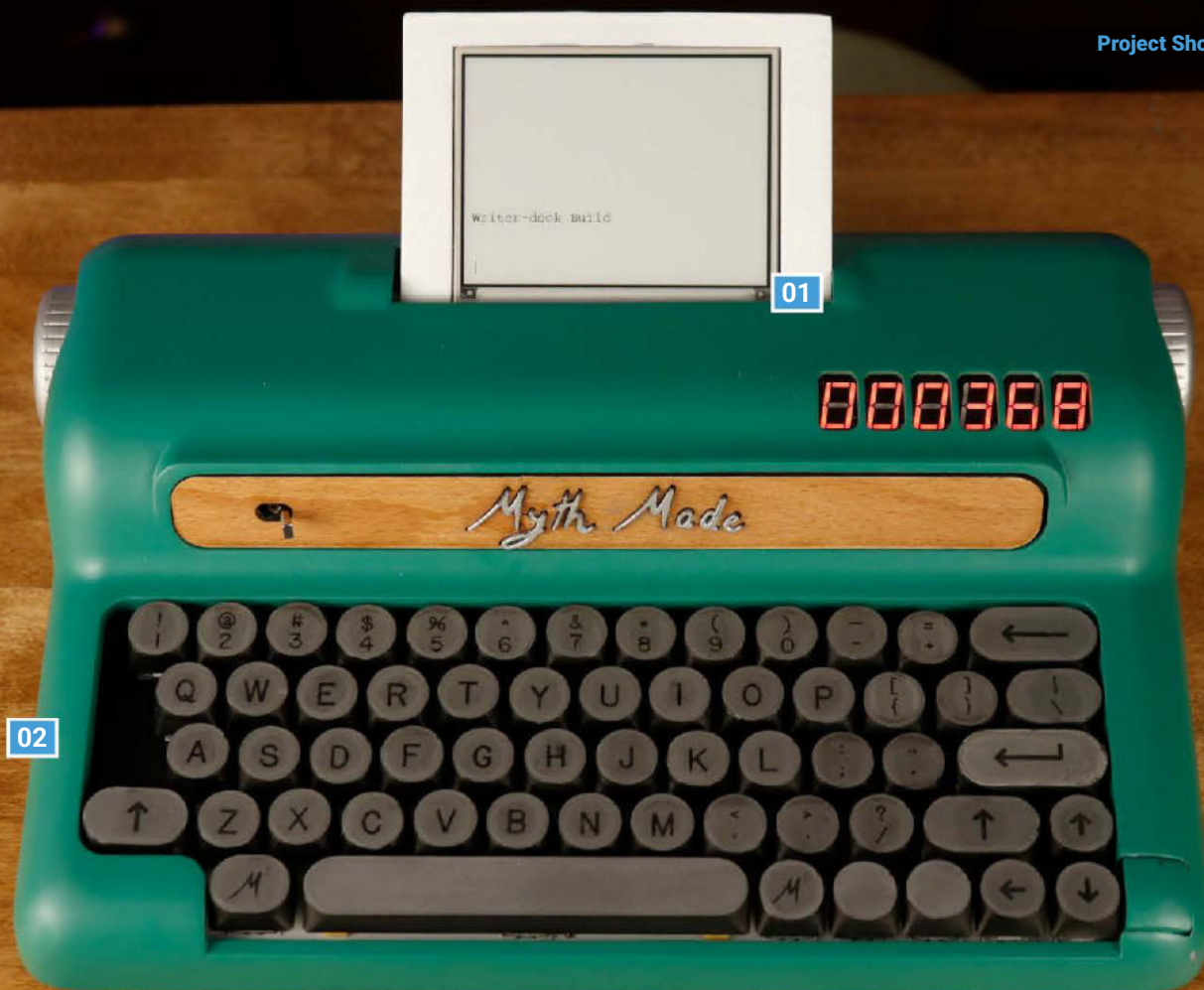
Typewriters have been making a surprising comeback in recent years, partly helped by high-profile fans such as Taylor Swift and partly because of a growing desire among writers to escape digital distractions. When maker Jamie decided she wanted to make the writing process more engaging, however, she was driven more by aesthetics. As such, in the process of creating her own typewriter from scratch, she produced a device that captures the beauty and mechanical feel of vintage machines while replacing paper with the convenience of a screen.

“Aesthetics are probably way too important to me, and I love surrounding myself with beautiful objects that feel like they are from a book,” says Jamie, explaining that she wanted to try writing her own stories but felt uninspired when sitting down at her computer. “Old-fashioned typewriters have all the charm I wanted, but I found them far too impractical for actually writing, so the only logical solution was to build my own [e-paper] typewriter!”

She began by looking at what already existed. “That’s when I discovered the Zerowriter,” she says. Zerowriter is a beginner-friendly DIY e-paper typewriter project built around a Raspberry Pi Zero 2 W computer. It works with any USB keyboard and a Waveshare 4.2-inch e-paper module. “I chose Zerowriter because it was easy to find, well documented, and felt approachable to modify,” Jamie adds. “This was good because I had never done anything like this before.”

Key to success

With a foundation to build on, Jamie also began to work on the look of the project. “I collected references of vintage typewriters, sketched my ideal design, and then recreated it using CAD software,” she recalls. She had a clear idea of the features she wanted before she started modelling, although she hadn’t figured out how to make them all work at that point. “The essentials were a word-count display, a scroll wheel, and a mechanism that let the screen roll up and down like a sheet of paper,” Jamie says. She also needed to identify a keyboard.



Most of the extra features came from studying a real vintage typewriter

"I wanted it to look like a classic typewriter, but all of the 'typewriter-style' keyboards I found looked a bit cheap and gimmicky," she says. "Instead, I bought a good mechanical keyboard with satisfyingly clicky switches and replaced all of the keycaps with custom 3D-printed round ones." She painted the tops of the keycaps using paint markers, then used laser-engraving to add the labels. "I also covered several of the keys to make the keyboard appear narrower, and even hid one of the keys inside the border of the typewriter so it disappeared into the design."

That key was the right arrow one and Jamie had to be creative. The keyboard she'd purchased was too wide for her printed typewriter, so she needed to trim the sides. Trouble is, to ensure it fitted

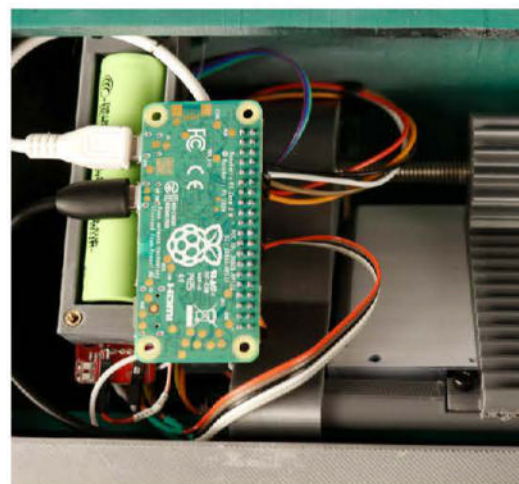
well, she would have needed to chop off the right arrow key. By creating a cutout in the case to accommodate the key, the keyboard not only slotted in perfectly, it also ensured the overall aesthetic wasn't spoiled. Even so, the arrow keys caused another problem:

"Zerowriter is open source and that made the whole project possible. The Python code looked relatively simple to edit and, at first, I only needed to support my different hardware, such as the scroll wheel and the word-count display. But what I didn't expect was that the existing software didn't support the arrow keys I had built into the keyboard. That ended up becoming a much bigger programming challenge than I had planned and, in the end, I used AI to help me solve it."

01. The 4.2-inch e-paper screen moves up and down, just like a piece of paper would

02. Jamie is tempted to fill the small left-side gap with a 3D-printed writing companion

▼ The project is based on Zerowriter, an open-source project, and Jamie has uploaded her own files for free





Ink-credible extras

Jamie's project is certainly eye-catching. Due to its size, the typewriter body had to be printed in halves and the sections were fused together using ABS-acetone slurry. Once the exterior was smoothed, Jamie painted it green and inserted the keyboard before adding the other components. "Most of the extra features came from studying a real vintage typewriter and asking myself which elements I could recreate in a modern way," Jamie reveals.

Of all the features, the rack-and-pinion mechanism that raises and lowers the screen is her favourite. "It makes the [e-paper] display feel like a real sheet of paper," she says. The mechanism allows a rotary motion to be converted to a linear motion. It is made up of a circular gear and a linear track, and it's controlled by a hand wheel fitted to the side of the typewriter. Once the screen is in position, the user can type away. An LED display capable of showing six digits then updates with the current word count.

"The word-count display was partly practical and partly psychological," Jamie says. "I'm very goal-oriented, so being able to watch the number climb is surprisingly motivating. It also just looks really nice." She says none of the individual features were especially difficult to implement on their own. "The real challenge was fitting them all into a single project," she explains. "Every new feature meant more CAD work, more electronics, more code and more troubleshooting."

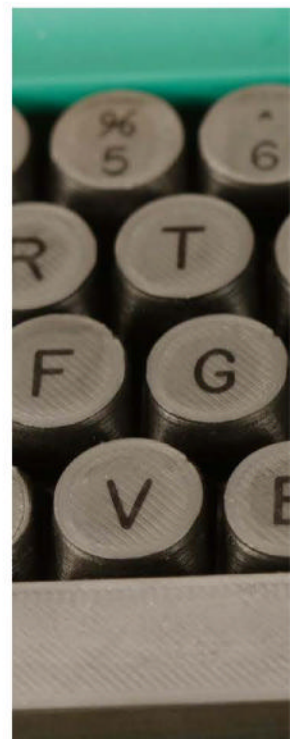
The write way

This meant the project became a much larger undertaking than Jamie had expected, but her foundational choices ensured the finished device runs smoothly. "Zerowriter already runs on a Raspberry Pi Zero 2 W using Raspberry Pi OS Lite, which is incredibly power-efficient: a single 18650 battery lasts for ages," she explains. "My favourite feature is being able to develop over Wi-Fi. Instead of constantly taking the typewriter apart to plug in a USB cable, I could edit the code remotely and immediately test the changes."

She's considering some tweaks. "The one feature everyone keeps telling me to add is a classic typewriter 'ding' whenever you press **ENTER**, so I might have to give in now and do that," she says. In the meantime, she's just enjoying using the device and it's succeeded in inspiring her.

"It turns out that if you spend time building yourself a typewriter, you feel slightly obligated to actually write on it," Jamie says. "But honestly, it really does make me want to sit down and write. It doesn't magically make me a better writer, it just makes writing more inviting. When something is beautiful and fun to use, I'm much more likely to actually use it, and I think that's true for pretty much everything I build." 🍷

▲ Jamie couldn't resist putting her YouTube channel @themythmade across the front of her device in shiny silver cursive



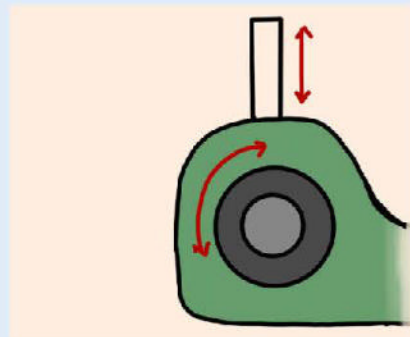
Quick FACTS

- The typewriter took Jamie two months to make
- It's based on the Zerowriter open-source project (rpimag.co/zerowriter)
- Words appear on an e-paper screen rather than paper
- No typewriters were harmed: the body is 3D-printed
- It runs off a battery, making it entirely portable

▼ With little screen clutter and comfortable keys, the typewriter works perfectly well as a distraction-free drafting device



How you type on it



1. After rolling up the e-paper screen using the hand wheel on the side of the typewriter and flicking the lever-style power switch, Jamie says the typewriter opens straight to the document that you're currently working on, allowing you to instantly get your creative juices flowing.



2. You type on a modified USB keyboard that has been fitted with 3D-printed typewriter-style keycaps. Jamie wants to improve the typing latency because it can take a while for words to appear. She believes the problem is in the software rather than the e-paper display or Raspberry Pi.



3. As you type, the six-digit display updates your live word count. You can edit using the arrow keys, copy and paste text, and scroll back through your writing using the scroll wheel. When you're finished, you can simply email the document to yourself directly from the typewriter.

In-Vehicle Wrong-Way Driving Detection System

Innovative thinking combined with a Raspberry Pi computer and AI could save hundreds of lives. By **David Crookes**



Maker

Rohan Parthipan

Rohan is a Statistics and Machine Learning student at Carnegie Mellon University who's developed an AI-based solution for wrong-way driving.

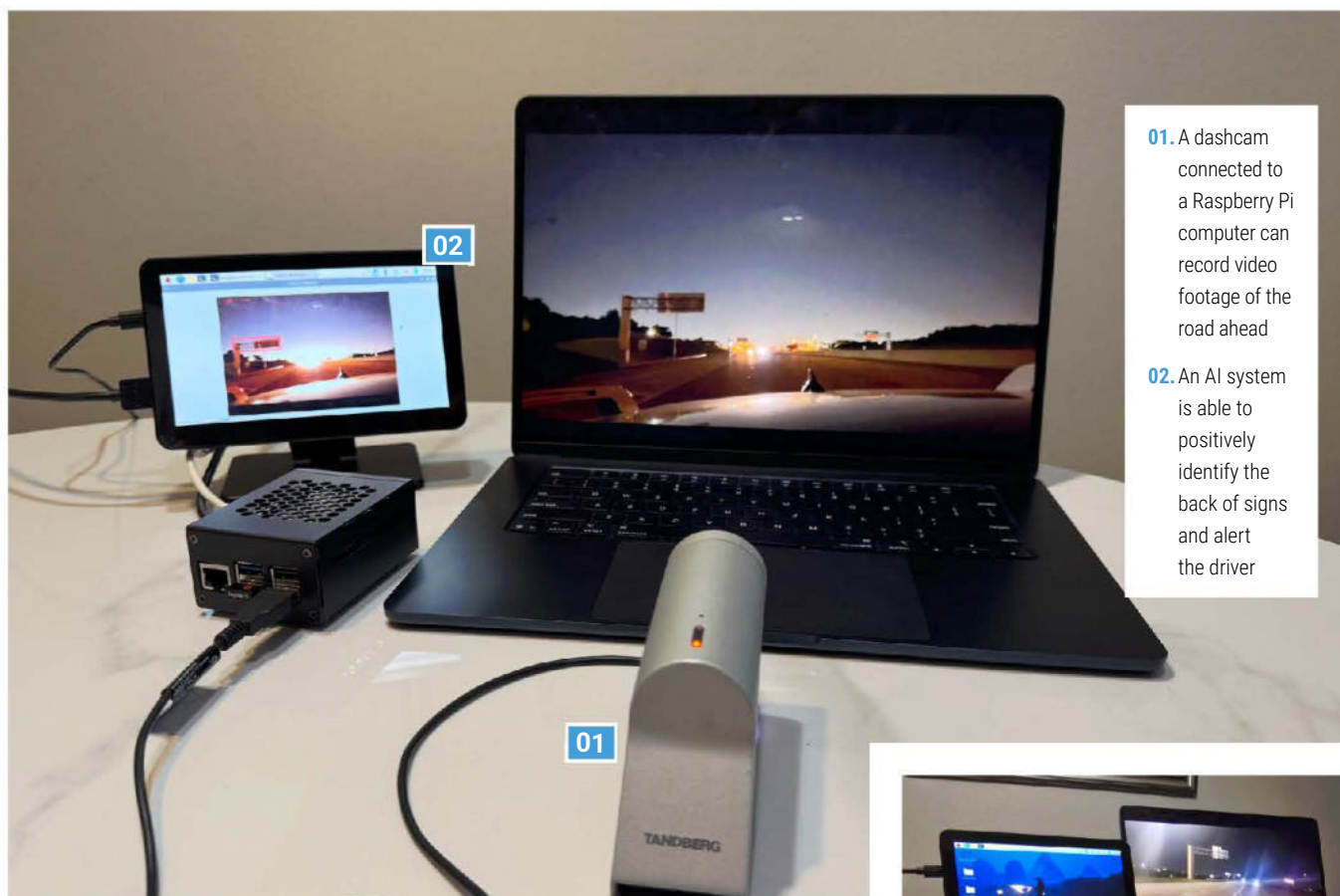
rpimag.co/stopwwd

In November 2023, Rohan Parthipan was on his way home from school in Dallas, Texas when he witnessed the aftermath of a horrific car crash. "Seeing the wreckage first-hand made me realise that behind every headline about a traffic fatality are real people, families, and lives that can change forever in an instant," he says, adding that it had been caused by a driver going the wrong way down the road.

The incident left a lasting impression on him. "As I researched the issue further, I was surprised to learn how frequently these crashes occurred in Dallas, often claiming the lives of innocent drivers through no fault of their own," he recalls. "The more I learned, the more I felt that these tragedies shouldn't simply be accepted as unavoidable accidents. That's when I asked myself: if our cars can warn us about something as simple as not wearing a seatbelt, why can't they also warn a driver when they are travelling in the wrong direction?"

With that in mind, Rohan considered developing a low-cost warning system capable of being fitted in any vehicle. "With so much technology already dedicated to improving safety, such as emergency braking, forward collision warnings, and lane-keeping assistance, it seemed like there should be similar technology to help prevent wrong-way driving," he tells us.

When researching the market, however, he discovered wrong-way driving prevention technology did exist but he found it to be flawed. "I saw that current solutions relied on roadside infrastructure, such as 'Do Not Enter' and 'Wrong Way' signage, as well as thermal or camera-based monitoring systems at critical highway entry points," he says. "Despite these systems, the number of wrong-way crashes has continued to occur, particularly at night and often involving drivers who are intoxicated." He reckoned he could create an improved system.



01. A dashcam connected to a Raspberry Pi computer can record video footage of the road ahead

02. An AI system is able to positively identify the back of signs and alert the driver

Signs of danger

One of the main issues, he found, was that the external warnings were, in many cases, proving ineffective because drivers were not noticing the signs or reacting in time. He also learned that it took about ten minutes for police to intervene whenever a wrong-way driver was detected by traffic

After much thought, Rohan envisioned a system that would still rely on signs but, instead of reading the front of them, he realised wrong-way driving could be quickly detected if his device could read the back. By combining a Raspberry Pi computer with a camera and using real-time computer vision running entirely on

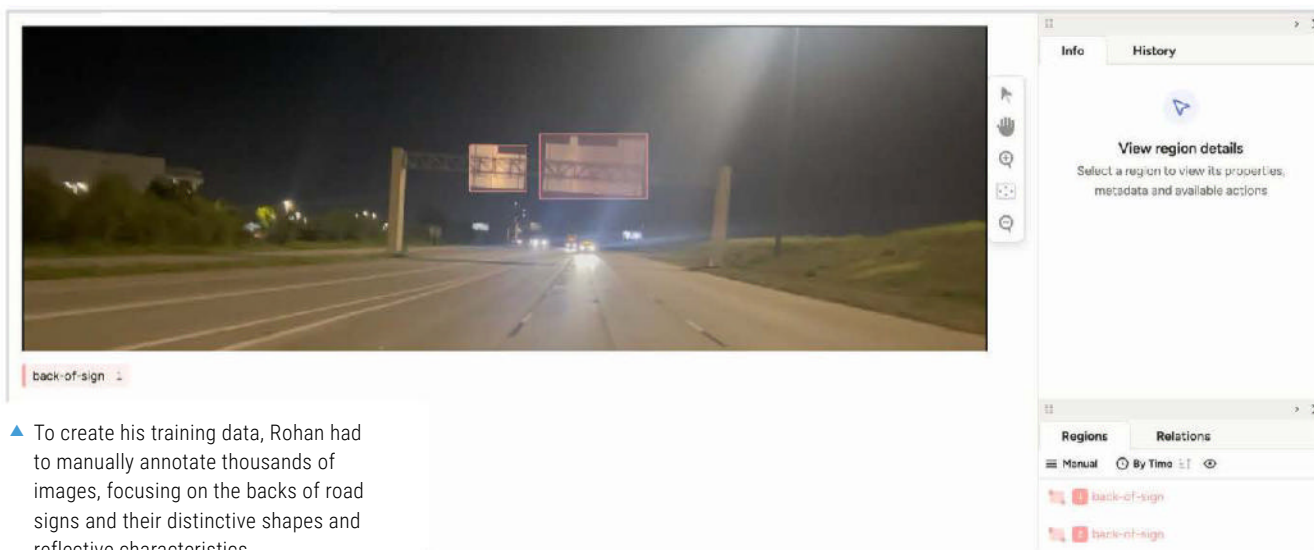


I focused on night-time driving since many fatal wrong-way crashes occur between 11pm and 2am

management centres, so he decided that his system would seek to detect wrong-way driving as early as possible and immediately alert the driver. “This would increase the likelihood of a quick and corrective action by the driver,” Rohan says.

device, he soon worked out that he could produce a system that could recognise the back of road signs and trigger an in-vehicle alert in about 30 to 45 seconds. It proved to be more effective than his other ideas.

▲ Rohan is considering potentially using a GPS/GNSS HAT to improve detection accuracy by monitoring the driver’s location and ensuring they’re travelling the right way



▲ To create his training data, Rohan had to manually annotate thousands of images, focusing on the backs of road signs and their distinctive shapes and reflective characteristics

“I’d decided to use computer vision for my in-vehicle alert system and my original idea was to identify oncoming traffic by detecting the fronts of vehicles but, by the time a driver clearly sees oncoming headlights, they are already only seconds away from a potentially fatal collision,” Rohan explains. “I then considered using the solid yellow lane markings that separate oncoming traffic lanes – on most highways it should always be on the left and, if it’s on the right, the vehicle is going the wrong way. But when I reviewed night-time and poor-weather footage, I found that lane markings are often faded, covered by glare, or partially invisible, making this option unreliable.”

Detecting the back of road signs created the best results. “When driving on highways, a wrong-way driver always sees the backs of road signs, while a driver going in the correct direction sees the front,” Rohan says. “I built a sliding window algorithm that requires multiple detections of the backs of road signs within a certain time interval before confirming a wrong-way event and triggering an audio alert to the driver.”

On the edge

A Raspberry Pi computer works well for this project. Rohan wanted to use edge computing rather than rely on

the cloud, and Raspberry Pi has more than enough computational power to allow all processing to take place on-device. The computer makes use of images taken using a connected HD camera and it analyses each video frame as it’s captured, processing them temporarily in memory before immediately discarding them. “This eliminates latency from internet transmission and ensures the system continues to function even in areas with no connectivity such as rural highways or tunnels,” Rohan says. “It also helps preserve privacy.”

To ensure that the system works accurately, Rohan needed to train his model and he says he used several hours of video footage representing the perspective of a wrong-way driver. “Obviously, it would not be safe or legal to intentionally drive the wrong direction on a highway to collect video footage,” he explains. “To address this, I mounted a camera on the rear of a vehicle facing backward and collected footage while driving normally on highways. During data processing, the video was reversed to closely replicate the visual perspective that a wrong-way driver would experience.”

In total, the dataset consisted of between 300 and 400 miles of North

Texas highways, which amounted to thousands of video frames of the back of many different types of road signs which he then manually annotated. “I focused on night-time driving since many fatal wrong-way crashes occur between 11pm and 2am,” Rohan says. He also took into account glare, signage clutter, and poor visibility so that the model would be reflective of different environmental conditions.

Rohan continues to refine his device and he says he is prioritising reliability, making improvements so that he can reduce false positives. This, he adds, is more important at the moment than reducing the detection time – in any case, to be accurate, the system needs to detect the back of signs multiple times to eliminate alerts from signs on cross streets, nearby roadways, and adjacent lanes.

“In future iterations, I plan to extend the model to also detect front-facing road signs and compare the ratio of front-facing to back-facing detections over time. A consistent dominance of back-of-sign detections would provide a stronger and more reliable signal of wrong-way driving,” he says. “Combining both signals would further reduce false positives and improve overall system robustness.” 📺

Quick FACTS

- Wrong-way driving causes approximately 650 fatalities each year in the US
- Rohan has created a \$100 in-vehicle alert system using Raspberry Pi
- It seeks to quickly spot the rear of multiple road signs
- Drivers are then alerted and prompted to take corrective action
- Rohan hopes governments and automakers will take the idea on board

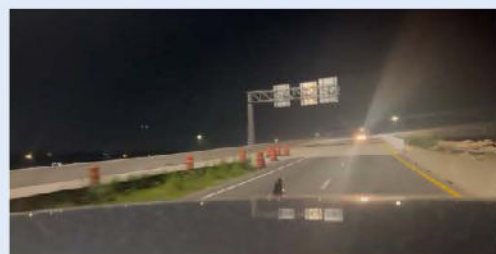


- ▲ Fast processing is important because vehicles travel on the highway at speeds of up to 70mph. On-device processing eliminates delays

Detecting the signs



1. There are many signs around the United States and the rest of the world warning motorists that they could be about to travel in the wrong direction. Despite that, drivers may not notice or respond to these signs, so Rohan believes an in-vehicle warning system is the answer. **Image credit:** Coolcaesar, Wikimedia.



2. Rather than get in-vehicle technology to detect the front of road signs, Rohan collected lots of footage of the back of them. He then used the images as training data for an AI system which sees a Raspberry Pi computer running a custom-trained EfficientDet computer vision model.



3. A forward-facing HD dashcam connected to a Raspberry Pi gathers video which is processed on the device. Instead of reacting to a single detection, an AI algorithm waits for multiple detections within a short time interval before confirming a driver is going the wrong way. **Image credit:** Balder.de, Wikimedia.

LesionIQ

This device can be used to diagnose skin cancer in remote areas. By **David Crookes**

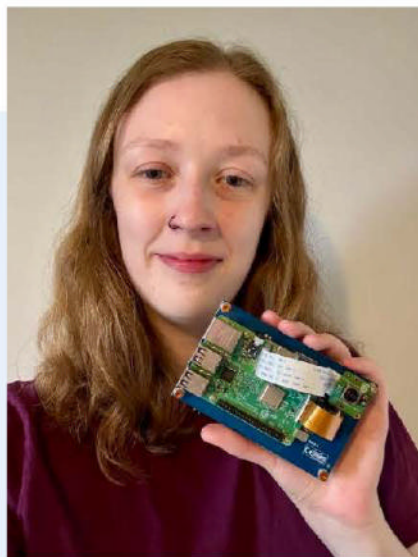


Maker

Tess Watt

Tess Watt is a PhD Student in Healthcare Technologies at Heriot-Watt University in Edinburgh, Scotland.

hw.ac.uk



Summer is now at its peak (in the Northern Hemisphere) but, while most of us enjoy basking in the sun, it comes with an inherent risk. The number of diagnoses of basal cell carcinoma, the most common skin cancer, continues to rise while cases of melanoma skin cancer – the most dangerous of them all – are reaching record highs. “According to The Skin Cancer Foundation, skin cancer is the most common cancer globally,” says scientist Tess Watt. “Despite this, if melanoma is detected and treated at an early stage, its five-year survival rate is 99%.”

► Tess has been demonstrating the system at Heriot-Watt’s advanced health and care technologies suite

◀ Tess has been working with academics at London South Bank University, Edinburgh Napier University, and the Foundation for Research and Technology – Hellas in Greece

With this in mind, Tess, a PhD candidate in the School of Mathematical and Computer Sciences at Heriot-Watt University in Edinburgh, has created an early detection system. She’s developed a set of AI tools designed to diagnose skin cancer and other skin conditions using a Raspberry Pi 3 Model B computer attached to a small camera. The aim has been to produce a low-cost device that can be used by patients living in remote areas of the world. It allows skin conditions to be monitored from home without the need for an internet connection.



**Warning!****Medical advice**

Please seek advice from registered medical experts when diagnosing health issues.

01. A photograph is taken of a skin lesion using an interface designed to be as user-friendly as possible

02. The system is a demonstration of TinyML – machine learning running on lower-powered, non-connected devices

Quick FACTS

- The project aims to reduce delays in skin cancer diagnosis
- Patients could check suspicious skin lesions from home
- It can operate offline, using AI to analyse patients' skin complaints
- Photos are compared to thousands of reference images
- Potential problems will be flagged and assessed by medics

Machine learning can then identify patterns that the human eye cannot

Making a diagnosis

Tess's system uses machine learning to analyse images of skin lesions. "Machine learning has overcome most of the challenges faced in traditional methods of skin lesion classification by analysing many images at once, more accurately than human dermatologists," she says. "Machine learning can then identify patterns that the human eye cannot and can therefore be more accurate and objective."

To assess a skin complaint, patients would be asked to take a photograph of the affected area. Since the skin lesion image datasets are stored on the Raspberry Pi computer, her program is

able to analyse the photograph in real-time before comparing it to thousands of images stored in the dataset. The idea is that the information would be shared with a GP and a decision made over potential treatment. "Current legislation [in the UK] requires a medical professional to validate the outputted diagnosis," Tess says.

Global impact

Tess envisages LesionIQ being used in rural Scotland and she is in talks with NHS Scotland to gain ethical approval. But it could also become a vital tool for remote communities across the world, particularly given Raspberry Pi devices are so widely used and affordable. "I am

often asked why I chose not to deploy my AI model on a smartphone, and this is because smartphones are costly and not widely available/used in developing countries where access to the Internet is also limited," she explains.

The device certainly looks promising. Currently, it's proving to be 85% accurate in diagnosing skin cancer and Tess says this can be improved. "I am working with the small amount of publicly available skin lesion datasets available and I am working to create and access new datasets soon which are larger and more diverse," she reveals. This is proving to be the greatest challenge, however.

"There is a lack of diverse skin lesion data available and the landscape of clinical AI is still in its infancy," Tess adds. "My future plans are to create/source more diverse data and conduct a study to test this device in a real-world setting." She hopes the device will be well on the path to being used by real-life patients before 2030. 📺

Robot arm lamp

By Nikolay Tyulkin

rpimag.co/Watti

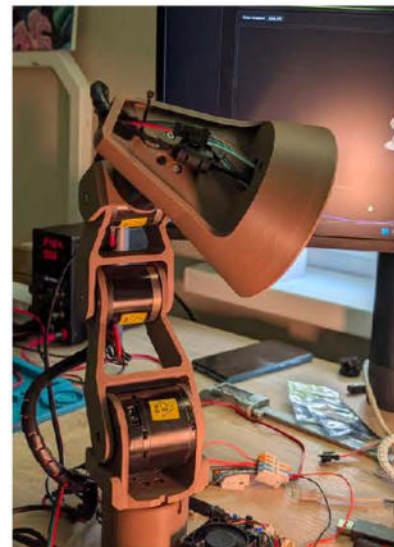
We've never met a lamp that we couldn't simply reposition by reaching out a hand and moving it a bit, so we don't need this moving, robotic lamp controlled by Raspberry Pi 5. We don't need it, but we do very much want one – it's an engineering masterpiece, it looks great, and... what more do you need?

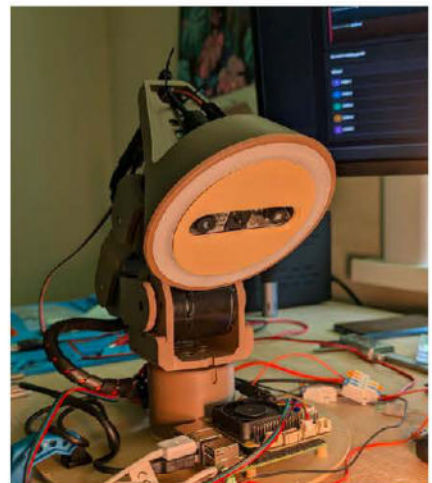
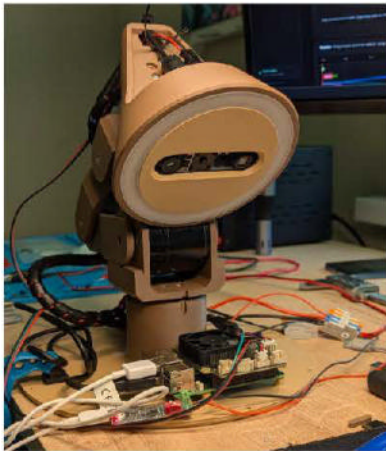
The lamp uses RobStride BLDC servo motors powered at 24V, connected over CAN. In case you're not familiar with CAN, it's a standard adopted by the automotive industry that enables devices to talk to each other without having to send all data through a central computer. In the early days of development, the robot lamp's creator Nikolay Tyulkin controlled its movement from his Windows PC; however,

he's recently ported the software over to Raspberry Pi 5, so motor control, lighting, and animation playback (for its movements) are all now handled locally rather than needing a separate computer.

The animations are still created on a separate computer. Each joint has its own timeline with position keyframes, velocity, torque limits, and controller parameters, and finished animations are sent to the lamp and played back from there. The light itself is provided by an addressable LED ring with a 3D-printed diffuser inside the head, and the brightness and colour of the LEDs can also be programmed as part of the animations.

- Watti (as Nikolay has christened his creation) uses ROS 2 on Raspberry Pi 5 to control its movements





The One Ring

By **MaterialFuture5319**

rpimag.co/TheOneRing

Keept it secret, keep it safe. Generations have been brought up with the *Lord of the Rings* books, the creepy cartoon, and the superb trilogy of films with their extended scenes of Sean Bean liberating Osgiliath. But few have made the One Ring themselves.

You don't need to cast this shiny object into the fire to make it reveal its true nature: the nearby presence of fire, or a hot cup of coffee, is enough to activate the (TMP390A2DRLR) thermostat circuit. A PFET transistor is used to turn on tiny surface-mounted LEDs on a PCB that runs around the inside of the ring, with 3D printed translucent filament to diffuse the light. It looks like some kind of elvish, but it's actually the language of Mordor, which we shall not utter here.

This project is slightly beyond our purview, as it does not rely on microcontrollers of any sort. However, it's a beautiful piece of craftsmanship and uses enough electronics that we can reasonably claim kinship with the creator, who has put the schematic of the One Ring up on Thingiverse for anyone to follow, be they elves, dwarves, men or even hobbits.

► One ring to rule them all, one ring to find them, one ring to bring them all, and in the darkness, bind them!





Pi-Cam-IR-A

By Keith Mallonee

rpimag.co/Pi-Cam-IR-A

This beautiful camera is a mobile infrared Pi-Cam-IR-A, featuring a Raspberry Pi Zero 2, 1.44-inch LCD screen, and Raspberry Pi NoIR Camera Module housed in a 1950s Kodak Duaflex II. That's enough, quite frankly – you had us at '1950s Kodak'. But there's more. It hosts its own web server, so you can upload pictures without having to return to your home network. And Raspberry Pi Zero handles HDR merging, so you can get the hyper-real effect of high dynamic range without having to upload images to a separate computer. The maker, Keith Mallonee, tells us: "I've always been what I consider a process-artist, being that I'm less interested in the finished refined product and more the process to get there and how adjustments change the outcome. For me, as a digital artist, that's what drew me to [cyanotype] kit. I did not have the Duaflex before I started. I built the original Pi-Cam-IR-A in a tin soup can and then shopped around eBay for something I thought/hoped it would all fit in. I started with two older cameras and the Duaflex ended up being the ideal candidate to modify."

► Great gear won't make you a great photographer. But it will make you much, much cooler





Astrolabe

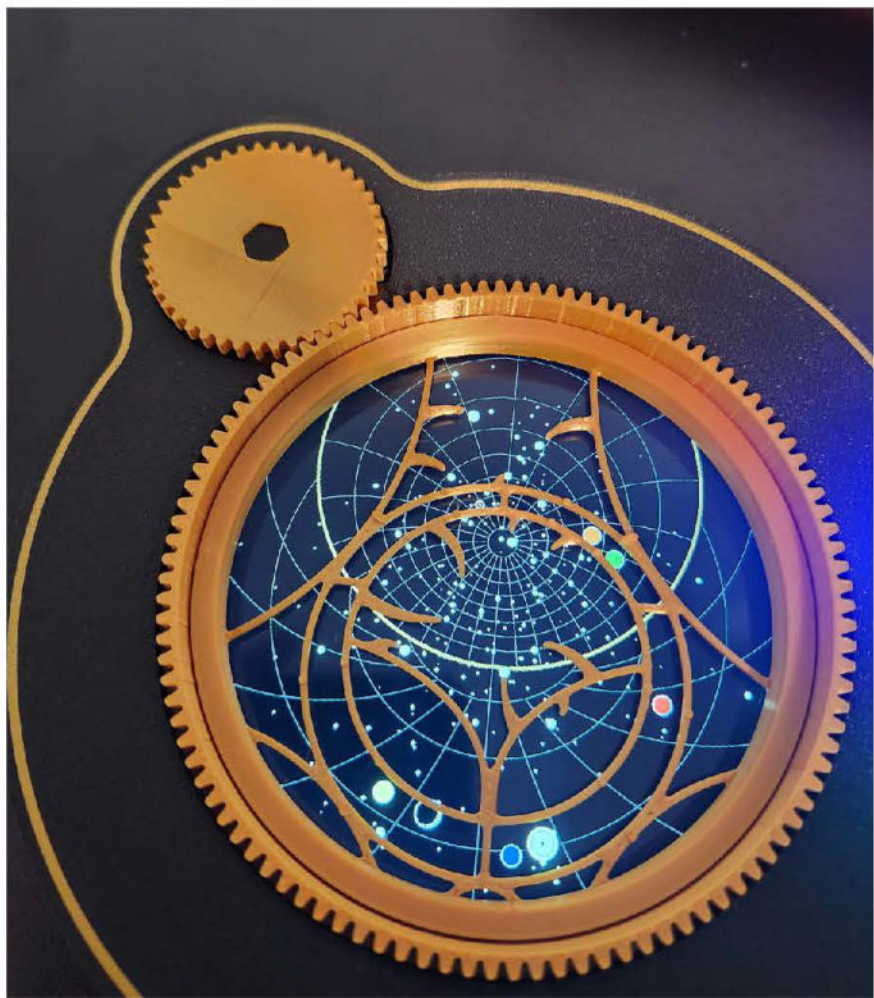
By Albert Chu

rpimag.co/Astrolabe

An astrolabe is a historical scientific instrument which was used to tell the time, map the sky, and find directions. First used in Ancient Greece, it was a kind of analogue computer for navigators. Fast-forward to the modern age, and Albert Chu has made one that uses an ESP32-S3 development board to calculate and display astronomical information on a 4 × 4 inch LCD panel. Give it a time and date, and it'll give you the co-ordinates of seven planets: the Sun, the Moon, Mercury, Venus, Mars, Jupiter, and Saturn. (The scientifically minded will note that not all of these 'planets' are, in fact, planets.)

