Get spooky with **Raspberry Pi Pico this Halloween**

Go under the sea to test **water quality and see fishes**

Create a **voice-controlled biscuit camera**

Raspberry Pi

Official Magazine
#170 | October 2026

DIGITAL DISPLAYS

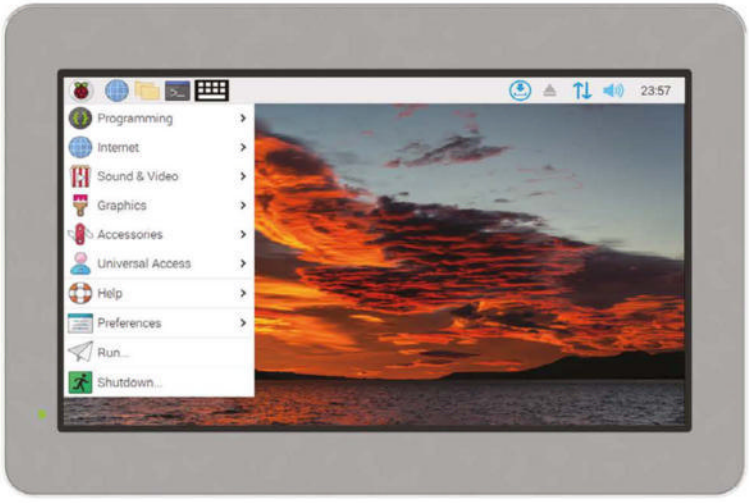
Building info boards with Raspberry Pi



£7.99
ISSN 2977-4403 (Print)

Industrial Raspberry Pi
ComfilePi





UL LISTED

Powered by Raspberry Pi

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

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

Visit www.comfiletech.com

© copyright COMFILE Technology, Inc. ALL RIGHTS RESERVED

COMFILE TECHNOLOGY

Welcome to Raspberry Pi Official Magazine



Editor
Lucy Hattersley
Lucy recently went to Paris to watch The Cure on a jumbotron from the back of a massive crowd. The show was amazing. The display was probably powered by Raspberry Pi as well.

rpimag.co



Many real-world displays are powered by Raspberry Pi. It's a low-power, internet-connected computer with a range of display output options that can be relied upon to provide information.

This month we decided to take a deep dive into the world of digital displays and how you can integrate this technology into your life. Create your own information boards, weather displays, and screen-connected devices (I particularly like Nik's Simple Writer project).

Raspberry Pi has a range of display products, from the Touch Display 2 to the new Smart Display Module, and third-party products like e-paper displays provide ultra low-power, long-term information.

So next time you spot a flight board, a bus stop display, or a shopping centre screen, remember: there's probably a Raspberry Pi behind it. And with this issue in hand, you'll know exactly how to build one yourself.

Lucy Hattersley – Editor

Sixfab

Complete Edge AI Hardware Stack for Raspberry Pi

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

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

START HERE



from $63

SIXFAB AI HAT+ FOR RASPBERRY PI 5

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

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

EXPAND



from $190

SIXFAB EDGE AI EXPANSION BOARD

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

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

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

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

DEPLOY



from $650

SIXFAB ALPON X5 AI

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

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

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

SIXFAB × ULTRALYTICS ACCELERATION PATH

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

Raspberry Pi Design Partner

Intelligented by DEEPX

ultralytics | Sixfab

Scan to browse and pre-order

Contact: hello@sixfab.com

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

Contents

World of Raspberry Pi

008 New Compute Module 5 Programming Jig released

010 Subscribe to Raspberry Pi Official Magazine

Project Showcase

012 Sensea

016 MirrorBallBot

020 The Transmission

024 5-DOF Robot Arm







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





Top Projects

026 Red Raven cyberdeck

028 Blossom

030 Accessible book reader

032 Bird recogniser

034 3D print showcase

Feature



036 **Digital Displays:**
Build your own info panel

Tutorials

050 Make a touch-enabled flight information display

056 Build a distraction-free Touch Display 2 writing machine

062 Run Ultralytics YOLO on Raspberry Pi with OpenVINO

068 Create a voice-controlled Custard Cream Camera

076 Upcycle a cheap colour printer

082 Add tech to a doll house – part 2

088 Simple electronics with GPIO Zero – part 11

Feature



094 **A Raspberry Pi Pico Halloween:**
Be scared by the tiny but mighty Pico





Reviews

102 **Only the best:** robots

108 Powered by Raspberry Pi

112 **Ten amazing:** computer vision projects

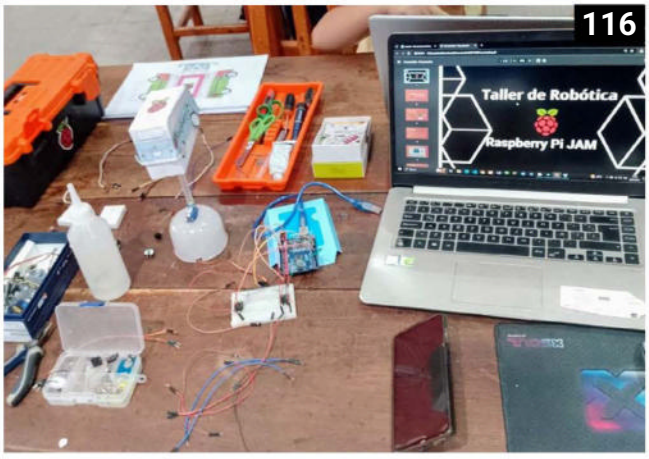
Raspberry Pi Community

116 Interview: Eliana Otrera

118 This Month in Raspberry Pi

122 Your Letters

124 Community Events Calendar



Competition

126 **Win** Raspberry Pi Development Kit for Compute Module 5

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.

Introducing the Raspberry Pi Compute Module 5 Programming Jig

Provision Raspberry Pi Compute Module 5 at scale with the Raspberry Pi Compute Module 5 Programming Jig, on sale now. By **Tom Dewey**



◀ The Raspberry Pi Compute Module 5 Programming Jig

Making a single device is hard; you have to prepare your hardware, your software, your documentation, and how you're going to sell it. Once you clear all of those barriers, you stare down a new realm of problems – how do you make the next thousand?

For developers designing products based on Raspberry Pi Compute Module 5, we've made that problem space smaller with the new Raspberry Pi Compute Module 5 Programming Jig: a single-head provisioning system that programs your Compute Module 5 with an operating system and security configuration (**rpimag.co/cm5jig**).

Simple, repeatable provisioning

The Programming Jig simplifies the provisioning workflow by taking the bulk of the configuration steps out of your

Compute Module 5 Programming Jig is priced at $600 and is available to order now



▶ Releasing the clamp to insert a Compute Module 5

hands; after setting up the jig's software and connecting the ports you require, all you need to do is insert the Compute Module 5 into the module bay and press the clamp shut. No further interventions are required – the status LED will tell you when the unit has been programmed.

On the back of the Programming Jig, you'll see a small set of ports, including Ethernet for the jig and the Compute Module 5 you want to program, and a USB programming port to manually update the jig software. There are also two activity LEDs – one for the jig and one for the module – that signal activity.

To go even faster, you can connect both the JIG ETH and DUT ETH ports to the same network.

Built around Raspberry Pi OS

To automate many aspects of production – including Secure Boot implementation, full disk encryption, and bare operating system installation — the Raspberry Pi Compute Module 5 Programming Jig needs capable software. So, we built it around our operating system (OS) construction tool, rpi-image-gen (**rpimag.co/rpi-image-gen**), and our automated provisioning software, rpi-sb-provisioner.

The jig's OS ties in to other Raspberry Pi software, notably Raspberry Pi Imager. The recently released v2.0.11 promotes the Compute Module writing functionality to public availability, and this can be used to customise the jig's OS to your requirements — including configuring Wi-Fi, security, and user settings.

Using the same OS components as our other platforms makes a wide range of customisation scripts and functions available for provisioning purposes. More than that, you also get the benefit of ongoing development and new features in Raspberry Pi OS, Raspberry Pi Connect, and rpi-sb-provisioner.

Available now

The Raspberry Pi Compute Module 5 Programming Jig is priced at $600 and is available to order now through our global network of Approved Resellers (**rpimag.co/cm5jig**).

You can find instructions for the Programming Jig on our documentation page (**rpimag.co/cm5jig-docs**).

Our rpi-image-gen and rpi-sb-provisioner tools are available on the Raspberry Pi software sources page (**rpimag.co/software-sources**).



▲ The rear panel houses the Wi-Fi/Bluetooth, power, USB boot, dual Ethernet, and status LEDs

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**

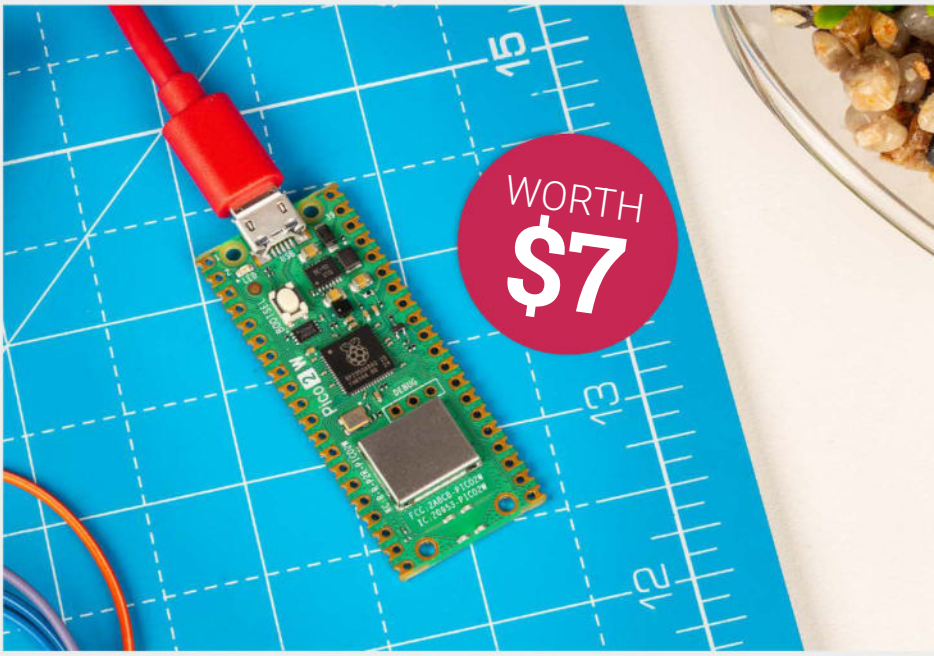
Subscribe for £10 is a UK-only offer. The subscription will renew at £15 every three months unless cancelled. A choice of free Raspberry Pi Pico 2 W or Pico 2 is included with all subscriptions.

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

Sensea

A trio of young makers created a fantastic Raspberry Pi 3-based underwater camera that highlights water quality and organisms. **Rosie Hattersley** is mighty impressed



Maker

Unboxed STEM Club

The club comprises three young makers and their adult mentors who enjoy coding, science, and engineering.

rpimag.co/sensea

Unboxed STEM Club consists of three friends aged 10 to 13 who meet up each weekend along with two enthusiastic dads. The boys' defining approach is to ask lots of questions, "look at problems as connected puzzles with many possible answers, and keep trying hands-on solutions even if things go wrong at first.

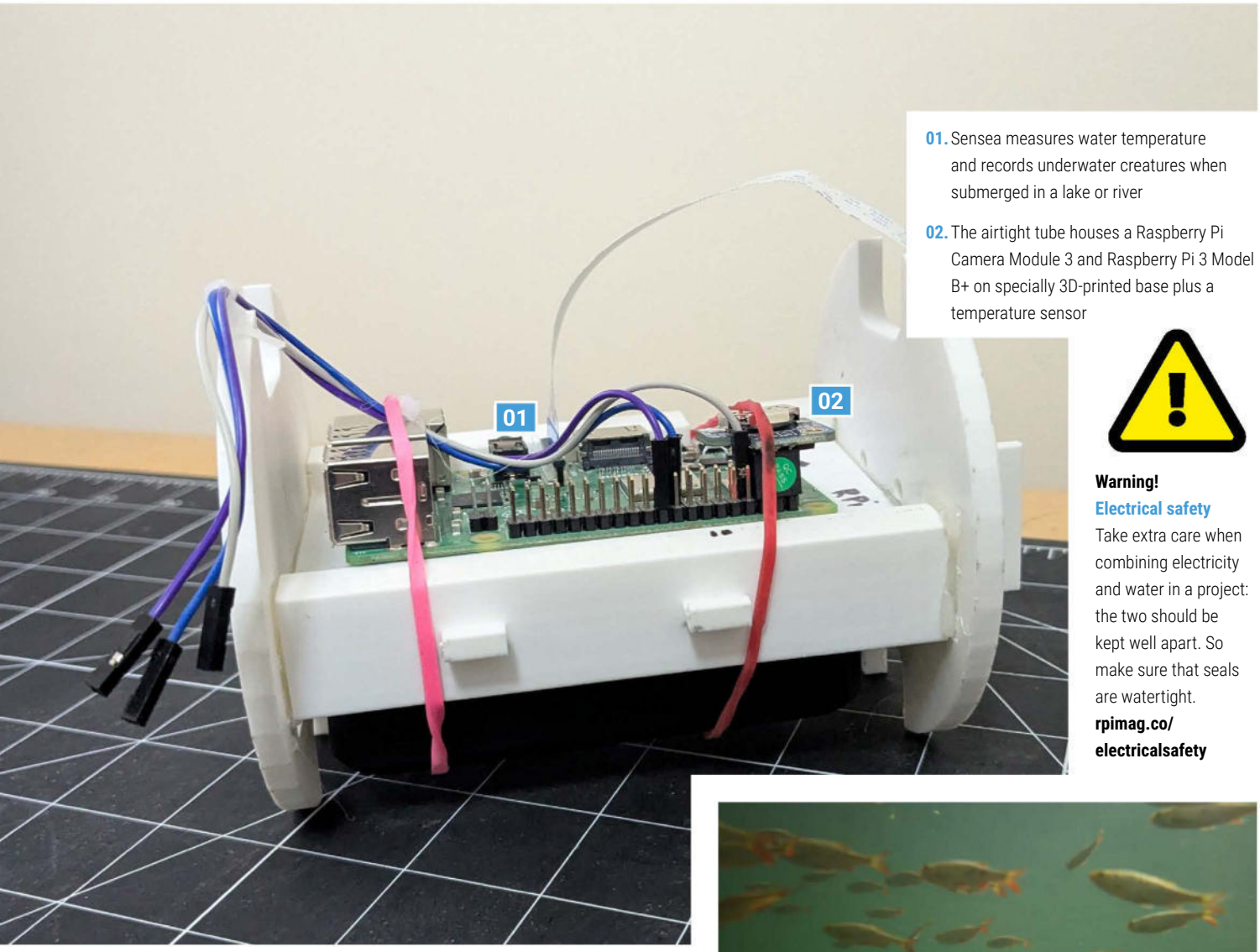
Pinak, Vihaan, and Yuvan and their mentors Kshitij Varma and Dr Amol Patil were excited by the idea of tackling a hands-on science, engineering, and coding project. They were inspired by the idea that we often talk about protecting oceans and rivers but rarely have the tools to measure what's happening below the surface and in the depths.

Raspberry Pi had huge appeal, being a real compact computer that they could code themselves and use with a wide variety of sensors, as well as being able to directly attach and control a Camera Module. Although they hadn't used Raspberry Pi for STEM Club projects before, they realised it was an obvious choice for their citizen science aims and came with an impressive online resource: thousands of willing and helpful fellow Raspberry Pi users. The fact that the whole build would come in at around £130 and could largely be built by the boys with some adult supervision set it apart from most underwater explorers, in terms of both complexity and cost.

What lies beneath?

Sensea was conceived as "a low-cost DIY underwater explorer that would reveal what's going on below the surface of our local ponds and rivers", and prove that you don't need expensive equipment to investigate it. For context, the boys' agitated pleas for a GoPro fell on deaf ears, so they built a custom camera rig instead, based around Raspberry Pi 3B+. This version has the advantage of low power consumption and can run for longer off a standard 5V USB power pack, as well as affordability. The team hoped the project might catch the imagination of other hobbyist makers, so these were important practical considerations.

The most important aspect of all was how to keep the electronics dry. The container needed to be completely



01. Sensea measures water temperature and records underwater creatures when submerged in a lake or river

02. The airtight tube houses a Raspberry Pi Camera Module 3 and Raspberry Pi 3 Model B+ on specially 3D-printed base plus a temperature sensor

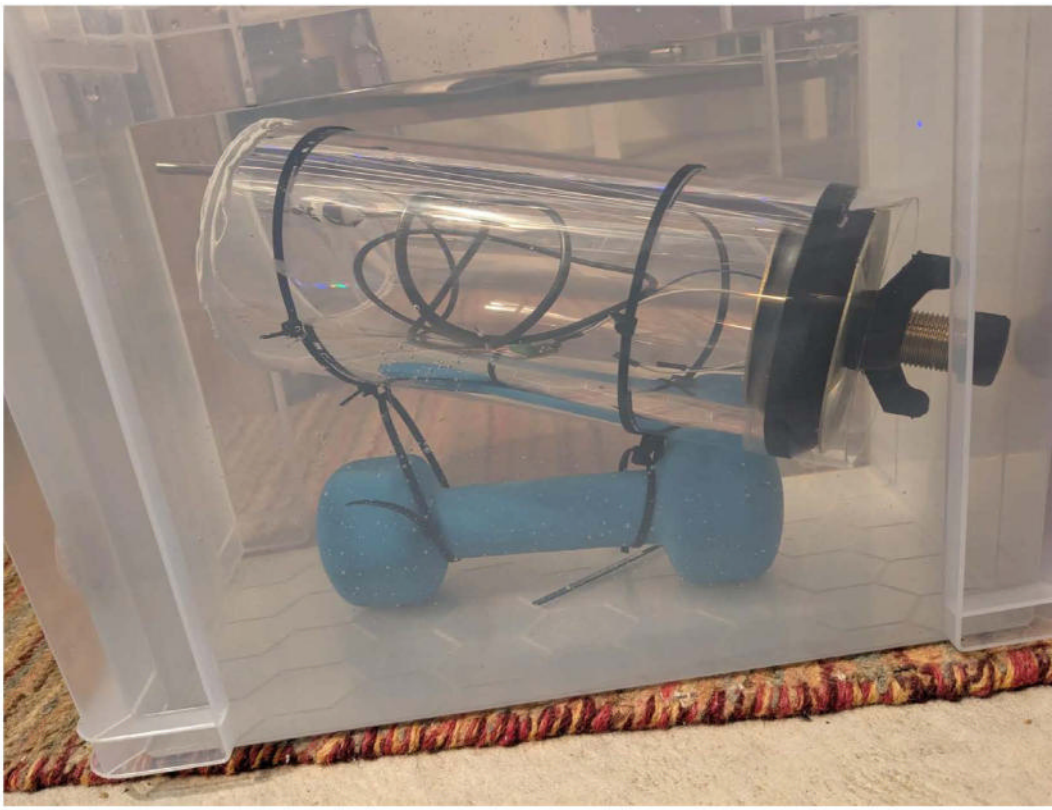
Warning!
Electrical safety
Take extra care when combining electricity and water in a project: the two should be kept well apart. So make sure that seals are watertight.
rpimag.co/electricalsafety



▶ Fish captured with Sensea

waterproof every time, yet still allow sensors to measure the surrounding water and enable the camera to see clearly through a transparent window. It also needed to be opened and resealed to insert delicate sensors and rechargeable batteries, making for quite an engineering challenge! They came up with a sealed acrylic tube that acts as a transparent pressure chamber that keeps the electronics dry, but allows the camera to see through its end cap. One end is permanently sealed with a 10.5 cm acrylic disc bonded using marine epoxy. A waterproof cable gland is drilled through this end to allow the DS18B20 temperature

A leak test regime sees the fully sealed Sensea tube and cable submerged in a bucket of water for five minutes and checked for rising bubbles



▶ Rather than using propellers, Sensea sinks to the river or lake bed using a 2kg dumbbell as ballast

probe cable to pass through while maintaining a watertight seal. At the other end is an expanding drain plug of the sort used by plumbers to check for leaks.