- Unbelievably, this is Albert's first hobbyist electronics project

Albert's done a superb job with his implementation of this ancient tech, and has left a trail of instructions on GitHub for brave travellers to follow in his footsteps: he's included build instructions, a bill of materials, and a guide on how to use the Arduino IDE to flash the firmware onto the device. 🍷





3D print

If you love 3D printing, you've also got to learn to love sanding, filling, sanding, priming, and more sanding

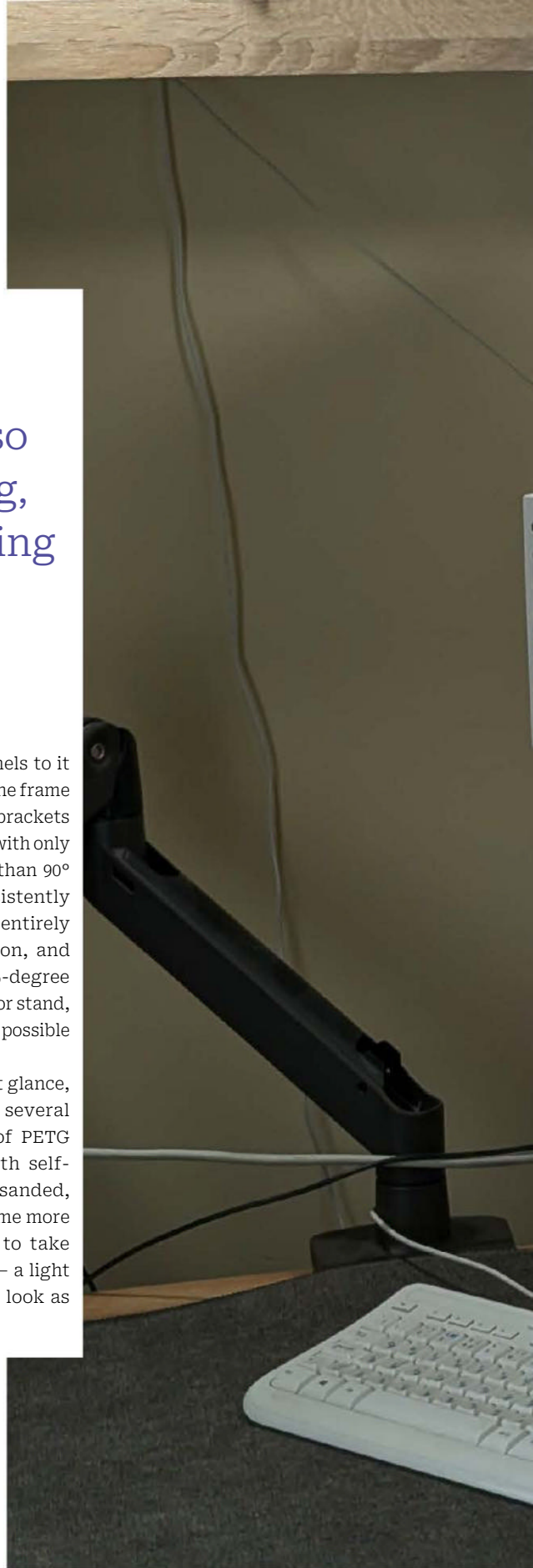
rpimag.co/TTU

In a world where we can customise anything, the humble computer screen has remained boringly rectangular. But it doesn't have to be this way, as Artur (aka selfdotcom) demonstrates in this beautiful build: the Tall Tower Unit (TTU). He calls the look and feel of this piece 'cassette futurism'; like steampunk, an imagined reality that grafts the look and feel of the late Victorian age on to modern equipment, cassette futurism takes the plastic aesthetic of eighties and early nineties tech and applies it to modern equipment, proving that we humans will get nostalgic over any piece of obsolete data storage format.

Despite appearances – the LEDs, the buttons, the cassette window, and the floppy disk slot – this is purely a case for a monitor in portrait orientation. Artur started with a wooden frame, and

added a load of 3D printed panels to it that he had designed himself. The frame used plastic 3D printed corner brackets – these were essential, because with only limited tools, any angle other than 90° would be difficult to get consistently right. 3D printing solved this entirely self-inflicted design restriction, and enabled Artur to add a few 45-degree angles on the back of the monitor stand, so that it would fit as far back as possible into the corner of a desk top.

You wouldn't know it at first glance, but the case was printed in several different parts (other than of PETG filament), and assembled with self-tapping screws. Artur then sanded, filled, and sanded and filled some more until the surface was ready to take several thin coats of RAL 7035 – a light grey chosen to give the same look as generic OEM hardware. 🍷





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.

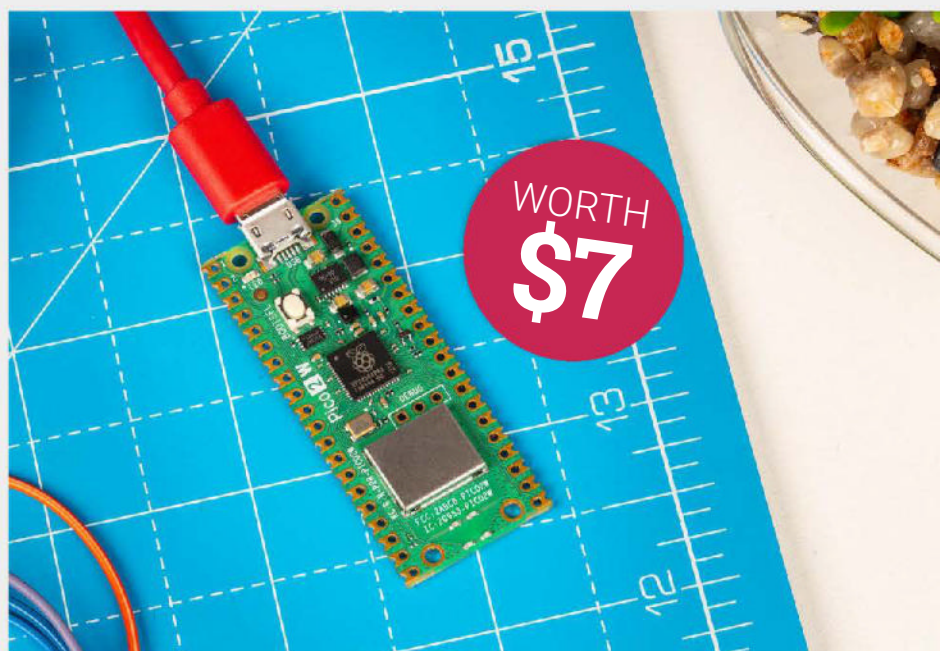
SUBSCRIBE TODAY AND GET A **FREE** Raspberry Pi Pico 2 W (or Pico 2)

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**

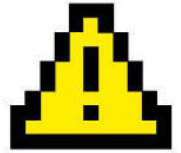
 Available on the
App Store

GET IT ON
 **Google Play**



➔ **The quickest and easiest way to
get gaming with Raspberry Pi**

By Rob Zwetsloot

**Alert!****Video game copyright**

Video game files are protected by copyright law. Be sure to use ROM files that have been released with the owner's blessing, or modern homebrew games designed to be shared. There are lots of legal options, discussed at the end of this feature.

YOU'LL NEED:

- Raspberry Pi 4/5 (2GB minimum)
- USB-connected controller or keyboard
- 32GB microSD card
- USB-C power supply
rpimag.co/powersupply
- HDMI cable
- TV/display

The modern games industry seems to be a mess right now. With lay-offs across big studios around the world, and an indie market getting ever more crowded, it's easy to conclude that the industry needs to slow down a bit.

Never fear though, as there are thousands of excellent video games spanning the last 40+ years to play, many of which you probably haven't had a chance to try. Thanks to years of community development and support, it's incredibly easy and fast to set up a Raspberry Pi-based retro gaming console to play these games too.

So let's head to a yesteryear of blast processing and emotion engines, and discover some new-old favourites...

**It's incredibly easy and fast
to set up a Raspberry Pi-
based retro gaming console**

Install Recalbox

For Raspberry Pi, Recalbox is the easiest way to play retro games, and it can be installed to the microSD card from another PC using Raspberry Pi Imager. This can be downloaded from: rpimag.co/imager.

Once you've installed it, launch it and select your Raspberry Pi device, then click **Next**. Scroll down on the **OS** screen and click on the **Emulation and game OS** category, then click **Recalbox**, then the Recalbox listed for your Raspberry Pi, followed by **Next**.

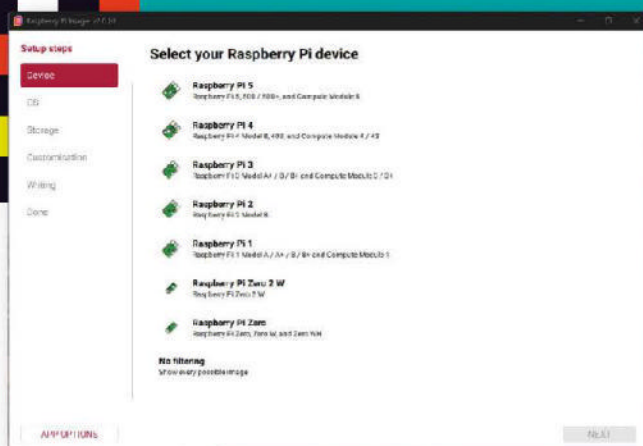
Choose your microSD card and click **Next**. You can skip customisation for now with the conveniently named **Skip Customisation** button, and finally you can use **Write** to start the installation process. It will take a few minutes to install, after which Imager will check if the installation succeeded.

Set up Recalbox

After a successful write to the microSD card, you can put the card into your Raspberry Pi. Plug your Raspberry Pi into your display, plug in your controller, and finally plug it into the power supply to boot into Recalbox.

You'll first be shown some screens talking about the features of Recalbox as it performs its final bits of set up, and then you'll be greeted by an interface showing various games consoles that Recalbox can emulate – that is, it can play the games of these classic consoles and retro computers. Recalbox is pretty clever and will likely allow you to use your game controller right away to navigate the menus, although you can use the settings menu to change up the layout of the buttons if they don't feel right to you.

From here you can also hook up Bluetooth controllers, connect to Wi-Fi, modify the look of the UI, and even the way games are shown on screen if you wish as well.



▶ Raspberry Pi Imager is a quick way to install operating systems to Raspberry Pi computers

Network install

If it's more convenient to plug in an Ethernet cable to your Raspberry Pi than try and find a microSD card reader for your PC or laptop, you can use the network install functionality to skip using a different computer: rpimag.co/netinstall

▼ Recalbox categorises games by system, and only shows those you have games for

Add games

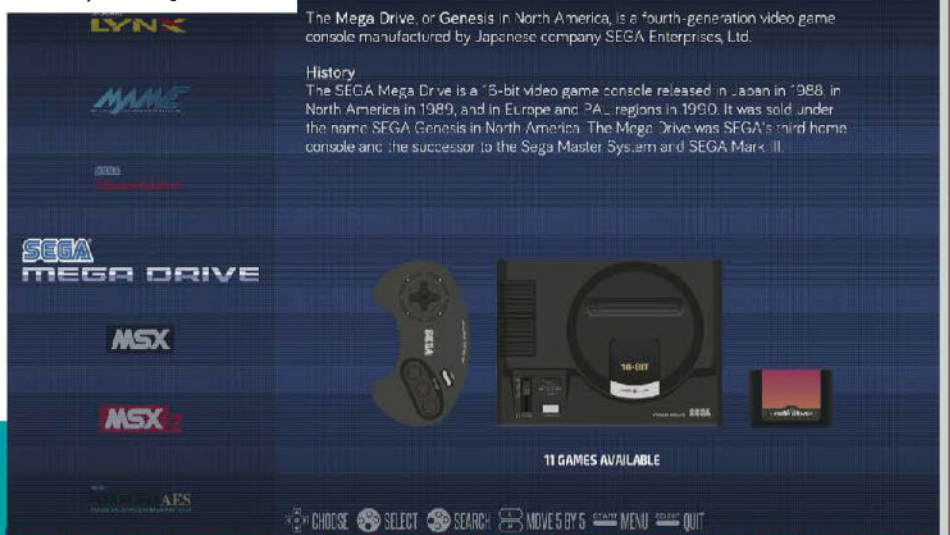
Games for old consoles are generally stored as ROM files – some free ones will already be on Recalbox when you install it. There are a few ways to add ROMs you already own, one of which is to load up a USB stick; another is to head to the share directory on your microSD card on which you installed Recalbox.

The USB stick method is fairly simple – plug one into your Recalbox and it will ask if you want to **Initialize** or **Move the Share**. The former makes it so all game ROMs are read from your USB drive, while the latter moves a good portion of the file system to your USB drive. For now, you can use **Initialize**. The downside here is that you need the USB stick plugged in at all times to play your ROMs; however, it's easy to unplug to take to another computer and transfer ROMs to the correct console folder in **recalbox > roms**.

The Mega Drive, or Genesis in North America, is a fourth-generation video game console manufactured by Japanese company SEGA Enterprises, Ltd.

History

The SEGA Mega Drive is a 16-bit video game console released in Japan in 1988, in North America in 1989, and in Europe and PAL regions in 1990. It was sold under the name SEGA Genesis in North America. The Mega Drive was SEGA's third home console and the successor to the Sega Master System and SEGA Mark III.



Move the share

Using the **Move the Share** function when plugging in a USB stick moves the entire contents of the share folder onto it. You'll still be able to put your game ROMs and other files on there, and you'll also find other folders for more advanced emulation aspects that will now live on the USB drive instead of on the microSD card.

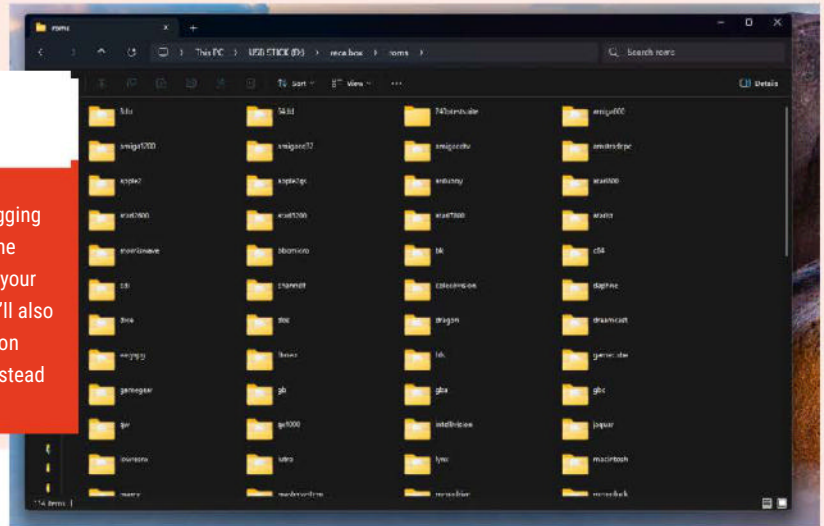
You can get to this share directory without moving any folders in two other ways: over the network, or simply plugging the microSD card into another PC (make sure to turn your Recalbox off before taking the microSD card out of your Raspberry Pi, though).

If you've set up the Wi-Fi or plugged in an Ethernet cable, Recalbox will automatically share a folder called **recalbox** over your local network. Open the file explorer on your system, head to Networks and look for **recalbox** listed in the folder. Open this folder, and you'll find the share folder.

From here you'll find several folders, but the one we're looking for is **roms**. Inside this are folders for the various systems that Recalbox can emulate – like with the USB drive, just drop your ROM into the correct folder.

When plugging the microSD card into another system, the process is similar. Open the microSD card after you've plugged it in and your computer recognises it, and head to the share folder.

Once you've added a game, go back to the menu, head to UI Settings and hit **Update Game Lists**. You can also just restart Recalbox.



▲ Game files are stored in folders named after the system they play on. Do you know your psx from your gbc?

Recalbox comes with some homebrew games

Read-only memory

The term ROM is used to describe the file that contains a retro game. The term comes from 'read-only memory', as older games would be stored on physical ROM chips in a cartridge or embedded into the mainboard of arcade machines.

Games from disc-based systems are technically not ROMs, and instead you'll find them called ISO images. This comes from the .iso file type for optical disc images, which contain all the data that would be on an optical disc. People will sometimes use the terms ISO and ROM interchangeably for disc-based systems.

Windows problems

Some Windows PCs may have issues accessing the share folder over the network. There are various solutions in the Recalbox Wiki: rpmag.co/recaltut

Sourcing your games

Having a fancy retro games machine is one thing, but without having anything to play on it, it would seem a bit pointless. While Recalbox comes with some homebrew games, not every emulator does.

The good news is that the lack of a Super Nintendo cartridge slot on Raspberry Pi doesn't mean you can't play some of your favourites from yesteryear. Cashing in on nostalgia, many companies re-release retro games and console collections on PC via **GOG.com** and Steam. Secretly a lot of these re-releases just include the files for the original game and an emulator to play it on, and in many cases we can use these files – or ROMs – on our own emulation machine.

Buying the classics

Due to the way these various game re-releases are distributed, online store fronts like Steam and **GOG.com** do not have an easy way to filter for them by default. However, the communities using these stores have stepped up to the challenge and maintain lists of games and collections that include retro game ROMs.

GOG lists

rpimag.co/gogroms

User WinterSnowfall maintains a list of all known releases on **GOG.com** that include ROMs that will work in other emulators. Many of them have ROMs just in the files you download from GOG; however, some will require extra tools to extract the games, which you use at your own risk.

To get to the folder of downloaded files from GOG, right-click on a game from the **Installed** tab, and then click on **Manage installation > Show folder** to open up the folder in your file explorer. Different games will include the ROMs in different locations in their file structure, so you may need to do a web search to find their exact location.

Steam lists

rpimag.co/steamroms1

rpimag.co/steamroms2

The first link here is a detailed guide on several collections that include ROMs from user Mykola1453, whereas the second link is for a Steam Curator group with lists of games and game collections with accessible ROMs that have handily called themselves Legal ROMs. There is some overlap on the lists, although unfortunately both have not been updated since 2024. Still, they're good places to start.

To get to your Steam files, right-click on a game in your **Library** tab, then go to **Manage > Browse local files** to open the folder. The ROM files may not be in an obvious location of the file structure; however, a web search should help you locate them.

Delisted titles

For various reasons, some collections and games on these lists are no longer available for purchase – on the bright side, if you did purchase them in the past, you can still download these games and grab the ROMs that come with them. This includes the popular SEGA Mega Drive & Genesis Classics Collection released in 2010 that had 58 games included.

Lists of games and collections that include retro game ROMs

Classic highlights

Name: Garou: Mark of the Wolves
Platform: Neo Geo
Release: 1999
URL: rpimag.co/garou

During the 1990s, Capcom and SNK were fierce rivals creating fighting games for the arcades and home consoles. One of SNK's most notable releases during this time was Garou: Mark of the Wolves, considered one of the greatest fighting games of all time – and for good reason. With a varied cast of characters, robust fight mechanics, and an unforgettable soundtrack, it's still played to this day by dedicated fighting gamers.



- ▶ Garou: Mark of the Wolves is a sequel to The King of Fighters and Fatal Fury series
- ▶ The 1990s truly were the golden age of point-and-click adventure games

Name: Rocket Knight Adventures
Platform: Mega Drive/Genesis
Release: 1993
URL: rpimag.co/rocketknight

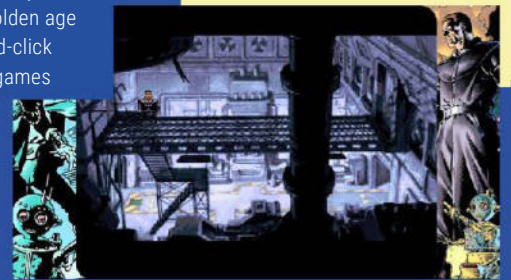
Back in the early 1990s, animal mascots were all the rage thanks to a certain quartet of turtles, and that concept made its way to video games by way of a certain speedy hedgehog. Konami's take was Sparkster, a possum rocket knight who had games across various 16-bit consoles. This game is an action platformer where you must fight enemies on a 2D plane with your sword that also fires energy bolts, and is part of a collection with other games in the series.



- ▼ The Sparkster series also made its way to SNES

Name: Beneath a Steel Sky
Platform: Amiga
Release: 1994
URL: rpimag.co/shadowsteel

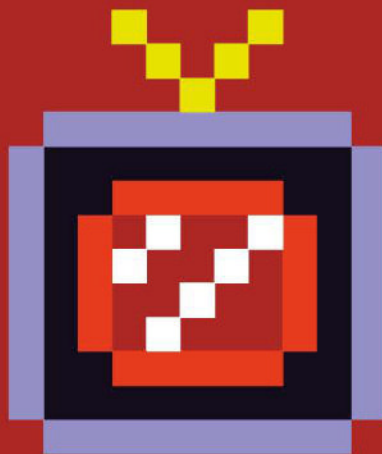
From the makers of the Broken Sword games comes the critically acclaimed point-and-click adventure Beneath a Steel Sky, set in a dystopian cyberpunk future where you must explore the world around you and talk to the denizens there to uncover and face the corruption plaguing Union City. This is part of a collection of other classic adventure games.



Independently distributed ROMs

Not everyone re-releases games on **GOG.com** or Steam, and creators in the past have sold or simply just given away their old games as ROMs online.

Legendary game dev Jeff Minter of Llamasoft does just that on his website minotaurproject.co.uk; however, some resources like this prove to be temporary as licences inevitably change hands.



Homebrewed newer games

If you want to play new games on these old systems, you'll be happy to know that people regularly make new games for old consoles and release them as ROMs for emulators. These are called homebrew games, and are unofficial/unlicensed games usually released long after the console has ceased manufacture.

Retro game preservationist, historian, and game developer K.G. Orphanides (who has written many retro gaming articles for this magazine) maintains lists of homebrew titles available on Itch.io – a site for indie, amateur, and hobbyist game devs alike. Here you'll find a wide range of experimental, silly, and even critically acclaimed games to suit just about any taste.

Nintendo consoles

- Game Boy: rpimag.co/itchgb
- Game Boy Color: rpimag.co/itchgbc
- Game Boy Advance: rpimag.co/itchgba
- NES: rpimag.co/itchnes
- SNES: rpimag.co/itchsnes

8-bit computers

- Commodore 64: rpimag.co/itchc64
- ZX Spectrum: rpimag.co/itchzxs

SEGA consoles

- Mega Drive/Genesis: rpimag.co/itchmd
- Game Gear and Master System: rpimag.co/itchms
- Dreamcast: rpimag.co/itchdc

Sony consoles

- PlayStation Portable: rpimag.co/itchpsp
- PlayStation 2: rpimag.co/itchps2

Other systems

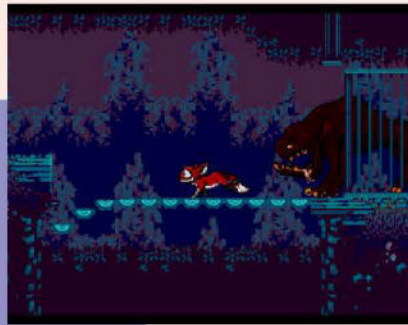
- Atari Lynx: rpimag.co/itchlynx
- Arcade: rpimag.co/itcharc
- DOS: rpimag.co/itchdos
- PC Engine: rpimag.co/pce
- Amiga: rpimag.co/itchamiga

People regularly make new
games for old consoles

Homebrewed highlights

Name: TANGLEWOOD
Platform: Mega Drive/Genesis
Release: 2018
URL: rpimag.co/tanglewood

This very successful platformer is still available as a Mega Drive ROM on Itch (and other places), but has since been ported to many modern consoles – it's a true indie success story. It's a platformer where you play as a fox as you make your way through a beautiful fantasy world as night falls.



▲ Find unique ways to defeat monsters on your perilous journey



Name: Far After
Platform: Game Boy
Release: 2022
URL: rpimag.co/farafter

A throwback action-RPG with extremely cute pixel graphics, Far After shows how Game Boy games could have evolved if people were still making games for it – and you'll find that a lot with homebrew games as they use modern coding techniques and tricks to get systems to look and perform better than in their heyday.

Name: What Remains
Platform: NES
Release: 2018
URL: rpimag.co/whatremains

In Japan, the NES/Famicom was home to several visual novels that never made their way over the West (until recently), so it's no surprise to see homebrewed visual novels for the console. In this game, you play as a skateboarding teen who discovers a mysterious game for their new NES (very meta), launching a quest to save the world.



◀ The game leans into the slang of the time, or at least a parody of it

AI generative picture frame

Create ever-changing and uniquely strange artwork and learn about Python processes and threads



Maker

Rob Miles

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

robmiles.com

DOWNLOAD
THE FULL
CODE:



[rpimag.co/
ai-ink-display](http://rpimag.co/ai-ink-display)

YOU'LL NEED

- Raspberry Pi (from Raspberry Pi Zero 2 W upwards)
- microSD card – between 4GB and 64GB. You will need a larger card if you want to install multiple models on the same machine
- Display – the author used a 7.3-inch e-paper display from Pimoroni (pimoroni.com)

Memory conservation

This version of Stable Diffusion gets around the limited memory of small devices by creating the image from smaller tiles that are stitched together. This makes it possible to create results on a machine with only 512MB of random access memory (RAM).

AI image generation lets you create a picture from a text prompt. By building random prompts from a pool of keywords, we can create themed images which update as fast as your Raspberry Pi can compute new versions. In this article, you'll discover how to do this using an application that lets you view images, change themes, and configure the system from your browser. You'll also find out how Python processes and threads underpin the operation of the code. All the software is available for download at rpimag.co/ai-ink-display. The repository also contains full instructions for installation and configuration.

Figure 1 shows a generative picture frame showing an image created from the prompt “an astronaut, surrounded by wildflowers, epic, delicate atmosphere, high detail, beautiful composition”. The Stable Diffusion program fed the prompt into a neural network trained on many thousands of tagged images. The network started with an image containing random noise and turned it into a recognisable picture by repeatedly refining the contents.

Signals from noise

The Stable Diffusion program makes repeated passes over an image as the picture is refined and detail added. **Figure 2** shows this in action. The prompt is shown at the bottom, and above are thumbnails for each successive pass. The picture used ten passes which took a total of 90 minutes to complete on a Raspberry Pi 4. Further passes would add more detail.



Generative origins

This project was inspired by the article 'Generate images with Stable Diffusion on a Raspberry Pi 5 1GB' in issue 164: rpimag.co/164. You can follow the instructions in the article to get Stable Diffusion running on your machine. The author did this and then remembered he had an e-paper display he could use with a Raspberry Pi 4 to make an ever-changing picture frame.

◀ **Figure 1:** The image is displayed on an e-paper device. You can use an HDMI screen instead

▼ **Figure 2:** You can ask Stable Diffusion to save the results of successive passes

QUICK TIP

The number of passes you need is different for each model. The Stable Diffusion program has a download option which is used to load models onto your device.

The program didn't make the picture in **Figure 2** because it knows what a camera looks like; instead, a neural net inside a large data model maps the words 'camera' and 'cliff' onto similarly tagged image elements and uses their structure to modify the starting image.

The picture frame application generates random prompts and feeds them into a local copy of the Stable Diffusion program. It then displays the results. The software will run on machines as low-powered as a Raspberry Pi Zero 2 W. It doesn't require a network connection to work (although one can be useful for configuration and viewing images). Let's look at how you use it.



"newspaper photograph of an old camera, on a windswept cliff, twilight glow, uplifting, carefully balanced composition, high detail, beautiful composition"

Generates random prompts and feeds them into a local copy of Stable Diffusion

Drawing a bicycle

The image creation process is not perfect. The software will quite happily create dancers with three or more legs, and hands with an incorrect number of fingers, but the author has learned to accept this and rather appreciates the highly visible reminders that artificial intelligence is not infallible. And some of the results are very funny.

- ▶ Performing extra passes will not fix this problem, but you can try adding more detail to the text request

▼ **Figure 3:** Click the Live button to make a full-screen display of the images as they are generated

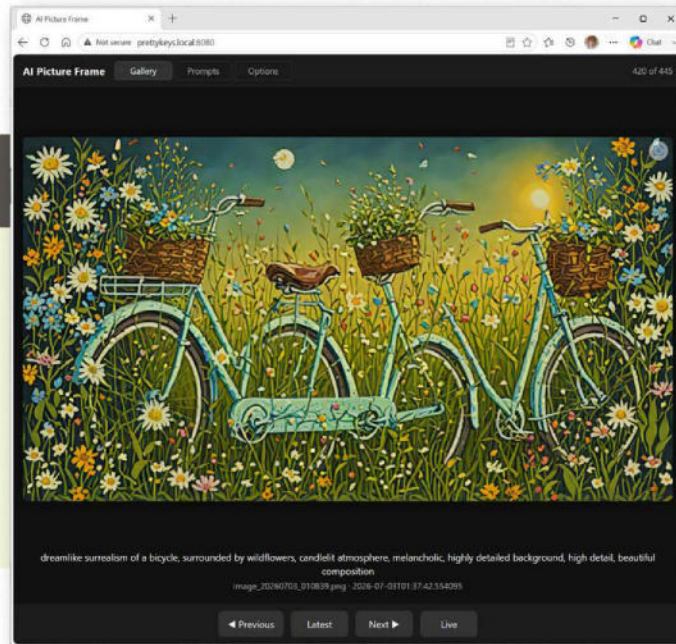


Figure 5 shows the configuration options. You can select an HDMI or e-paper display. The Pimoroni displays are fitted with buttons which can be used to step through previous images or move to the latest one. If your machine has a keyboard, you can use that to navigate the display. You can also set how generated images are fitted to the display. The application works well and the author has used it on a variety of systems to create interesting and ever-changing displays. Now, let's look at how it works.

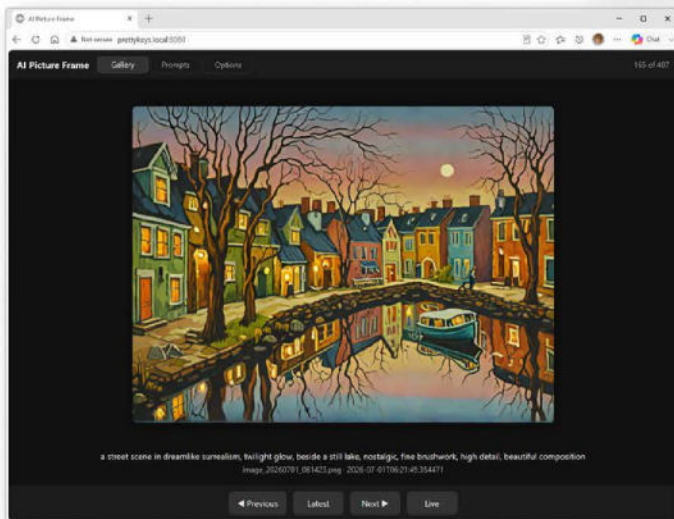
Python in control

The Stable Diffusion software is not written in Python. It is a C++ program that you build from its source code during the install process (details of how to do this are in the repository for the project). The author could have created an AI picture frame application by adding more code to the Stable Diffusion source, but this would have been a lot of work, particularly as he wanted to make the program host a web page that could be used to view the images and configure the application. So instead, he wrote a Python program that creates the prompt string, runs the Stable Diffusion program as a separate process, and then presents the results. Python is extremely useful for in situations like this. It can start and stop programs and view the results using the `subprocess` module. Let's look at how we do this.

```
import subprocess
```

The first thing we need to do is import the `subprocess` module. Now that we've got that we can create a new process.

Now we can use the `Popen` method provided by `subprocess` to create a variable called `sd_process` which will refer to the running process.



The AI picture frame application

Figure 3 shows the web page hosted by the AI picture frame application when it is running. You can navigate to this using a browser running on a machine connected to the same network as the picture frame and use it to view images. Use the URL `machinename.local:8080`. Clicking the Prompts button at the top of the page displays the prompt editing page.

Figure 4 shows the web page used to specify the topics for picture generation. At the bottom of the page, you can set an additional hint which applies to all images. Not all the items have the effect that you might expect, but it is worth experimenting with them. The Options page lets you configure the display and input options and provides access to the settings for the Stable Diffusion program.

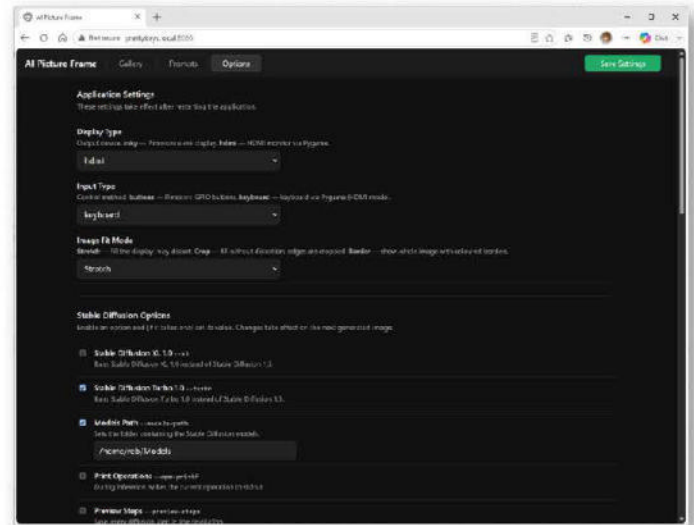
Turbo and XL Stable Diffusion modes

There are three versions of Stable Diffusion that you can use to make images. They are selected by setting flags on the options page you can see in **Figure 5**. If you don't specify the version that you want, you will get Stable Diffusion version 1.5. If you select either of the two other options, you can use Stable Diffusion XL 1.0 or Stable Diffusion Turbo 1.0. Each of these has a characteristic style and each requires its own model file. You can find out how to download models in the repository for this article. The Stable Diffusion Turbo version allows you to set the resolution of the output; the other versions always make an image which is 512 pixels square.

QUICK TIP

You can use this technique to run any program on your system, including system commands.

- ▶ **Figure 5:** All the settings are stored in a JSON (JavaScript object notation) file
- ▼ **Figure 4:** The program picks a random template and then selects a random word from the topics for each item in the template



```
sd_process = subprocess.Popen(
    ["/sd", "--prompt", "a lighthouse",
    "--output", "out.png"],
    stdout=subprocess.PIPE,
    text=True
)
```

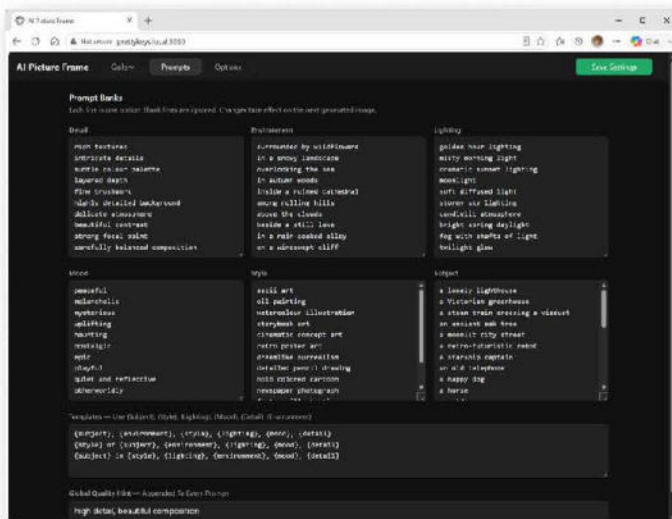
The statement above uses the **Popen** function in **subprocess** to start a process running Stable Diffusion. You'd run this Python code from the folder containing the **sd** executable. The **Popen** function is given details of the command to be performed as an argument containing a list of strings that describe the command. What you see above is the equivalent of typing the following at the command line:

```
$ ./sd --prompt "a lighthouse" --output "out.png"
```

This command asks Stable Diffusion to draw a lighthouse and put result in a file called **out.png**. We could add other strings to specify the model location and the version of Stable Diffusion to use.

After the list of command elements, we have two further arguments to the **Popen** function which allow a Python program to 'see' what the **sd** program is doing. When Stable Diffusion runs, it displays status message on the console. We can configure the **sd_process** to send these messages into our Python program as it runs. This is super-powerful. You can make a Python program that runs a command and sends you an email if the command fails with an error message.

```
stdout=subprocess.PIPE,
text=True
```



Start and stop programs and view the results using the subprocess module

The lines here are the last two arguments to the `Popen` function that is creating the process. The first line sets the `stdout` argument to the value `subprocess.PIPE`. This tells `Popen` to capture messages from the application. The second line sets the argument `text` to `True` so that `Popen` delivers the captured messages as text. At this point, the process is running and generating output. Now we need some code to do something with the output messages:

```
for line in sd_process.stdout:
    print(line)
```

The statements above create a loop which repeatedly reads messages from the `sd_process` output stream (`process.stdout`) and prints them on the console. Every message that `sd` displays will appear on the console where the Python program is running. We could add code to inspect the message strings and detect errors, but for this demonstration the messages are just printed. It is important to remember that the Python program will get messages from the application as it is running, not when it finishes.

QUICK TIP

It is possible for a Python program to send messages to the standard input of a process too, but we don't need this for our solution.

The output stream of a process is closed when the process ends. So, the `for` loop will complete when the `sd` application has finished. We can use the `wait` function on the process. This pauses program execution until the process has ended:

```
sd_process.wait()
```

If you think about it, we don't really need to use `wait` because the `for` loop above will only complete when `sd` has finished. However, if you create a Python program that doesn't read the process output, you will need to use `wait`. Our code can now check the return code from the process:

```
print(f"Exit code: {sd_process.returncode}")
```

QUICK TIP

You can start multiple applications simultaneously.

The code here just prints the `returncode` value, but we could use an `if` construction to make our program do different things depending on the outcome. The convention is that the value of the `returncode` property will be 0 if the application finished successfully. Any other value will be an error code. Some error codes are defined for a particular application; others are set by the operating system. The AI picture frame application repeatedly runs the Stable Diffusion application in this way. You can find the code that does this in the `generator.py` source file.

We need another form of multitasking too

Thread bearing

Now we know how a Python program can run at the same time as an application it is controlling. But we need another form of multitasking too. At the same time as Stable Diffusion is running, we need the picture frame application to respond to user inputs and host a website which allows users to view the images in their browser and change the image prompt. We could run each of these components as a separate application, but it would be very difficult to coordinate their operation. So, we use another multitasking feature of Python: the thread. Let's look at how threads work, starting with a function that we might like to run as a thread.

```
import time

def worker(name):
    print(f"{name}: starting")
    time.sleep(1)
    print(f"{name}: done")
```

The `worker` function above accepts a string as a parameter. It prints out starting and done messages and uses the `time.sleep` function to wait for a second when it runs. A program calling the `worker` function is paused for a second because that is how long the function takes to complete and return. However, we could run this function as a thread:

```
import threading
t = threading.Thread(target=worker,
args=("Thread-1",))
t.start()
print("Worker has started running")
t.join() # wait for it to finish
print("Worker has finished")
```

The code above creates a thread called `t` which runs the `worker` function at the same time as the main program. The thread is given an argument (in this case the string `"Thread-1"`) which the `worker` function prints when it runs. When the program calls the `start` method on the thread, the `worker` will start running. When the program calls the `join` function on the thread, the program will wait until the thread finishes. We can improve our understanding of what is going on by looking at the output from this program:

```
Thread-1: starting
Worker has started running
Thread-1: done
Worker has finished
```

The program creates the thread and calls `start` on the thread. The worker function prints the message `"Thread-1: starting"` and then goes to sleep for one second. The main program prints the message `"Worker has started running"` and then calls the `join` method on the thread to wait until the `worker` has finished. When the `sleep` in the `worker` completes, the function prints `"Thread-1: done"` and returns, ending the thread. The `join` method then returns to the main code, which prints `"Worker has finished"` and then ends the program.