The setup proved reliable and, crucially, dependably waterproof. Nonetheless, Sensea's makers implemented a leak test regime that sees the fully sealed Sensea tube and cable submerged in a bucket of water for five minutes and checked for rising bubbles. Once this critical test has been done, the tube is lowered into the water in its protective cage.

Getting the buoyancy and descent right was another challenge. A 2kg bone-shaped dumbbell, selected after many experiments over size and weight, ensures the Sensea sinks beneath the river surface ready to start recording what its Raspberry Pi Camera Module 3 can see.

Exploring the murky depths

The Camera Module captures 10-second underwater video clips and logs time-stamped water temperature readings from a waterproof DS18B20 sensor. The team settled on a manual fixed focus, since autofocus was likely to struggle when encountering suspended particles in the river. "Fixing the focus gives much sharper and more stable footage," they explain.

The setup also includes a PCF8523 real-time clock that records accurate time-stamps even without an internet connection. Video capture is triggered via a Python script called **sensea_logger.py**. This also measures the water temperature and ensures everything is written to Raspberry Pi's microSD card for post-submersion retrieval.

For their initial deployments, the team compared temperature measurements from the River Cam and a nearby lake (Todd's Pit), clearly revealing differences between the two environments. Future additions may include a pH sensor for pollution monitoring, a depth or pressure sensor, and even AI-powered species identification. A similar setup could be used to monitor air quality, greenhouse conditions, or surface water from a floating buoy. Since we met the boys, the project has gone on to win Instructables' Sensors Contest. Congratulations on a richly deserved win.

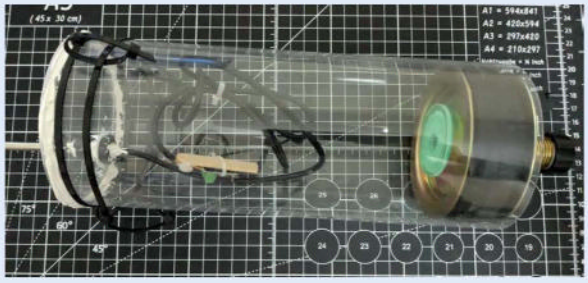
Quick FACTS

- The camera is set to 0.35 on the lens scale
- This fixed focus setting is ideal for subjects 30 cm–1m away
- Fixed focus results in sharper, less fuzzy photos in the murky waters
- They aim to add an LED for use in murkier waters
- A turbidity sensor to measure water clarity is another planned addition



▼ Team Unboxed STEM Club celebrate their successful debut at CamJam in Cambridge

Build your own Sensea



1. To create your own Sensea, you will need a clear plastic tube and a plumber's drain plug to create an airtight seal. This will allow Raspberry Pi Camera Module 3 to see and record its surroundings.



2. Raspberry Pi 3 Model B+ is powerful enough to record video, take photos, and save readings from the DS18B20 temperature sensor. Attach a USB battery pack to your Raspberry Pi so Sensea can operate in any location.



3. Glue one end of the tube shut and drill through the plug end so a cable with the temperature probe can fit through. Follow the code listed on the Instructable at **rpimag.co/sensea**. Always test your Sensea for waterproofness before deploying it!

MirrorBallBot

This trick uses no smoke but it does use mirrors. By **David Crookes**



Maker

Andrea Favero

Andrea is an engineer who began learning computer vision and programming. All of the files to reproduce his robot are stored on GitHub.

rpimag.co/mirrorballbot

Balance bots never cease to amaze. Their ability to monitor a ball and ensure it remains balanced on a platform demonstrates how robotics can combine sensors, programming, and precise motor control to great effect. One of the best examples we've seen has been created by Andrea Favero.

It keeps the machine compact by incorporating a mirror and uses computer vision to track the ball. There's a large touchscreen, a high-throughput camera, preset path trajectories, automatic setting of the ball colour, photoionization detector (PID) settings, sensorless homing of the steppers, and active cooling. It really is a feat of engineering.

But then Andrea is truly passionate about this project, which is the culmination of a lifelong fascination with the balance skills of street and circus performers. When he visited the 'Cité des sciences et de l'industrie' in Paris in 2019 and saw a large inverted pendulum demonstrator, he became motivated to build his own and this ignited his interest even further.

Bouncing ideas

Andrea began his MirrorBallBot project in the summer of 2024. Initially, he wanted to build a ball-balancing robot capable of bouncing a ping-pong ball. "I broke the project into distinct building blocks, forcing myself to focus on one at a time," he says. "These were camera tracking at high FPS, a stepper motor control, I2C communication, mechanical design, code integration, a graphical user interface (GUI), and finally, the bouncing code."

He spent six months on the project, but paused for a kitchen renovation. "The project lay dormant for almost a year," he reveals. "It was only in January 2026 that I restarted, tackling the mechanical design and the integration of the different blocks."

> *The solution came unexpectedly while looking in a bathroom mirror*

He dropped the bouncing ambition and concentrated on making a balance bot with a 25 cm platform monitored by a Raspberry Pi Camera Module 3 Wide – but realised this size would require a bigger robot. "I thought a larger 25 cm platform would make it easier to keep the ball from falling off," he explains. "But this required a tall robot to give the camera a large enough field of view."

He began to reflect on the problem. "The solution came unexpectedly while looking in a bathroom mirror," he says. "I noticed I could see my face in perfect focus, but the objects on the shelf below

01. The camera points at a mirror inclined by 27° to view the ball on the platform

02. A touchscreen is used to adjust the PID parameters and to control the ball



◀ Andrea has spent a lot of time finalising the fully 3D-printable design, simplifying the 3D printing and assembly

the mirror were out of focus. In that moment, I realised a mirror could increase the camera's focus distance without making the robot taller, solving the design problem elegantly."

Creating a robot

Pointing the camera at the mirror meant it could monitor a 60% wider field of view. This allowed Andrea to get on with creating a compact build, using a Raspberry Pi 4 as the central controller to handle computer vision, PID control, and the inverse kinematics calculations. Waveshare RP2040-Zero mini development boards were used for their small form factor and PIO features, used to generate precise quantities of motor steps at the required speed.

Andrea says he also used NEMA 17 stepper motors for their handling torque and excellent positional repeatability, as well as TMC2209 stepper drivers for silent operation, easy UART control, and StallGuard-based sensorless homing, eliminating the need for physical end-stop switches.

He created a custom PCB via PCBWay to connect three RP2040-Zero boards and three TMC2209 drivers together, eliminating a mess of wires and ensuring reliable connections. Elecrow provided him with a 7-inch DSI touchscreen for the GUI. This enabled another innovative feature: the ability to place a finger on the touchscreen and have the ball follow in real-time. By watching the display, the ball always lands beneath the user's finger.

Challenging times

There were many challenges. "My goal of high FPS initially failed because the latency was too high, making the robot unresponsive," he recalls. "Using parallel threading, where one thread captures while another analyses, was necessary but not sufficient. So I drastically reduced latency by setting the camera to manual mode, disabling white balance and HDR, reading directly from the camera buffer (limited to three images), and skipping frames for the display to save resources."

He also needed to find a way to use the RP2040 as an I2C slave because there was no standard MicroPython library for it. "I adapted a memory-register-based script from the Raspberry Pi forums (credit to danjperron) to create a custom communication protocol," Andrea adds. "The I2C bus runs reliably at 200kHz, and the connection is so robust that I eventually removed the acknowledgement returns from the Raspberry Pi 4's code."

To keep latency low, he split the RP2040's workload across its two cores. "Core 1 handles all I2C communication, validating packages and placing parsed instructions (direction, speed, steps) into shared memory. Core 0 processes this data and sends commands to the PIO. Access to the shared memory is protected by locked threading," Andrea explains.

The initial PIO implementation also used two separate state machines communicating via an interrupt. "The interrupt was too slow, causing them to get out of sync and leading to motor step-slipping," says Andrea. "After minutes of operation, the platform's position



▲ The user calibrates the ball (size and colour range) and the robot (auto-homing, levelling, and PID tuning) via the touchscreen GUI

Quick FACTS

- A camera captures the ball and provides a real-time image
- It processes at more than 110 frames per second
- Computer vision detects the ball's position on the platform
- A mirror increases the field of view of the platform by 60%
- Andrea has produced a maker manual with more than 150 pages

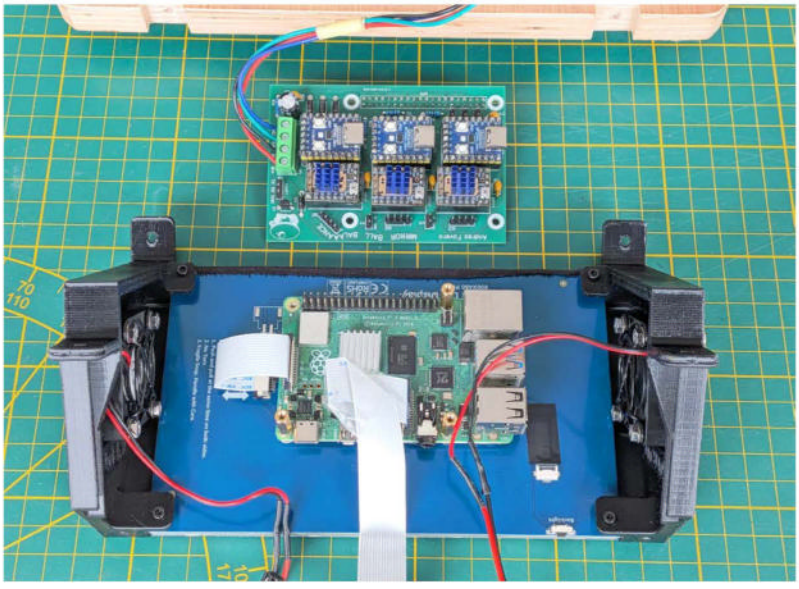


would drift unpredictably. The fix was to combine both state machines into one, sending a single 32-bit word that held two fields. The state machine would split this word internally, eliminating the need for the slow interrupt. The result is a zero-slip system, even after hours of continuous operation."

Such is Andrea's attention to detail, that he tackled calibration of the ball colour. "Different lighting conditions – sunny, cloudy, artificial – required different colour ranges for the same ball," he says. "I wrote a fully automatic calibration function to find the optimal colour range with the push of a (GUI) button. This function is the longest and most complex in the code, taking about 11 seconds to run, but it's far more consistent than manual tuning."

Andrea continues to fine-tune the project and he might eventually succeed in making the ball bounce. The project is certainly attracting a lot of attention. "It's a real-world example of multi-Raspberry Pi communication, high-speed camera vision, and distributed real-time control—all brought together in a DIY hobbyist project," he says.

▼ Raspberry Pi 4 is the robot's brain, sending commands via I2C to three RP2040-Zero boards controlling the stepper drivers

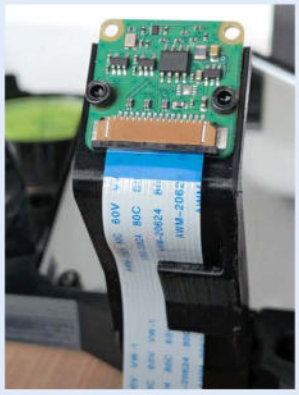


Robot balancing act

1. The robot operates in a continuous loop. And it has two main modes: a static mode, where it holds the ball at a fixed point and reduces motor current to cool down when idle; and a dynamic mode (PATHS), where the target position continuously changes, making the ball follow a trajectory.



2. The Raspberry Pi Camera Module 3 tracks the ball's coordinates. The Raspberry Pi 4 then calculates the platform inclination needed to roll the ball towards its target position, sending movement instructions (direction, speed, steps) to the three RP2040-Zero boards via a custom I2C protocol.





3. Each RP2040-Zero uses its PIO state machines to control a TMC2209 driver, moving the stepper motors precisely. The GUI provides graphical feedback and allows the user to switch between modes including Free Path, where the ball follows the user's finger on the touchscreen in real time.

The Transmission

It's a phone that allows you to call your future self. By **David Crookes**



Maker

Hillary Leung

Hillary is a creative technologist who is obsessed with dissecting, subverting, and reassembling the familiar.

rpimag.co/transmission

When best-selling author Jennae Cecelia posted her poem, *I Met My Younger Self for Coffee*, it sparked a cathartic TikTok trend. Soon, many people began posting videos of themselves having conversations with their prerecorded, past selves, exploring everything from relationships to careers.

For creative technologist Hillary Leung, however, the trend raised a more intriguing question: what if, instead of looking backwards, we could have a conversation with the person we are still becoming? "There's something incredibly eye-opening about speaking to your 'self' as if you're speaking with a different person," she says. The concept also resonated with her own winding path through life.

> *I wanted the object to both blend in and stand out*

Having studied law as an undergraduate, Hillary moved into UX design, car sales and, most recently, creative technology. That journey has sometimes left her questioning where she's headed next. Drawing on the idea of the hero's journey, she began imagining a physical project that could help people seek guidance.

"What if I had an oracle telephone where I could speak with my future self? The one I aspire to be? What would she say?," she asks. The idea soon evolved into a time-warped, futuristic vintage telephone. "Naturally, the 'how' followed; how do I want users to experience it? And how do I make it all work with existing technologies?," Hillary says. The result is an interactive, personal experience that invites people to reflect on their past, present, and future.

Dial an emotion

Hillary wanted users to be able to pick up the phone's receiver, dial an emotion, and answer a series of reflective, spoken questions. This would give people a moment to pause and consider how they're feeling, what's causing their uncertainty, and what they might want from the future.

Upon hanging up, the answers would be processed using a large language model (LLM) and the phone would ring, letting people listen to a voice sounding like their own giving friendly advice. This fundamental concept remained stable throughout the project, but the technical implementation evolved as Hillary



01. The device is made up of three antique telephones stacked on top of each other asymmetrically

02. A Raspberry Pi 4 is used to process the audio and to communicate with the external APIs

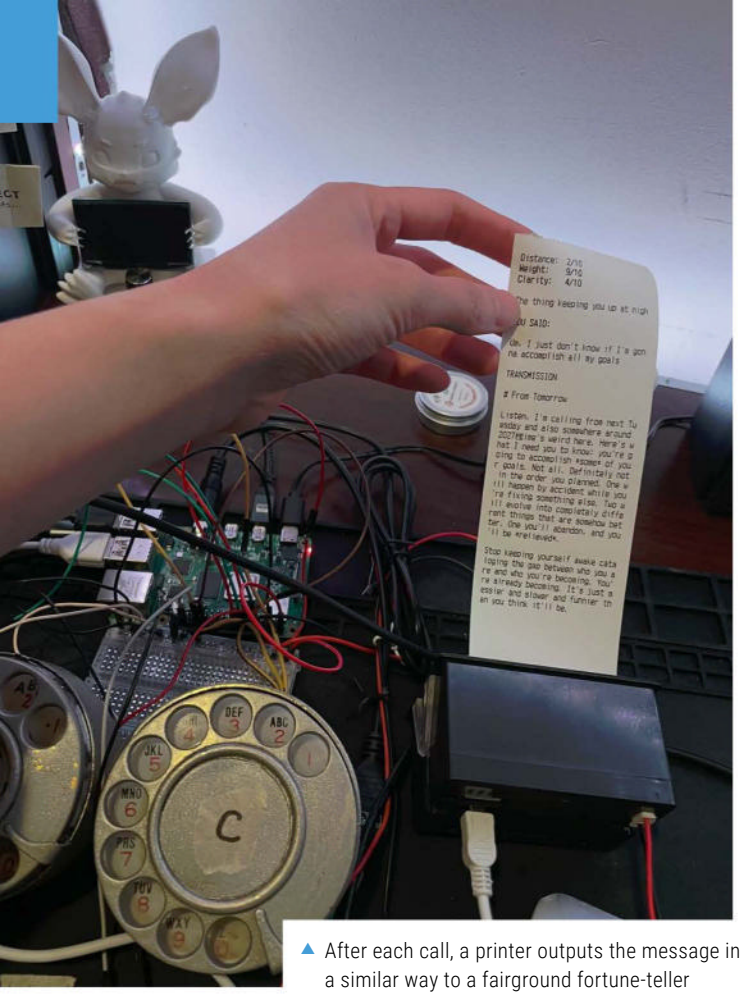
tested various components, libraries and workflows.

Hillary also wanted her project to be a work of art. "I wanted the object to both blend in and stand out – like it looks familiar, but there's something off about it," she says. To achieve this, she began sourcing antique rotary telephones on eBay. When the phones arrived, she took them apart and spent time figuring how they worked. Then she began identifying the extra components she'd need.

"I used a combination of Google, large language models, and trial and error to figure out what components I needed to create the experience," she reveals. "I then decided to use a Raspberry Pi 4 computer for this project because I needed something with enough RAM and processing power to handle the audio-processing pipeline and communicate with external application programming interfaces (APIs), including LLMs."

Making good calls

To make the project unique, Hillary decided to combine the telephones she'd bought. "I stacked them asymmetrically, and I rendered their antique design in futuristic silver, to make something recognisable feel strange," she says. But although it took her a long time to clean and spray the exterior parts



▲ After each call, a printer outputs the message in a similar way to a fairground fortune-teller



▲ All of the components are entirely hidden within the telephone housing, making for a neat project

Quick FACTS

- The phone turns self-reflection into a conversation
- Responses are processed via a chain of AI technologies
- The phone generates personalised guidance
- A cloned voice delivers a response
- It sounds like the user's 'future self' is speaking back to them

before putting them together, many more hours and days were spent putting everything together.

It was the first time Hillary had worked with Raspberry Pi and, for some time, she tried to do so without a keyboard and monitoring, accessing the computer from her MacBook remotely via SSH (Secure Shell) instead. This caused a fair few problems, not least because she couldn't see or tackle the kernel panic error that had been causing her a fair few headaches. Hillary was also learning Linux and using the terminal without any prior experience.

Despite all of that, slowly but surely, the project came together. Hillary connected the phone's speakers, microphones and rotary dials to a Raspberry Pi 4. She opted to use ElevenLabs' speech-to-text and text-to-speech models as well as Claude's LLM, both proving to be ideal for processing the user's messages and delivering a response.

"The telephone runs on a Python script, and the experience pipeline uses speech-to-text, an LLM, text-to-speech, and voice cloning to generate a relevant response from the 'future self' while making it sound like the user," Hillary explains.