QUICK TIP

The `demos` folder in the repository for this project contains copies of the programs shown here.

Threads are very useful as a way of breaking an application down into a set of cooperating elements. However, the simple threading that we have just seen is not as useful as it might be because once a thread has been created, it is impossible to stop it:

```
def evil_worker():
    while(True):
        print("Har Har")
```

If the `evil_worker` function above was started in a thread, it would repeatedly send messages and consume CPU time until the main program completed execution. Note that it would not 'bring the computer to its knees' because Python repeatedly swaps between active threads, but it would consume processor time and slow everything else down. For this reason, the `worker` threads in the AI display are created in a slightly different way, by extending the `Thread` class. This technique gives us a way of stopping a `worker` thread after we have started it. The `Worker` class below shows how we do this:

```
class Worker(threading.Thread):
    def __init__(self):
        super().__init__()
        self._running = True

    def run(self):
        while self._running:
            print("working")
            time.sleep(0.5)

    def stop(self):
        self._running = False
```

The code above defines a class called `Worker` which is an extension of the `threading.Thread` class. We call `Worker` the child class and `threading.Thread` the parent. The child class picks up everything from its parent and can add its own properties and behaviours on top. So, a `Worker` can do everything a `Thread` can do, plus whatever we add to the `Worker` class.

The `__init__` method in `Worker` runs when an instance of the `Worker` class is created:

```
def __init__(self):
    super().__init__()
    self._running = True
```

The code above shows the `__init__` method in `Worker`. It uses the Python provided `super()` method to call the `__init__` method in the parent `threading.Thread` class. This makes sense. To make a `Worker`, we need to make a `threading.Thread` instance too. The `__init__` method in `Worker` also creates a data member in the `Worker` instance called `_running` which is set to `True`. We will set the value of `_running` to `False` to stop the `worker` task. The `Worker` class defines a `run` method which contains the code that the thread will perform.

```
def run(self):
    while self._running:
        print("-- working")
        time.sleep(0.5)
```

The code above shows the `run` method in our `Worker` example. The `run` method here displays a message every half second. We stop a `worker` thread by setting the `self._running` property to `False`. When we do that, the loop in the `run` method will end and the thread will finish.

```
def stop(self):
    self._running = False
```

When a program wants to stop the thread from running, it calls the `stop` method. Let's look at a complete example:

```
w = Worker()
w.start()
print("Worker has started running")
time.sleep(2)
print("Stopping the worker thread")
w.stop()
w.join()
print("Worker has stopped")
```

The code here shows how we would create and control a `Worker` instance called `w`. The first statement creates the thread. The second statement starts the thread running. The main thread then sleeps for two seconds and then stops the `worker` thread. It then uses the `join` function to wait for the `worker` thread to finish. The output from this code looks like this:

```
Worker has started running
-- worker working
-- worker working
-- worker working
Stopping the worker thread
Worker has stopped
```

Above you can see the output produced by this program. The `worker` managed to complete three loops within the `run` method before it was stopped by the main one. If we changed the time that the main method sleeps (by increasing the value 2 in the program above) the `worker` would run for longer.

Process vs thread: fight!

Processes and threads allow Python programs to work on several tasks at once. They are slightly different, though. A process is a running program created in the operating system of your computer which runs independently of your Python program. The computer operating system decides when a process runs. If the AI display program is stopped, the Stable Diffusion process that it started will continue to run. However, if we stop a Python program containing multiple active threads, all executing threads will be stopped too. It is rather hard for a Python program to communicate with an application started via a process. However, threads are running alongside each other within Python. They can use events to send signals to each other and pass data using shared variables. Both processes and threads are useful, and the AI picture frame is a great way to explore how they are used. The code is well documented (the author thinks) and would repay careful study. It is also quite fun to use. 🍷

Raspberry Pi *Essentials*

AI Projects with Raspberry Pi

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



Lucy Hattersley,
Brien Jepson

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

Test AI Camera imaging models with data injection

Use Sony's data injection tool to determine the quality of AI Camera models



Maker

Lucy Hattersley

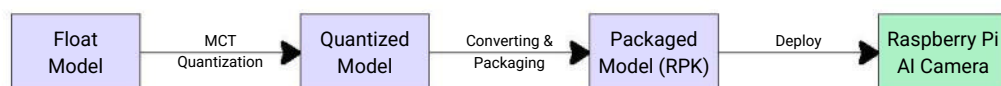
Lucy is editor of *Raspberry Pi Official Magazine*. She's busy testing out AI Camera on Gus, her toy gorilla.

rpimag.co

YOU'LL NEED

- Raspberry Pi 5
rpimag.co/raspberrypi5
- Raspberry Pi AI Camera
rpimag.co/aicamera

▼ **Figure 1:** A typical model development flow for AI Camera



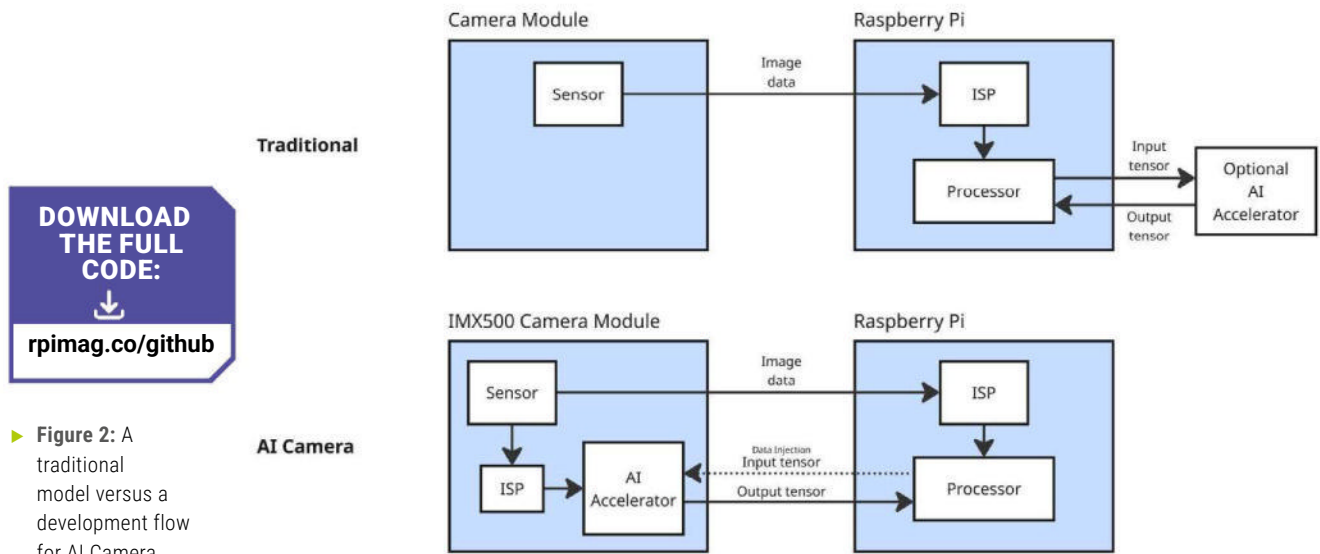
When you're developing a computer vision application for the Raspberry Pi AI Camera, evaluating your model in the real world is essential to guarantee the quality of your application. This is where data injection comes into the picture.

Data injection allows you to evaluate the performance of AI models running inside Raspberry Pi AI Camera. This tutorial explains why data injection is so valuable, how to do it, and how to apply it to your model.

The Raspberry Pi AI Camera is a powerful attachment that enables you to create vision AI applications. The on-board IMX500 sensor is capable of running neural network models to perform real-time image inferencing. Typical uses include placing bounding boxes around detected elements, and labelling them.

Models deployed to AI Camera are packaged into an RPK format. A typical model development flow for the AI Camera would look something like **Figure 1**. You have a floating-point model in your preferred framework (e.g., PyTorch) that is ready for deployment to the AI Camera. The framework typically provides baseline performance and accuracy metrics. The objective is to deploy this floating-point model to the AI Camera while maintaining performance metrics as close as possible to the original model.

The memory constraints of Raspberry Pi AI Camera (8,388,480 bytes for firmware, network weight file, and working memory) requires quantisation of the floating-point model, compressing the model size. A starting point on



how to use Model Compression Toolkit (MCT) for quantisation can be found here (rpimag.co/mct-tutorial). Again, one should have an idea of model performance and accuracy metrics after the quantisation step.

Finally, the compressed model should be converted to the IMX500 AI accelerator format. Running the IMX500 Converter and IMX500 Packager will make sure all different layers in your model are compatible with the architecture of the IMX500 and will package the model into a firmware file (.rpk) that can be loaded at runtime onto the camera. You can run these steps manually (rpimag.co/aicam-deploy) or use Modlib's models (rpimag.co/aicam-custommodels) to automatically convert, package, and deploy the quantized model to the AI Camera.

The goal of data injection is to evaluate the performance of the RPK model (quantized, converted, and packaged) that is running on AI Camera hardware. So you can compare the performance of the model running on the edge with the performance of the quantized model and the floating-point model.

Data injection on the AI Camera

Unlike traditional camera module setups, where sensor image data is processed on the host device (or an optional external accelerator), the IMX500 camera module uses its internal image signal processor (ISP) to preprocess sensor data into an input tensor that is ready for the IMX500 AI accelerator (see **Figure 2**). The Raspberry Pi receives the model's output tensor from the AI Camera which, depending on the model, may already be post-processed. This architecture offloads all compute-intensive operations, making the Raspberry Pi's resources fully available for your application.

Our goal during data injection is to prepare an input tensor from an image in our dataset, and inject it into the IMX500 AI accelerator so we can evaluate the performance of our packaged RPK model.

The goal of data injection is to evaluate the performance of the RPK model

First data injection example

To prepare the Raspberry Pi for data injection, first disable the AI Camera cache. Update the `/boot/firmware/config.txt` file:

```
$ sudo nano /boot/firmware/config.txt
```

Adjust the `camera_auto_detect` setting (on line 17 by default) to `0` and add the extra text to manually add the IMX500 camera and disable the cache:

```
# Automatically load overlays for detected
cameras
camera_auto_detect=0

# IMX500 camera, disable cache
dtoverlay=imx500,cam0,bypass-cache
```

Now perform a reboot: `sudo reboot`.

Undo edits

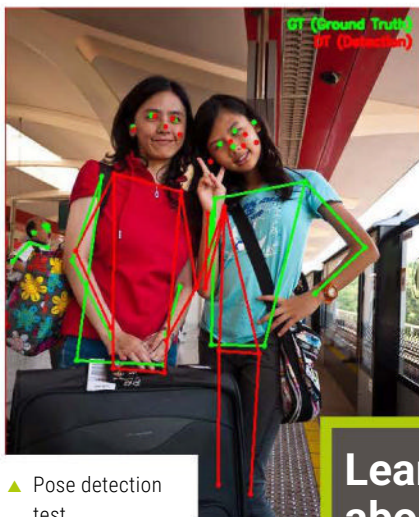
When you have finished testing data injection and want to return to the original inference settings, undo this by editing `/boot/firmware/config.txt`, remove `dtoverlay=imx500,cam0,bypass-cache`, restore `camera_auto_detect=1` and reboot.



Warning!

Random art

Image generators produce random artworks and although there are 'not safe for work' checks built-in, you may still get images that are NSFW. Children should be supervised.



▲ Pose detection test

To run data injection, you can either use Modlib's Data Injection APIs directly or use the evaluator tool in the Sandbox, which integrates those APIs and supports scriptable evaluation across every zoo model as well as extensions for benchmarking your own custom model.

You'll need a program called uv created by Astral to use data injection. Open a terminal window and enter:

```
$ curl -LsSf https://astral.sh/uv/install.sh | sh
```

Now we'll clone the aitrrios-rpi-sandbox repository and fetch all assets and example images. The evaluator looks for a **dataset/ground-truth** under **./examples/assets/** when using zoo models. To populate them:

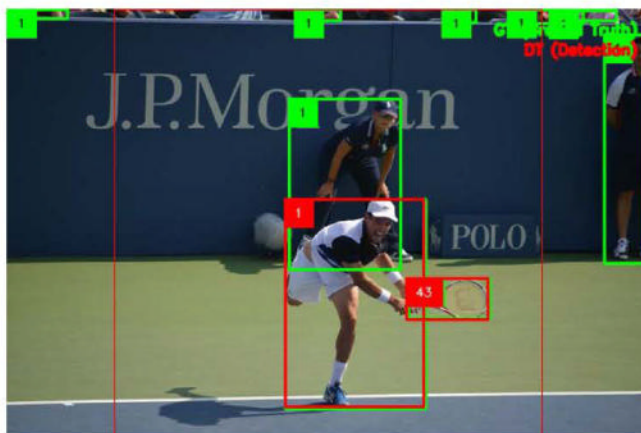
```
$ git clone https://github.com/SonySemiconductorSolutions/aitrrios-rpi-sandbox
```

Navigate into the **evaluation** folder:

```
$ cd aitrrios-rpi-sandbox/examples/evaluation
./fetch_modlib_assets.sh
```

Then run data injection:

```
$ uv run evaluator aicamera -m yolo11
--display_image
```



◀ The ground truth (green) is manually added so image detection quality can be tested

Learn more about uv

Astral's uv is an extremely fast Python project manager written in Rust. It is a single tool that replaces pip, pip-tools, pipx, poetry, pyenv, twine, and virtualenv.

rpimag.co/uvsetup

When the script has finished, you will get output results, like this:

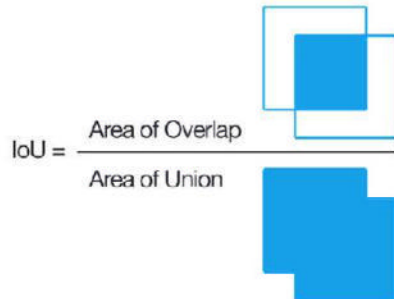
```
Average Precision (AP) @[ IoU=0.50:0.95 | area=
all | maxDets=100 ] = 0.397
Average Precision (AP) @[ IoU=0.50 | area=
all | maxDets=100 ] = 0.631
Average Precision (AP) @[ IoU=0.75 | area=
all | maxDets=100 ] = 0.391
Average Precision (AP) @[ IoU=0.50:0.95 | area=
small | maxDets=100 ] = 0.335
Average Precision (AP) @[ IoU=0.50:0.95 |
area=medium | maxDets=100 ] = 0.395
Average Precision (AP) @[ IoU=0.50:0.95 | area=
large | maxDets=100 ] = 0.963
Average Recall (AR) @[ IoU=0.50:0.95 | area=
all | maxDets= 1 ] = 0.284
Average Recall (AR) @[ IoU=0.50:0.95 | area=
all | maxDets= 10 ] = 0.451
Average Recall (AR) @[ IoU=0.50:0.95 | area=
all | maxDets=100 ] = 0.456
Average Recall (AR) @[ IoU=0.50:0.95 | area=
small | maxDets=100 ] = 0.370
Average Recall (AR) @[ IoU=0.50:0.95 |
area=medium | maxDets=100 ] = 0.400
Average Recall (AR) @[ IoU=0.50:0.95 | area=
large | maxDets=100 ] = 0.975
COCO evaluation results: [0.39703574 0.63055813
0.39108911 0.3350135 0.39534653 0.96262376
0.28428571 0.45142857 0.45571429 0.37 0.4
0.975 ]
```

What we can gather from this is that our yolo11 model achieves moderate performance just shy of 40% across all areas (AP@0.50:0.95 = 0.397). However, there is a sharp increase (over 96%) on detection with large objects (AP = 0.963, AR = 0.975).

This kind of information can be invaluable when determining which model will be most effective with your real-world values.

- **Figure 3:** The formula for determining the Intersection of Union: Area of Overlap/Area of Union. The result is a number between 0 (no overlap) and 1 (complete match).

Credit: By Adrian Rosebrock, rpimag.co/pytorchiou

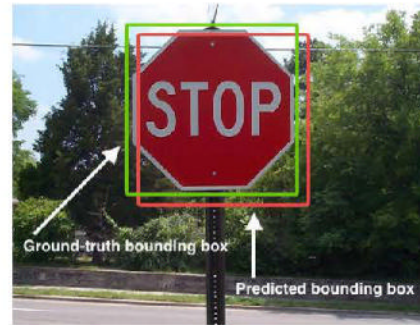


Let's decode the output:

- **AP (average precision).** A score between 0 and 1 (the value at the end of the line, i.e: = **0.397**). This is a measure of the number of predictions that meet the ground truth at that IoU (intersection of union) threshold. Typically, the higher the threshold, the lower the AP as it is more difficult to detect in a stricter bounding box.
- **IoU (intersection over union).** IoU is a metric that evaluates the extent of overlap between two bounding boxes. It measures the ratio of the Area of Overlap/Area of Union (see **Figure 3**). A higher IoU score implies a more accurate bounding box prediction (see **Figure 4**). The test runs ten AP and AR tests ranging from 0.50 IoU to 0.95 IoU.
- **AR (average recall).** The number of elements in an image accurately identified.
- **Maxdets (maximum detections).** The maximum number of objects to be detected in an image.
- **Area = small, medium, large.** The COCO dataset categorises objects by ground-truth area of the bounded object (not the overall image size). Small: area < 32² (32 × 32 box); Medium: 32² to 96²; Large: > 96².



- ▲ **Figure 4:** A higher IoU score indicates a closer match between the ground truth of the bounding box and the detection
- rpimag.co/pytorchiou**



- ◀ A ground truth bounding box (green) and predicted bounding box (red)

Ground truth

The 'ground truth' is the reality of the image created manually by a human, who will look at an image and draw a bounding box around each item and label them manually. The predictions are measured against 'ground truth' values.

This blog post on picsellia.com has a good explanation of COCO metrics: rpimag.co/cocometrics.

For a full list of models to test: `uv run evaluator list-models`.

You can run data injection of other models using `uv run evaluator aicamera`:

```
$ uv run evaluator aicamera -m squeeze
--display_image
```

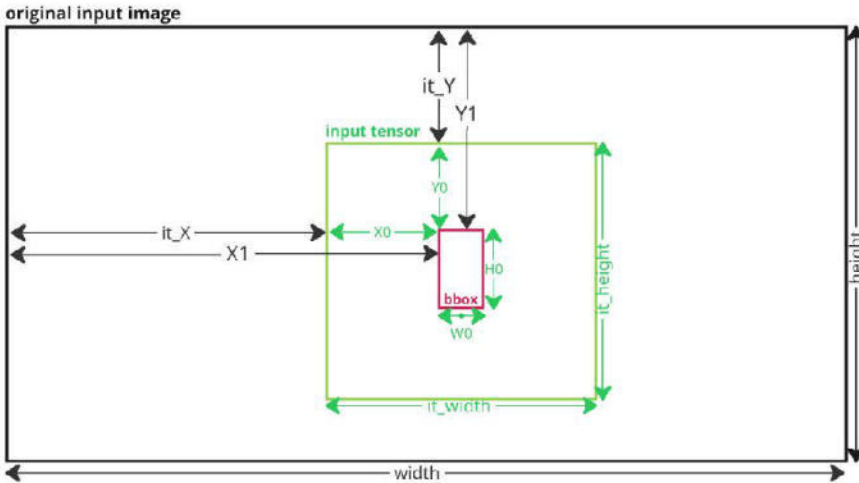
Note that different models are tested with different ground truth and compute metrics depending on the model. Some, such as deep, are tested with pixel accuracy and others, such as squeeze, are tested with per-class accuracy.

For a full list of available arguments, run: `uv run evaluator aicamera -h`.

Order of operations

Every data injection example follows a similar order of operations:

1. Pre-process the input image to an input tensor available for injection in the IMX500 AI accelerator.
2. Inject the input tensor and post-process the output tensor according to the model specification. This automatically happens by the `device.inject` method and Modlib's definition of models and `Model.post_processor`'s. Just like running the AI Camera in normal operation.
3. Evaluate the resulting detection against the ground truth and compute performance/accuracy metrics for our model.
4. Visualise the resulting detections with respect to the original input image. Find visualisations against the ground truth in `tmp/annotated_images` or enable the `--display_image` flag during data injection to see the result directly.



◀ **Figure 5:** The region of interest

To cope with any resizing influences, all these bounding box coordinates are (by design) internally normalised to the width and height of its coordinate frame: $(x0, y0, w0, h0) = (X0/it_width, Y0/it_height, W0/it_width, H0/it_height)$. Additionally, we define the ROI to contain how the input tensor coordinate frame relates to the coordinate frame of the

Region of Interest (ROI)

The Region of Interest (ROI) contains the information on how the coordinates of your resulting detections relate to your original image and is used during visualisation of the output detections. We can explain using a general object detection bounding box (**Figure 5**).

Let's say the preprocessing of your model requires cropping the original image to a certain ROI indicated by it_X , it_Y , it_width , it_height . After creating the input tensor (resizing, normalisation, etc.), the model's output bounding boxes will, as a result, be defined in the coordinate system of the input tensor $(X0, Y0, W0, H0)$. So, when we visualise/evaluate the resulting model detections against the original input image, it is important to convert the detection coordinates to the coordinate system of the original image $(X1, Y1, W1, H1)$.

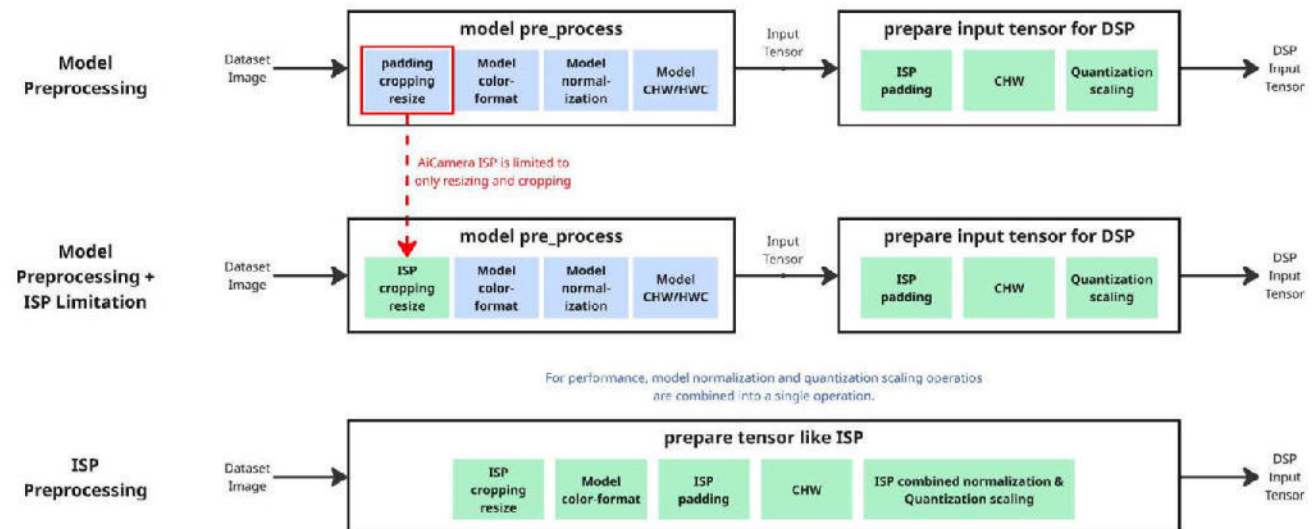
original input image. $ROI = (it_X/width, it_Y/height, it_width/width, it_height/height)$. Defining and storing the ROI like this will simplify calculation later on.

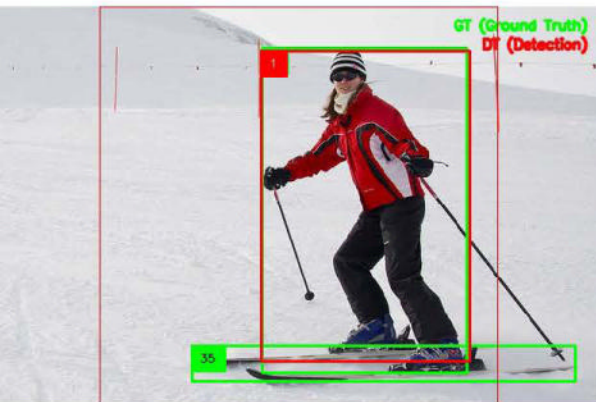
So, transforming the boxes to the coordinate frame of the original image would become: $x1 = X1/width = (it_X + X0)/width = it_X/width + x0 * it_width/width = ROI[0] + x0 * ROI[2]$. The same holds true for the other coordinates:

$$\begin{aligned} x1 &= ROI[0] + x0 * ROI[2] \\ y1 &= ROI[1] + y0 * ROI[3] \\ w1 &= ROI[0] + w0 * ROI[2] \\ h1 &= ROI[1] + h0 * ROI[3] \end{aligned}$$

While this example was using object detection bounding boxes, similar reasoning applies for keypoints in pose estimation, and every pixel in a segmentation mask. Scaling detections like this is not applicable in a classification task.

▼ **Figure 6:** The different model preprocessing methods: Model Preprocessing; Model Preprocessing + ISP limitation; ISP Preprocessing





◀ The ground truth bounding box (green) and detection box (red)

Understanding preprocessing and ISP limitations

So far, we've used data injection with ISP preprocessor limitations to prepare our image as DSP input tensors (= input tensors in the format the IMX500 AI accelerator accepts) and use it for data injection. (Default: `--prep isp` for AI Camera and `--prep like_isp` for the Interpreter) This mimics the functionality of the ISP on board of the AI Camera module, as shown in the image above that illustrates data injection. However, there are multiple different ways of converting the sample image to an input tensor that will impact the resulting performance. It is important to understand the different methods, and choose the one that matches your expectations. See **Figure 6**.

1. Model preprocessing

This model preprocessing method offers the best possible accuracy with data injection:

```
$ uv run evaluator aicamera -m yolo11 --prep
model --display_input_tensor
```

You can choose to follow the preprocessing methods specified by the float/quantized model. Since the original AI model was trained, validated, and tested with this preprocessor, it is expected that this will result in the best possible performance when running data injection.

Model preprocessing process

The typical preprocessing process looks like this:

- Padding, cropping and resizing to the input tensor size. Following your model specification and whether or not to preserve the original aspect ratio of the sample image.
- Convert the image colour format to the RGB/BGR colour format of your model.
- Apply model normalisation ($x_{norm} = mean * x + std$ using `mean` and `std` values defined by your model).
- Convert to the channel order defined the framework format of your model. Typically `HWC` for TensorFlow, and `CHW` for PyTorch models.

Prepare input tensor for DSP

The typical process for preparing the input tensor for digital signal processing (DSP):

- **ISP padding:** The IMX500 expects a dedicated memory block for the input tensor. The input tensor might be padded with extra 0s depending on the model input tensor width and height.
- **CHW:** IMX500 firmware always requires a CHW channel order.
- **Quantisation scaling:** $x_{quant} = scale * x_{norm} + shift$. A converted model expects the input tensor range (`int8` or `uint8`) to be the quantized input tensor – compared to the quantized model which still expects normalised float values and does the quantisation scaling in the first layers of the model.
- After applying both steps, one gets a resulting DSP Input Tensor that the IMX500 AI accelerator will accept for injection.

2. Model preprocessing + ISP limitation

Result that matches ISP behaviour and can still be compared with the result of the quantized model.

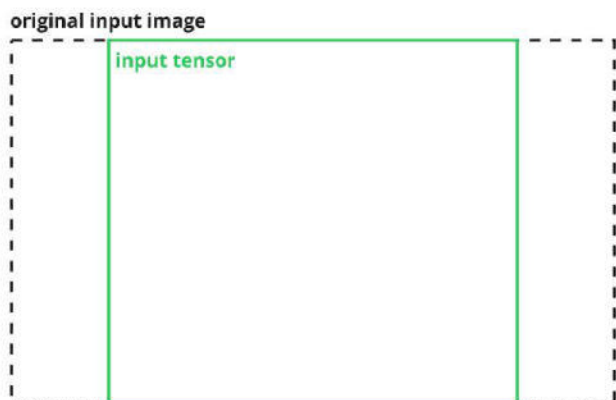
```
$ uv run evaluator aicamera -m yolo11 --prep
like_isp --display_input_tensor
```

The ISP module on the AI Camera has a critical limitation. While it allows for resizing and cropping, it does not provide any functionality for padding the sensor image. This means that it is impossible to follow a model preprocessor that requires to pad the input image (e.g. like YOLO).

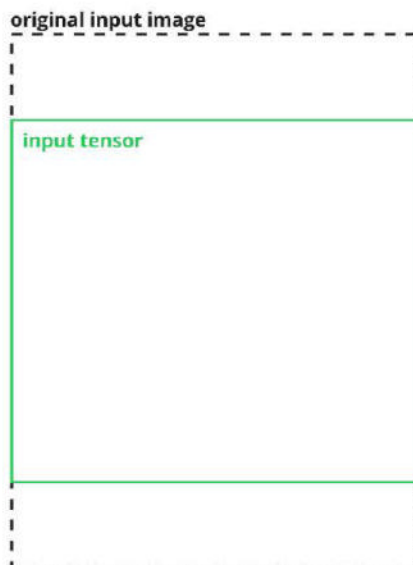
To integrate this limitation, one can swap out the resize function of the model preprocessor with the default behaviour of the ISP and AI Camera device in Modlib. The `--prep like_isp` option does not include extra padding for specific models that require the aspect ratio to be preserved. Instead, the effect depends on whether the model requires `preserve_aspect_ratio`.



▲ An object labelling test



▲ **Figure 7:** Resize the model to the input tensor



preserve_aspect_ratio=False. Resize the entire image to the input tensor size directly, ignoring the original aspect ratio.

preserve_aspect_ratio=True. Centre-crop the image (no padding) to match the aspect ratio of the model's input tensor size. Cropping preserves spatial relationships but may result in some edges being lost. To mitigate the effect on the data injection result, choose dataset images where the aspect ratio is close to the aspect ratio of the model input tensor. Alternatively, match the AI Camera sensor aspect ratio of 1.3342.

Resize to the model input tensor size (see **Figure 7**).

3. ISP preprocessing

The result that best matches the actual ISP behaviour. However, no intermediate input tensor available for framework inference.

```
$ uv run evaluator aicamera -m yolo11 --prep isp
--display_input_tensor
```

In practice, the AI Camera ISP module does not perform separate operations. Model normalisation and quantisation scaling are combined into a single operation to avoid the need for allocating memory of the intermediate input tensor.

In short:

```
x_norm = (x - mean) / std # 1. Normalisation:
q = x_norm * scale + shift # 2. Quantisation/
scale:
```

These normalisation and quantisation scale operations can be expanded to:

```
q = A * x + B where A = scale / STD, B = -scale
* MEAN / STD + shift
```

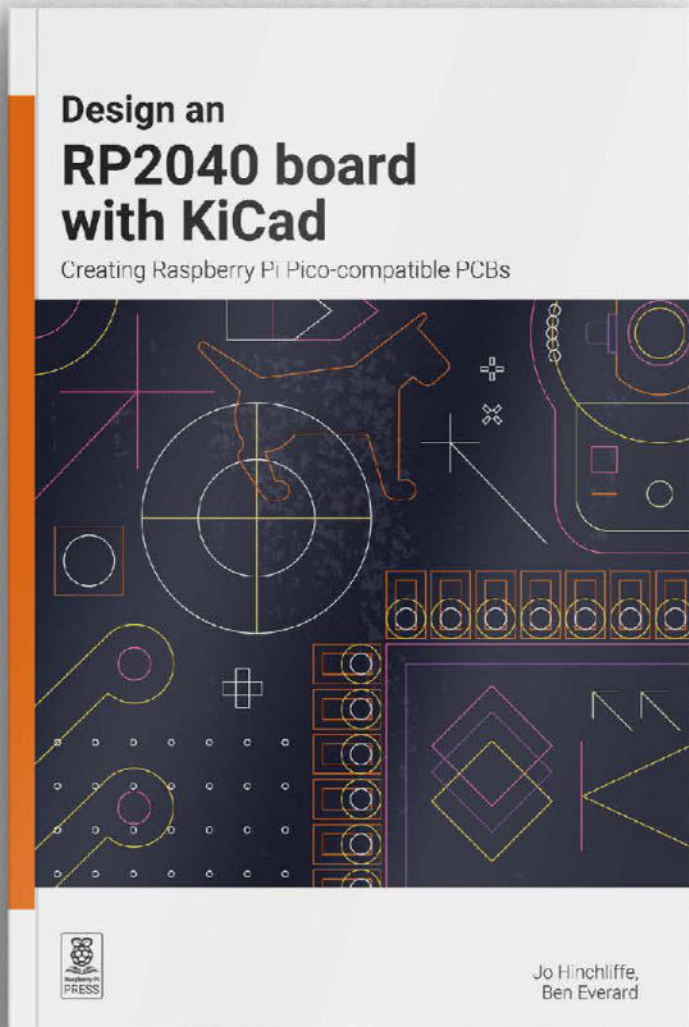
Please read the comments in data injection documentation (rpimag.co/datainjectiondocs) on how this is done in practice. 🟩

Custom models

Data injection isn't limited to models from the zoo. One can use it to validate your own models on the AI Camera. Point the evaluator at a custom implementation with **--model-file <path>** and override artifacts via **--model-path**, **--dataset**, and **--ground-truth**. For more information, check out the evaluator tool (rpimag.co/datainjeval), Modlib's Data Injection (rpimag.co/modlibdatainj), and more IMX500 training and evaluation tutorials on GitHub (rpimag.co/imx500tutorials). Data injection will make sure one can iterate confidently over your custom models and ship camera applications that hold up in the real world.



◀ This test detects performance against multiple objects



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

Build a digitally sovereign second brain

Let's build a system to keep information and notes organised, searchable, synchronised, and backed up, using Raspberry Pi



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

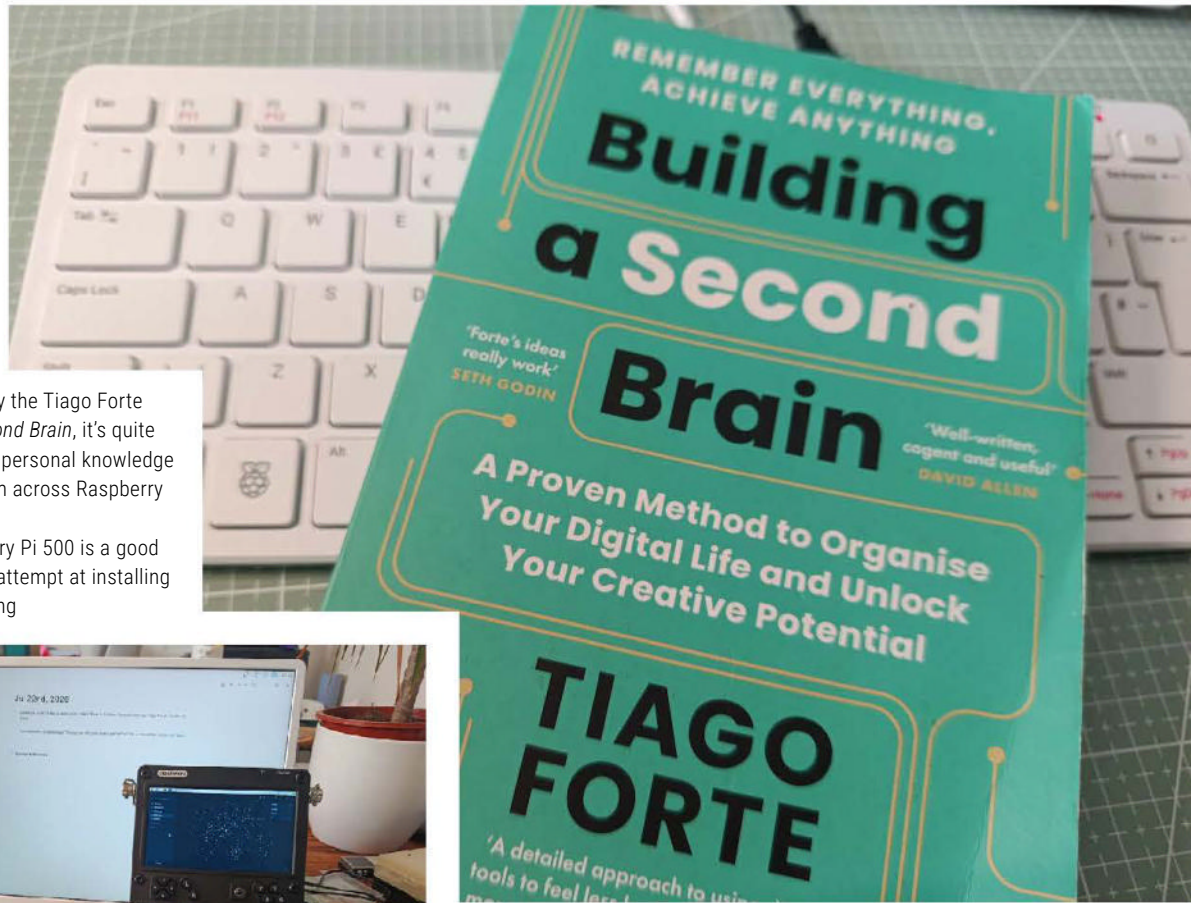
- ▼ Logseq running on Raspberry Pi OS; this is a 'graph view' which shows all your tagged information and also the connections between the tags



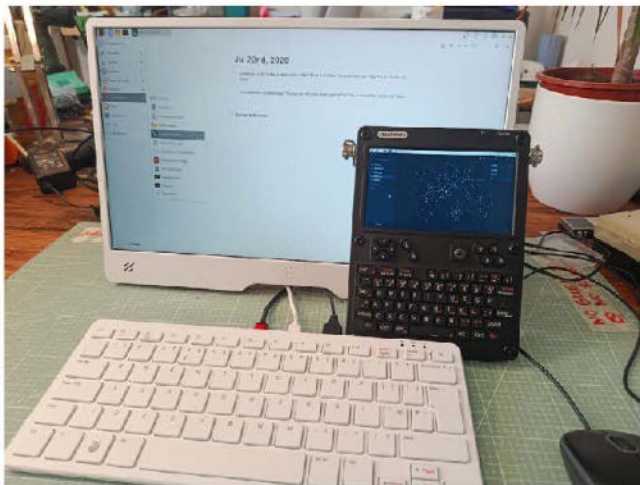
A while ago, we read the book *Building a Second Brain* by Tiago Forte (Figure 1). It starts with the idea that we are all a little overloaded and bombarded with information and how overwhelming it can be to not be able to retrieve the information that we need at a later time. The foundation of a Second Brain is to increase the capture and cataloguing rate of information, and the book delves into lots of custom systems to tag, organise, prioritise, and retrieve information. This Personal Knowledge Management (PKM) idea feels similar to the ideas of David Allen and the older but still brilliant 'Getting Things Done' system.

At the heart of these PKM systems is the concept of ubiquitous capture: the idea that as a start we need to capture the information of interest. This is also an idea in journalling and is basically being proactive about note-taking. The question then is how to store these bits of information. There are lots of options.