Creating a keepsake

There were, however, some problems. The dials sometimes misread the pulses because the mechanical contacts produced extra electrical signals when switching, an issue that can be resolved by adding a debouncing mechanism. "A few of the phone's original parts also had to be replaced: the handset's microphone and the clapper solenoid inside the bell because of their high voltage requirements," Hillary says

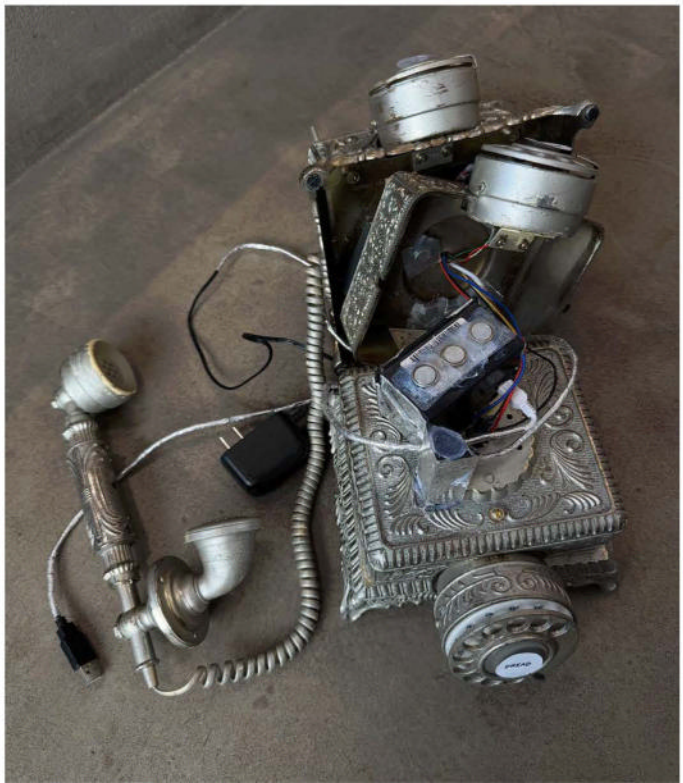
There were some programming setbacks too. "The library installations broke my system packages and corrupted the environment, forcing me to reflash again and rebuild everything inside a virtual environment instead," she adds. Yet Hillary persevered and she says most of the terminal commands had become ingrained in her memory by the end.

Since she was using more than one telephone, she was able to have three usable dials and this allowed her to attach

an emotion to each one and allow it to be answered on a scale of 1 to 10 by simply rotating the dial. Hillary also included a printer to allow users to see the spoken AI-generated message from their 'future self' transcribed onto receipt paper.

This proved to be a lovely finishing touch to the project, but Hillary says she's not quite done with it just yet. "I think it would be nice to add a 'replay' and 'reprint' button for the last 'future self' message. If possible, I'd also like to consider running everything locally to avoid issues with internet speed and API calls, while improving data privacy."

In the meantime, she's been showcasing the device at events such as Open Sauce in the San Francisco Bay Area, California, where it's been enthusiastically received. Hillary is also sure to want to discuss it with her future self.



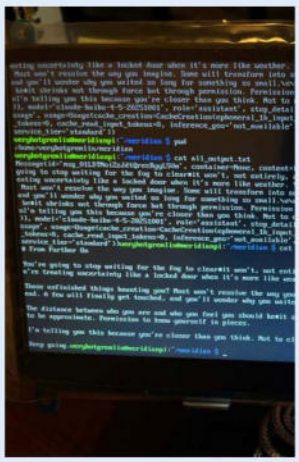
▼ When purchasing each of the telephones, Hillary pulled them apart for analysis before retro-fitting and spray-painting them

Dial-a-discussion

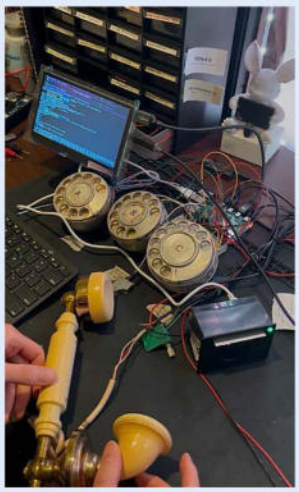
1. To use the Transmission, a user simply needs to pick up the phone, listen to the dialling tone, and turn the three dials. Each one of the dials represents a different emotion and the user has to rate how they are feeling on a scale of 1 to 10.



2. Once the emotions have been registered, three questions written by Hillary are delivered through the handset's speaker. The user answers each of the questions in turn by speaking into the microphone. Once done, the user is prompted to put down the receiver.



3. The phone will then ring and when the user picks up the receiver again, a voice resembling their own is used to address the messages that were left, followed by a printed receipt of their future self's response. The user data is deleted after each session.



5-DOF Robotic Arm

One young maker's amazing project. By **Rob Zwetsloot**



Maker
Malik
A 16-year-old electronics and science enthusiast from the French Alps who dreams about working for CERN – he's even had an internship there.

rpimag.co/5dof-arm

Who hasn't had a spark of inspiration mid-build for something much greater?

"Honestly, the whole thing started by accident," Malik, a teenage maker, tells us. "I wasn't setting out to build a robotic arm – I just wanted to build a gripper, with two big 13kg·cm servos and thermoplastic polyurethane (TPU) foam fingers for better grip. Once it was done, I thought: well, now I need the robot to go with it."

Malik didn't want to create a robot arm that felt like a toy, so decided to go all in with NEMA stepper motors and some industrial stepper controllers to create his own, custom, robot arm.

> *I love this project because of how much it's taught me, and how fast*

▼ The lab is fully kitted out to test every aspect of the arm



Iterative design

"[Once I got the stepper motors] I started designing my first cycloidal gearbox prototypes," Malik explains. "Once I had a proper working gearbox, I built the robot around it. The problem was, I dived in head-first like a beginner – no calculations, no simulation – and I hit a wall almost immediately: the motors couldn't lift the arm. So I rethought the whole thing, the architecture, the look, the motors. I moved to a big NEMA 23 for the shoulder, a smaller one for the elbow, and NEMA 17s for the other three axes, and designed new gearboxes for the 23s. Those were a real challenge — they have to output several tens of newton-metres of torque – so this time there were a lot of tests, calculations, and simulations."

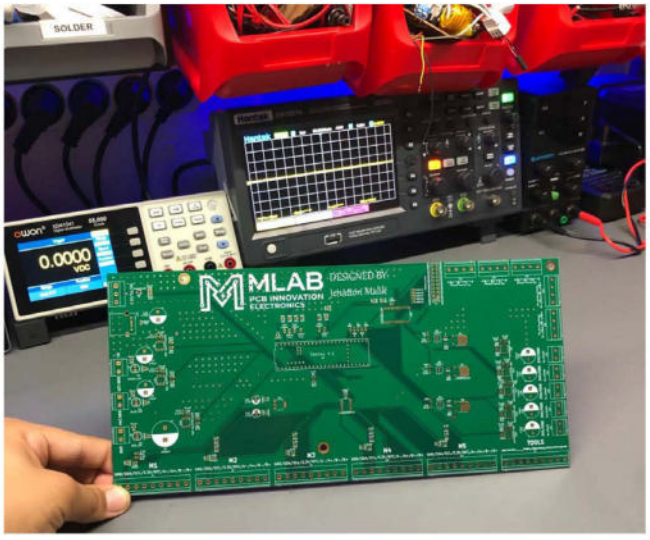
Malik reveals that he has two cardboard boxes filled with parts after a long build process (he reckons it's been about a year of building), thanks to a continuous iteration process with events like the aforementioned one.

"I love this project because of how much it's taught me, and how fast," Malik says. "Mechanics, with all the physics and maths that come with it. 3D printing, learning to work with technical filaments. And of course electronics – I designed a six-layer control board built around an Arm Cortex-M7, with a Raspberry Pi 5 running the software and the heavy computation. For AI, I'm planning to add a Raspberry Pi AI HAT, designed for computer vision. Alongside it I kept



01. The robot arm is a very serious bit of engineering

02. You can tell by the big red stop button



▶ The new controller board for the robot arm, ready to install

building smaller projects too: a discrete transistor-level 555, analogue and power circuits, complex digital stuff."

Still in development

While the arm is very impressive, there's still a decent amount of work to do.

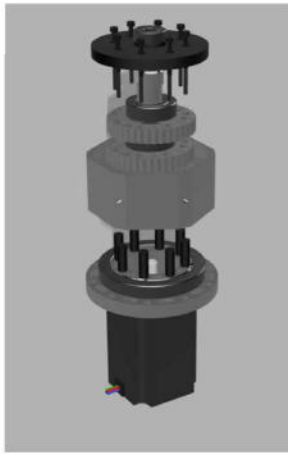
"Mechanically, it works – the arm moves and the gearboxes drive it," Malik tells us. "I've just received the main control board, and I still need to buy the last few components to populate it. For now the arm runs on prototype electronics, which only let me drive one axis at a time until the board is fully assembled."

A lot of 3D-printed parts are soon to be upgraded with machined metal parts too, it seems. It's a great illustration of how one thing can lead to another, and in the end you have an industrial-level robot arm in your room at only 16.

Quick FACTS

- The arm has a reach of 76 cm and five degrees of freedom
- The metal parts are being machined out of aluminium
- A Teensy 4.1 microcontroller aids with motor control
- The arm requires a beefy power supply – 24V/25A
- Malik has stress-tested parts to destruction

▼ Planning out the gearbox for a NEMA 23 stepper motor



Red Raven cyberdeck

By **Independent-Music320**

rpimag.co/red-raven

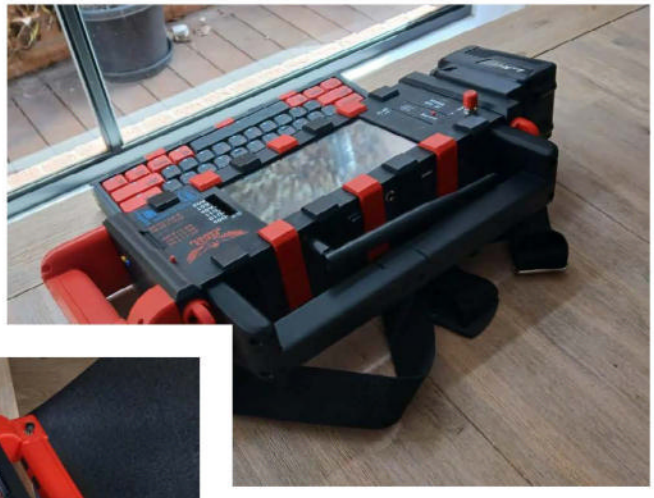


We really love a good cyberdeck build, and this one from Reddit user Independent-Music320 caught our eye for a few reasons. First is the retractable mechanical keyboard; many cyberdeck builds go down the route of using a Bluetooth keyboard – a sensible option as it cuts down on wiring, which can get in the way of a hinged case. Then there's the battery: again, there's a de facto standard with power for beginner cyberdecks, and that's to use a USB power pack. This build takes a different approach, using a 20V battery from the maker's electric drill. The choice of battery, keyboard, and colour scheme combine to give a sense that this is a tool rather than a toy. And while having the battery on one edge apparently makes the build a little heavier on that side, it looks so good that it just doesn't matter.

Rather than the usual Raspberry Pi at the heart of this cyberdeck, this one uses a LattePanda V1 which had been gathering dust. For those not familiar with this platform, LattePanda is a single-board computer that uses the Intel Atom processor, meaning that it can run the Windows operating system. Which is great if you like the sort of thing, but in a cyberpunk future we'd much rather be using some sort of Linux. Which brings us nicely to the other hacker-friendly features: there's an Arduino Leonardo and a Raspberry Pi Pico built in, with the GPIO pins of each device nicely broken out on the top of the cyberdeck for hardware tinkering on the move.







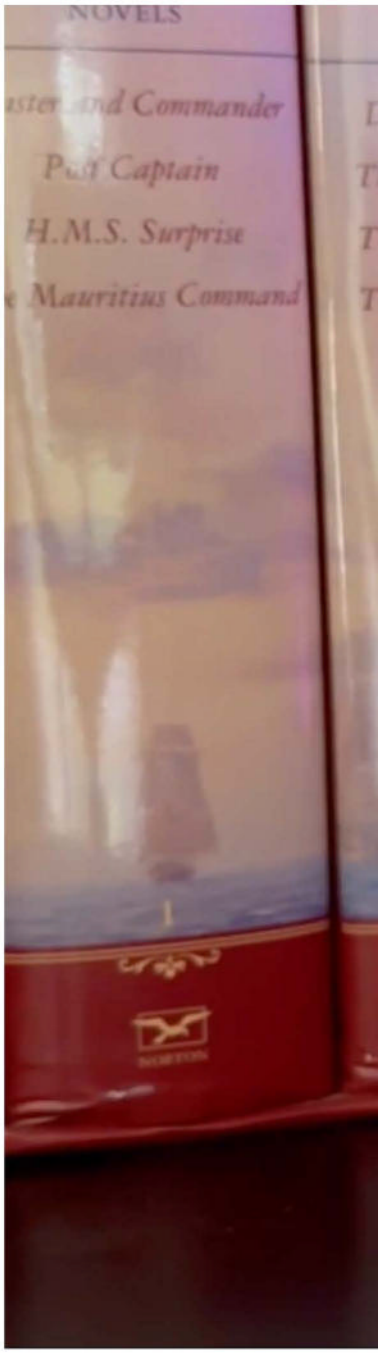


◀ Red and black are officially the two coolest colours

Blossom

By **Dave Karsak**

rpimag.co/blossom



Physical computing is, in this writer's opinion, the best way to learn computing full stop. Making text appear on a screen is just nowhere near as exciting as making an LED flash, or controlling a sequence of LEDs, or making a stepper motor move when the user presses a button. You're learning the fundamentals of programming, but without spending all your time staring at a screen.

Blossom, an interactive art piece that's controlled over your local network from within your web browser, is a great piece of physical computing. It's built on a Raspberry Pi Pico 2 W and a 16-light NeoPixel ring, and acts as its own web server, so anyone can control it from **blossom.local** on a local network – or if you're feeling ambitious, you can program any device to control it.



▲ Blossom is open-source hardware – everything is documented

Accessible book reader

By **lhm0**

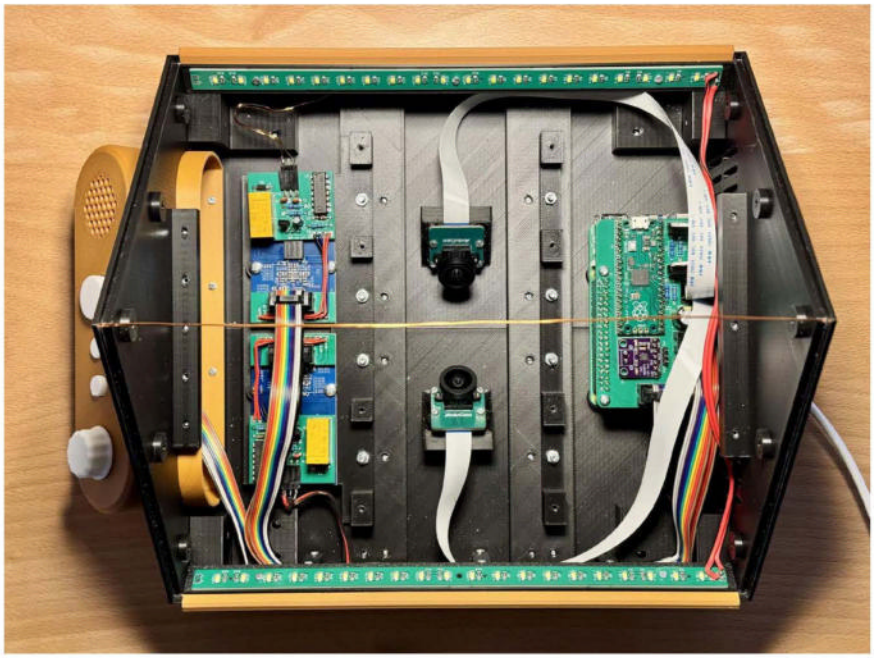
rpimag.co/book-reader

We take in so much information visually that it can be hard to imagine how difficult it would be if we were to lose our eyesight. Braille books are expensive, and screen readers are fine for websites, but it's a bit harsh to expect someone to have to give up on their book collection just because they can't see as well as they used to.

Enter maker lhm0 with their accessible book reader. This is a complicated build featuring a lot of electronics and custom PCBs – we're including it here because it's extremely impressive. It uses, among other components, Raspberry Pi 5 and Raspberry Pi Pico, two camera modules, NFC tag readers, and a system for outputting audio.

The book sits open, spine up, with the pages to be read lying flat along two glass sides of the reader; this places them in a position where the text will be in focus for the cameras. Optical character recognition software on Raspberry Pi 5 'reads' the words, then a text-to-speech engine running in the cloud sends an audio signal to the audio amplifier board and the words are read out on the built-in speaker.

▶ Read your book collection out loud with an advanced build using Raspberry Pi 5





Bird recogniser

By **Arne Giacomo Munthe-Kaas**

rpimag.co/bird-detector







We recently brought you a bird detector, which used Raspberry Pi to grab local sightings of birds over the web, and illustrated the findings on an e-paper screen. This build goes one further, in that it uses a microphone to identify birdsong, and gives the user an e-paper illustration of the bird life that's going on outside.

This bird identifier uses Raspberry Pi 5, an Inky Impression 13.3-inch e-paper display, and a USB microphone. The frame will draw whichever birds it has identified recently, using the brilliant BirdNET-Go service. Everything runs locally on Raspberry Pi, with no need for internet access.

BirdNET-Go works worldwide (we've seen it before in use identifying birds in Maine, USA), but it's best stocked with images of European birds (for now). Arne has identified 37 birds so far, out of a database of around 2,600.



▲ Arne uses a USB lavalier mic taped under the roof outside with the cable run in through the window

3D print

A camera with buttons that go click. What a treat!

rpimag.co/digitalcamera

Much like computer graphics in 1990s action films, touchscreens aren't inherently awful, but their overuse is terrible. What we need is a return to chunky, clicky buttons that have one function and one function only. Like the Unix computing philosophy, where each application does one thing, and one thing well, our brains instinctively respond to a device that does one thing.

This Tiny Film Camera by Suveen Ellawela is a great example of a device that's better for having fewer functions. It's a digital camera built on Raspberry Pi Zero 2 W and Camera Module 3. There's a button to take videos, a button to take photos, and there's a slider switch to enable the user to choose between three looks (warm, cold, and black and white). There's even a buzzer to let you know when the camera has booted up, and when you've taken a photo. And that's it. Suveen started out using Claude Code to, as he puts it, 'vibe CAD' the design, but ended up using online CAD tool Onshape to design the 3D printed parts.




SUV

DIGITAL DISPLAYS

Building your info screen with Raspberry Pi. By Lucy Hattersley





▶ Assemble your digital display project with Raspberry Pi

If you've seen a modern digital display in the wild, the chances are good that there's a Raspberry Pi behind the screen.

Raspberry Pi offers affordable compute power, great high-quality visual performance to run multiple 4K displays, HDMI/DSI flexibility, touch display input support for interactive displays, and GPIO output to handle triggered events.

From a two-inch touchscreen to a giant bank of 4K video walls – our one board can handle it all.

In this feature, we're going to look at how Raspberry Pi has quietly taken over the world of digital displays, and how you can use screens big and small for your own presentations. Whether you want to build a train time information screen, advertising board, event information screen, or just show off some photos at a show-and-tell event, we've got your display option here.

Raspberry Pi has quietly taken over the world of digital displays

Which Raspberry Pi

All Raspberry Pi computers are capable of outputting video in some capacity, and picking a model is largely a question of determining which display(s) you want to use and what the end purpose will be.