There's nothing wrong with a pile of sticky notes (or index cards which are the author's favourite) and a pen; however, you can quickly end up with a huge pile of notes which need to be physically stored and searched. Of course, lots of people have moved to software applications. Many of the popular operating systems for mobile devices will include some form of note-taking application, and there are many that require some form of account and will use the cloud to store your data, so it's a nice challenge to build a system that doesn't store your information on another person's computer. Finally, the solutions we explore here use open-source projects so we can examine the code should we need to maximise our privacy and security.



- **Figure 1:** Inspired by the Tiago Forte book *Building a Second Brain*, it's quite possible to create a personal knowledge management system across Raspberry Pi devices
- ▼ **Figure 2:** A Raspberry Pi 500 is a good target for your first attempt at installing Logseq and Syncthing



At the heart of these PKM systems is the idea of ubiquitous capture

The solution we are going to explore is centred around Logseq (logseq.com), a privacy-focused open-source knowledge management system. It's a really rich piece of software with more features than most people will use, and we'll only really cover the basics, although the basics are incredibly powerful. Later in the article we'll use Syncthing, another open-source project, to synchronise our second brain created in Logseq across multiple devices. Syncthing (syncthing.net) is a peer-to-peer continuous file synchronisation tool which means that it can sync data without passing that data through the cloud/a company's server, it also encrypts and decrypts the data with the data encrypted in transit, so you are nice and secure whilst the data is moving.

It's totally OK though to just install Logseq on one machine and have a play with it to see if it can meet your needs.

We began by installing Logseq on a Raspberry Pi 500 (**Figure 2**), but note that the procedure outlined below is the same for any Raspberry Pi board; so, for example, we have this system currently set up on the Raspberry Pi 500, a Raspberry Pi Zero, and our Uconsole which contains the Raspberry Pi CM4 board.

Currently, the only option to install a build of Logseq is via Snap. So, if working from a fresh Raspberry Pi OS installation, we'll need to install the Snap package manager. If you already have Snap installed on your system, then you can skip to installing Logseq.



◀ **Figure 3:** When you first open Logseq, you land on an informative demo page

To install Snap and then Logseq, open the Terminal application and, at the prompt, enter:

```
$ sudo apt update
```

We can then install the latest version of Snap using:

```
$ sudo apt install snapd
```

We then need to reboot Raspberry Pi 500:

```
$ sudo reboot
```

Once the machine has rebooted, open the Terminal application; we need to install a Snap package called ‘snapd’:

```
$ sudo snap install snapd
```

With Snap installed and set up, we can then go ahead and install Logseq using:

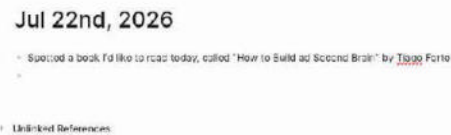
```
$ sudo snap install logseq --channel =og
```

With Logseq now installed, you will be able to launch it using the desktop applications menu at Accessories > Logseq Desktop in the drop-down. On launch, Logseq will open a demo page and you’ll see a journal-style entry for the current date (**Figure 3**). In this entry is a basic tutorial on how Logseq works and some of its functions.

In Logseq terminology, when you create a collection of notes and potentially other assets, you insert them into a ‘graph’. Scrolling on the current demo page, you should find a section where Logseq wants us to specify a folder it can use to contain all our information. You can use an existing folder, but it’s probably better to create a new folder. Click to ‘add a graph’ and then give your folder the title that you want your second brain system to be called; we went with ‘2nd_Brain’, rather unimaginatively. This OG version of Logseq will, in this specified folder, store all the data you input into the system. It essentially creates Markdown files and also stores any uploaded assets (PDFs, pictures, audio, etc.) to an assets subfolder. This means that you could manually

copy this entire folder and move it to another machine and open it in another installation of Logseq.

With the file specified and a new graph created, let’s put some basic information into our second brain system. Whenever you open Logseq, it will create an empty journal page for the current day. Each of these journal dates will become a page item in our system. So let’s type something in the currently empty journal page.



▲ **Figure 4:** Adding a note without a tag is fine, but the only way to rediscover that tag is to either remember the date it was made or the exact contents

As an example, we typed a note: ‘Spotted a book I’d like to read today, called “How to Build a Second Brain” by Tiago Forte’ (**Figure 4**). Now imagine we wrote a note about a different book every day. Currently the only way to return to this piece of information in the future is if we remember the date or use the search function. If we’d forgotten the name of the book, however, we could only really search for terms like ‘book’, which might generate a lot of returned results. What we need is to tag our information. Tagging is the secret sauce of Logseq that makes it incredibly useful.



▲ **Figure 5:** Adding a tag creates a new page for each tag in which tagged information is listed and linked

To add a tag to a piece of information, we place the tag term inside a two pairs of square brackets. Let’s return to the line we entered earlier and write at the end of the line (although tags can be placed anywhere in text) ‘[[Books To Read]]’ and then press **ENTER**. Note that the tag text will become a clickable link (**Figure 5**). If we click that tag, the link then takes us to a new page, as every tag we add to Logseq becomes a new page. On

the new **[[Books To Read]]** page, we will see the reference to our line, ‘... “How to Build a Second Brain” by Tiago Forte’. If we click the back arrow in the upper toolbar, we can navigate back to our journal. Let’s add a second line, ‘Another book I’d like to read, “Design an RP2040 Board with KiCad” by Jo Hinchliffe’. This line will always be listed in the current journal page, but again we can add the **[[Books To Read]]** tag as we did previously and it will be populated to that tag page. If we write a book title every day into separate journal entries, so long as we use the **[[Books To Read]]** tag, they will all appear on the tag page (**Figure 6**). Note that in the image we made a typo in the book title; in a way, this doesn’t matter, but be sure to spell your tags correctly!

However, tags are free and unlimited, so it’s worth tagging information widely with lots of tags and think about how you might remember them in the future. As an example, in our second brain Logseq setup we have some Customs and Excise tariff numbers; these numbers are unique for different types of goods that you sell and it can be quite hard when faced with a new item to research and identify the correct tariff number as they are quite complex and there are a lot of them. It’s definitely the type of information you want to keep note of rather than have to discover every time. If we consider how we might think about these numbers, it’s pretty clear to tag them as **[[Tariff]]**, but sometimes we might think **[[Customs]]** or we might want the tariff number for a particular product or project so we might use the **[[Product Name]]** as a tag. Adding multiple tags to items of information increases your chances of finding your way back to the info when needed and the same piece of information will appear regardless of which tag is the one you remember or use in the future.

We need to move on to other aspects of the system, but it’s worth noting that you can type ‘/Upload an asset’, or rather start by writing ‘/U’ and it will autocomplete, and you can then add files of pretty much any type to an entry. You can tag these then in the usual way and the asset will be copied to the assets folder in your designated directory. This means you don’t have to worry about copying and pasting items, or losing links to items as they are all stored in the folder. Finally, note that you can embed links, videos, and even can highlight and annotate PDFs in Logseq, tagging the exact part of a PDF to return to later.

Books To Read

2 Linked References

Jul 22nd, 2026

- Spotted a book I'd like to read today, called "How to Build a Second Brain" by Tiago Forte
- Books To Read
- Another book I'd like to read "Design an RP2040 Board with KiCad" By Jo Hinchliffe
- Books To Read

Unlinked References

▲ **Figure 6:** The ‘Books To Read’ tag page begins to collect all the inputs listed with that tag

So we have the basis of our second brain setup, but we’d like it to be accessible on more than one device. Our approach for this will be to install Logseq on another device, and then we can use Syncthing to synchronise folders so that our second brain folder is on both devices and up to date (**Figure 7**).

To install Syncthing on Raspberry Pi, it’s best to first run:

```
$ sudo apt update
$ sudo apt upgrade
```

...to make sure our system is up to date. Then we need to add the ‘apt-transport-https’ package to the apt service by running:

```
$ sudo apt install apt-transport-https
```

We then need to store the Syncthing GPG keys on Raspberry Pi in our **keyrings** directory. In a nutshell, this means that the Syncthing packages we install are legitimate and are sourced from the right place. To do this, run the following command:

```
$ curl -s https://syncthing.net/release-key.txt | gpg --dearmor | sudo tee /usr/share/keyrings/syncthing-archive-keyring.gpg >/dev/null
```

We can now add the Syncthing repository with:

```
$ echo "deb [signed-by=/usr/share/keyrings/syncthing-archive-keyring.gpg] https://apt.syncthing.net/ syncthing stable" | sudo tee /etc/apt/sources.list.d/syncthing.list
```

We can then update the packages once again with:

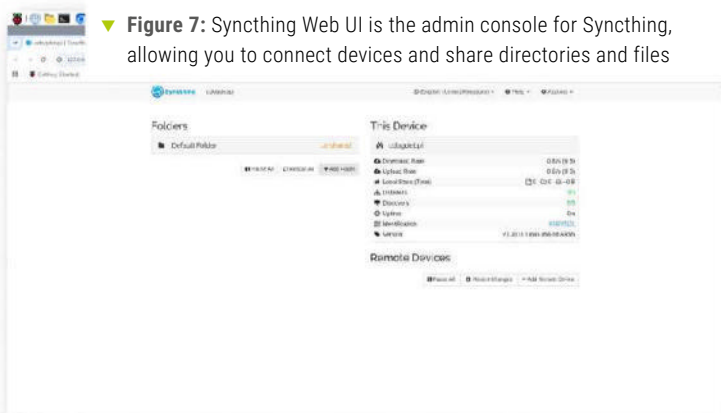
```
$ sudo apt update
```

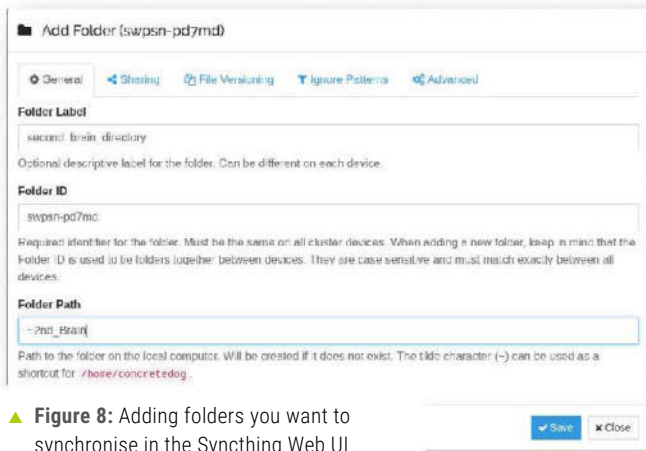
And then finally install Syncthing with:

```
$ sudo apt install syncthing
```

Once it’s installed, you will find two Syncthing items in the

▼ **Figure 7:** Syncthing Web UI is the admin console for Syncthing, allowing you to connect devices and share directories and files





▲ **Figure 8:** Adding folders you want to synchronise in the Syncthing Web UI

applications menu: one is a simple ‘Start Syncthing’ which sets the service running; the other is ‘Syncthing Web UI’. First, double-click the ‘Start Syncthing’ application, then double-click the ‘Syncthing Web UI’. This should now launch the default web browser and will land on a default Syncthing admin page for the device. From here, we can set up files and directories for synchronisation.

First let’s click the ‘Add Folder’ button. In the ‘Add Folder’ dialog box (**Figure 8**), we can add a folder label – we could add, for example, ‘second_brain_directory’ as we are going to share the directory we made for our Logseq second brain files. Leave the Folder ID with the default generated identifier string and then in the last input, Folder Path, set the exact folder path to match where the Logseq folder you made is. So for example, if you made your Logseq folder in the default user directory of a Raspberry Pi and you called it ‘2nd_Brain’ you can input the tilde (~) character (which means the same as /home/username) and write `~/2nd_Brain`. Clicking Save sets this file to be shared.

On another device with Logseq and Syncthing installed (for example, we used our Uconsole), on the same Web UI page you can click on the short name identification in the ‘This Device’



▲ **Figure 9:** Adding devices to your Syncthing setup is straightforward

area. Clicking this launches a dialog with a QR code and also a long-form ID string for the device (**Figure 9**). This is a 56-digit string with dashes. Back on the Raspberry Pi 500, you can click the ‘Add Remote Device’ button and type in the long device ID; note that you don’t need to

add the dashes or spaces, just the long string. If you leave the Device Name section empty, it will populate with the random generated device name on the other device, or you can add a more descriptive name such as ‘Uconsole’ or ‘Pi Zero here.

QUICK TIP

Note that you don’t have to be on the same network for Syncthing to synchronise your files.

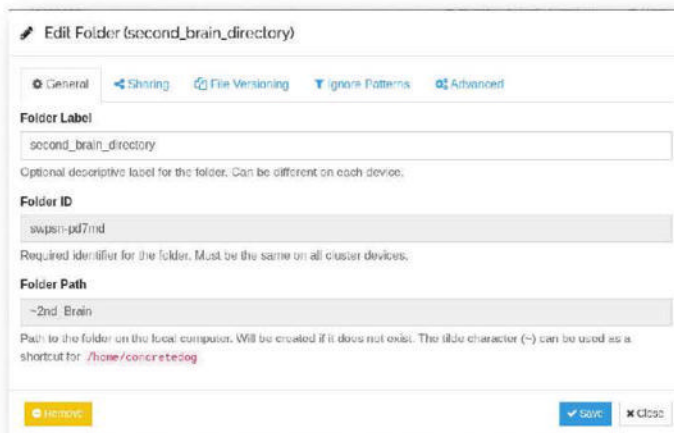
Finally, by clicking on the folder we created to share earlier on the Raspberry Pi 500 and then clicking the Edit button, we can move over to the Sharing tab in the dialog and we should be able to see the added remote device we just added (**Figure 10**). If we tick the box to share with that device, we can then click Save. Give everything a couple of minutes to work through the networks and then check on the other device; you should see a notification that the other device wants to share a folder with you. Accept the folder and follow the instructions to set up a path to a directory to synchronise that folder to. All that then remains is to open Logseq on the second device and click “add graph”, navigating to the synchronised folder we just made. This means if we add or change anything in Logseq on our first device (pi500) or our second device (Uconsole) then these changes appear on both devices keeping our Logseq second brain system up to date. You can of course repeat this process sharing and synchronizing your **2nd_Brain** Logseq folder onto as many devices as you want.

Careful now!

Note that synchronising folders is not a backup! If you accidentally delete part or all of your Logseq folder content, this will propagate through Syncthing and delete it everywhere! A solution to this is to, on one or more devices, set up a system that periodically backs up your folder to another folder.

To do this, we will create a small Bash script that will copy a target folder to a different folder that we won’t sync. We’ll then use the cron service to make this script run periodically so that we don’t have to think about it. You can do this on any of the Raspberry Pi devices, or indeed more than one; however, it would be most useful on a device which can be left always on. In our system we have it set up on a Raspberry Pi Zero W which

▼ **Figure 10:** Editing a file setting to share with specified devices



runs headless and is an ‘always on’ device in our workspace. However, we’ll set this up here on the Raspberry Pi 500, as it’s easy to do this by typing commands into the terminal.

To begin, launch the Terminal application and in the root directory of your Raspberry Pi device, let’s launch the nano text editor to write a Bash script:

```
$ sudo nano
```

In nano, type the following small Bash script, replacing **LOGSEQ_Directory_Path** with the path to your Logseq folder. If your folder is in the home directory, you don’t need any leading edge slashes, so to back up our ‘2nd_brain’ folder in the home directory to ‘backup_2nd_brain’ is really straightforward to input.

```
#!/bin/sh

rsync -avh LOGSEQ_Directory_Path
Backup_Directory_Path
```

Save this file with a name like **backup_script.sh**.

We need to set this file as executable, so in the same working directory as the script, we need to run:

```
$ sudo chmod +x backup_script.sh
```

We can now test the **backup_script.sh** script by, in the same directory, executing it with:

```
$ ./backup_script.sh
```

You can then use the File Manager (or the Terminal) to check if the files have been copied to the newly created backup folder. Finally, we can automate this script using a cronjob. We can do this by editing a file called **crontab**. Type:

```
$ crontab -e
```

...in the terminal. If this is the first time you have done this, it might ask what editor you would like to use; select nano from the list. In the now open **crontab** file, there is a long section of comments that give a great overview of how the cron

system works (**Figure 11**). At the end of the file, we can add a line which has the following structure: minutes, hour, day of month, month, day of week, followed by the command. You can leave slots empty using a ***** and so you can construct values for different automated periods. So, for example:

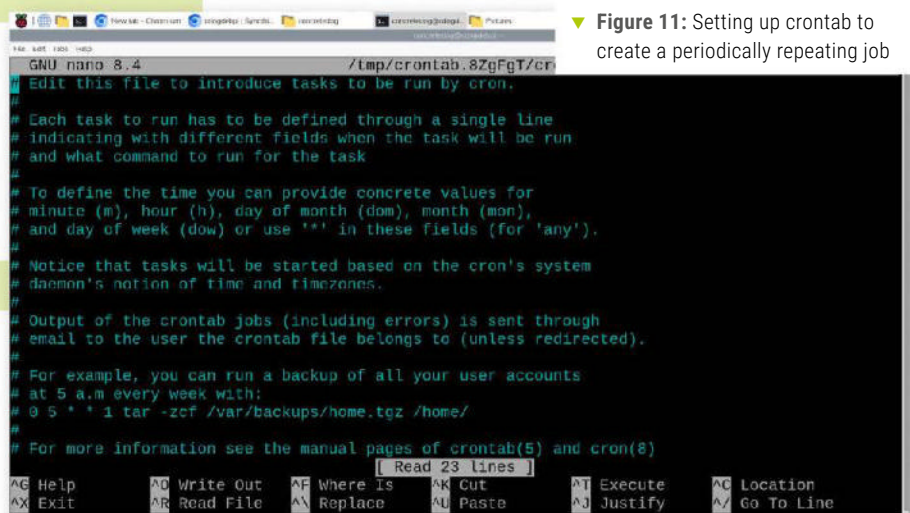
```
46 10 * * 1 /home/concretedog/backup_script.sh
```

...would execute the **backup_script.sh** script every Monday at 10.46 am. Note that if you wanted to run the script once a day, you’d simply leave the day of week value empty, such as:

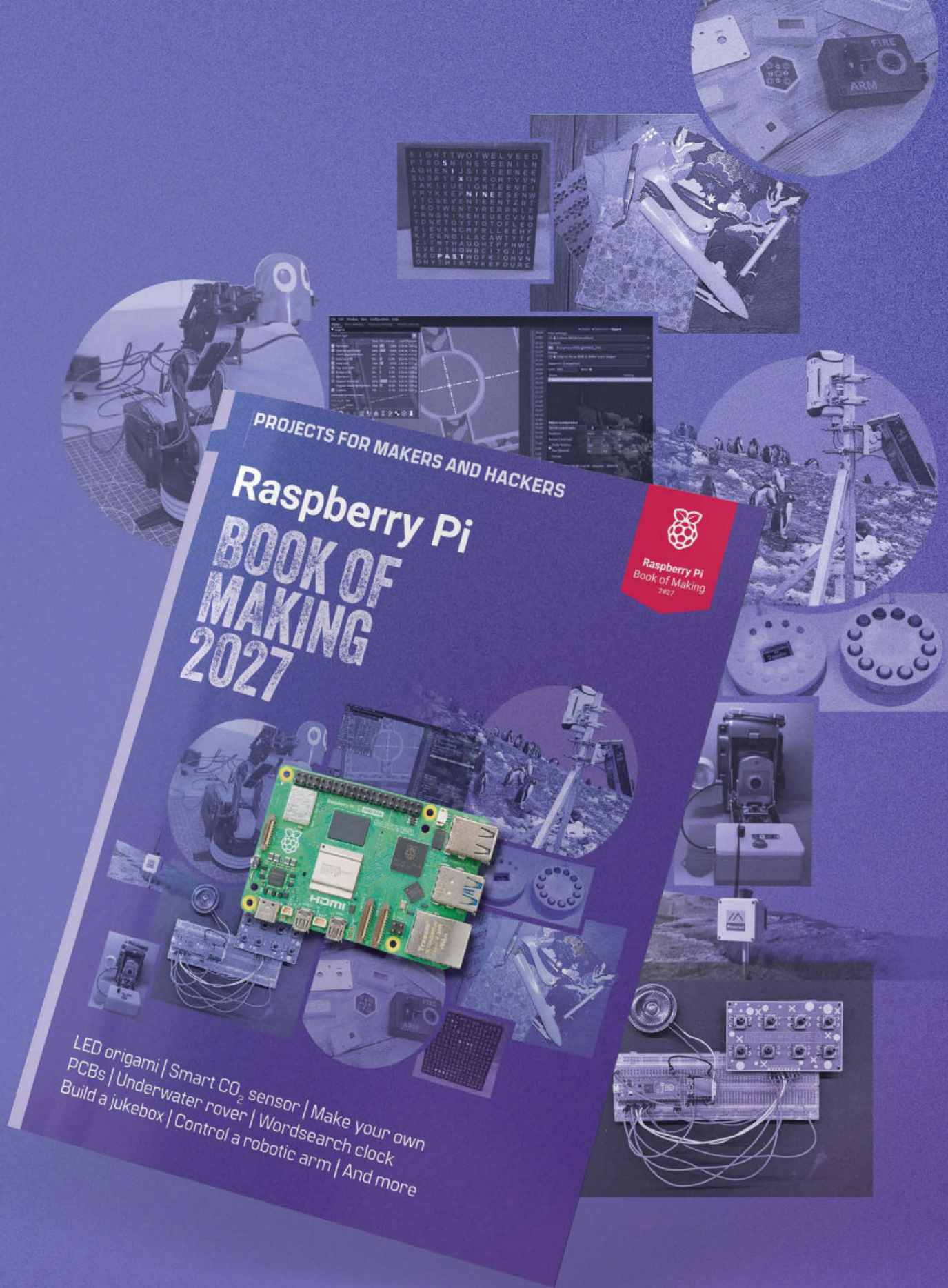
```
30 10 * * * /home/concretedog/backup_script.sh
```

Of course, you can extend and enhance the system; it would be great to back up the folders to an attached SSD, or you might like to explore how to add monitoring and logging to your cron system. Logseq and Syncthing are available for most desktop distributions, and they are also ported to smartphone operating systems, meaning you can carry your private, secure, and backed up second brain with you always! 📁

If you accidentally delete part or all of your Logseq folder content, this will propagate through Syncthing and delete it everywhere!



▼ **Figure 11:** Setting up crontab to create a periodically repeating job



PROJECTS FOR MAKERS AND HACKERS

Raspberry Pi BOOK OF MAKING 2027



LED origami | Smart CO₂ sensor | Make your own
PCBs | Underwater rover | Wordsearch clock
Build a jukebox | Control a robotic arm | And more

Raspberry Pi

BOOK OF MAKING 2027

Technology isn't just something you buy — you can build it yourself! We've rounded up a few of our favourite makes previously published in Raspberry Pi Official Magazine, which should keep you entertained for an afternoon, a weekend, or longer.

- Build a simple sensor robot from scratch
- Experiment with long-range communication platform Meshtastic
- Turn a toy skeleton into a jumping, dancing robot
- Design, prototype, and manufacture a custom PCB
- Control a robot arm
- Add sensors and LEDs to pretty much anything you want!

Buy online
rpimag.co/bookofmaking2027

Attach a microscope to your Raspberry Pi

Investigate a tiny new world and get a better view of your work with simple, cheap microscopes



Maker

Rob Miles

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

robmiles.com

QUICK TIP

Make sure the microscope is switched on before you connect it to another device.

Choosing a microscope

There are lots of slightly different versions available with slightly different screen sizes and buttons, but they all seem to be based on the same configuration. Their prices change on an almost daily basis, but with patience you should be able to pick one up for around £20 on your favourite e-commerce site. If you want to store images, you will have to buy a microSD card.

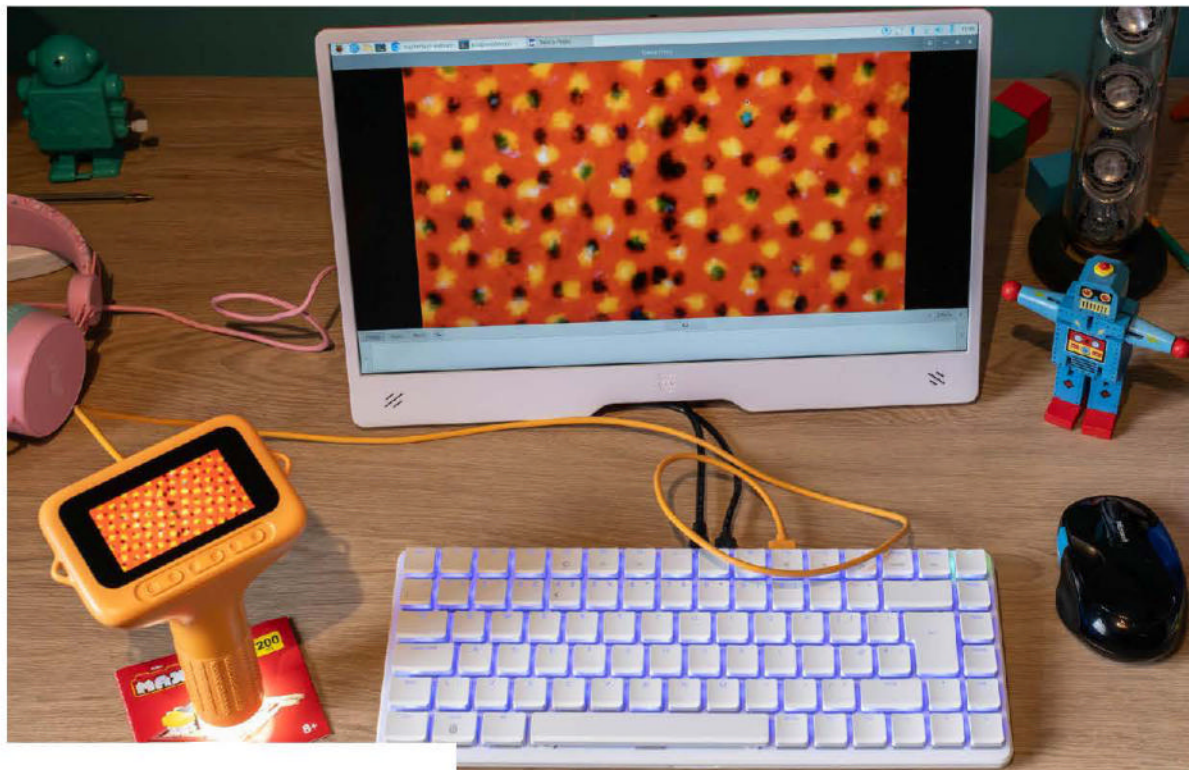
Toys can be a great place to find useful technology. Here, we'll show how you can connect low-cost optics to Raspberry Pi and get a good view of little things.

Figure 1 shows a digital handheld microscope attached to a Raspberry Pi. The microscope provides 1000 times magnification and has a 3-inch display. It is showing the pattern of coloured dots in the orange colour on a printed leaflet. The microscope can capture images onto a microSD card (which is not included). It is powered by an internal lithium battery that is charged via a USB-C connection. It is great fun to use and makes a super toy.

The microscope lens is surrounded by white LEDs which can be varied in brightness. If you have something which is backlit (for example, a film negative on a light table), you can turn the lights off completely. You turn the ring on the end of the microscope to

adjust the focus. The maximum resolution of the image is 1920×1080 , which is good enough for a video monitor. Captured images are encoded as JPEG and tend to be around 800kB in size.

A menu appears on the microscope if you plug it into a Raspberry Pi (other computers are available). The menu asks you how you want the microscope to connect to the host. It can appear as a USB mass storage device, allowing you to load images off the card. Alternatively, the microscope can provide a USB video feed in the same way as a standard webcam. This is very useful if you want to show lots of people what the microscope is seeing.



► **Figure 1:** The microscope is at the bottom left of the picture

Install a program that displays the camera output

Connecting a microscope to a Raspberry Pi

You can use any of the USB sockets on your Raspberry Pi to connect the microscope. Once it is plugged in and configured as a camera, you can check for a video device using the command `lsusb` at the command prompt. You should see a device called 'Generic HD video' in the list that appears. If you don't see this, make sure that you selected 'View on Computer' from the menu which appears on the microscope when you plugged it into Raspberry Pi.

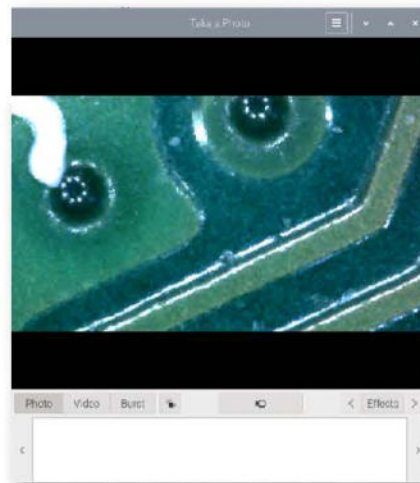
Once you've got the microscope connected, you need to install a program that displays the camera output on the screen of your Raspberry Pi. There is a program with the fantastic name of `cheese` that does this.

Figure 2 shows `cheese` in action, displaying a view of a Raspberry Pi Pico circuit board. You install it with the following terminal commands:

```
$ sudo apt update
$ sudo apt install cheese
```

You can then use the command `cheese` in order to start the program running.

```
$ cheese
```



◀ **Figure 2:** Make the app full-screen for the largest view

Microscope menu machinations

The microscopes offer a menu interface which is navigated using the buttons on the device. This can be confusing and counter-intuitive; for example, the power button might be used to select items on the menu. The microscopes are supplied with manuals which you should read very carefully before using.

► **Figure 3:** These are also sold as 'coin microscopes'

Over Magnified

The first kids' microscope the author bought worked very well, but he found the magnification excessive for some applications. He wanted a wider field of view for tasks like soldering and adding surface-mount components. Fortunately, there are devices that have lower magnification and can work over a wider range of subject distances.

The microscope shown in **Figure 3** comes with a desktop stand that is height adjustable. It is used in a very similar way to the handheld microscope, but it has a lower 500× magnification and a much wider range of focus adjustment. It can also be used as a handheld device and has LED lighting.

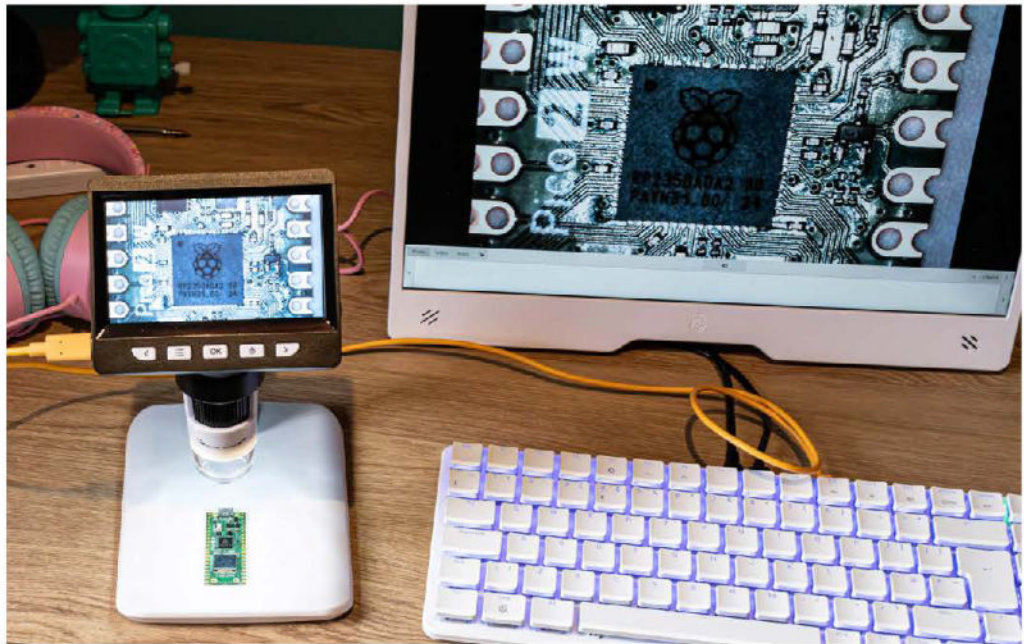
Microscopes and photography

Some old film cameras have ground glass screens for focusing. The camera projects a preview of the image to be taken onto a sheet of frosted glass. The image is not very bright and is hard to see. It turns out that you can use a microscope as a 'viewfinder' by pointing it at the ground glass image. All you need is a hood to fit around the end of the microscope and keep out external light.

Figure 4 shows the hood being used to focus an old camera on some treetops. It is a bit unwieldy to use; you need to make sure that the microscope is focused on the ground glass before adjusting the camera's focus. However, the microscope does manage to pull an image off the back of the camera which would otherwise be very hard to see. You could also use this approach to digitise photographic negatives, although the resolutions of the pictures would not be that great.

Close-up fun

The author has found the microscopes fun to use and properly useful. They are another example of how cheap it is possible to make complex technology and would make a splendid present for a someone wanting to explore things on a small scale. 🍷



◀ **Figure 4:** The design for the hood is available at rpimag.co/microscopehood

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

Commanding the internet

Yes, it is possible to get online from the command line. Here's how...



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](#)

Try the retro world of the command-line web browser

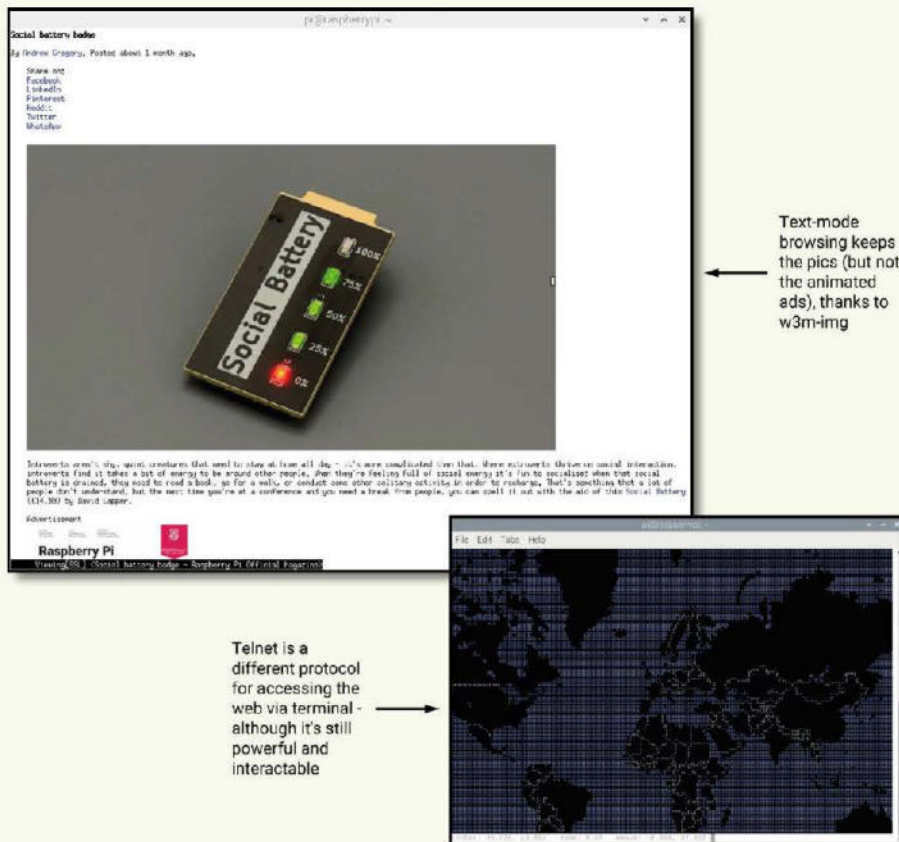
Now that you're getting comfortable at the command line, and perhaps finding it faster for some tasks, you may want to increase the amount of time you spend there, trying to improve your productivity on everyday tasks. Here we survey internet software – a diverse field, covering clients for protocols, specific services (YouTube), and tasks (search), as well as general web browsers, mail clients, and even surprising uses of the venerable Telnet client.

Don't get carried away: command-line alternatives vary from essentials you'll see often used in scripts and Raspberry Pi projects, to less satisfactory alternatives you'd only use when there is no GUI available – so don't plan to switch all of your Facebook and Instagram use to the terminal. We concentrate

here on internet interactions that will be useful for your Raspberry Pi project, but sometimes you may be using the command line on Raspberry Pi and just need to look something up on the web, so welcome back to the Internet of Text.

These days, Raspberry Pi is a very capable desktop machine. However, some websites are so dependent upon JavaScript that for all the memory that they hog on Raspberry Pi, you can only access them via Chrome (or Firefox). For others, try the retro world of the command-line web browser.

In the early 1990s, if you needed to browse the web from a command-line environment, you probably used a text-mode



Conquer the Command Line 3rd Edition out now

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



browser called Lynx. More than a quarter of a century later, it's still in use! If not being able to see any pics makes the web pointless for you, consider the advantages: minimal tracking, no distractions from reading the text, and quicker page loads. Screen reader users and people with mobility impairments can benefit from the simpler keyboard navigation, and as pages can be dumped to stdout, you can pipe it through other command-line applications:

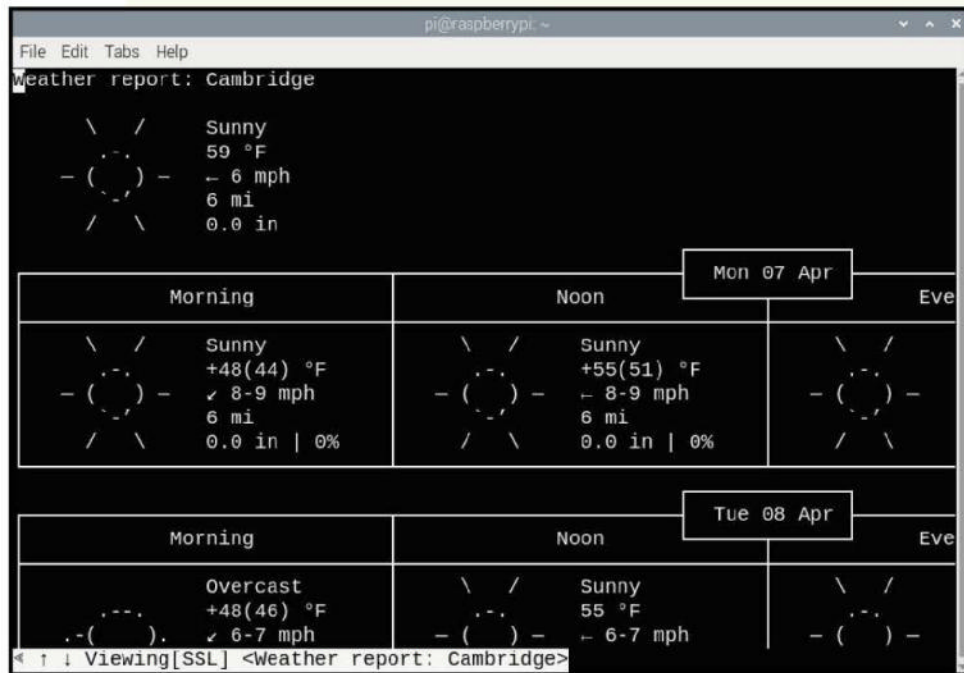
```
$ lynx -dump https://duckduckgo.com/?q=gpio+bash
| tail -n 30
```

Lynx isn't the only choice in command-line browsing - you can install the links2 and w3m packages with `apt install`; the latter is even usable from within the Emacs text editor. Install xterm (an alternative to the default Terminal application in Raspberry Pi OS) and w3m-img, and you can even view images when browsing from the terminal.

Commanding the web

Telnet - short for teletype network - is the protocol developed at the end of the 1960s to provide two-way text communication between computers (install it with `apt install`). It dates from an era without security concerns, and was long ago replaced by SSH

- Command-line browsing is all about the information – mostly text, but not entirely



as a means of connecting directly to other machines over open networks, but the Telnet client remains useful for interactively querying services such as mail or web for testing, by connecting to the appropriate port on the server, then issuing commands in the service's protocol:

```

$ telnet example.com 80
$ GET / HTTP/1.1
$ HOST: example.com

```

...which (after pressing **ENTER** again) should dump a webpage onto the terminal. It's a basic but useful diagnostic tool.

Surfraw

As we have seen earlier, you don't need a web browser to download files if you use **wget** and **curl**. Another case where specialist commands seem most appropriate is searching. Looking things up from the command line can be done directly with a nifty little program called **surfraw**, which uses helpers called **elvi** to tackle different search tasks – see them all with:

```

$ surfraw -elvi

```

As a good citizen of the command-line world, surfraw pays due regard to your time and your tendons, and can be called simply with **sr**.

```

$ sr google -results=2 raspberry pi clojure gpio
$ sr translate peloton
$ sr gutenberd dickens
$ sr wikipedia permaculture
$ sr bbcnews tim peake
$ sr lastfm springsteen

```

The results are displayed in your browser, which you can set in **/etc/xdg/surfraw/conf**:

```

def SURFRAW_text_browser /usr/bin/lynx
defyn SURFRAW_graphical no

```

Play button

YouTube videos also won't work from a text mode browser, but it's easy enough to grab them with **yt-dlp**, a fork of **youtube-dl**. You can grab the code from **rpmag.co/ytldp** and compile it using the information from part 13 of this tutorial series in the previous issue (**rpmag.co/168**).

Once you have, you can try downloading a video with:

```
$ yt-dlp
https://www.youtube.com/watch?v=ouhXh8thnQM
```

Each downloaded video is saved as an WebM file which, thanks to the built-in video player's (VLC) ability to output video through a choice of libraries, can even be rendered in ASCII through the terminal, using `DISPLAY= cvlc -V aa video.webm`. The `DISPLAY=` before the command makes VLC unable to use the GUI, forcing it into text-mode output (you don't need it if you're connecting over SSH or from the text console).

Communication

Command-line apps are available for other means of internet communication, from IRC to Jabber, and Mutt is a powerful enough email client to keep people (who otherwise don't spend much time in the terminal) using it. Serious work gets done in a shell, but coders keep churning out command-line apps that are just for fun, too.

MapSCII is a Braille and ASCII map renderer for your console, using OpenStreetMap data, written by Michael Straßburger (rpimag.co/mapscii). To connect from a remote computer:

```
$ telnet mapscii.me
```

ASCII Star Wars appeared at the end of the last century, and is still available at asciimation.co.nz. Ironically, you'll need a modern graphical browser to view it.

Some sites are designed to look good in text-mode browsers. A favourite is wttr.in, the weather website. Enter your location, and open in any of the text-mode browsers that we've mentioned; for example:

```
$ w3m wttr.in/Liverpool
```

That's all for now. Enjoy using the command line! 🍷

Make email more manageable

If you struggle to keep up with email using a webmail interface, challenge yourself to use Mutt for two weeks: the command line may let you finally tame your inbox.

- No spare memory to open a YouTube link? Download the video to watch later

```
pi@raspberrypi: ~/Videos
File Edit Tabs Help
pi@raspberrypi:~/Videos $ yt-dlp https://www.youtube.com/watch?v=ouhXh8thnQM
[youtube] Extracting URL: https://www.youtube.com/watch?v=ouhXh8thnQM
[youtube] ouhXh8thnQM: Downloading webpage
[youtube] ouhXh8thnQM: Downloading tv client config
[youtube] ouhXh8thnQM: Downloading player 73381ccc-main
[youtube] ouhXh8thnQM: Downloading tv player API JSON
[youtube] ouhXh8thnQM: Downloading ios player API JSON
[youtube] ouhXh8thnQM: Downloading m3u8 information
[info] ouhXh8thnQM: Downloading 1 format(s): 248+251
[download] Destination: We are Raspberry Pi [ouhXh8thnQM].f248.webm
[download] 100% of 8.92MiB in 00:00:01 at 6.49MiB/s
[download] Destination: We are Raspberry Pi [ouhXh8thnQM].f251.webm
[download] 100% of 1.69MiB in 00:00:00 at 3.47MiB/s
[Merger] Merging formats into "We are Raspberry Pi [ouhXh8thnQM].webm"
Deleting original file We are Raspberry Pi [ouhXh8thnQM].f248.webm (pass -k to keep)
Deleting original file We are Raspberry Pi [ouhXh8thnQM].f251.webm (pass -k to keep)
pi@raspberrypi:~/Videos $
```

Create an LED thermometer

Read a temperature sensor and display its readings on an LED bar graph



Maker

Phil King

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

philkingeditor.com

YOU'LL NEED

- 1× solderless breadboard
- 1× MCP3008 ADC chip
- 1× TMP36 temperature sensor
- 5× LEDs (red, yellow, green)
- 5× 330Ω resistor
- 1× 1µF ceramic capacitor
- 11× pin-to-socket jumper wires
- 8× pin-to-pin jumper wires

Continuing the theme of analogue inputs from last issue, we'll use the MCP3008 analogue-to-digital converter (ADC) again and this time hook it up to a temperature sensor. We'll display the current temperature on the screen, then add some LEDs and use GPIO Zero's handy `LEDBarGraph` class to show the temperature in a bar graph style.

Enable SPI

As in part 9 of this series, last issue (rpimag.co/168), 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 will get better results if you enable full SPI support. See the 'Enable SPI' section in part 9 for more details.

Connect the ADC

If you still have the MCP3008 wired up from part 9, leave it in place, straddling the central groove of the breadboard. As noted before, there's quite a lot of wiring required; connect the jumper wires as in **Figure 1**. Two go to the '+' power rail, connected to a 3V3 pin; two others are connected to a GND pin via the '-' rail. Four legs of the ADC are connected to GPIO 8 (CE0), 10 (MOSI), 9 (MISO), and 11 (SCLK).

Add the sensor

Now that the ADC is connected to the Raspberry Pi, you can wire devices up to its eight input channels, numbered 0 to 7. **Figure 1** shows how to connect a TMP36 analogue temperature sensor. It's vital that this is wired up correctly, otherwise it'll overheat. With its flat face towards you, the left-hand leg is for power, so

To help stabilise the readings which might otherwise be erratic, we'll add a capacitor

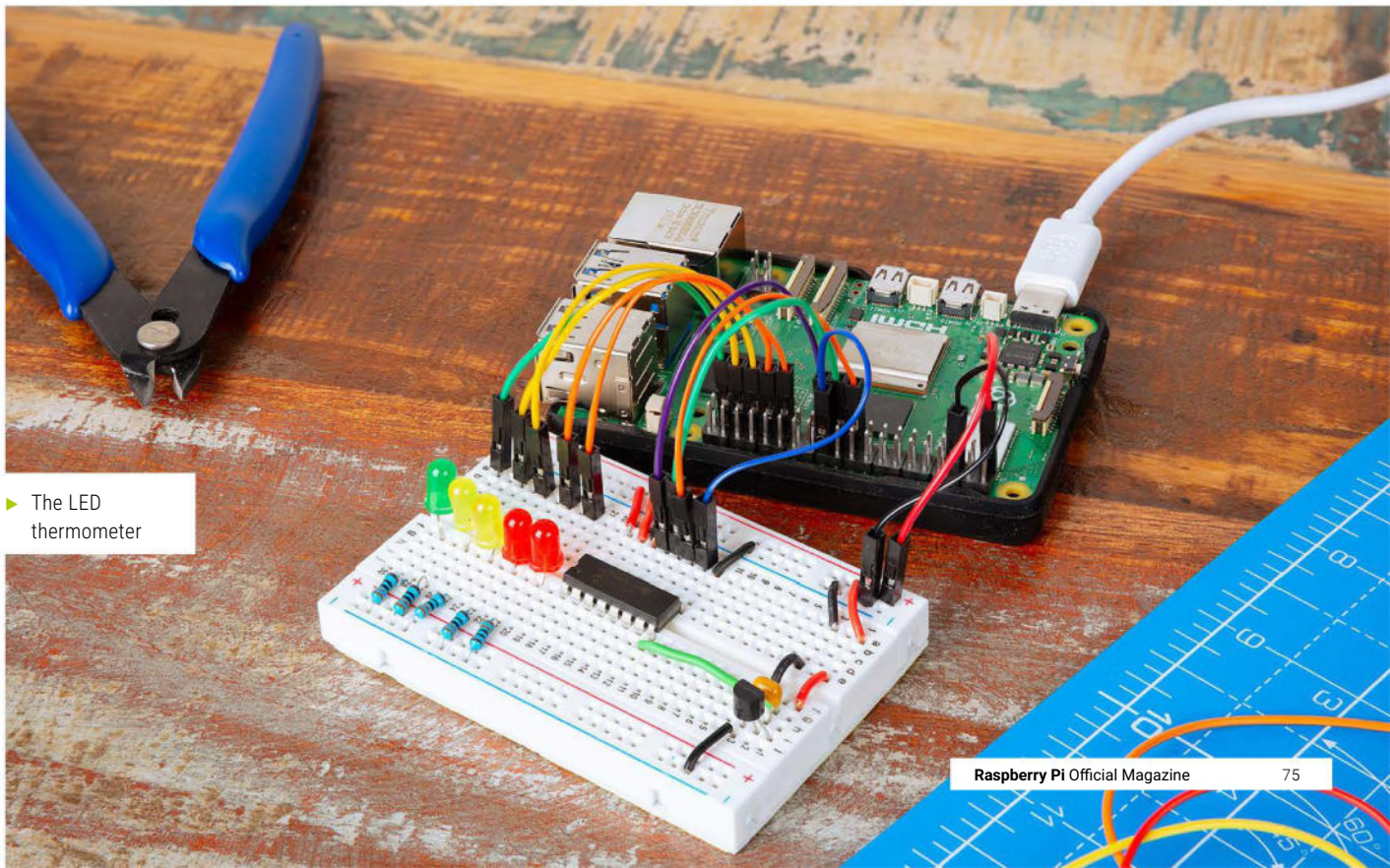
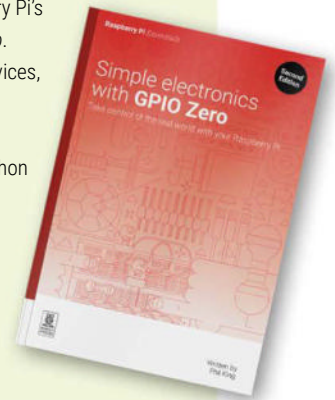
connect this to the '+' power rail. The right-hand leg is connected to the '-' ground rail. Its middle leg is the output; here we're connecting it to channel 7 (the nearest one) of the MCP3008. Finally, to help stabilise the readings which might otherwise be erratic, we'll add a capacitor to link its output and ground legs.

Simple electronics with GPIO Zero

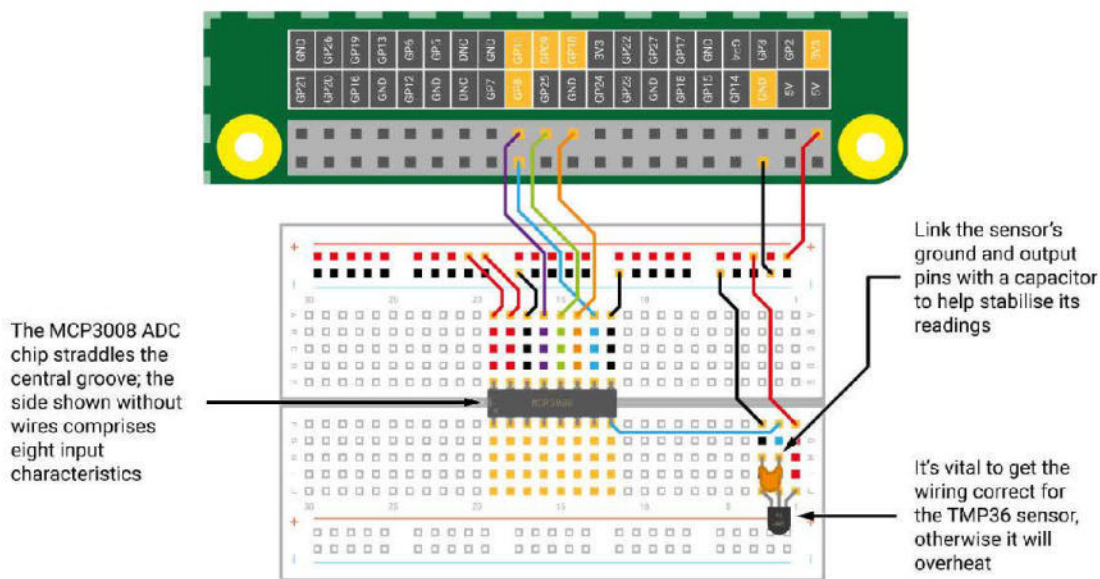
This article is an extract from Raspberry Pi's book, *Simple electronics with GPIO Zero*.

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 LED thermometer



◀ **Figure 1:** Wiring up the sensor

Take the temperature

We can now read the sensor's value in Python. Create a new file, then save the following code as **test_tmp36.py**, and run it.

```
from gpiozero import MCP3008
from time import sleep

def convert_temp(gen):
    for value in gen:
        yield (value * 3.3 - 0.5) * 100

adc = MCP3008(channel=7)
for temp in convert_temp(adc.values):
    print(f"The temperature is {temp}C")
    sleep(1)
```

At the top we import the **MCP3008** class from GPIO Zero, then the **sleep** function from the **time** library. Next, we define a function that converts the sensor reading into degrees Celsius. We then set the **adc** variable to channel 7 of the MCP3008. Finally, we use a **for** loop to display the converted temperature, updating it every second. In this example, we're reading the **adc** object's **values** property, which is an endless source of sensor readings; the loop will continue until you stop the script. The values are fed to the **convert_temp** function, which is a *generator function*, a special type of function that can be used as the source of a **for** loop. Rather than using the **return** statement to return a single value, **convert_temp** uses the **yield** keyword, which causes Python to treat it as a generator rather than a regular function.

Note: If you've just been handling the sensor, it might take a little while to settle down to the ambient temperature.

LED bar graph

Next, we'll add our line of five LEDs to the circuit, as in **Figure 2**. From green to red, we've connected their longer legs to GPIO 26, 19, 13, 6, and 5. Create a new file, enter the following code, and save it as **test_graph.py**.

```
from gpiozero import LEDBarGraph
from time import sleep
graph = LEDBarGraph(26, 19, 13, 6, 5, pwm=True)
graph.value = 1/10
sleep(1)
graph.value = 3/10
sleep(1)
graph.value = -3/10
sleep(1)
graph.value = 9/10
sleep(1)
graph.value = 95/100
sleep(1)
graph.value = 0
```

At the start, we import the **LEDBarGraph** class from GPIO Zero; this will enable us to use the LEDs to display a bar graph, saving a lot of complex coding. We set the **graph** variable to our LEDs on the GPIO pins and enable PWM so that **LEDBarGraph** can use brightness levels to represent intermediate values, providing a more precise display. We then set **graph.value** to various fractions between 0 and 1 to light the relevant number of LEDs from green to red, including partially lit ones for precision. Note that if the value is negative, it will light the LEDs from the other end, red ones first.

Display the temperature

So, we've got our temperature sensor and LED bar graph set up; let's combine them to display the temperature on the LED bar graph. In Thonny, create a new file, enter the code below and save it as **graph_temp.py**.

```
from gpiozero import MCP3008, LEDBarGraph
from time import sleep

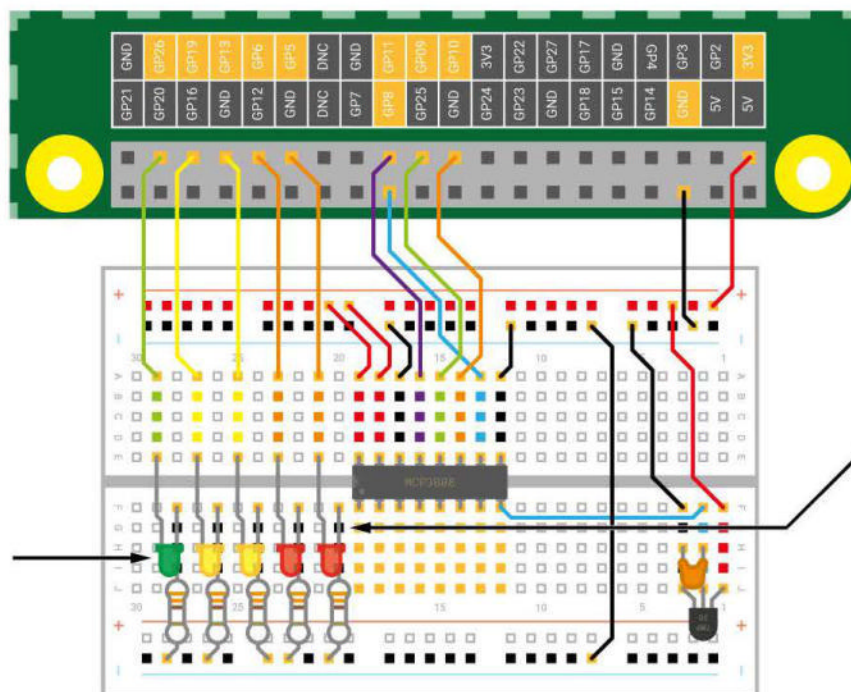
def convert_temp(gen):
    for value in gen:
        yield (value * 3.3 - 0.5) * 100

adc = MCP3008(channel=7)
graph = LEDBarGraph(26, 19, 13, 6, 5, pwm=True)
for temp in convert_temp(adc.values):
    bars = temp / 40
    graph.value = bars
    sleep(1)
```

At the top, we import GPIO Zero's **MCP3008** and **LEDBarGraph** classes, along with the **sleep** function from the **time** library. As in our original code, we then define a function to convert the temperature sensor's readings to degrees Celsius. We set the **adc** variable to channel 7 of the MCP3008 and **graph** to our LEDs' GPIO pins, setting **PWM** to **True**. Finally, in our **for** loop, we add a **bars** variable to determine how many LEDs are lit in the bar graph. In this example, we've divided **temp** by 40, which is around the maximum temperature for the UK, so if it gets to 40°C, all the LEDs should light up fully. Naturally, you can adjust this number to suit your own location's climate. When ready, run the code and see those LEDs light up to show the current temperature. 🍷

*We define a function
to convert the sensor's
readings to degrees Celsius*

We use five LEDs to display a bar graph relating to the temperature; each LED is connected to a different GPIO pin



◀ **Figure 2:** Adding the LEDs

Each LED's shorter leg is connected to the ground rail via a resistor, to limit the current

An introduction to predictive models

Learn about two foundational machine learning techniques: linear and logistic regression



Maker

Brian Jepson

Brian is the publisher at Raspberry Pi Press. He pursues a passion for making computing and technology accessible to people of all ages and backgrounds. Brian also enjoys creating electronic devices that interact with the physical world. Outside of work, Brian collaborates with like-minded artists, technologists, and non-profits in Rhode Island to create experiences that get people excited about STEM/STEAM careers and hobbies.

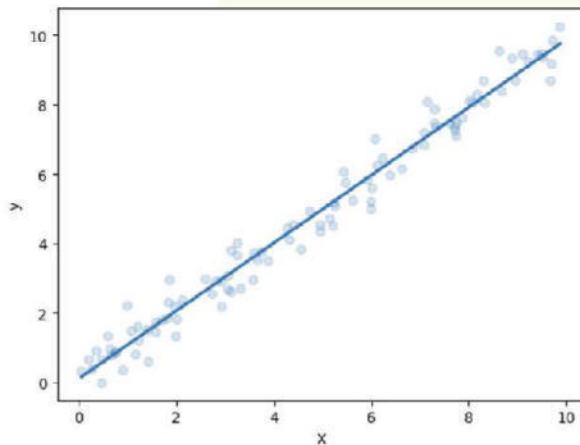
rpimag.co

This tutorial introduces some fundamental machine learning techniques that will help you understand modelling and inference: *linear* and *logistic regression*. Both techniques model the relationship between data that you already know (one or more *independent variables*) and data that you want to predict (the *dependent variable*). To build these models, you need historical data for both sets of variables. Using that data, these techniques produce models that, given new values for the independent variables, attempt to predict the value of the dependent variable.

Different methods work best in different situations. A linear regression model is used when changes in one variable are associated with consistent changes in another. For example, if one variable increases, the other tends to increase (or decrease) in a steady way. If you plot these values on an X/Y graph, the points won't usually form a perfect straight line, but they will cluster around a line that represents the model's best estimate of the relationship between the variables.

Logistic regression is used in a different kind of situation: when the value you want to predict is not a number, but a *category*. Sometimes this means a binary outcome such as yes or no, true or false, or one or zero. It can also answer the question "what group does this belong to?" Instead of predicting a value directly, logistic regression predicts the likelihood of a particular outcome.

Model the relationship between data you already know and data you want to predict



When building any model, it's helpful to inspect graphs of historical data to see whether a relationship between variables might exist. This becomes more challenging when you have multiple independent variables, since a simple X/Y plot is no longer possible. Fortunately, as you'll see later in this guide, there are tools and techniques that can help you determine whether meaningful relationships are likely to exist.

Figure 1 shows a plot of some sample data, using *x* (the horizontal axis) for independent variables and *y* (the vertical axis) for dependent variables. This data lends itself well to linear regression. In logistic regression, the relationships fit an S-shaped, or *sigmoid curve*. This can make patterns harder to spot by eye than in linear regression. While you can use plots to help guide your intuition, such relationships are usually not as immediately obvious as straight-line trends.

Install Python modules

Before you can install the Python modules needed for this tutorial, you'll need to create a virtual environment and activate it. For more details, see rpimag.co/venv.

```
$ python3 -m venv ~/.virtualenvs/Regression
$ source ~/.virtualenvs/Regression/bin/activate
```

After you've activated the environment, you'll need to install a few modules (you will only need to do this once for each environment) with:

```
$ pip install jupyterlab pandas seaborn
scikit-learn tqdm nbtime
```

DOWNLOAD
THE FULL CODE:



rpimag.co/aiproj-resources

◀ **Figure 1:** A plot that shows a linear relationship

The first of these modules provides the *Jupyter Lab* browser-based environment for creating interactive *notebooks*, which consists of *cells* that contain your code and documentation. The Jupyter Notebook web-based interface runs entirely on your Raspberry Pi.

The other modules that you installed are used extensively with Jupyter:

- **pandas** – An open-source framework for data analysis.
- **seaborn** – This is a module for data visualisation. You'll use it to create charts and graphs.
- **scikit-learn** – This module offers a set of tools for predictive analysis, including support for linear and logistic regression.
- **tqdm** – This is a fast progress bar implementation for Python. It can draw progress bars at the Terminal or in your notebooks. Because you'll be running scripts that take a while to do their thing, **tqdm** can take the mystery out of how long a process will take to complete.
- **nbtime** – This module helps you compare changes in notebooks visually and can be used by the **git diff** command to compare changes you make against our original.

Collect some data

Before we can make some predictions, we need something to analyse. Your Raspberry Pi is always busy doing something and has some interesting metrics you can measure. Suppose you want to predict the temperature of your Raspberry Pi CPU – this will be your dependent variable. You'll need to pick some independent variables. But how do you know which ones? You can start by picking some likely candidates: CPU speed, CPU voltage, CPU utilisation (in percentage), and load average (the average number of processes using or waiting on the CPU).

You can use the **vcgencmd** command to measure the CPU speed, temperature, and voltage. The **psutil** Python module can measure load average and CPU percentage. If your virtual environment isn't active already, open a Terminal window on your Raspberry Pi, then activate it:

```
$ source ~/.virtualenvs/Regression/bin/activate
```

Next, install the `psutil` module, just in case it's not already installed:

```
$ pip install psutil
```

You can find this program in the GitHub repo ([rpimag.co/ai-projects-raspberrypi](https://github.com/rpimag/ai-projects-raspberrypi)). Use `git clone` to check it out, then the `cd` command to change to the `ai-projects-raspberrypi/ch03` subdirectory. The data collection script is `collect_cpu_data.py`. First, it imports the modules needed to run this script:

```
import pandas as pd
import psutil
import os # for os.popen to run vcgencmd
commands
import re # for regex to extract numbers from
vcgencmd output
import time # for time.sleep to control sampling
rate
from tqdm import trange # for progress bar
```

Next, you need to define some functions to get data out of `vcgencmd`. The first one, `extract_number()`, uses a *regular expression* to extract numbers from the output of `vcgencmd`.

In the following code, the regular expression `=(\d+(\.\d+)?)` first matches an equals sign (`=`). Then, the first item in the outer set of parentheses matches one or more digits: `\d+`. The `\d` matches any digit, and the `+` after the `\d` modifies the expression such that it matches one or more of the preceding (one or more digits).

Regular expressions

A regular expression, also known as a regex, defines a syntax for matching patterns within text. Regular expressions are a sublanguage of their own, and you'll encounter them in languages such as Python, JavaScript, and even in the Terminal shell. For more on regular expressions, see rpimag.co/pyregex.

```
# vcgencmd uses a format like "temp=45.2'C", so
we need to
# extract the numbers with decimal points if
any.
def extract_number(s):
    match = re.search(r"=(\d+(\.\d+)?)", s)
    return round(float(match.group(1)), 4)
```

The expression in the inner parentheses is similar, but it matches a decimal point followed by one or more digits. The `\d` sequence matches any digit, but because the full stop (`.`) is a special character in regular expressions (it matches any character), we escape it by prefixing it with a `\`.

The parentheses are used to combine matches (or parts of them) into numbered groups. By wrapping `\.\d+` in parentheses and adding the question mark operator (`?`) after the closing parenthesis, it makes that entire expression optional, which allows the regex to match `=123` as well as `=1.23`. By wrapping *everything* but the equals sign in parentheses, we can refer to just the numeric part of the match with `match.group(1)`. In the case of `volt=1.23`, group 0 would be the equals sign and everything after it, group 1 would be `1.23`, and group 2 would be the match from the inner set of parentheses (`.23`).

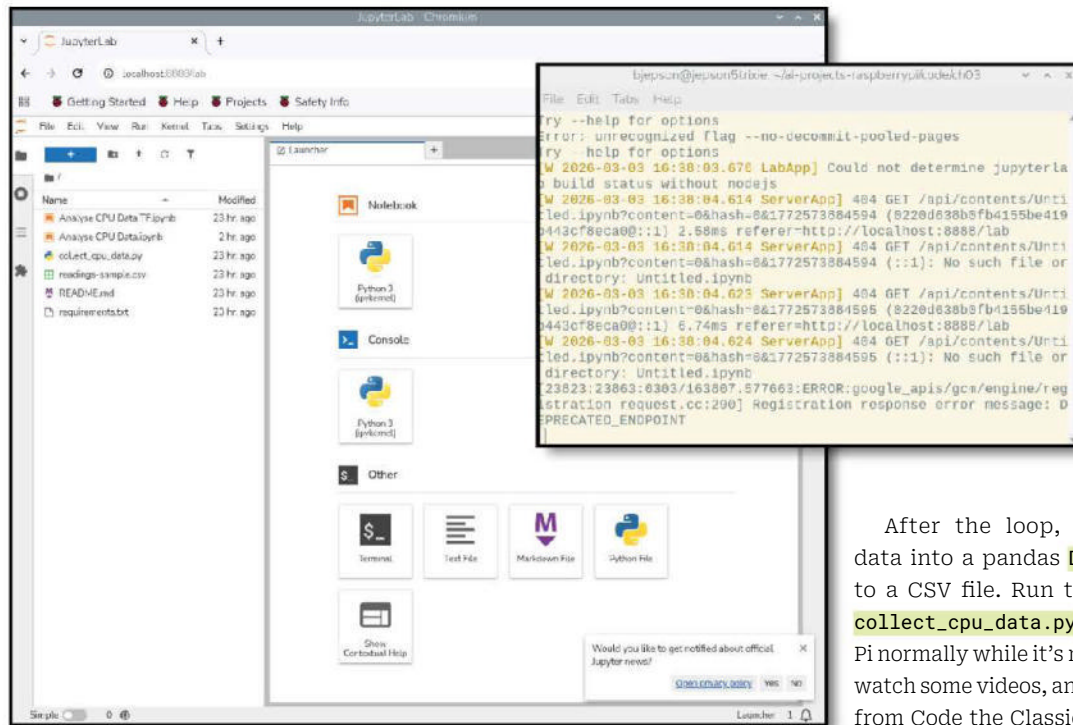
The remaining functions use `os.popen()` to run `vcgencmd`, then they call `extract_number()` to get the numeric result before returning it. The `cpu_speed()` function divides the result by one billion (10 to the 9th power, or `10**9`), before returning it. This converts the value (in Hz) to GHz.

```
def cpu_speed():
    val = os.popen("vcgencmd measure_clock
arm").read()
    return extract_number(val) / 10**9

def cpu_temp():
    val =
os.popen("vcgencmd measure_temp").read()
    return extract_number(val)

def volts():
    val =
os.popen("vcgencmd measure_volts").read()
    return extract_number(val)
```

Finally, we're ready for the main part of the program. The first line in this section checks to see if this script was run directly



◀ **Figure 2:**
Jupyter
Lab

After the loop, the script stores the data into a pandas **DataFrame** and saves it to a CSV file. Run this script with **python collect_cpu_data.py**, and use your Raspberry Pi normally while it's running: browse the web, watch some videos, and play some of the games from Code the Classics (you can install these from the Recommended Software utility).

(rather than being imported as a library). Next, it sets a progress bar (**pbar**) using **tqdm**'s **trange()** function. Then it runs through a loop for each value in **pbar**, and collects the data as an array, appending that array to the array of samples. Each time through, the script updates the progress bar with the most recent set of readings and then pauses for a quarter second.

```
if __name__ == "__main__":
    numsamples = 2500
    samples = []
    pbar = tqdm(numsamples)

    for i in pbar:
        cpu_pct = psutil.cpu_percent()
        load_avg = round(psutil.getloadavg()[0],
2)

        r = [speed(), volts(), cpu_pct,
            load_avg, temp()]

        samples.append(r)
        pbar.set_description(
            f"{r[0]:.2f}GHz {r[1]:.2f}V {r[2]}%
{r[3]} {r[4]}C")
        time.sleep(0.25)

    df = pd.DataFrame(samples)
    df.columns = ['cpu_speed', 'volts',
                  'cpu_pct', 'load_avg', 'temp']
    df.to_csv("readings.csv", index=False)
```

The idea here is to simulate what a normal user would do on a Raspberry Pi. Be sure to occasionally step away from the computer for a minute every now and then, so that you can include some baseline readings.

On a Raspberry Pi 3 or Zero, you will probably find that running this script consumes enough computing resources to greatly affect the readings. You may want to try adjusting the sleep time to a half a second (**0.5**) on one of those devices.

Each time you run the script, it will overwrite the **readings.csv** file. There is a sample file, **readings-sample.csv**, that contains results from running this on a Raspberry Pi 5.

Create your notebook

Now you're ready to create a Jupyter notebook. As with other examples, you can download a ready-to-run notebook from the GitHub repository ([rpimag.co/aiproj-resources](https://github.com/rpimag/aiproj-resources)). However, we'll go through how to create the notebook step-by-step so you can learn how each part of it works.

Make sure you have a Terminal window open and that you've activated the **Regression** virtual environment. If needed, **cd** to the same directory where you created the sample data file.

Run the command **jupyter lab**, and after a few seconds, your default web browser should open to the Jupyter Lab page, shown in **Figure 2**. You'll most likely see a few harmless error messages in the Terminal window while the notebook is running.

The Jupyter Lab user interface is divided into two main panes. The left pane is a collection of vertical tabs, which defaults to the File Browser, which shows a list of files in your current directory. The right pane contains the Launcher tab as well as any notebooks that you have open. You can find the Jupyter

SSH/headless users

If you are running your notebook on a Raspberry Pi over a Secure Shell (SSH) connection, you can still open the notebook from a browser on your local computer. When you connect to your Raspberry Pi over SSH, add the argument `-L 8888:localhost:8888` to your `ssh` command.

This will tell `ssh` to configure a tunnel between your computer and the Raspberry Pi, forwarding TCP port 8888 from the Raspberry Pi to your computer. Port 8888 is Jupyter's default port. When using a tunnel, add the `--no-browser` argument to the `jupyter lab` command. That way, Jupyter will display the URL you should use to open the notebook and won't try to launch a browser.

If you already use port 8888 on your Raspberry Pi for some other purpose, you can use a different port number for your `ssh` tunnel argument and add `--port <number>` to the `jupyter lab` command.

menu just above those two panes. It includes several menus: File, Edit, View, Run, Kernel, Tabs, Settings, and Help.

To create a new notebook, you can click **Python 3 (ipykernel)** in the Notebook section of the Launcher. You'll be greeted by an empty notebook with a blank cell. You can type some Python code into this cell. Might we suggest `print("Hello, World!")`? You can run the code in the currently selected cell by pressing **SHIFT+ENTER**.

Jupyter Notebooks in Visual Studio Code

Although Jupyter usually runs in your web browser, Visual Studio Code can run Jupyter Notebooks. Unfortunately, the Thonny IDE cannot run Jupyter Notebooks. To run the notebooks in Visual Studio Code, make sure you've installed the Python extension and Jupyter extension from Microsoft ([rpimag.co/vscodepy](https://marketplace.visualstudio.com/items?itemName=ms-python.python) and [rpimag.co/jnvscode](https://marketplace.visualstudio.com/items?itemName=ms-toolsai.jupyter)), and that you have the Jupyter Notebook (a file ending in `.ipynb`) open.

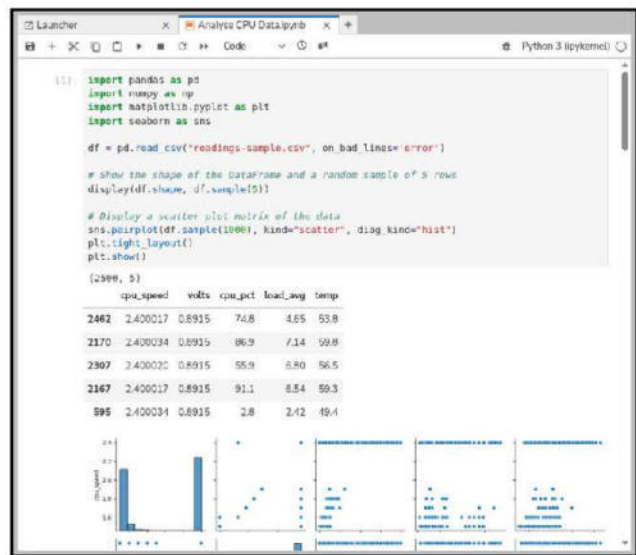
If you've downloaded the example code from the GitHub repository, you can open the `ch03` subdirectory to find the examples for this tutorial. If you'd rather try typing in all the code, press **CTRL+SHIFT+P** to open the Command Palette, start typing **Jupyter**, and choose **Create: New Jupyter Notebook**.

Click the Python menu on the right side of the toolbar (just above the editor) and click **Select Another Kernel**. You should see a list of all the virtual environments you've created. Pick the one named **Regression**. On Raspberry Pi OS, the Visual Studio Code Python extension will look for virtual environments in the `.virtualenvs` folder in your home directory.

Linear regression

In linear regression, the dependent variable has continuous values, meaning it can take on any value within a range. For example, the temperature of a Raspberry Pi 5 CPU can range from about 0°C to roughly 85°C (you could push it higher, but the computer would not be happy). When you obtained the current temperature using `vcgencmd measure_temp`, the result was some number within that range.

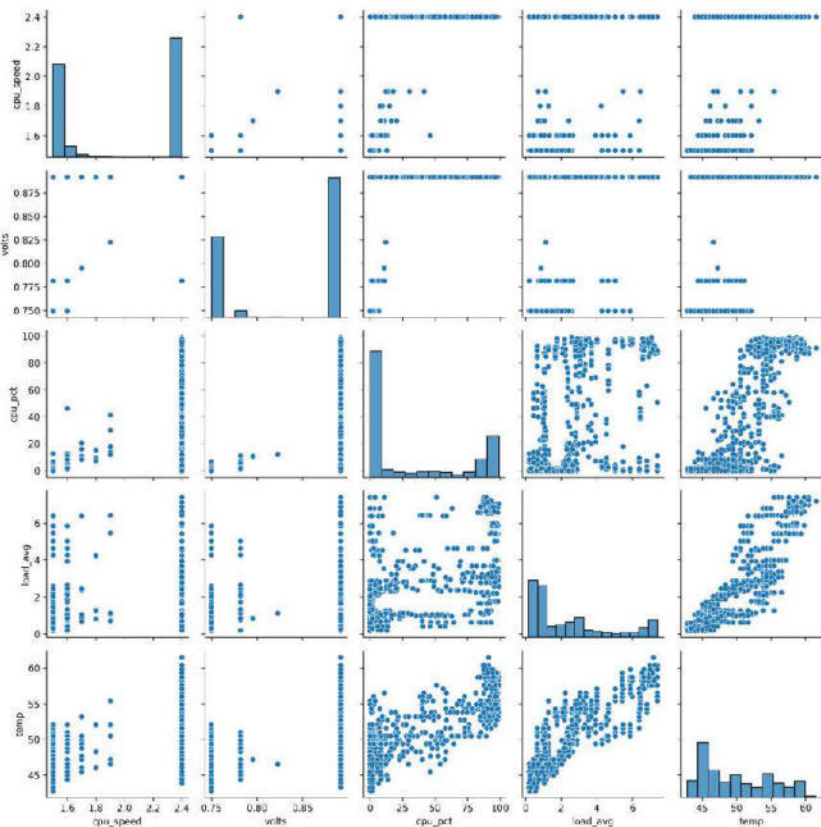
If you started Jupyter Lab in a local clone of the GitHub repository, you'll see one or more notebook files (with the `.ipynb` file name extension) in Jupyter's **File Browser**. You can open the first regression analysis notebook now: double-click **Analyse CPU Data.ipynb**. The notebook will open, as shown in **Figure 3**.



▲ **Figure 3:** Opening the regression notebook

Load and inspect the data

The code in the first cell imports all the modules used in this notebook and then opens the sample CSV file (if you created your own CSV file, change the file name if needed). It stores the contents of the file in a `DataFrame` object called `df`. Next, the notebook prints the shape of the data (2500 rows by 5 columns) and generates a *pair plot*, which is a table that shows the relationship between variables as well as the distribution of individual variables.