Raspberry Pi 5 and Raspberry Pi 4 can both run a 4K display at 60 frames-per-second (FPS). However, Raspberry Pi 5 can run two displays at 60 FPS, and it also features two Display Serial Interface (DSI) connectors. Raspberry Pi 4 can run two 4K displays but at 30 FPS and has only a single DSI port.

Raspberry Pi Compute Module 5 has matching functionality to its computer counterpart, but lacks physical ports. Instead of the usual HDMI, USB, and Ethernet ports, Compute Module features two 100-pin connectors on the bottom of the board: these connect to dedicated hardware that breaks out the sockets required. A Raspberry Pi Compute Module IO Board is used during development, which breaks out every port on Compute Module to match its Raspberry Pi computer counterpart.

To this end, Raspberry Pi has recently designed a Smart Display Module that utilises Compute Module hardware. It features a range of input/output (I/O) options and, crucially, slots inside matching displays produced by Sharp and other manufacturers that have products with an Intel® SDM specification slot. This provides a professional, and/or industrial, solution for display technology with compute power that can be accessed remotely, and quickly removed and upgraded.

Older, and more affordable, Raspberry Pi technology is also powerful in this area. Earlier Raspberry Pi computers such as Raspberry Pi 3, or Raspberry Pi Zero and Zero 2, are capable of running displays with 1920 × 1080 resolution.

Both Raspberry Pi 4 and 5 can run a 5-inch and 7-inch Touch Display 2, but you will need a Raspberry Pi 5 to run the newer Touch Display 2 – 10-inch that we feature in several tutorials.

Still, if you don't have any Raspberry Pi 5 requirements like dual 4K display or Touch Display 2 – 10-inch, choosing an older Raspberry Pi can be an affordable solution: especially with RAM demand still constraining supply.

Raspberry Pi digital display compatibility

Model	Max output	HDMI type	DSI	Displays
Raspberry Pi 5 / CM5	4K60 × 2	micro HDMI	2 × MIPI CSI/DSI	HDMI, Touch Display 2, Touch Display; Smart Display Module (CM5 only)
Raspberry Pi 4B / CM4	4K60 × 1; 4K30 × 2	micro HDMI	1 × MIPI DSI	HDMI, Touch Display 2 (5-inch, 7-inch); Touch Display
Raspberry Pi Zero 2 W	1080p60	mini HDMI	N/A	HDMI
Raspberry Pi Zero / Zero W	1080p	mini HDMI	N/A	HDMI
Raspberry Pi 3 and earlier	1080p	HDMI	1 × MIPI DSI	HDMI, Touch Display 2 (5-inch, 7-inch); Touch Display

Raspberry Pi Smart Display module is a new product designed around Compute Module 5

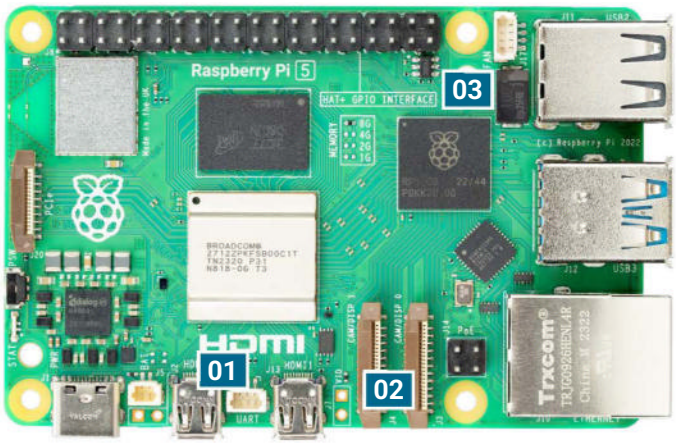
How much RAM?

If you are planning to use Raspberry Pi with the Raspberry Pi OS in graphical user interface (GUI) mode, we recommend 2GB RAM. However, many display projects run a Raspberry Pi in headless mode to minimise system overhead.

If you are just displaying video or image information in a kiosk format, then a 1GB model in headless mode will work fine. But if you wish to have more complicated compute interaction, such as live data display or touch screen input, then 2GB is more appropriate.

Raspberry Pi

Raspberry Pi computers add compute power to standard HDMI displays and feature direct Touch Display integration.



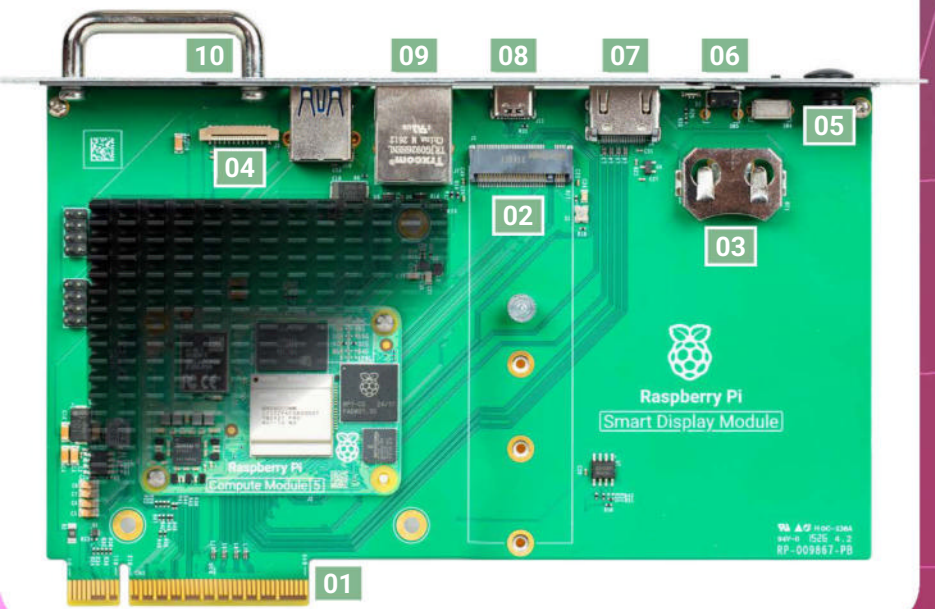
01. Raspberry Pi 5 has two micro HDMI ports, each capable of outputting 4K video at 60fps. Raspberry Pi 4 also has two ports and can output 4K at 30fps

02. The DSI ports are used to drive display devices such as Raspberry Pi Touch Display. Raspberry Pi 5 shares its DSI ports with Camera Serial Interface (CSI) ports and can support two displays

03. Raspberry Pi 5 also features four USB Type-A ports, Ethernet, and 40 GPIO pins to support a range of input/output actions

Smart Display Module

Raspberry Pi Smart Display Module is a new product designed around Compute Module 5. It is designed to fit directly into compatible displays.



01. The Intel SDM connector attaches the Smart Display Module to a compatible display. It provides data and power transfer, enabling the Raspberry Pi SDM to operate without external power

02. The PCIe 2.0 x1 interface enables fast peripherals such as NVMe drives and AI accelerators to be connected to the screen

03. A built-in battery connector enables Raspberry Pi SDM to run a real-time clock (RTC) and retain date and time information when the system is powered down

04. A CSI/DSI port provides Raspberry Pi Camera Module support, or you can add a second Touch Display

05. An optional antenna module enables Raspberry Pi Smart Display Module to connect to wireless networks

06. A rocker switch enables you to boot between Normal and USB modes, enabling easy debug via a USB connection

07. HDMI and USB Type-A ports are provided for remote docking

08. A USB-C port is made available for servicing

09. An Ethernet port provides reliable internet access

10. A milled handle enables Raspberry Pi Smart Display Module to be easily removed

Pick your display

Not all digital signage projects have equal demands. Some require huge screens, other require personalised displays with touchscreen interaction. Here are your main options

Smart Display Module & SDM display

Cost: $30 (not including CM5 from $67.50; compatible display from $1500+)

rpimag.co/sdm

If you want to deploy a range of large-scale, industrial displays, then Raspberry Pi SDM and compatible Intel SDM displays is the system to investigate. **Use cases:** Flight information systems, retail and corporate signage, and industrial displays.





4K HDMI display

Cost: From £100

rpimag.co/dell27

Most 4K monitors come with desktop stands or VESA mounts. Attach a Raspberry Pi to the HDMI port and you can turn it into a digital display and deploy it into any environment. **Use cases:** Retail and small business signage, video display, photo display.



Micro screen

Cost: $9

rpimag.co/waveshare-oled-pico

Sometimes a tiny display is all you need. If all you want is to display a small piece of information, then Waveshare's 1.3" OLED Display Module for Pico is just the thing. **Use cases:** Compact status/data readouts on Pico-based projects; sensor values, clocks, tiny user interfaces.

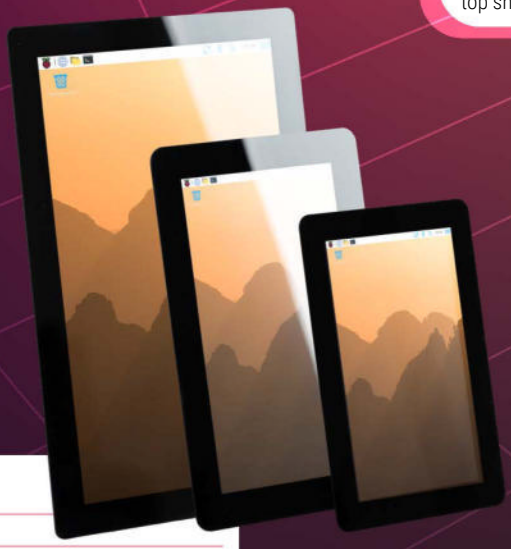


Raspberry Pi Monitor

Cost: $100

rpimag.co/monitor

The standard monitor is perfect for quickly attaching a Raspberry Pi and placing on a desktop. It features a VESA mount for wall display. **Use cases:** Desktop use or integration into projects and systems, table-top show-and-tell video and image presentations.