◀ **Figure 4:** The pair plot from the notebook

However, for `cpu_speed` and `volts`, the connection between them and `temp` is not as clear. If you look at the distribution of both these variables (the upper-left plot and the intersection of the second column and row), you'll see that extreme high and low values make up most of the data.

While this makes it hard to ascertain the relationship, it doesn't mean there isn't one.

You can use another type of visualisation, a heatmap, to display the value of a useful metric known as the *Pearson correlation coefficient*. This metric measures the linear correlation between two variables in the dataset, and ranges from 1 at the high end (perfectly correlated) to -1 (perfectly *negatively* correlated, meaning that as one value increases, the other decreases). The heatmap makes it easy to pick out high and low values.

Condition your data

Before you create the heatmap, it's worth *conditioning* the data a bit. For time series data like this dataset, a rolling average helps smooth out the data. This next cell applies a rolling average to produce a `DataFrame` named `df_adjusted`. It recalculates each value as a 5-point rolling average using the average of each value and the four values that precede it. This means that the first four values in each variable can't be computed and are replaced with a missing data marker known as NA (not available). The code in this cell drops those values and then displays the heatmaps for both the original `DataFrame` (`df`) and `df_adjusted`. Be sure to run each cell by selecting it and pressing **CTRL+ENTER**.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv("readings-sample.csv",
                 on_bad_lines='error')

# Show the shape of the DataFrame and a random
# sample of 5 rows
display(df.shape, df.sample(5))

# Display a scatter plot matrix of the data
sns.pairplot(df.sample(1000), kind="scatter",
             diag_kind="hist")
plt.tight_layout()
plt.show()
```

Click in the cell to select it, then run it with **CTRL+ENTER**. You can see the complete pair plot in **Figure 4**. The distribution plots appear where the row and column labels refer to the same variable. Distributions start in the top left and continue diagonally down and across. The relationship plots appear in all the other intersection points. We're most interested in the last row, which shows the relationship between temperature and all the other variables. It should be easy to imagine a line that follows the plot of `load_avg` vs `temp`. With a little more effort, you can imagine such a line for `cpu_pct` (you don't have to imagine it, however – simply change the `kind` argument to `"regplot"` and it will draw a line).

```
# Add a 5-point rolling average to smooth noise
df_adjusted = df.copy()
df_adjusted = df_adjusted.rolling(5).mean()

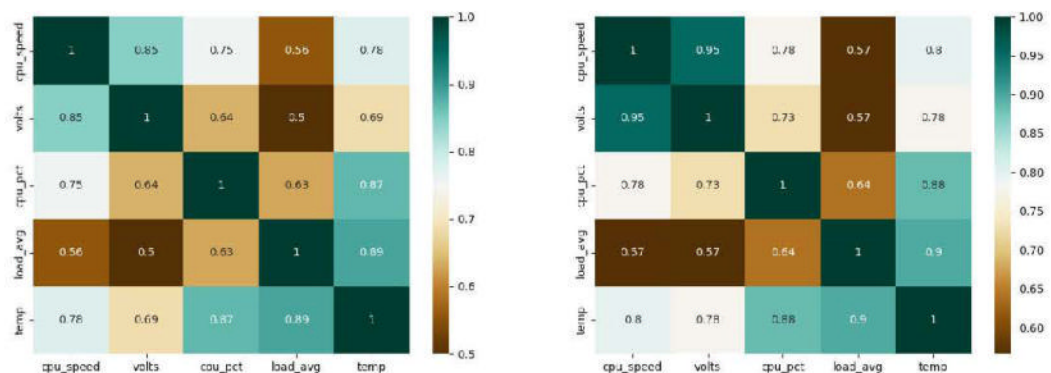
# Because of the rolling average, the first 4
# rows will be NaN
# values. We can drop these rows.
df_adjusted = df_adjusted.dropna()
print(df_adjusted.shape)

# Display correlation for the original and
# adjusted DataFrames.
sns.heatmap(df.corr(), annot=True, cmap='BrBG')
plt.show()
sns.heatmap(df_adjusted.corr(), annot=True,
            cmap='BrBG')
plt.show()
```


The heatmap uses a colour palette known as BrBG, which uses shades of brown and green/blue to clearly highlight the differences in data. Each cell is also labelled with its correlation coefficient, as shown in **Figure 5**. You can see that the correlation improves noticeably with the rolling average applied. Notice that the values on the diagonal are always 1, which makes sense: the values of a variable must be in perfect positive correlation with themselves.

But how should you interpret this correlation heatmap? There are two things to look for here. First, you want to choose independent variables (also called *features* in this kind of analysis) that correlate well to the dependent variable. Second, you may want to avoid using features that are tightly correlated to *each other*. In this heatmap, you can see that **volts** and **cpu_speed** are very well correlated, with a correlation coefficient of 0.95. It's not necessary to discard one of these features, but you need to keep an eye on them as you get further along.

► **Figure 5:**
Correlation heatmaps for the original data (left) and for the data with a 5-point rolling average (right)



Perform the regression

With the data prepared, we're now ready to run the regression. The next cell in the notebook contains a single function named **do_regression()**. Before the function itself, it imports all the modules needed and then defines the function. Perhaps unsurprisingly, this is the function that will handle all the steps needed to actually run the linear regression.

The function's two arguments are a pandas **DataFrame** and a list of features. The first thing the code does is to separate the independent and dependent variables into **X** and **y**. **X** is a **DataFrame** containing just the features columns. The **y** variable is a pandas **Series**, which is essentially a single column from a **DataFrame**. It contains the values of the **temp** (temperature) column.

Next, the function uses the **train_test_split()** function to separate the data into training and test sets. You'll use the training data to train the model, and the test data to evaluate how well it's performing. **X_train** consists of about 75% of the rows from **X**, while **X_test** consists of the remaining rows. The

same goes for **y_train** and **y_test**, but those values are taken from **y**. The code sets a **random_state** to an arbitrary number, which guarantees that **train_test_split()** returns the same results every time you run the code. This is useful if you want to experiment with the code a bit and ensure that you're working with the same training and test data set each time.

Next, the function creates a **LinearRegression** model and *fits* it to the training data. Fitting is the real work of the regression model: it determines a straight line that best fits the relationship between **X_train** and **y_train**. After that, it creates a set of predictions from the **X_test** data and stores it in **y_pred**. Because you separated the training and test data, you have a set of **y** values that are known to be correct, stored in the **y_test** variable.

The next step is to plot the predictions against the actual values and see how close the model came. The function creates a plot of actual vs predicted temperature and uses a scatter plot style with alpha blending of 20%. This means that each point in

the plot will have 20% transparency, making it easier to spot where multiple points overlap because they'll appear darker than outliers.

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import
train_test_split
from sklearn.metrics import mean_squared_error,
r2_score
from IPython.display import HTML

def do_regression(df, features):

    # Separate the independent and dependent
    variables.
    X = df[features]
    y = df.temp
```

```
# Split the data into training and test
sets.
X_train, X_test, y_train, y_test = \
    train_test_split(X, y, random_state=42)

# Create and fit the model.
model = LinearRegression()
model.fit(X_train, y_train)

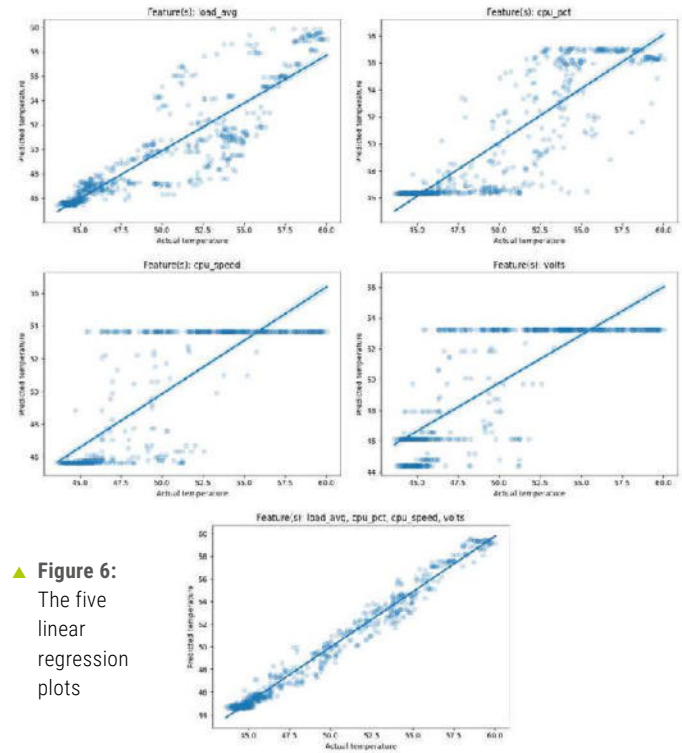
# Plot the test data vs the model's
predictions.
y_pred = model.predict(X_test)
sns.regplot(x=y_test, y=y_pred,
            scatter_kws={'alpha':0.2})
plt.xlabel('Actual temperature')
plt.ylabel('Predicted temperature')
plt.title(f"Feature(s):
          {' , '.join(features)}")
plt.show()
```

The next cell specifies the features and calls `do_regression()` for each one. After it's done that, it makes one last call to `do_regression()`, this time with all the features. This last one is a multiple regression model that predicts a single dependent variable from multiple independent variables.

```
# Specify the independent variables, also known
as "features".
linreg_features = ["load_avg", "cpu_pct",
                  "cpu_speed", "volts"]

# Build a single-variable regression model for
each feature,
# then build a multiple-variable model for all
the features.
for fc in linreg_features:
    do_regression(df_adjusted, [fc])
do_regression(df_adjusted, linreg_features)
```

Figure 6 shows the five plots resulting from this. The top four plots show the actual versus predicted temperature from the single-variable regression model. If the model's predictions match the actual values perfectly, the dots would line up perfectly with the diagonal line. Of the four, `load_avg` is the closest, but it's far from a straight line. The bottom plot shows the multiple regression model for all four features. This looks much better than any of the single-variable regressions.



▲ **Figure 6:**
The five
linear
regression
plots

Graphs and visual intuition alone are an insufficient basis for evaluating a model's performance. Fortunately, there are some helpful metrics we can use to evaluate a model. There are two useful metrics we can use to determine how well the model is performing: the *mean squared error* (MSE) and the *coefficient of determination* (also known as R^2).

MSE calculates the differences between actual and predicted values, squares them, and takes the average. The nice thing about squaring the differences is that it accentuates large errors, so even a few outlier values can increase the error score. The lower the MSE, the better. R^2 is a metric that ranges from zero to one and is based on a formula that assesses how well the predictions and actual values compare. You want this as close to one as possible.

You can add the following code at the end of `do_regression()`. If you're editing this code on the fly, be sure to re-run the code in the cell to pick up the new definition of the function. (The sample code in the GitHub repository will already have this code in it.)

```
# We want a low mean squared error and a
coefficient of
# determination ( $R^2$ ) as close to 1 as
possible.
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
display(HTML(f"<b>Mean Squared Error:</b>
             {mse}"))
display(HTML(f"<b> $R^2$  score:</b> {r2}"))
```

Feature(s)	Mean Squared Error	R ² score
load_avg	5.0797	0.7881
cpu_pct	5.3420	0.7772
cpu_speed	7.6249	0.6820
volts	8.4697	0.6467
All of them	0.4735	0.9803

Run the cell that calls `do_regression()` again; in addition to the graphs, you'll see the above output for each model. Here, it's abundantly clear that the multiple regression model performs best, confirming our earlier interpretation of the graphs.

Earlier, we said that when you fit a linear regression model, it determines a straight line that best fits the relationship between independent and dependent variables. If this sounds a lot like a mathematical linear function ($f(x) = ax + b$), it should, because it is! You can go back to the cell with the definition of `do_regression()` and add the following. Run it, and the following cell again, and you'll see some numbers you can play with (these lines of code are already in the example program in the GitHub repository):

```
# Show some model stats
display(HTML(f"<b>Coefficients:</b>{model.coef_}"))
display(HTML(f"<b>Intercept:</b>{model.intercept_}"))

# Pick and test a few samples
test_val = X_test.sample(3)
prediction = model.predict(test_val)
display(HTML(f"<b>Test samples:</b>{test_val.to_html(index=False)}"))
display(HTML(f"<b>Predictions:</b>{prediction}"))
```

This new code displays two constants for the linear function: one or more coefficients (a); and the intercept (b). It also displays three randomly sampled values and the model's predictions. Here's the output from one of our test runs:

```
Coefficients: [1.99871962]
Intercept: 45.05711954089134
Test samples:
load_avg
0.33
1.28
1.66
```

Let's use the linear function to calculate the predictions by hand. You can add an empty cell and paste this code into it, or better yet, work it out on paper, with a pocket calculator, or for extra points, a slide rule! Here, we're using some sample data for `load_avg`:

```
a = 1.99871962
b = 45.05711954089134
for load_avg in [0.33, 1.28, 1.66]:
    print(load_avg * a + b)
```

The results are as follows, which matches the predictions!

```
45.71669701549134
47.61548065449134
48.37499411009134
```

How can we do this with multiple regression? We can use basically the same approach, but the equation is more complex. Here's some sample output for the multiple regression model:

```
Coefficients: [1.19572801 0.05043958 1.1171309
6.78998038]
Intercept: 37.480108349477504
Test samples:
load_avg.    cpu_pct    cpu_speed    volts
6.836        87.44        2.400028    0.8915
0.300        0.54         1.500017    0.7500
1.140        90.14        2.400026    0.8915
Predictions: [58.79895482 44.63426452
52.12427348]
```


Here's how you'd run that first prediction by hand:

```
b0 = 37.480108349477504 # Intercept
b1 = 1.19572801
b2 = 0.05043958
b3 = 1.1171309
b4 = 6.78998038
X = [6.836, 87.44, 2.400028, 0.8915]
print(b0 + b1*X[0] + b2*X[1] + b3*X[2] +
      b4*X[3])
```

This gives us 58.79895484947271. Given the vagaries of how floating-point numbers are stored and retrieved, this is as near as makes no difference.

Logistic regression

In a logistic regression analysis, the dependent variable has categorical values. For example, if you ask “Is the CPU voltage above 0.825 volts?”, you get a no or a yes (a zero or a one). The answer to this question will never be 0.5, even though the model may estimate a 50% probability that the voltage is high. In practice, these probabilities are used to make a yes-or-no decision by choosing a cutoff point where a “no” turns into a “yes”, such as 50%.

None of the variables in our dataset have a categorical value, but you can synthesize one easily. Suppose you want to indicate whether the processor voltage is high or low. You'll just need to define a threshold, perhaps 0.825 volts, after which you'd consider the voltage to be high. The next cell in the notebook imports some modules that you'll be using, defines a function to calculate the voltage level, and then creates a new **DataFrame** (**df2**) that replaces the **volts** column with a **voltage_level**. It then displays a correlation heatmap for the new dataset.

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import
LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.metrics import
ConfusionMatrixDisplay

def calculate_voltage_level(x):
    return 1 if x >= .825 else 0

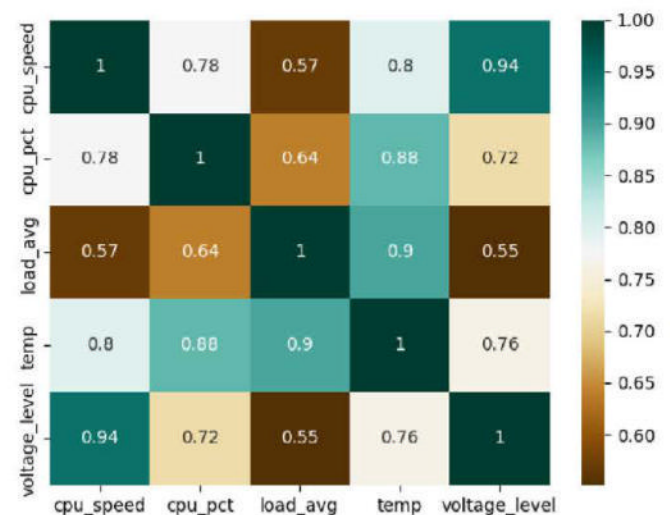
df2 = df_adjusted.copy() # make a copy before
                        changing anything
```

```
# Convert cpu voltage into a low/high indicator.
df2["voltage_level"] =
df2.volts.apply(calculate_voltage_level)
df2.drop("volts", axis=1, inplace=True) # remove
                                         volts column

# Visualise correlation
sns.heatmap(df2.corr(), annot=True, cmap='BrBG')
plt.show()
```

As with the linear regression, we should look for features in **Figure 7** that correlate to the dependent variable (voltage level), but don't necessarily correlate too closely to one another. CPU speed, CPU percentage, and temperature correlate well to voltage level, but they are all somewhat well correlated to one another. Of these, temperature correlates the highest to both other features, so we can probably safely eliminate temperature. You can also turn to domain knowledge to construct an argument for eliminating temperature: in general, CPU speed is fully or in part controlled by the voltage level, so we wouldn't need a predictive model to determine one from the other.

Here's the code for running a simple logistic regression between the selected features and dependent variable. As with the linear regression, the code isolates the X and y values and splits the data into training and test data:



▲ **Figure 7:**
Correlation matrix
for the new
dataset

```
# Prepare the data
features = ["cpu_speed", "cpu_pct"]
X = df2[features].values
y = df2.voltage_level.values
X_train, X_test, y_train, y_test = \
    train_test_split(X, y, test_size=.2,
                    random_state=42)

# Scale the data and run the regression.
logreg = make_pipeline(StandardScaler(),
                      LogisticRegression())
logreg.fit(X_train, y_train)

# Generate predictions based on the test inputs
y_pred = logreg.predict(X_test)

# Evaluate the model with a confusion matrix
cm = confusion_matrix(y_test, y_pred)
disp =
ConfusionMatrixDisplay(confusion_matrix=cm,
                      display_labels=logreg.classes_)

disp.plot()
plt.show()
```

Unlike the linear regression, this code uses the `make_pipeline()` function to create a two-part workflow: first, the data is scaled using scikit-learn's standard scaler, which transforms the data such that it has a mean (average) of 0, and a standard deviation of 1. So, for example, if we scaled the values 0, 10, and 100 with `StandardScaler`, we'd end up with -0.82, -0.59, and 1.41. These three numbers average out to zero, and if you calculate the standard deviation, you'll get 1.

The second stage of the pipeline runs the logistic regression. The call to the `fit()` function runs both pipelines in order. Once that's done, the model is complete, and the code creates a set of predictions based on the test inputs (`X_test`) and stores them in `y_pred`. You can evaluate the model by comparing the predictions to the correct values in `y_test`.

When working with logistic regression, you'll use different evaluation methods than what you used for linear regression. One of the most intuitive options is the *confusion matrix* (you may also want to look into scikit-learn's *classification report*).

You can see the confusion matrix in **Figure 8**. From upper left, moving clockwise, the cells are true negatives, false positives, true positives, and false negatives.

Standard scaler, standard deviation

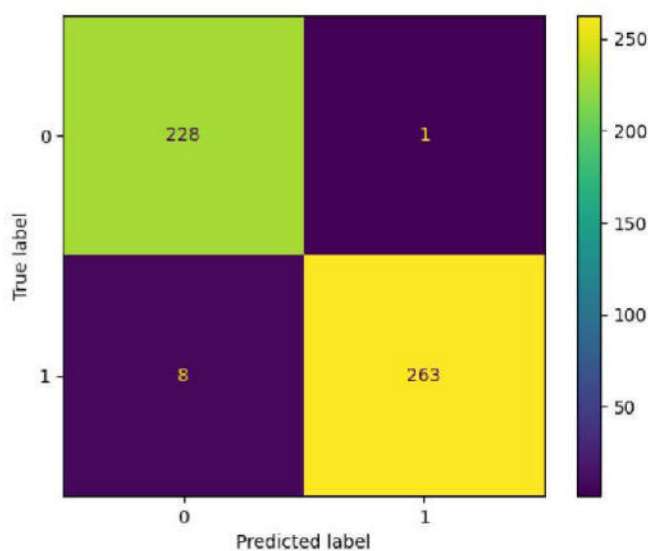
You don't need to calculate the standard deviation, but if you wanted to, you'd subtract the mean from each value, square the result, add them all up and divide by the number of items in the list. Since the mean is zero, that's one less step:

$$-0.82^2 + -0.59^2 + 1.41^2$$

$$0.67 + 0.35 + 1.99 \approx 3$$

Of course, 3 divided by 3 gives you 1, showing how `StandardScaler` makes good on its statistical promise while maintaining the proportional distance between data points. In both the original and the scaled data set, the distance between the second point and the first point is 10% of the distance between the last and first points.

▼ **Figure 8:** The confusion matrix



1. A true negative occurs when both `y_pred` and `y_test` agree that, for a given input from `X_test`, the result is 0. There are 228 of these in the confusion matrix.
2. A false positive occurs when `y_pred` (incorrectly) predicts a result of 1 where `y_test` has a 0. There's only one false positive.
3. You get a true positive when `y_pred` and `y_test` agree, but in this case, they agree on a result of 1. There are 263 of these.
4. A false negative, of which there are 8, is when the model incorrectly predicts a result of 0, but the real value (`y_test`) is 1.

You can tell immediately that the model is doing quite well at predicting 0 (low voltage threshold) but is not quite as good at calculating when your Raspberry Pi has crossed the high voltage threshold (1). Another important detail is that the total number of low and high values is relatively well-balanced. There are 229 low values in the test data (total count of true negatives and false positives) and 271 high values (total count of true positives and false negatives). It's reasonable to conclude that the model performs well.

It's reasonable to conclude that the model performs well

From regression to neural networks

A linear or logistic regression is essentially equivalent to a single-layer neural network. It takes inputs (features, or independent variables), applies a model to those inputs, and returns predictions. As with a logistic regression, a neural network layer can use an *activation function* to convert continuous probabilities into a 0 or 1.

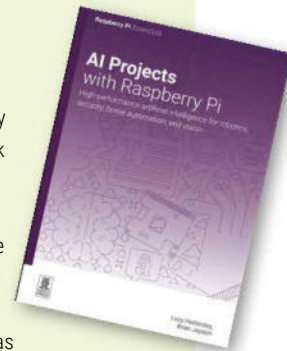
In machine learning, neural network layers are stacked on top of one another with many hidden layers in between the input layer and output layer. Further, in neural networks, you can have different kinds of activation functions, and combinations of input, hidden, and output layers allow neural networks to generate more complex models to help make better predictions.

Although linear and logistic regression can't model the complex situations as neural networks, they are easier to interpret and adjust. As with every tool, your choice of linear regression, logistic regression, or neural network will depend on what problem you are trying to solve. 🟢

AI Projects with Raspberry Pi

This tutorial is part of our latest book, *AI Projects with Raspberry Pi*. This book explains how you how to run AI models directly on the CPU and on the Raspberry Pi AI HAT+ and AI HAT+ 2 neural network accelerators. You'll learn how to identify and categorise objects, recognise and generate speech, translate language, use generative AI models to create images or text, and train models to make predictions from sensor readings, such as temperature and acceleration.

rpimag.co/aiprojectsbook



Doll's house dec and tech

A smart little home with a dose of tiny tech



Maker

Nicola King

Nicola is a freelance writer and sub-editor. This month she's wishing it was this quick and easy to get her actual house decorated inside and out and up to scratch.

@holtonhandmade

YOU'LL NEED

- A plain or blank mini 'house' of some kind (such as rpimag.co/hcdollhouse)
- Scissors, pencil, and ruler or measuring tape
- Paint and a paintbrush/es for outside décor
- Flooring material
- External roof covering (optional, as you could just paint it)
- Appropriate glue and glue palette or brush
- Wall covering for internal walls (we used rpimag.co/hcbluwallpaper)
- A few items of furniture (e.g. a small table, a telephone etc.)
- Tiny FX microcontroller (rpimag.co/tinyfx)
- Mini speaker
- PIR Stick (rpimag.co/pirstick)
- Battery pack or power bank

QUICK TIP

Choose a furniture paint for the outside of the house that doesn't require primer or sanding to save some time and effort.

Over the next couple of craft/tech tutorials, we are going to escape from the real world for a while and go small... doll's house small, to be exact. Having purchased an inexpensive, plain wooden doll's house, we are going to craftily decorate it, and then add a few tech touches to inspire you to maybe do the same with your own unique creation. This is potentially a long-term project that you can come back to, tinker with, and improve on very gradually. We plan on making a space for it, and popping back on a regular basis to do a bit of, well, home improvement. (The fact that our actual hallway is in more pressing need, with new tins of paint literally ready to go, is by the by!)

Paint job

The very first task is to paint the exterior of the house with whatever colour you wish. We chose to use a chalky furniture paint which works well on wood, but there is really a broad spectrum of options to choose from for outside décor. You'll need different-sized brushes so that you can do some of the more delicate painting in corners and around window frames with a thinner tip. Paint in sections, leave to dry ideally outside on a sunny day, and then complete the next section. Don't worry about getting paint strokes or drips on the roof, or even on the inside walls, as we are going to cover those with papers – or you could of course paint them with a different colour. Have some sandpaper ready after the paint has dried so you can smooth any areas that need some finessing. If you want to paint inside the house, now is also the time to do that before any papers are added.



Warning!

Solvents

Always use paint and glue in a well-ventilated space.

rpimag.co/solvents

- ▲ The interior is a work in progress; by the next tutorial, it should look a lot better
- ◀ Painted and papered, our doll's house is starting to come together: it's compact and bijou in a lovely location



QUICK TIP

As doll's houses are so small, if you select too many colours when decorating rooms, it can look messy. Keep it simple.

*Potentially
a long-term
project you can
come back to*

QUICK TIP

If you have some wallpaper left from a recent home décor project, keep it and use in your doll's house, as you will only need small offcuts.

- We've chosen to use specific doll's house materials; however, you can really use any papers or paints to decorate

The magnetism of miniature

The appeal of doll's houses is wide, as both children and many adults enjoy downsizing. For children, who obviously like to play and pretend, the attraction is obvious, but the adult interest in them is worth investigating a little further. Reasons why adults enjoy them might include:

1. Escape from the everyday world to a place where you are in control of everything. If you want to try out interior design ideas that might be too bold for your own home, then this is somewhere to let creative expression rip.
2. Nostalgia is a huge factor, connecting people with childhood memories, and the idea that they are making something that can be handed on to future generations.

3. The satisfaction of small – there is something very appealing to our natures about detailed miniatures. Perfection in a tiny form really makes people smile.
4. Collectability – a number of adults just enjoy collecting items to add to their doll's house or houses. It's a hobby like any other, and for those with the space for it, many hours can be spent perfecting the look, content, and layout of doll's houses. We are using a very inexpensive base for our house, but doll's houses can cost hundreds and thousands of pounds. The following link will take you to a website showing some doll's houses that are hugely expensive, intricately detailed, and simply breathtaking: mulvanyandrogers.com.





▲ We'd love this parquet-style flooring in our own house!

QUICK TIP

Much like when painting walls in your own home, keep a damp rag on hand to mop up drips when slapping the paint on.

Paper caper

So, let's now look at adding the roof effect, flooring, and wallpapers. It's optional to add these, as you could just paint them, and add your own artwork if you wish, but we purchased some papers in the hope that it would speed up the process a little. This author honestly found it easier to paper the feature wall in her lounge than these tiny doll's house room walls and floors. You really need to implement any wallpapering skills you may have, including brushing out those air bubbles, and cut your papers as accurately as you can, so measure twice and cut once. Have a little brush on hand to help you smooth the papers whenever needed. A craft knife is also useful to cut away excess paper.

We found it easiest to paste glue onto the appropriate area of the house, and then attach the papers, but see what works for you. We did the roof first, followed by walls, then floors. We painted the window side of the walls, as cutting holes in the paper would have been a huge labour of love. Leave to dry and then we're ready to start adding some items of furniture; we borrowed some from a Sylvanian Families set, which fitted the 1:24 scale well. Playmobil items might work well too.

Dog and bone

We also added some tech in the form of a ringing phone and barking dog. As it makes it so easy to output sound to a tiny speaker, we opted to use Pimoroni's RP2040-powered Tiny FX board, triggering the telephone ring sound with its BOOT button. To make the dog bark when motion is detected in the same room, we added a PIR Stick connected to the board's three-pin Sensor port.

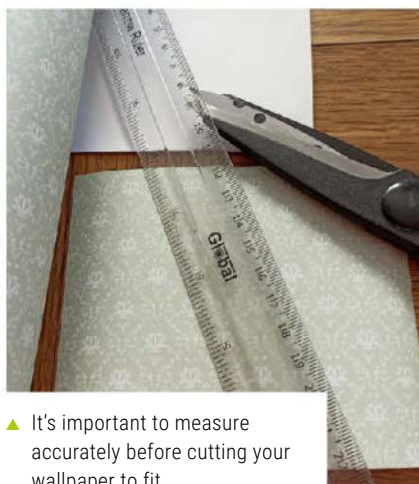
For the sounds, we downloaded a couple of examples from **Freesound.org**: an old-fashioned telephone ringing (rpimag.co/phonering) along with a dog barking/growling (rpimag.co/dogbark). We edited both in Audacity (audacity.org) to trim any silent parts and increase the volume where needed, then exported them as mono WAV files with the names **ring.wav** and **bark.wav**. We uploaded them to the Tiny FX using Thonny IDE's Files tab on a connected computer.

Written in MicroPython, and making use of the `tiny_fx` library, our code (see the **bark_ring.py** listing) is fairly simple. In a `while True:` loop, we first check for a press of the BOOT button; if so, and a sound isn't already playing, we trigger the phone sound. Next, we check whether the PIR Stick has sensed any nearby motion; if so (and no sound is playing), we trigger the dog bark sound. The use of the `motion_active` variable prevents the code from re-triggering the barking sound repeatedly while someone remains in front of the sensor.

Saving the code as **main.py** makes it run automatically whenever the Tiny FX is powered up – we used a 3×AA battery pack for this, but a USB power bank would also work.

Project pause

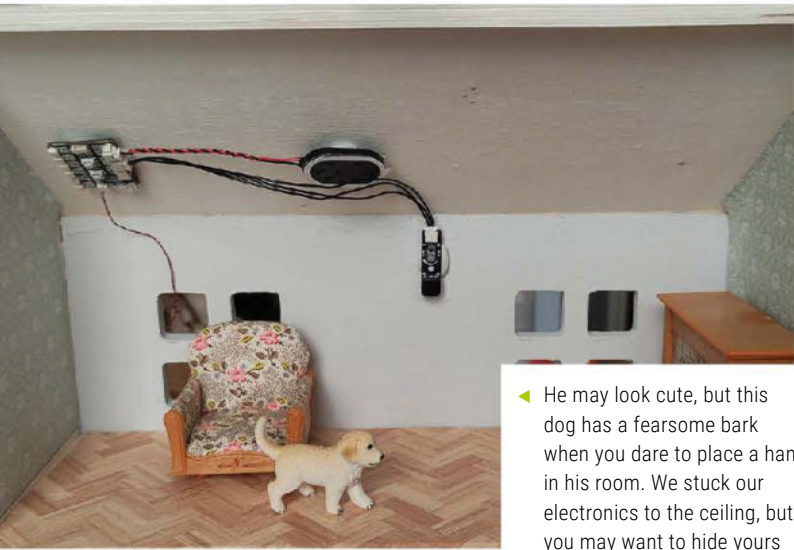
This is almost one of those projects that will never be fully finished, as you can keep tweaking it and changing things as you wish. Make it unique and see what you have in your craft and tech stashes that can add some new elements, and remember that there is also no pressure to make it a modern home; you can adopt a very retro take if you want, as we have with our old-style telephone. In fact, we think a vintage look is a fitting way to escape the real world. In our next tutorial, we will hand-make a few home comforts and add some more tech to the mix. ▣



▲ It's important to measure accurately before cutting your wallpaper to fit

▼ Decorating a doll's house is as messy as doing it for real, so take appropriate measures to protect surfaces





◀ He may look cute, but this dog has a fearsome bark when you dare to place a hand in his room. We stuck our electronics to the ceiling, but you may want to hide yours

bark_ring.py

> Language: Python

DOWNLOAD
THE FULL CODE:



rpimag.co/github