Touch Display 2

Cost: From $40

rpimag.co/touchdisplay2

Available in 5-inch, 7-inch, and a new 10-inch version, the Touch Display 2 uses the Display Serial Interface (DSI) port on Raspberry Pi to connect a screen directly. Raspberry Pi Touch Display 2 is ideal for interactive projects. **Use cases:** Tablets, entertainment systems, home automation dashboards, robotics interfaces, gaming systems, and information dashboards.

E-paper display

From: £50

rpimag.co/inky

An e-paper display might not have the brightness or frame rate of a digital display, but it looks stylish and has dramatically lower power requirements – the screen keeps showing its last image when powered off. And it's easy to read, even in sunlight. These displays are often deployed in office environments. **Use cases:** Information boards, retail price labels, name tags, calendars.



Cases and mounts

Many screens are sold without a case, or with a simple stand. You'll need to surround and mount your display and position your Raspberry Pi. Here are some options



◀ The rear of a Raspberry Pi Monitor has four screw-holes that conform to the VESA standard

What is VESA?

VESA pops up a lot when mounting displays. It is a family of standards for mounting flat-panel displays, televisions, and other displays to stands or wall mounts. You can also buy VESA-compatible cases to attach Raspberry Pi to the back of displays. The official Raspberry Pi Monitor is VESA compliant with a 100 × 100 mm layout. Larger screens may use 100 × 200 or 200 × 200 (or larger) depending on the weight of the display.

rpimag.co/vesa



VESA Wall Mount for Raspberry Pi monitor

Cost: £3

rpimag.co/monitor-vesa

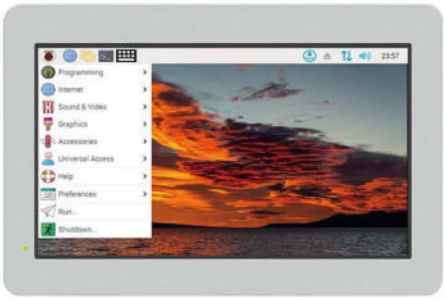
This full-metal bracket enables you to wall-mount the official Raspberry Pi Monitor (sold separately) using its 100 × 100mm VESA points. It also works with other 100- and 75mm-standard VESA. It offers horizontal angle adjustment and cables pass through a central cut-out.

ComfilePi

Cost: From $399

rpimag.co/comfilepi

Comfile offers a series of touch panel systems called ComfilePi. These cases offer strong protection against electrical surges, ESD, and static electricity, making Raspberry Pi suitable for direct use in industrial environments. Panels come with CE, KCC, and UL certification.





These cases offer strong protection against electrical surges

Argon Industria HMI 5C

Cost: £27

rpimag.co/argon-industria-5c

Argon is well-known for its consumer cases. This industrial-spec, solid aluminium enclosure for Raspberry Pi 5 and Touch Display 2 – 5-inch is built for VESA-mounted deployments. Its lipped profile and foam gasket protect against dust and knocks, while an internal I/O PCB reroutes HDMI/USB-C and converts micro HDMI to full-size. A range of optional Argon Industria accessories enable you to add NVMe storage and RS232/RS485 IoT interfaces.





Pimoroni Pibow Frame

Cost: £17

rpimag.co/pibow-frame10

A tidy three-layer desktop frame for Raspberry Pi Touch Display 2 (not included). The two legs offer multiple viewing angles. The touchscreen and all Raspberry Pi ports stay fully accessible. Available in Noir Black or Coupé Red..

Pimoroni OmniVESA

Cost: £5

rpimag.co/omnivesa

A neat, low-cost clear plastic mounting plate that attaches to the back of a bare Raspberry Pi, or one in a case. This gives it a larger base to support 75mm and 100mm VESA spacings. Comes with nylon thumbscrews, spacers, and nuts.



Aluminium VESA Raspberry Pi 5

Cost: £10

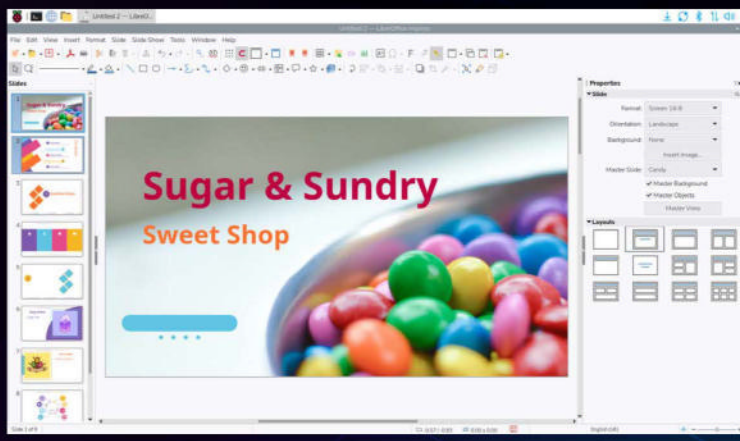
rpimag.co/alum-vesa

A highly protective, all-aluminium case for Raspberry Pi 5 (not included) with passive cooling via upper and lower vents, all side ports accessible, and a soft-gasket power button with indicator light. Two 75mm VESA wings are included for mounting behind a monitor or display.



Quick start display options

If you want to set up a Raspberry Pi in an enthusiast environment, here are some ways to quickly display content



Chrome browser

If Raspberry Pi OS is in GUI mode, you can open a Chrome browser, navigate to your website or YouTube, and press **F11** to enter full-screen mode. A slightly more professional option is to open Terminal and start Chromium in Kiosk mode. For example:

```
$ chromium --kiosk --incognito magazine.raspberrypi.com
```

This is ideal for dashboards, Google Slides, web forms, or online media playback. The downside is fragility: a crashed tab or lost network can take your system offline.

LibreOffice Impress

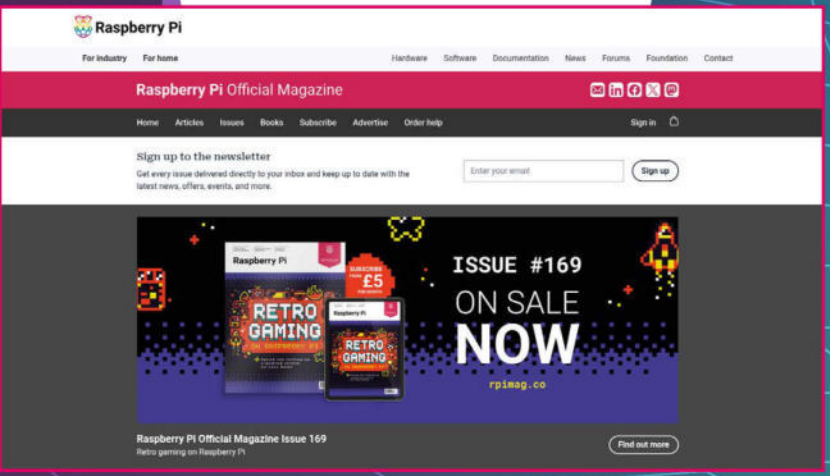
LibreOffice Impress can run a presentation in continuous kiosk loop mode (View > Slide Show, with automatic timed transitions). This is a low-effort way to display slide decks, noticeboards, or event schedules without writing any code.

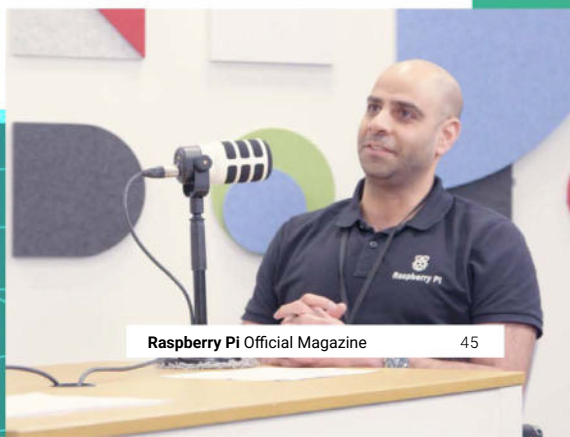
VLC video

A dedicated video-loop display, driven by VLC, plays an MP4 video or looped playlist full-screen at boot. This is the most reliable option for pure video content: trailers, ambient visuals, information reels, and so on. Because the media player is lightweight and rarely crashes compared with a browser, it is preferable to using a service such as YouTube. Start VLC from the command line and point it to your video; for example:

```
$ cvlc --fullscreen --loop --no-osd --no-video-title-show your_video.mp4
```

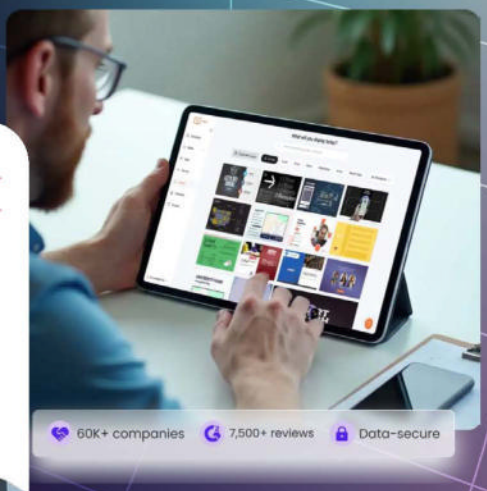
Using the `cvlc` command is preferable to `vlc` because it loads without any GUI overhead. This is cleaner and more memory-efficient.





Industrial display solutions

If you want to deploy a screen, or many screens, there are content management system (CMS) providers out there who use Raspberry Pi. If you are looking to install displays in an industrial or professional environment, these are the people who can help



Yodeck

yodeck.com

Yodeck was founded in 2014 and is Raspberry Pi's largest partner for digital signage. Yodeck offers a Raspberry Pi-based media player called Yodeck Player, with a digital signage solution via annual subscription. It comes complete with 160 apps for integration