```
001. import time
002. from machine import Pin
003. from tiny_fx import TinyFX
004.
005. # Initialise Tiny FX hardware
006. tiny = TinyFX()
007.
008. # Configure the PIR Stick data pin
009. pir_sensor = Pin(TinyFX.SENSOR_PIN, Pin.IN)
010.
011. motion_active = False
012. time.sleep(3)
013.
014. while True:
015.     # 1. BOOT Button (Phone Ring)
016.     if tiny.boot_pressed() and not
        tiny.wav.is_playing():
017.         tiny.wav.play_wav("ring.wav")
018.
019.     # 2. PIR Sensor (Dog Bark)
020.     if pir_sensor.value() == 1:
021.         if not motion_active and not
            tiny.wav.is_playing():
022.             motion_active = True
023.             tiny.wav.play_wav("bark.wav")
024.         else:
025.             if motion_active and not
                tiny.wav.is_playing():
026.                 motion_active = False
027.
028.         time.sleep(0.05)
```

A word on scale

For those familiar with the film *This Is Spinal Tap*, you might remember a Stonehenge stage prop that was supposed to be 18 feet tall, but turned out to be 18 inches high due to, well... we'll call it human error. Scale matters.

It's very easy to think that any small items of furniture or décor will fit perfectly in your doll's house, but you do need to think about the scale here, or how big or small the doll's house is compared to real life. Making the assumption that anything would fit was the mistake that we made: we purchased a wooden chair which we thought would be perfect, but it dwarfed the room and looked somewhat ridiculous. So, you do need to experiment a little on this front once you have your house to understand what is going to fit perfectly inside it.

One of the most common sizes is 1:12, meaning one inch in the doll's house equals 12 inches in real life, and this would be a 'standard' house. Ours is actually way smaller than that, probably closer to 1:24 which would be a 'space-saving' doll's house, so we can't buy standard furniture. This isn't an issue, just something to be aware of, and we therefore need to be more creative and try different ideas out, including maybe making a few of our own items. You can actually go as small as 1:48, and probably smaller than that, which might test your eyesight. Moral of the tale is: don't mix your scales.

There are also practical points to consider related to scale choice: how much room do you have to house your house? What's your budget, and how much time do you have to spend on the project?

There are lots of places to find small-scale dolls' house furniture online, so have a browse and see what you can find based on the scale of your house. A couple of options might be: dollshousedirect.co.uk and rpimag.co/hcdollfurniture.

QUICK TIP

Leave some empty space in each room – if you completely fill it, the house will look cluttered.

► We bought the wrong-size furniture for our house!



Citizen science with Raspberry Pi

Science is all about taking an interest in the world around us and sharing our findings widely. Use Raspberry Pi to explore science at home...

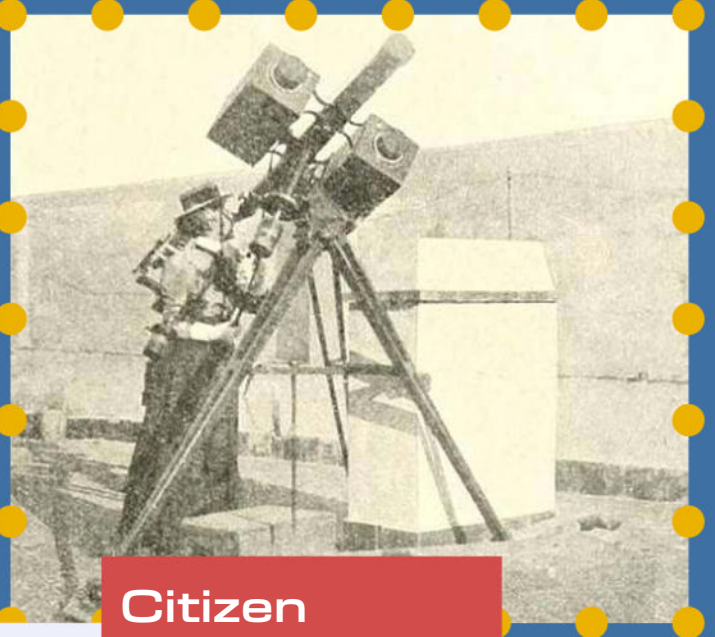
By **Rosie Hattersley**

Hedgehogs, moths, bats, birds, and kangaroos – Raspberry Pi cameras and computers are used to monitor, photograph and count them all, whether for personal inspiration and enjoyment or for wider environment accounting purposes.

Environmentalists make good use of Raspberry Pi cameras to send back photos from remote locations in a bid to manage livestock and crops and ensure food security and sustainability in the face of earthquakes, deforestation, and floods. Meanwhile, there's also plenty of enjoyment for us hobbyists in monitoring natural phenomena as well as animals and plants, and noting what thrives and where. Whether you want to tick off an unusual species or are simply curious about the great wide world, there's plenty of reason to record and photograph what your Raspberry Pi rig can see. With Raspberry Pi AI Camera smarts or High Quality Camera imaging, you can snap photos and videos, and potentially identify what's in shot too.

And it's not just about imaging. Raspberry Pi can connect to a wide range of sensors (rpimag.co/sensors): temperature, humidity, light, audio, water quality, soil quality; even the presence of humans (rpimag.co/humanpresence24ghz).

"Environmentalists make good use of Raspberry Pi cameras"



Citizen science matters

Perform science at home

Exploring the world around us is part of our natural curiosity. Humans and animals have been doing it for millennia, long before anyone thought to give it a label such as science or botany. Arguably, science has always relied on citizens to show interest in a specific topic and conduct research and observations. It's what the likes of Gilbert White were doing back in 1789, with his seminal botanical studies, and Charles Darwin, who minutely followed the life cycles of insects and plants. Victorians were avid hobby scientists and students of nature, with hothouses specifically designed to recreate the ideal growing conditions for plants they'd admired overseas and brought samples back to the UK to propagate. Famously, less financially secure fossil enthusiast Mary Anning (rpimag.co/mary-anning) had to polish and sell the ammonites she found on Dorset beaches to well-heeled palaeontologists who wanted them for their own cabinets of curiosity, and women in general were often prevented from becoming members of organisations such as the Royal Astronomical Society, preventing pioneers such as astronomer and astrophotographer Annie Maunder from joining and largely remaining amateurs: rpimag.co/annie-maunder.

Thankfully, everyday science studies are within the grasp of ordinary folk these days. In fact, the term citizen scientist – coined by MIT in 1989 in reference to community groups studying and reporting on environmental hazards – recognises the important contribution of enthusiasts in scientific research, matching amateur data gathering to larger trends.

Microsoft's paper on the planetary computer, launched in 2020 as a partial response to the UN's Global Assessment on Biodiversity, says "it is abundantly clear that the world needs greater access to environmental data to assess, diagnose and treat the natural systems that society depends on" with data powered by machine learning cited as "a game changer" (rpimag.co/cleansociety).

Citizen scientists around the world can play a critical role in planning for and mitigating the effects of climate change and diminishing water supplies in the face of ever-greater demand for energy. As the projects we cover here show, getting involved in the nitty-gritty of the planet's numbers can also be very rewarding.

▲ Maunder captured images of the Milky Way and solar coronas back in 1900

Credit: English Heritage

Space projects

Turn your attention to the stars with out-of-this-world projects

Science United

scienceunited.org

You may have heard of SETI@home – the search for extraterrestrial intelligence project that you could ‘loan’ your computer in the interests of international research. SETI is no longer actively looking for participants, but there are other projects that could benefit from your Raspberry Pi’s passive computing power. Many of these are also based around University of California Berkeley’s BOINC software (rpimag.co/boinc). Choose the area of research you would like to support, from astronomy to AI, chemistry to earth sciences, and Science United attaches your computer to related projects. Simply create an account, choose your area of interest, and specify whether your computer offers low, standard or maximum computing power.

Rosetta@home (rpimag.co/rosetta-at-home) is another distributed computing project you can join and run through Raspberry Pi 64-bit OS. It also uses the BOINC app or platform.

NASA Science

rpimag.co/nasa-sci

NASA’s website has details of nearly 50 citizen science projects that involve counting and identifying specific objects and which can be done from any internet-connected device. Lunar Melt and other space-based projects involve tagging photos of rocks, craters, and boulders on the surface of the Moon and then submitting findings via the project portal. The project looks at the impact of asteroids hitting the Moon’s surface, creating craters and sometimes melting rocks to cause lunar melt flows. Other NASA Science projects include options to study high-altitude clouds to track how they have changed over time. While this can be done with a smartphone, it’s probably more fun to use an All Sky Camera and your Raspberry Pi. We’ve featured several All Sky Camera builds over the years, including this autoguider one: rpimag.co/autoguider.





Compute for Science

• BOINC lets you help cutting-edge science research using your computer. The BOINC app, running on your computer, downloads scientific computing jobs and runs them invisibly in the background. It's easy and safe.

• About 30 science projects use BOINC. They investigate diseases, study climate change, discover pulsars, and do many other types of scientific research.

• The BOINC and Science United projects are located at the University of California, Berkeley and are supported by the National Science Foundation.



START COMPUTING!

To contribute to science areas (biomedicine, physics, astronomy, and so on) use Science United. Your computer will help current and future projects in the areas you choose.

[Join Science United](#)

Or download BOINC and choose specific projects. This will let you participate in competitions and systems like Gridcoin.

[Download BOINC](#)

Citizen science with Raspberry Pi

Learn

- Science projects
- Science publications
- User manual
- Add-ons
- Web resources

Communicate

- Message boards
- Help
- Email lists
- Report bugs

Help

- Translate
- Test
- Publicize

Computing power - Certificate - Poll - Teams

Scientists:

- Create and maintain a BOINC project.
- Run Docker and Autodock jobs with BOINC Central.

Programmers:

- Visit the BOINC github repo

Contact - Papers - Graphics - Events - History

News

BOINC client 8.2.4 released

The 8.2.4 version of the BOINC client has been released for all platforms. Download it here. Release notes are here. The new version supports Docker apps - an important step for BOINC.

19 Jul 2025, 21:48:02 UTC - [Details](#)

UK Fireball Alliance and Global Meteor Network

rpimag.co/fireballs-nz

Astrophotography is a perennial favourite Raspberry Pi project and lends itself well to citizen science. Although many thousands of celestial objects are tracked and regularly photographed, meteors and shooting stars are less predictable. We've covered projects by both the UK Fireball Alliance and Global Meteor Network, which have locked on to, and traced, meteorites and the places they fell to Earth, including the fairly famous Winchcombe meteorite. The UK Fireball Alliance keeps enthusiasts up to date and explains how to build your own meteor camera. You can also view a world map of meteor sightings: rpimag.co/meteormap. An example from 2024 in New Jersey saw a meteor crash-land in someone's bedroom. The fragments were collected and able to be studied thanks to contact with the American Meteor Society: amsmeteors.org. Should you happen to see a fireball (a meteor burning up on encountering the Earth's atmosphere), you can report it at rpimag.co/fireball-report.

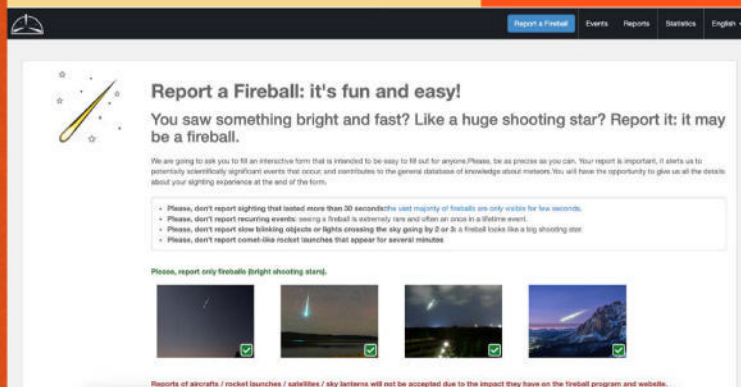


"Astrophotography is a perennial favourite Raspberry Pi project"

Radio astronomy

rpimag.co/meteor-beacon

Radio astronomy can be used to spot and track celestial objects and phenomena that might otherwise be invisible or go undetected. If it gives off radio waves, it can be detected using Raspberry Pi (or another approach) and an SDR (software-defined radio). It's a lot less weather-dependent than other forms of stargazing and noctilucent cloud observing. This is a great option if you don't want to go to the trouble and expense of an astrophotography setup, but like the idea of seeing – and hearing – meteors via an amateur radio network.



Keep an eye on critters

Watch over wildlife from the wilderness to your back garden

Pick up a penguin

rpimag.co/penguin-watch

Lots of citizen science involves counting and logging numbers of a specific species, also known as biologging. Zooniverse has been running projects of this sort for more than a decade, most notably PenguinWatch, in which citizen scientists counting penguins from their own computer or Raspberry Pi screens directly helped evidence the number of penguins. The ruggedised cameras were designed by Arribada, an environment technology non-profit which fitted them with Raspberry Pi Zero computers. By recording penguin numbers, Zooniverse's citizen scientists were able to measure the fishing impact on penguin numbers around the Antarctic's South Sandwich Islands and South Georgia, persuading the governments of the islands to designate them and the surrounding seas as protected marine sites.

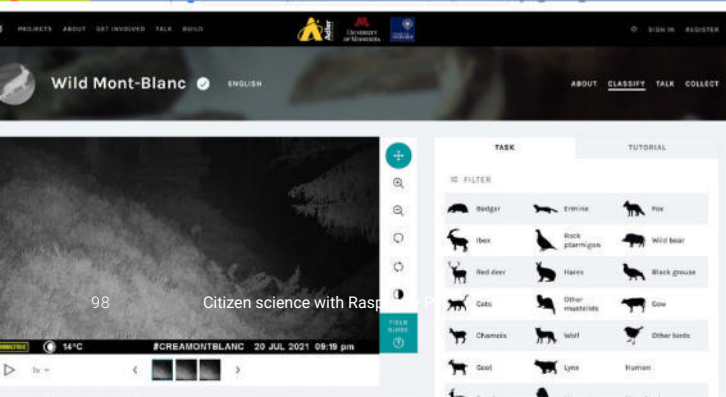
While penguins can still be counted, other current Zooniverse projects you might like to join include marmots and mammals on the Mont Blanc massif. This is a more advanced identification project than the penguins one since there are a lot of possible creatures to identify and tag. However, there is a detailed identification guide along with tools to enhance and zoom in on the on-screen animals. This helps you narrow down the options and exclude any birds or animals that it can't be, based on whether the creature in the photo has wings, horns and so on.

BeeWalks and the National Biodiversity Network

beewalk.org.uk

A BeeWalk asks participants to regularly walk a transect or route and record their sightings each time. Check the BeeWalk website to see whether others are already recording and submitting data from your chosen area before you start, since ideally you want to be queen recording bee of your own patch so more areas are covered. Of course, if the bees you're recording are regular visitors to your garden, you could set up a Raspberry Pi camera to photograph and identify them.

Records submitted to BeeWalks become part of the National Biodiversity Network database, of which there are 147 different organisations collecting data. If there is a specific type of creature or plant for which you'd be interested in recording sightings, head to rpimag.co/biodiversity-database to see whether there's already a citizen science scheme available.

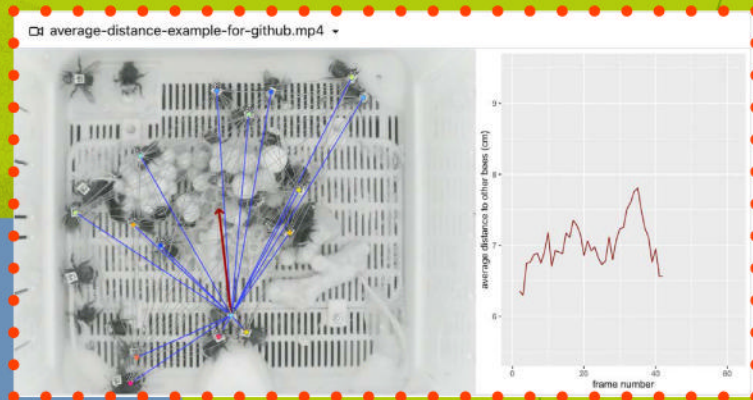


BeeMonitor and BumbleBoxes

rpimag.co/bumblebox-git

If you keep bees or know of a local colony you're keen to study, consider making your own BumbleBox. The system records a known colony or nest, automatically collecting and quantifying information about bees you spot and their behaviour, with ArUco tags used to plot their precise locations.

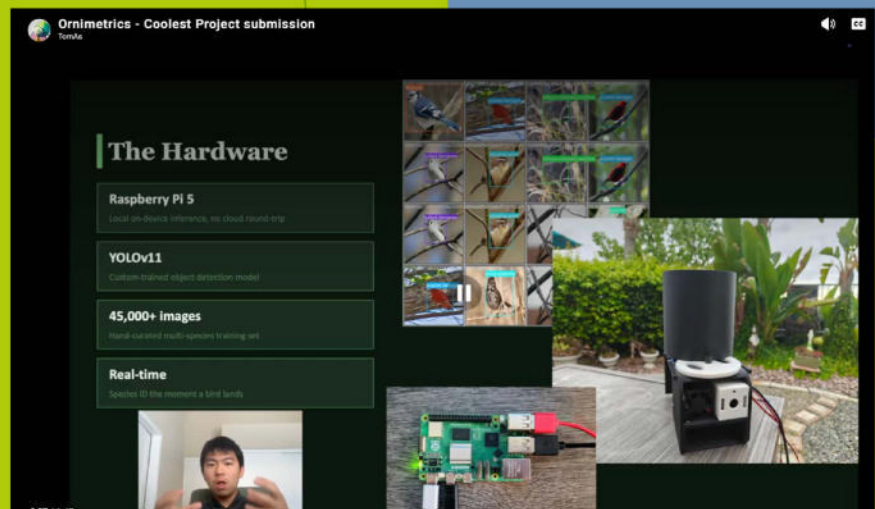
Build instructions for the 3D-printed nest arena and recording system based around Raspberry Pi 4 and HQ Camera mounted on the top of the Bumblebox with a 32GB SD card can be found in the GitHub repository.



You might also find some inspiration in this BeeMonitor project in which an apiarist undertakes a long-term study of generations of bees: rpimag.co/bee-monitor.

Ornimetrics AI-powered smart bird feeder

Audubon's bird surveys encouraging ornithologists to submit sightings are among the earliest examples of the public being asked to contribute to wildlife surveys. The National Audubon Society was set up in 1905 and has been recording birds in the US since the Christmas Bird Count in 1900. These days, you can take part in the annual RSPB Big Bird Watch or simply log sightings using a phone app such as BirdNET. We particularly like the acoustic-based bird identifier (rpimag.co/birdnet-pi) and applaud this fun project that seeks to feed only specific bird species, not the greedy ones that ruin bird table feed-times for everyone else: rpimag.co/ornimetrics-pi.





The Great Blue - Loker District (UK) - 17-05-2007 18:56:48

Build a bird box camera

rpimag.co/pi-bird-box

Nestbox Live (rpimag.co/pi-nestbox) is one of the most fun wildlife observation products around and even has its own YouTube channel.

If you prefer to make your own, this infrared bird box features a Raspberry Pi NoIR Camera Module, allowing you to observe birds even at night without disturbing them (since birds can't see infrared light). The box needs to be completely sealed so potential predators can't see there's something living inside, and to be sited several feet off the ground and away from the reach of anything that might attack chicks. You should test the night-vision camera module with your Raspberry Pi before installing it in the bird box, while the LED needs to be powered from the 5V connector rather than Raspberry Pi's own GPIO pins. The bird box can simply be a wooden Gardman model (around £10). Since these are quite small, you will probably need to adjust the camera's focal range so objects inside the box look sharp. Tape the Camera Module in place inside the bird box, hang it up in your garden, and complete the setup instructions on your computer to create a YouTube channel for your birdbox.

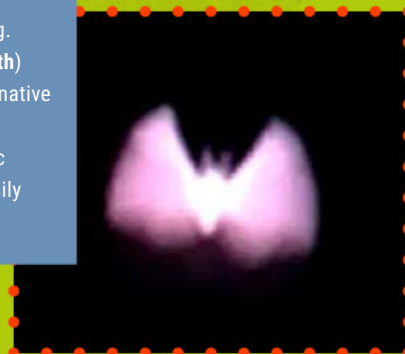
National Bat Monitoring Programme

rpimag.co/batinator

If you wish to, you can help the National Bat Monitoring Programme by sending them your data. This is a great option if you have a bat roost nearby and are monitoring bats arriving and departing with your Raspberry Pi setup. The annual Roost Count (rpimag.co/roost-count) takes place in June, but other types of recording are needed too and in many cases you can simply share your findings with officers in your local bat group (rpimag.co/bat-monitoring).

Martin Mander, aka The Batinator, offers a great Instructable for creating your own bat box at rpimag.co/batinator-guide while Extreme Kits sells a ready-to-go acoustic monitoring kit known as Bat Ears (rpimag.co/bat-ears).

Both bats and moths can be tracked using acoustic monitoring. AudioMoth sell an Open Acoustic Device (rpimag.co/audiomoth) capable of recording audible and ultrasonic frequencies. Alternative firmware for the AudioMoth can be downloaded (rpimag.co/audiomoth-usb) to turn it into a USB microphone for ultrasonic audio on a Raspberry Pi or other computer. You could also easily build your own insect monitoring rig: rpimag.co/diy-mothbox.



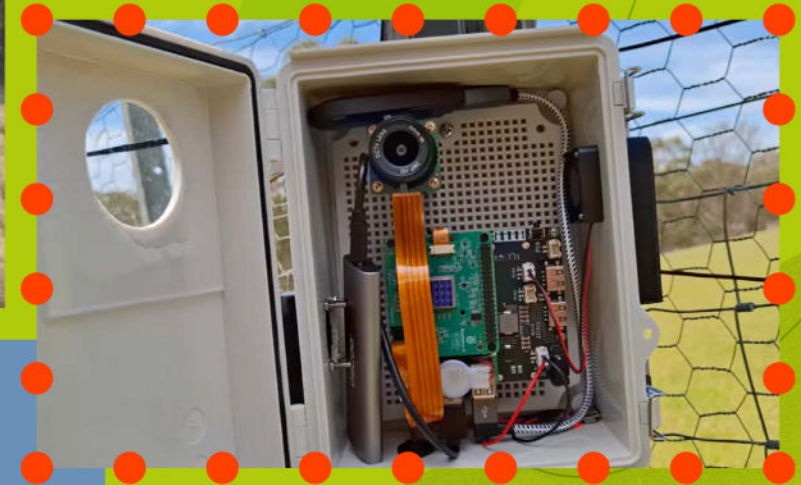


Off-grid camera trap

rpimag.co/ai-thermal-cam

We couldn't resist including this fantastic camera trap, not least because maker Luke Ditria decided to test out Raspberry Pi's image detection and machine learning abilities on kangaroos! As he explains, the camera trap differs from a trail camera in that it is looking for and recording specific species. The real-time clock on Raspberry Pi 5 schedules shutdown, and wake-up, times. There's a battery pack that powers the camera trap for around 14 hours between charges. The whole thing is designed for use in off-grid, remote locations with no wireless or 4G/5G coverage, but potential for LoRaWAN connectivity. Luke uses an extra-long set of pins so his project can accommodate a LoRa module on top of Raspberry Pi 5.

If Luke's kangaroo and mouse detector caught your imagination, it's worth watching the rest of the video for his detailed look at thermal cameras.

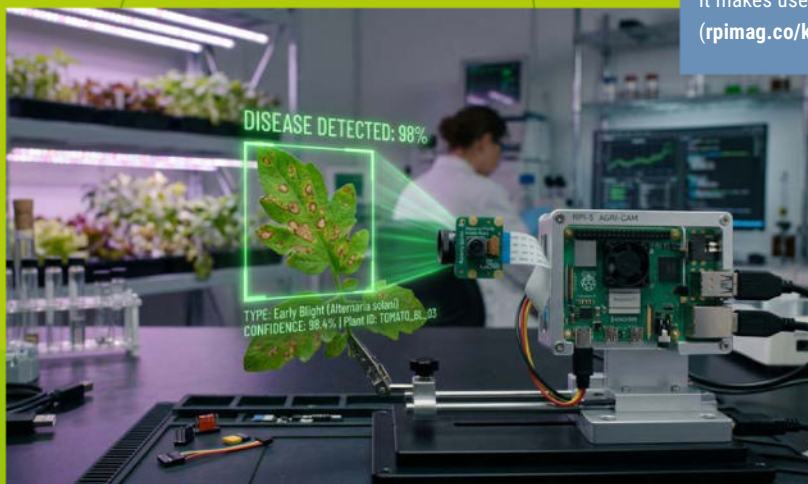


AI plant disease detection

rpimag.co/iot-plant-disease

This guide from Let's Learn IOT uses a Raspberry Pi 4 and Camera Module to build a real-time plant disease monitor that classifies 38 plant diseases. It uses live camera images and pushes the results to a Node-RED dashboard via MQTT. It makes use of the Kaggle PlantVillage dataset (rpimag.co/kaggle-plant-disease).

"Observe birds even at night without disturbing them"



Data capture

Use Raspberry Pi and sensors to monitor a changing world

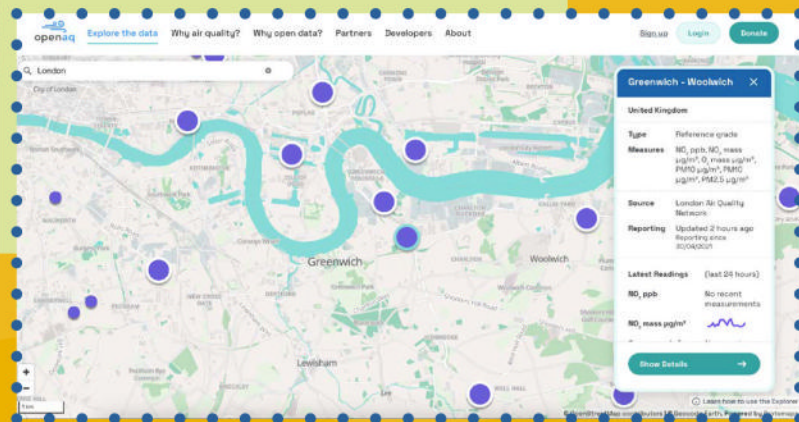
Pimoroni Enviro+

rpimag.co/enviro-pi

With rising temperatures and climate change come issues over air quality, pollution, and health. Since young people are more susceptible to poor air quality, it's a sadly apposite idea to get them to measure their immediate environment and then report their findings.

Developed alongside researchers at the University of Sheffield, this excellent HAT enables you to monitor the air quality in your home with sensors for the humidity, temperature, and air pressure as well as particulates and pollutants.

By following Pimoroni's instructions, you can use the Enviro+ to create an outdoor air quality sensor using Raspberry Pi Zero WH or another Raspberry Pi model (rpimag.co/enviro-luftdaten) and there's an option to submit data to an online community (rpimag.co/sensor-community).



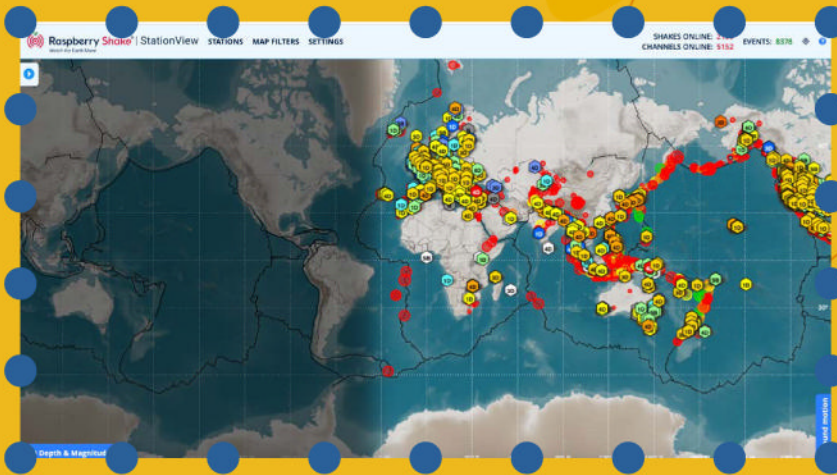
Handheld Air Quality Meter

rpimag.co/air-quality-meter

Arnov Sharma has created a handheld air quality meter which can use a Raspberry Pi Pico 2 connected to a gas sensor to display the current levels of a host of hazards. The Air Quality Meter was created with the intention of measuring the degree of air pollution in New Delhi.



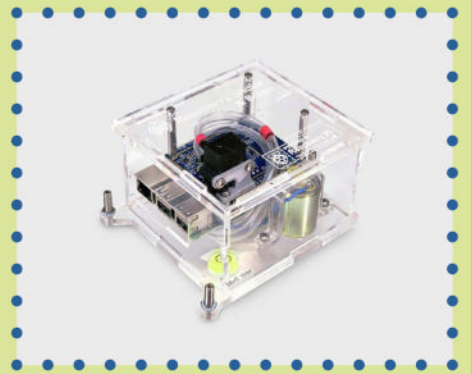
"Some of the most interesting citizen science projects have been done by younger Raspberry Pi users"



Raspberry Shake

[raspberrypi.org](https://raspberrypi.org/raspberryshake.org)

Raspberry Shake offers a fantastic way to learn about seismic events, pretty much as they happen. There is a worldwide network of seismographs tracking volcanic eruptions, explosions, tsunamis, and earthquakes as well as other ground-shaking events such as elephant stampedes. These are logged on a publicly accessible map, complete with locations, the size of the seismic event, and which type of Raspberry Shake captured the data about it. Schools, scientists, and hobbyists can all view and contribute data. Based around Raspberry Pi, the first Shake model was created in Panama to measure tremors on the slopes of Volcán Barú. Subsequent models have added extra features. A Raspberry Pi 3 or 4 computer oversees the geophone, accelerometer, and infrasound sensors to keep track of earth tremors.



Indoor Air Quality Asthma Monitor

rpimag.co/indoor-air-meter

Some of the most interesting citizen science projects have been done by younger Raspberry Pi users keen to find out about issues that directly affect them. Teenagers Katerina and Leonides decided to measure the impact of everyday pollutants in their home, which was especially important because Leonides has allergic asthma. A Fermion ENS160 air quality sensor helped them establish a five-point scale of total volatile organic compounds and CO₂ levels in their home over a four-month period. Cleaning products came a close second to perfumes and cosmetic creams as key asthma triggers, but most of the issues were resolved by improved ventilation in the house. Since submitting their experiment to Raspberry Pi via Coolest Projects, they have been working on a bracelet-worn version that can trigger an alert when airborne pollution rises.



ONLY THE **BEST**

Storage solutions

Give your Raspberry Pi extra storage and a speed boost

By **Phil King**

What a lot of great products we've got in store for you as we revisit this hot topic – which we first covered back in issue 150 (rpimag.co/150).

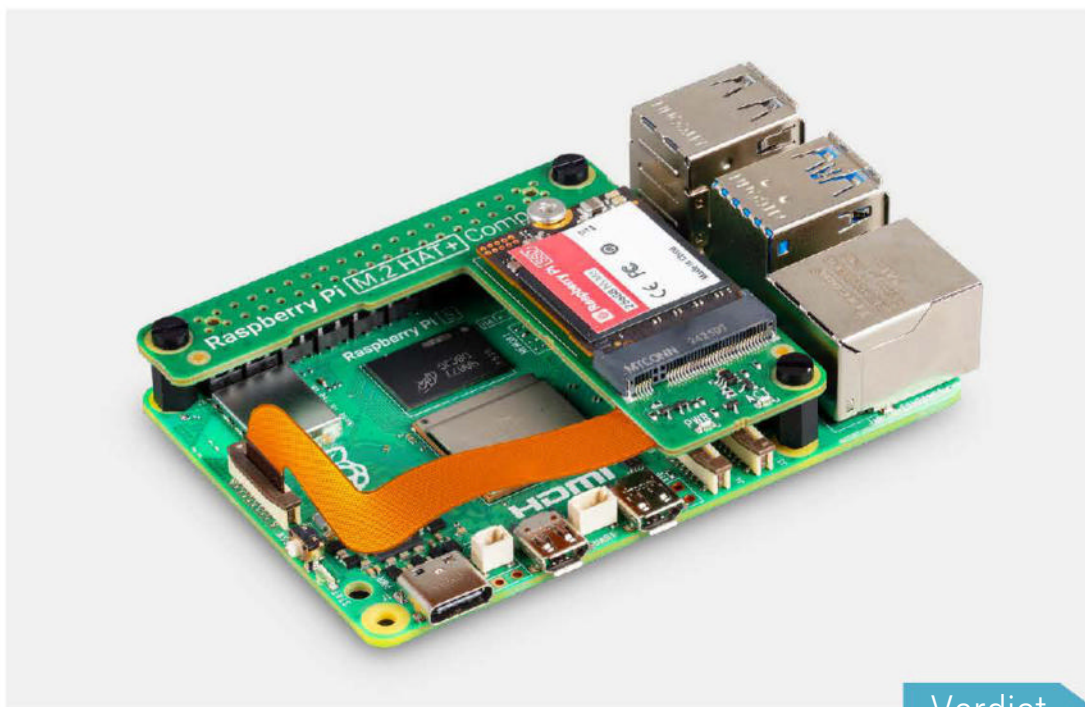
Since then, quite a few new storage-based devices have come onto the market.

While a microSD card is the default storage medium – as well as hosting the OS – for most Raspberry Pi computers, its limited read/write speeds can slow down the whole system. So, if you want to build a media centre or home server, say, you'll be looking for a speedier storage solution.

The obvious choice is a solid-state drive (SSD), due to its greater speed and capacity. With the PCIe port on Raspberry Pi 5, you can connect an NVMe SSD via an M.2 adapter board for faster data transfer. We take a look at a few adapter options here, including one in a case. We also try out the official USB Flash Drive and a board that enables you to repurpose an old SATA drive.

Raspberry Pi M.2 HAT+ Compact

Raspberry Pi | £14 / \$15 | raspberrypi.com



◀ The L-shaped M.2 HAT Compact sits fairly low on Raspberry Pi 5

Verdict

A compact option for SSD storage, it fits neatly inside the Official Case.

Launched in September 2025, this slimline version of the Raspberry Pi M.2 HAT+ is designed so that it will fit inside the Official Case for Raspberry Pi 5. As such, it doesn't include the full-size version's stacking header (used to give clearance for an Active Cooler), so it sits lower when mounted on Raspberry Pi 5 using the supplied stand-offs and screws. Cooling is supplied by the Official Case's integral fan.

While the L-shaped board features the same M.2 key slot as its larger sibling, it

can only accommodate the smaller 2230 size of NVMe SSD – such as the official Raspberry Pi SSDs (256GB, 512GB, or 1TB) – due to it being positioned sideways instead of longways. Instead of an FPC ribbon cable, a flex-rigid PCB (pre-soldered to the board) is used to connect it to Raspberry Pi 5's PCIe port.

Other than the form factor, it works in exactly the same way as the original M.2 HAT+, enabling you to use the SSD for speedy storage, with the option of booting the OS from it.

NVMe Base Duo

Pimoroni | £30 / \$33 | pimoroni.com

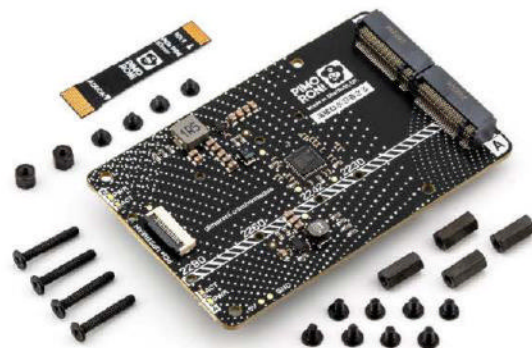
We've covered the original NVMe Base previously and were impressed by its slimline form factor and positioning underneath Raspberry Pi 5, thereby not obstructing access to its GPIO pins and ports, along with its ability to take an M.2 SSD up to 80mm long (size 2280).

The same is all true of the Duo version, except that – as its name suggests – you can connect two SSDs. The two M.2 key slots are positioned at the opposite edge

of the board than on the original, with the PCIe connector also moved to the centre of the other edge – a short, straight flexi ribbon cable connects it to Raspberry Pi 5.

One downside with the Duo is that it's limited to PCIe Gen 2 (not the faster Gen 3 available on the standard Base), and the total bandwidth of around 450MB/sec is shared between the two drives.

On the upside, for a neat setup, Pimoroni sells a black aluminium case to house Raspberry Pi 5 and the NVMe Base Duo.



Verdict

A good option for dual drives, although speed is reduced.

▲ It can hold two SSDs and sits underneath Raspberry Pi 5

NEO 5 M.2 NVME Case for Raspberry Pi 5

Argon / The Pi Hut | £23 / \$31 | argon40.com / thepihut.com



▲ A robust case with a special base to house an SSD

Last time we looked at the topic of storage in this section, we covered the Argon ONE V3 M.2 NVME PCIE Case. While that's still very much available, Argon also produces an alternative in the more compact NEO form factor.

Coming in three parts – two aluminium sections and a plastic base – the NEO 5 case looks great and feels very solid. The middle section features a fan next to curved fins for very effective cooling – as we discovered in our roundup of cooling cases in issue 156 (rpimag.co/156).

This special version of the NEO 5 case replaces the standard base with one (also available separately) containing a carrier board for an M.2 NVMe SSD – up to the 80mm 2280 size. The slot is located under a removable metal heatsink on the underside, with a thermal pad supplied to aid SSD cooling.

A handy booklet gives illustrated assembly instructions along with terminal commands to optimise power and boot and install Argon's control software.

Verdict

A compact case and SSD carrier in one – great value.

Raspberry Pi Flash Drive

Raspberry Pi | From £29 / \$30 | raspberrypi.com

As well as supplying official SSDs, Raspberry Pi has launched an official USB 3.0 flash drive. Available in two storage capacities (128GB or 256GB), it uses non-volatile flash memory to retain data when disconnected. It's a convenient way of transferring files, creating backups, or even running an operating system – you can install one using Raspberry Pi Imager.

With countless USB flash drives on the market, you may be wondering what makes this one so special. As well as looking 'flashy', with its shiny (and

super-tough) all-aluminium enclosure, it offers fast, reliable data transfer with a sustained write speed of 75MB/s on the 128GB version; 150MB/s on the 256GB one.

There are a couple of SSD-style bonus features too, accessible via terminal commands: SMART health reporting and TRIM operations support (to tell the drive which blocks of data are no longer needed so it can wipe them in advance). The Flash Drive will also enter a low-power state when idle.

Verdict

A smart-looking USB flash drive with some smart features.

▼ It looks and feels great and you can even use it to boot Raspberry Pi OS



X1008

SupTronics / The Pi Hut | £43 / \$57 | suptronics.com / thepihut.com



While you could go out and buy a brand new NVMe SSD to use with your Raspberry Pi 5, there are other options out there. SupTronics has launched a range of adapter boards to use with standard SATA hard drives or SSDs. So you could repurpose one you happen to have lying around – good for the pocket and environment.

The X1008 model features two connectors for different-size SATA drives – you can use both simultaneously. The large PCB (164.5 × 101.6mm) features an outlined section to mount your Raspberry Pi 5, a 2.5-inch SATA connector on top, and a 3.5-inch one on the underside.

You'll need a 12V barrel jack PSU (not supplied) to power everything, including Raspberry Pi. Also, note that the drive(s) can't be used to boot the OS, only for data storage. Still, it's a neat solution. Other SupTronics models include the X1007 (for two 2.5-inch SATAs), X1100 (2.5-inch SATA via USB), and X1009 (for up to five SATAs!).

Verdict

A clever way to repurpose old SATA drives.

◀ Connect a 2.5-inch and/or 3.5-inch SATA drive to your Raspberry Pi 5

Multifunctional Adapter (C) with M.2 Socket

Waveshare / The Pi Hut | £10 / \$13 | waveshare.com / thepihut.com

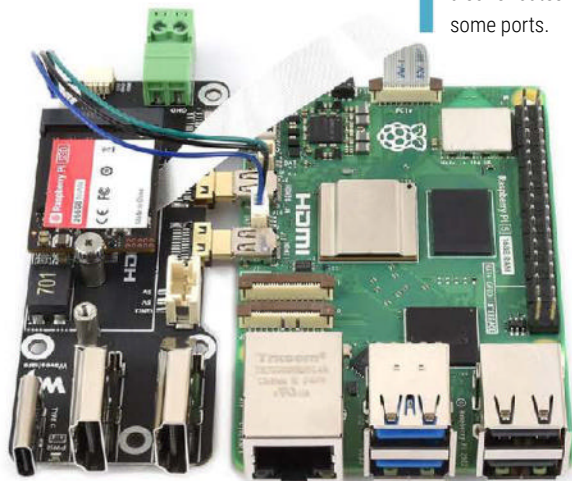
This is an alternative option for using an M.2 NVMe SSD with Raspberry Pi 5. Instead of mounting on top of the SBC, it plugs into the side of it, re-routing its USB-C power port and its two micro HDMI ports (to standard-size ones) to the same side as Raspberry Pi's USB-A and Ethernet ports. So it's good if you want an 'all out the back' arrangement.

Of course, the key feature is the M.2 key slot. It can accommodate a 2230 or 2242 size NVMe SSD. A ribbon cable is supplied to connect it to Raspberry Pi 5's PCIe port. There's also a three-wire connector cable to extend the SBC's UART debug port to one on the adapter.

To supply power to both the adapter and Raspberry Pi 5, you can either use a standard PSU with the USB-C port or a 5V supply connected to the adapter's green screw-terminal block. There's also a three-pin 5V/GND output to power other devices. ▣

Verdict

A compact M.2 adapter that also re-routes some ports.



▶ Plugging into the side of Raspberry Pi 5, this compact board can hold an M.2 NVMe SSD

MINI USB 2.0 MICROSD CARD READER

The Pi Hut | £3 / \$3 | thepihut.com

If you want to use a standard microSD card, you'll need a USB adapter to write the OS to it on a computer. This one is tiny, the card fitting into its metal plug, and comes with a cover and keychain.

- ▶ The smallest microSD card adapter we've ever seen!



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



Powered by
Raspberry Pi

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/poweredbypiiapply

Smart classrooms and even smarter remote camera rigs are among the plethora of products that are officially accredited by Raspberry Pi. By **Rosie Hattersley**

A **Powered by Raspberry Pi logo is a badge of honour that signals that a product has passed a slew of tests set by Raspberry Pi itself.** This month's showcase of accredited products gives a fair idea of the sheer breadth of uses to which some form of Raspberry Pi has been put. From a retro games arcade to a precision engineering aid to the Compute Module-enabled environment monitoring tools keeping track of industrial noise and emissions in Chile – and a quirky musical instrument based around bleeps, bloops and flashing lights – smart, open-source hardware and software controllers are at heart of them all.

Elsewhere this issue, we look at the popularity of citizen science and how placing cameras in remote locations

and sending data back to base can help us work out what's thriving and what's struggling. Canada's Zeitdice cameras have added LTE, solar power, and weather protection along with AI processing to provide developers and agriculturalists a real-time view of their investments.

The products we highlight here are just some of the hundreds that now sport the Powered by Pi badge. Each of them has 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, or taking a punt on a zany-sounding synth or self-assembling project. You will find information about the testing procedures involved in gaining Powered by Pi accreditation – along with how to apply – at pip.raspberrypi.com.

Zeitdice ZBeyond

Canada | zeitdice.com

Zeitdice creates visual sensing products that connect cameras to places that can be hard to reach and monitor. The company provides solar-powered and 4G LTE cellular networks over which it can beam back real-time photos and video feeds so the user can see what's happening even at far-flung locations. The ZBeyond smart camera's weather- and dust-proof 12MP IMX708 Sony image sensor has a 120-degree field of view and can capture 4K to 8K timelapse videos as well as offering motion-sensing and anomaly detection. The IMX708 is assisted by Raspberry Pi hardware and on-board AI-processing capabilities. For horticulturists, this might involve detecting growth abnormalities, pests, or nutrient deficiencies, while property owners or builders can check who is on site as well as creating a snazzy timelapse of their in-progress build.



Remote Bee

UK | remotebec.co.uk

Remote Bee is specially designed for CNC users to give them greater control and manoeuvrability. Built for ease of use, Remote Bee has a high-contrast screen built into a joystick-style controller displaying three axis coordinates so you know precisely where your tool is. There's a lockable dial for nippy axis changes, while quick-access buttons

take you straight to custom settings, pulse, and jog modes. The device is compatible with Duet 2 and 3 CNC controllers – a single line of code connects Remote Bee to Duet's PanelPort. Everything runs off a 32-bit Raspberry Pi RP2040 circuit board, which makes for straightforward firmware updates too.



Dog Paw

USA | dogpawmusic.com



The Dog Paw is a standalone playable grid instrument with weighted keys and a built-in synthesizer.

Thanks to Raspberry Pi 5, it is highly customisable and can import tracks that you can remix with ease. The Dog Paw has programmable dynamic keys and FX faders and there is a planned range of apps to go alongside the synth itself. The instrument has an isometric grid rather than a traditional piano-style keyboard and features a choice of scale and chromatic modes. The idea is that new musicians learn production as they play, while more confident players have plenty of means of musical expression including full MIDI and MPE control.

Iconic Arcade

USA | rpmag.co/iconic

Iconic Arcade is on a mission to bring much-loved arcade games to screens of all sizes, faithfully recreating the design and gameplay and selling them in flat-pack kits that games aficionados will no-doubt delight in assembling and firing up. A range of cabinet sizes allows Iconic Arcade to offer versions to suit different gaming scenarios such as a breakout room, individual players' homes, and games rooms. The company licenses ROMs of official game titles such as the legendary 1990s Street Fighter II and the 1980s classic BurgerTime (like Donkey Kong, but you play as Peter Pepper the chef). This has recently launched at 9/10th of the original cabinet size with a 24-inch touchscreen, plus an HDMI port so a second screen can be attached. The company promises that "combined with arcade controls, every jump, ladder climb, and burger drop feels just as you remember". Games run off Raspberry Pi 5 inside a distinct wooden cabinet and are Wi-Fi-enabled so players can stream their gameplay as well as receiving OTA updates. A further 15 arcade games are included alongside the headline title for each cabinet, so you get the full 'multicade' experience.

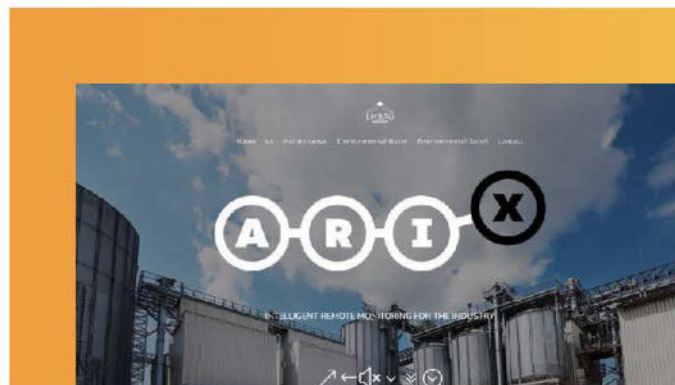


ARIX

Chile | arix.cl

E MBAU Ingeniería developed the ARIX system as an online environmental monitoring tool using Raspberry Pi Compute Modules to support industrial and field robotics. ARIX's variable monitoring makes use of Raspberry Pi's AI edge-processing abilities to assess environmental impact, specifically focusing on issues of environmental noise and odours.

As well as using connected sensors to detect existing output levels, and track industrial footprints, the Chilean company has successfully deployed ARIX to help clients remain within preset limits. It provides acoustic and ambient noise control along with maintenance and upkeep service contracts.



NEXINEO virtualised ICT classroom

Slovakia | nexineo.com



NEXINEO is a virtualised teaching environment that aims to revolutionise ICT in classrooms, not least by making technology in schools rather less expensive to install and run. Specialised software running on Raspberry Pi OS connects each workstation to a Windows-based NEO GO Server that acts as a hub and provides a virtual desktop environment within which each pupil can work. Each NEO workstation client device attaches to the back of a flatscreen monitor and typically runs on a Raspberry Pi 4. Each NEO Client uses a lot less energy than a standard desktop computer or workstation as well as being far more cost-effective to buy in the first place! The classroom teacher has control over access to the NEO desktop computers, locking screens as needed, sending out assignments centrally, as well as viewing a student's work area in order to prompt them or provide pointers. 📺

10 amazing:

printer projects

Create something physical with a Raspberry Pi

In a world of satnavs and digital tickets, it's not often we need to use a printer. When we do though, we're often reminded of how cursed these pieces of ancient technology can be. Luckily, as Raspberry Pi OS is based on Linux, it can fight these curses to create fun and imaginative projects. Thanks to its customisability as well, it's also great for the new world of 3D printing. Here's just ten ways folks are printing with Raspberry Pi.

01. BMOctoprint

3D printing friend

rpimag.co/bmooctoprint

Just because you can put OctoPrint on a Raspberry Pi with a small screen doesn't mean you can't put it in a replica of Adventure Time's friendly gaming robot.

06. Raspberry Pi Pico Ticket Game

Box of mysteries

rpimag.co/picoticket

How about breaking some ice by having a big red button print a question for the button-pusher to answer? This one uses a Pico in an old cigar box to do it, and we think it's neat.

02. Raspberry Pi Zero-Based Instant Camera

Point-and-print

rpimag.co/instantcam

With a thermal printer (the kind used for till receipts) and a Raspberry Pi Camera module, you too can create an instant camera! Albeit with a little less colour.

07. The Open Printer

Freedom to print

opentools.studio

A Raspberry Pi Zero-powered project to make printers more open and affordable for everyone. It's looking to crowdfund a proper launch soon.

03. Network printer

Upgrade older hardware

rpimag.co/netprint

Instead of buying a fancy new printer when yours is no longer supported, add a Raspberry Pi to make it usable over your network – no HP subscription required.

08. DIY Klipper Pad

A different controller

rpimag.co/klipperpad

"Do you want to upgrade your 3D printer to run at stupidly fast speeds and gain a load of functionality?" asks Ricky Impey. Yes, we do, and you can with this build.

04. OctoPrint Rig

Upgraded 3D printing

rpimag.co/octorig

Thanks to the very DIY nature of 3D printing, you can control them with many other devices as well – such as a Raspberry Pi with OctoPrint software installed on it.

09. AirPrint label printer

Apple, meet Raspberry

rpimag.co/airlabel

Using similar techniques to other wireless Raspberry Pi print servers, user sallovecroc has managed to make his label printer work with Apple's AirPrint functionality.

05. PolaPi-Zero

Tiny Polaroids

rpimag.co/polapi

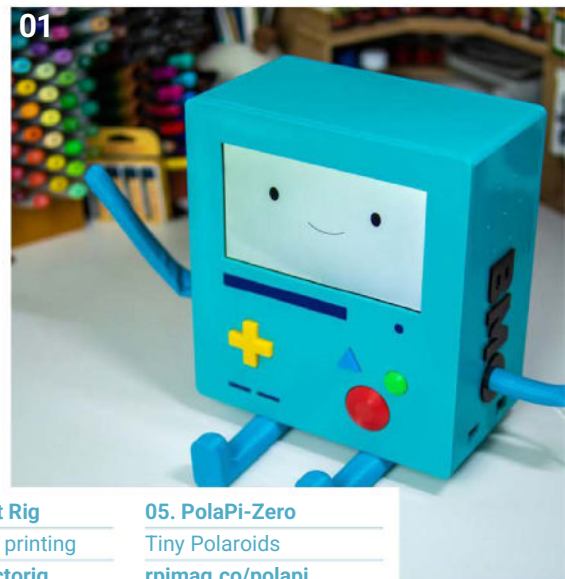
This very stylish, Polaroid-style camera uses a Raspberry Pi Camera Module like the HQ Camera Module, allowing for different lenses and focus if you so wish.

10. Retro printer

Translating old printers

rpimag.co/retroprint

Electrical engineer Rich Mellor had to get old machines working with modern printers, so invented a HAT that has an old Centronics port which turns Raspberry Pi into a middleman.



02



03



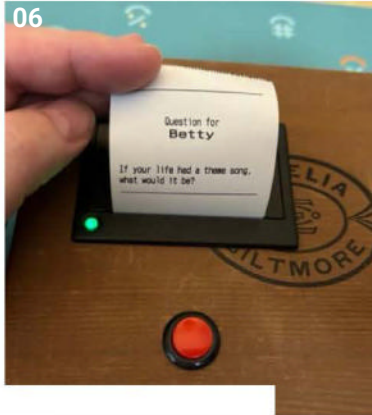
04



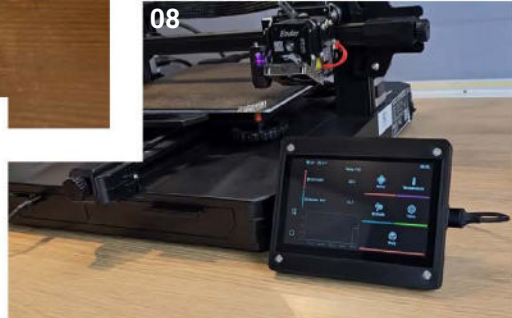
05



06



08



07

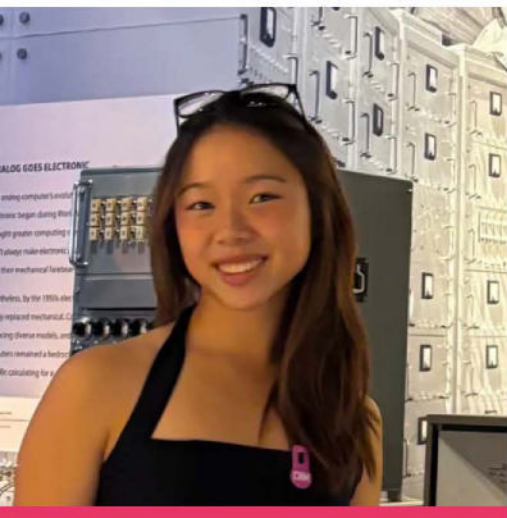


09



10





Sage Cadence

An electrical engineering student trying to provide the educational content she would have wanted

-  **Name** Sage Cadence
-  **Occupation** Student
-  **Community role** Tech influencer
-  **URL** rpimag.co/sagecadence

As we've mentioned a lot over these past 169 issues, Raspberry Pi was created to help get more younger students into computer sciences. While that has definitely worked, there are always a few that slipped the net – at least during school, that is. Sage Cadence discovered it when she entered higher education.

“It was through the space club at my university [that I discovered Raspberry Pi]: the Stanford Student Space Initiative,” Sage tells us. “Arriving at college, I was itching to get hands-on experience with engineering. I walked into the club meeting room on the first day when one of the upperclassmen handed me a box with a Raspberry Pi 4B... They told me it was my job to write some firmware for Ringo, and from that day forth began my long – and wonderful – journey with Raspberry Pi!”

What is your history with making?

I love making all sorts of things. For school and work I've done pretty 'serious' stuff such as circuit boards for satellites,

network systems for cars, and even chip design, but it's my personal projects that lean into the realm of whimsy and fun. I first went viral on Instagram for building the iconic Raspberry Pi Magic Mirror which tells you the time, weather, and gives you words of encouragement. For Valentine's Day I made a custom heart-shaped circuit board for my boyfriend; it featured blinking lights controlled by the best electronic component ever: the 555 timer!

▼ Sage is currently touring her workshop on creating cyberdecks





How did you become an online personality?

There were two gaps I wanted to fill. First, electrical engineering is hard and there are relatively few educational resources surrounding it. For instance, you have big educational companies dedicated to teaching people software (e.g. Codecademy), but there's no equivalent of this for hardware – in fact, many hardware videos online actually explain concepts incorrectly! I wanted to make educational content that would have helped me when I was learning. Second, there are very few women in electrical engineering or hardware in general – in my classes at university, in industry, and online. I felt that there was a need for more female voices in the space, so I became one of them.

How did you get into cyberdecks?

I had seen the r/Cyberdeck reddit a few years ago, but it was earlier this year that many cyberdeck videos began popping up on my feed. I was so excited to see many people building (and customising) their own decks! It brought a breath of fresh air to the conventional 'hardware look'. It was also bringing in new faces – people who had never touched a Raspberry Pi in their life were suddenly making these super-sick systems. I knew that the trend would be a great way to teach electronics to more folks, so I started making videos explaining the 'how to' of cyberdecks... at the end of the day, electronics is not a very flexible discipline – you either

I felt that there was a need for more female voices in the space, so I became one of them

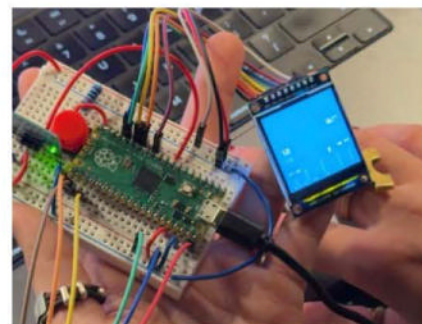
supply a steady 5V rail or suffer the consequences – and through my videos I hoped to bridge the gap between wanting to build something cool and actually being able to do it.

What are some of your favourite things you've made with Raspberry Pi?

It definitely has to be the satellite I helped build for Stanford's Student Space Initiative. And yes, this is a *real* satellite! We're launching later this year, which is super-exciting! The circuit boards on our satellite are designed around Raspberry Pi's RP2350B [chip], not excluding the one I worked on: the Attitude Determination and Control System (ADCS) board which controls the satellite's movements. I spent a lot of time diving deep into the RP2350B's capabilities and features, ultimately designing a system that integrated reaction wheels, sun sensors, and motor controllers. The RP2350B came at a time of desperation for our club – our previous chip turned out to have extremely bad radiation tolerance due to a variety of factors (manufacturing defects, the current phase of the solar cycle), and the RP2350B passed our radiation tests with flying colours. Huge shout-out to the Raspberry Pi team for supporting our club during the entire process! 🍷

Cyberdeck workshop

Sage is currently touring workshops on how to build cyberdecks, and tickets are available here: rpimag.co/cyberworkshop



▲ Pico is an invaluable tool for electrical engineers

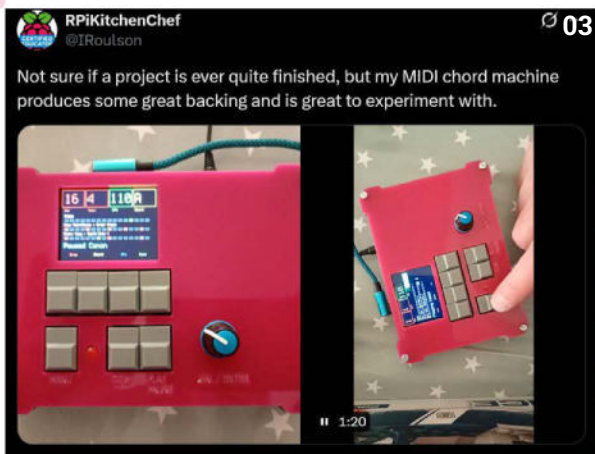
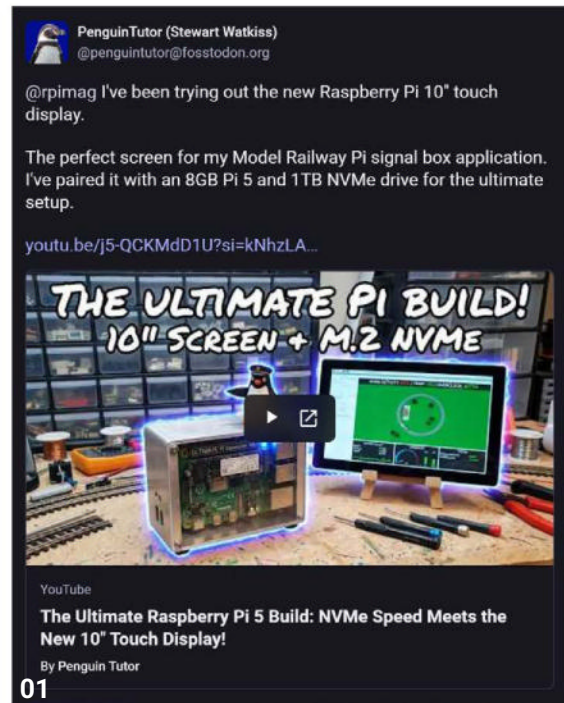
Maker Monday

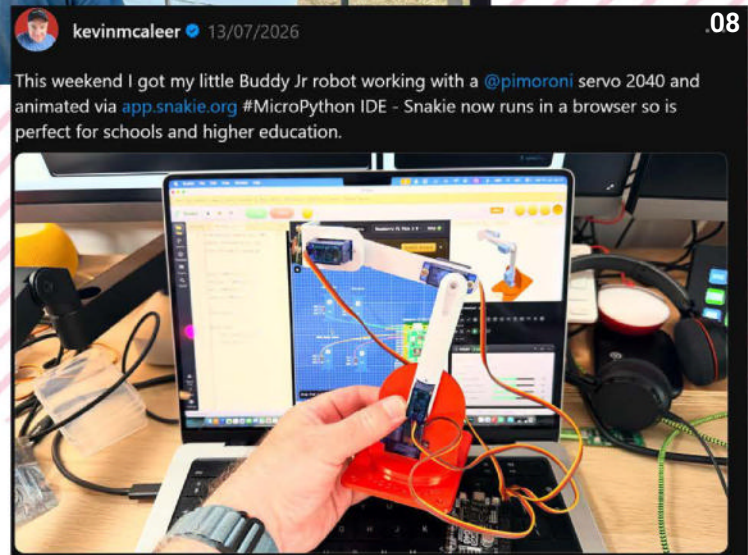
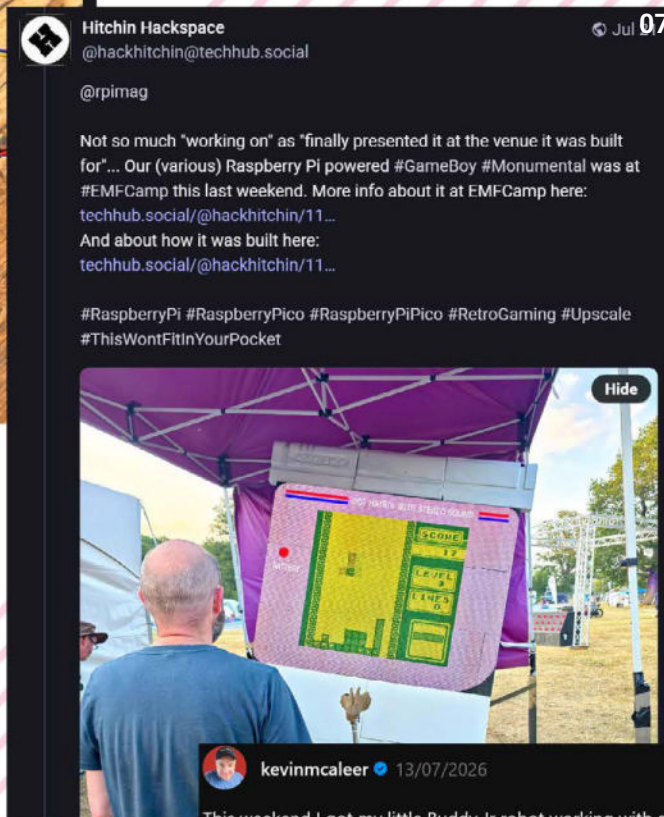
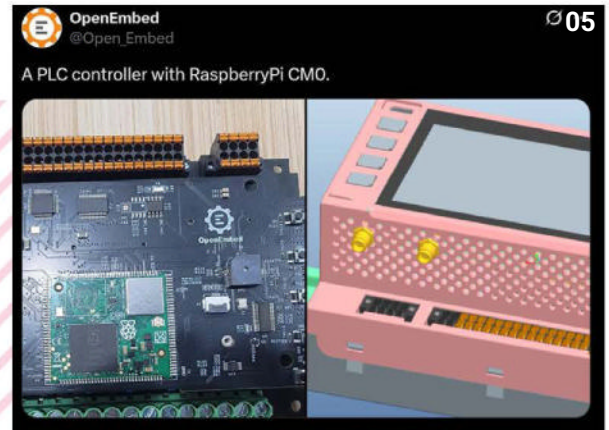
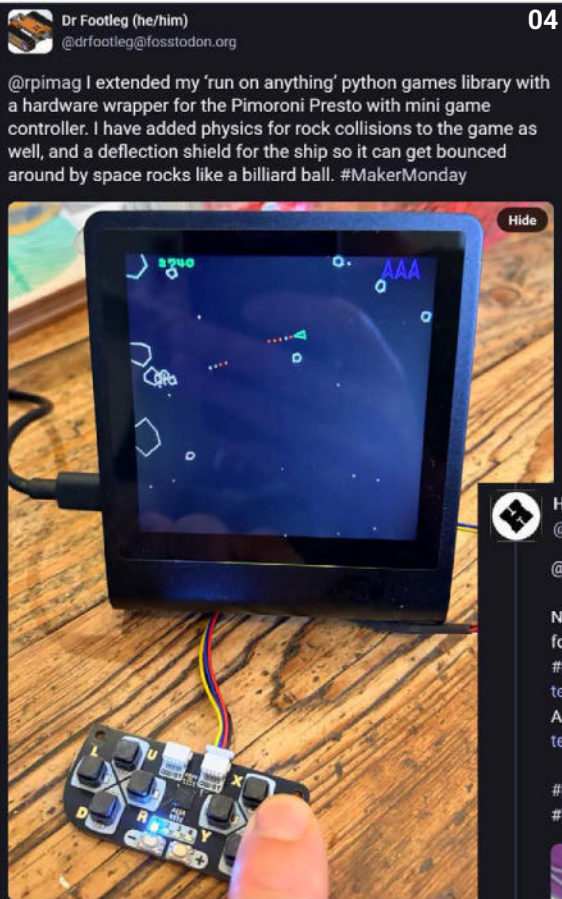
Amazing projects direct from social media!

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. Ten inches of model railway control – rpimag.co/169mm1
02. An AI assistant in a radio like this seems very Fallout – rpimag.co/169mm2
03. An important philosophical question for creators in any medium – rpimag.co/169mm3
04. Presto is taking the maker world by storm, it seems – rpimag.co/169mm4
05. Why buy a PLC controller when you can design one yourself? – rpimag.co/169mm5
06. So many LEDs for one little Land Rover – rpimag.co/169mm6
07. This might be the biggest working Game Boy we've seen so far – rpimag.co/169mm7
08. Say hello to snake, folks – rpimag.co/169mm8





Raspberry Pi at Open Sauce

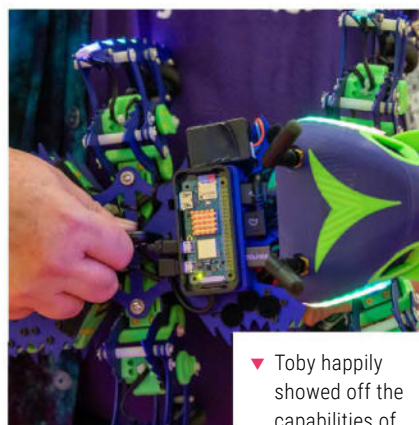
The 'mad science fair' in the Bay Area gets a visit from team Raspberry Pi

While it's only in its third year, Open Sauce has expanded significantly since its inception. Much like a maker faire, people set up stalls and show off their amazing builds for all to see. The team from Raspberry Pi were there too, checking out projects and showing off Raspberry Pi wares to boot.

▼ From left to right: Matt Richardson, Chris Boross, Samantha Snyder, and Toby Roberts made up the team at the event. Ashley Whittaker is also there, behind the camera



- ◀ We know all about flight radars from the last issue of the magazine (rpimag.co/168)
- ▼ There were plenty of robots to see at Open Sauce



▼ Toby happily showed off the capabilities of Raspberry Pi to other con-goers



Crowdfund this

Crowdfunding campaigns to keep an eye on

Isolated Data Acquisition Platform



This Raspberry Pi-based data logger "combines four differential analogue input channels, eight digital I/O lines, and integrated temperature and humidity sensing" according to the Kickstarter page, and is a standalone product and/or something you can add to a pre-existing Raspberry Pi that's used for monitoring.

► kck.st/4fvkU5L

IDO-EVB6208



A development board for Raspberry Pi Compute Module 4 and Compute Module 5 or a custom board based on Compute Module with a dedicated AI interface for a 1 TOPS NPU core. The dev board itself adds all the classic I/O ports along with Wi-Fi 6 and Bluetooth 5.4.

► kck.st/3S8iUsF

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



Your Letters



SLM

I was momentarily confused by the 3D printed metal watch body in last issue. It doesn't even look 3D printed! It's bizarre and amazing that you can go from 3D-printing a Benchy at home on a machine that cost a couple of hundred quid, to getting precisely made metal pieces created using the same design files.

David, via email

That's the magic of digital manufacturing. Once you've designed your parts, you can get things made at any scale, at a load of different sizes, in different materials, different colours, and different locations. You could have some parts made in one country, shipped to another country for assembly, then sent on to its final destination. That it looks so good is testament to just how good the SLM (selective laser melting) process is, and to the skill of the designer.



▲ It looks like machined metal, but the body of this watch was actually printed using a technology similar to resin printing

Pi-hole



▲ Add Raspberry Pi to your home networks and stop intrusive advertising

I wasn't expecting to see a Pi-hole tutorial in the latest issue – whatever next? BrewPi? Magic mirror? I must have first installed Pi-hole years ago. Don't tell me you've run out of tutorial ideas?

Robert, via email

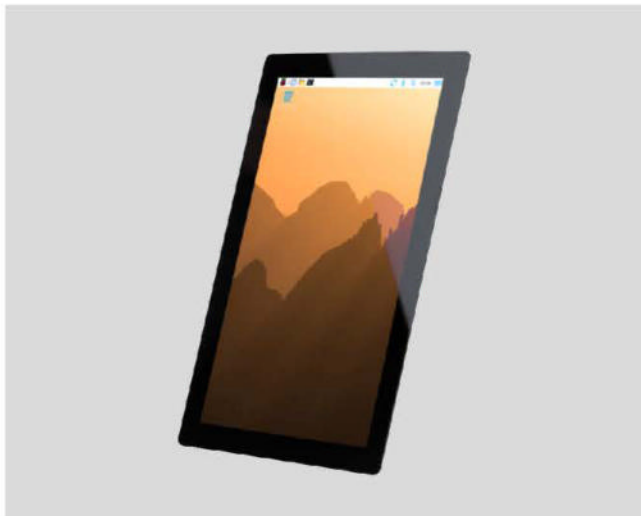
The thing with open-source software is that it's always being improved; Pi-hole may be old, but it's still around for a reason: it works brilliantly at keeping pointless ads off your network. And as pointless ads have got more common, Pi-hole has got even more useful. Likewise with magic mirrors, brewing beer, and all the old classic projects that people were doing with Raspberry Pi in the first couple of years since it first appeared: as the hardware available to us has got better, we can implement the original ideas in loads of different new ways. Keep playing – there's loads we can do!

10-inch display

A new bigger display is on the market, just in time for my wall-mounted calendar. Yes, I know it's going to be overkill, and an e-paper display would be perfectly adequate for something that only refreshes once a day, but it's an excuse to get a new toy.

Sarah, via email

The new 10-inch Touch Display 2 is a brilliant piece of kit – see page 8 for more details – but we wouldn't call it overkill. More, it's brimming with possibilities for your next project. We like to think of it as future-proofing: once you've got the calendar set up, you'll want to stream video over a local network, or set up a wildlife camera to keep an eye on what's outside while you're inside. We're having loads of fun with ours!



▲ Capacitive touchscreen, loads of pixels – what more do you want?

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. PLUG: Cannington Library Meetup

- 📅 **Tuesday 1 September**
- 📍 **Cannington Library, Perth, Australia**
- ▶ rpimag.co/plug169

The regular Cannington Library Catch Up. For people looking to learn about technology and who want to save their old laptops from going to the tip by installing open-source software. Bring your laptop or SBCs!



02. Breaking Breadboards Brighton September

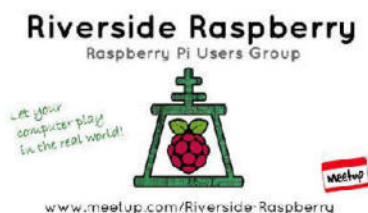
- 📅 **Tuesday 1 September**
- 📍 **Freedom Works Brighton, Brighton, UK**
- ▶ rpimag.co/bbb169

The Breaking Breadboards team are excited to announce their September event at Freedom Works in Brighton. This is the perfect opportunity for tech hardware enthusiasts in Brighton and beyond to connect, and share their passion for all things tinkering!

03. FTS Raspberry Pi Jam

- 📅 **Saturday 12 September**
- 📍 **Fruitful Hall, Ogun State, Nigeria**
- ▶ rpimag.co/ftsrpj169

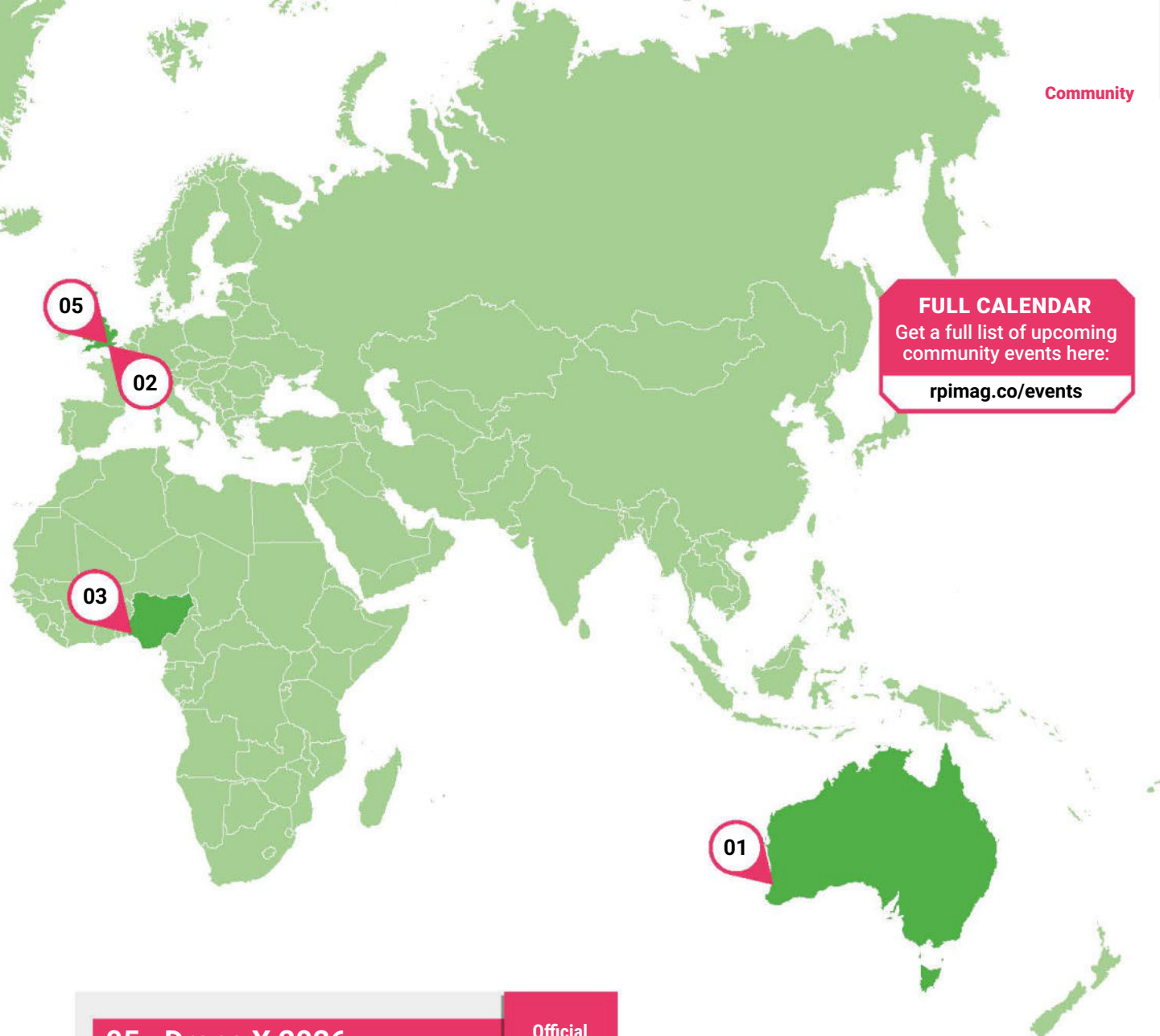
Join the community at Fruitful Hall for the Fruitful Raspberry Pi Jam in Ogun State. Discover Raspberry Pi 5's amazing features, this time around with artificial intelligence. Explore embedded systems with versatile devices that promote practical learning in coding, robotics, and AI. Plus, get a first look and touch the new Raspberry Pi 5 being connected to a Raspberry Pi Touch Display 2.



04. Riverside Raspberry Pi Meetup

- 📅 **Monday 14 September**
- 📍 **3600 Lime Street, Riverside, CA, USA**
- ▶ rpimag.co/rrpm169

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, they also cover topics about all kinds of maker technology.

**FULL CALENDAR**

Get a full list of upcoming community events here:

rpimag.co/events

05. Drone X 2026

Official
Raspberry Pi
Event



📅 **Tuesday 29 September to Wednesday 30 September**

📍 **Excel London, London, UK**

▶ rpimag.co/dronex26

The Raspberry Pi team is delighted to be returning to exhibit at Drone X in London, England, this September. There, you'll be able to meet the team and experience a range of Raspberry Pi technology.

You'll see how individuals and businesses in the UK and around the world use Raspberry Pi to support their projects and applications, and discover how Raspberry Pi can help you with your own solutions. You'll also be able to discuss how local partners can help with your projects, and learn about the technical, product, and compliance support services we offer to industrial customers.

Win 1 of 3

10-inch Raspberry Pi Touch Display 2

Last month, Raspberry Pi announced the brand new, much larger 10-inch Touch Display 2! It has a beautiful 1200 × 1980 IPS LCD screen and an 85-degree viewing angle. It's powered directly from Raspberry Pi too, giving it a very small footprint – and we have three of them to give away.



Head here to enter:

rpimag.co/win

Learn more:

rpimag.co/td210

Terms & Conditions

Competition opens on **26 August 2026** and closes on **24 September 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.

REACH A GLOBAL COMMUNITY

Advertise in **Raspberry Pi Official Magazine**

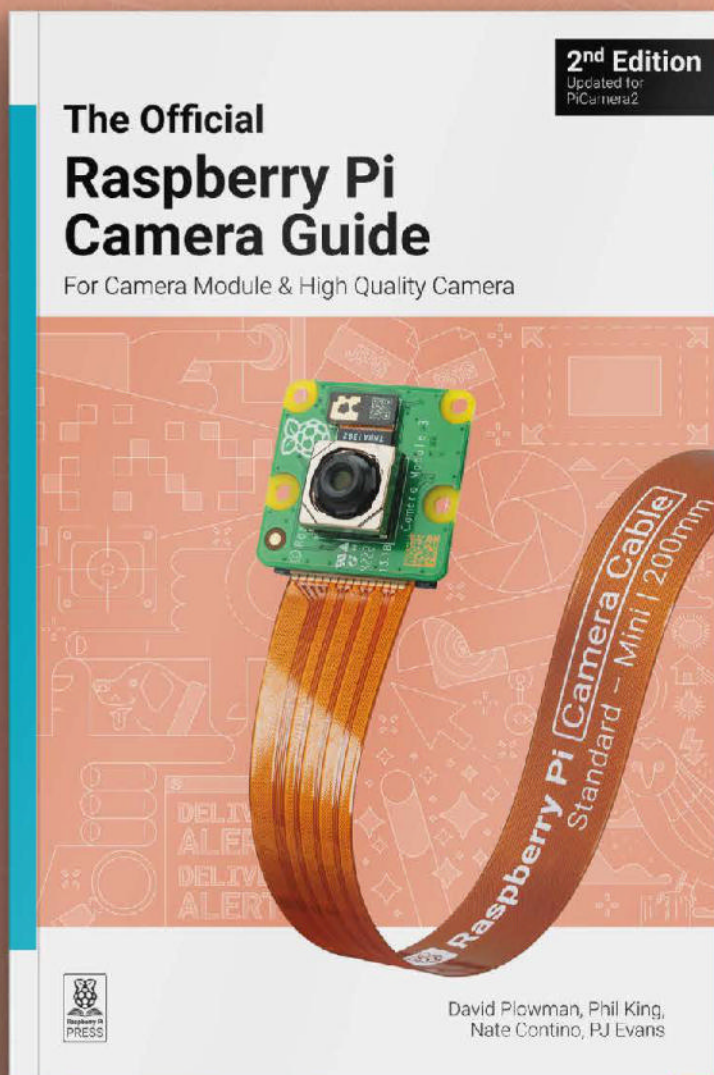
Our readers are passionate about technology and the Raspberry Pi ecosystem. From DIY enthusiasts to professional engineers.

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



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



Official Magazine
#170

Next Month

Digital Signage

Make a public display with Raspberry Pi

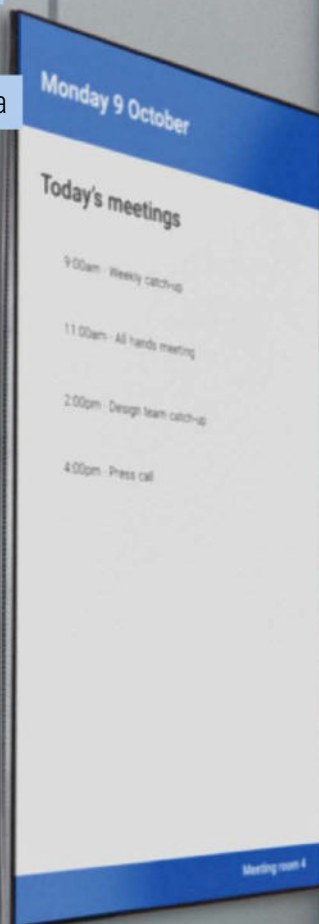
On sale 24 Sep

PLUS

Spooky Halloween Pico projects

Build a focus writing device

Discover a custard cream camera



Editorial

Editor

Lucy Hattersley
lucy@raspberrypi.com

Features Editor

Andrew Gregory
andrew.gregory@raspberrypi.com

Features Editor

Rob Zwetsloot
rob@raspberrypi.com

Sub Editor

Phil King

Advertising

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

Design

Head of Design

Jack Willis

Designers

Sara Parodi, Natalie Turner

Illustrator

Sam Alder

Brand Manager

Brian O Halloran

Contributors

David Crookes, Rosemary Hattersley,
Jo Hinchliffe, Nicola King, Phil King,
Brian Jepson, Rob Miles, Richard Smedley,
Ashley Whittaker

Publishing

Publishing Director

Brian Jepson
brian.jepson@raspberrypi.com

Director of Communications

Helen Lynn

CEO

Eben Upton

Distribution

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

Subscriptions

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



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

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)



New retro

Shifting times means shifting perspectives, but what about shifting definitions?

I've been playing video games for a long time. My earliest memory of this is very hazy, but it involves a maze game on BBC Micro that I played at my nursery school a couple of times – unfortunately, ‘maze game on BBC Micro’ doesn’t narrow anything down, so I don’t think I’ll ever identify it.

Anyway, I bring that up to say I’ve experienced a lot of changes in video games. From these 8-bit 1980s computers through 16-bit consoles, early 3D gaming, the birth of online games, and now the slow death of physical media.

*I know people
who still
haven’t come to
terms with the
N64 and PS1
being retro*

However, one thing has stayed consistent: we just don’t know how to define what retro games are.

Age is just a number

My take is simple: when I was playing on a GameCube in the early 2000s, the Super Nintendo was considered retro. Hot Topic was selling merch with an NES controller with the tagline ‘Classically Trained’ (I had a sweat band of it, which was the style at the time). Two console generations ago? 10–15 years old? Now that’s retro – which includes the Xbox 360, PS3, and Nintendo Wii right now. PS4 will soon be retro too...

Of course to some, that’s upsetting. It’s a reminder of the cruel passage of time. I know people who still haven’t come to terms with the N64 and PS1 being retro, and who insist no one was calling the NES retro until the late 2000s (which is, frankly, ahistorical. A lie. Rewriting history, even!), and that only pixellated games can be considered retro.

Answers on the silver screen

A sore point for me is folks trying to treat video games as film – something you see

with a lot of big cinematic games these days from a developer whose name rhymes with Pony. Video games are a different medium, and should be treated as such, not high-budget in-game cutscenes with the occasional button prompt. However, I’ve seen a recent push to classify retro games in a similar way as ‘classic movies’ are defined.

Classic movies tend to refer to movies from the Golden Age of Hollywood – from the mid-1920s to the 1960s. That’s a fixed time. There’s a similar definition in comics (Gold, Silver, Bronze, Modern, etc.), so perhaps we need to start making these definitions for games now too. Although what would we call the current age of games? I’m just hoping with all the upheaval in the modern industry that it’s not the downfall age. 🍷

Rob Zwetsloot – Author

Rob is features editor for *Raspberry Pi Official Magazine* and finally remembered to take a new profile picture that would not incur the wrath of brand manager Brian O Halloran.

rpimag.co

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:

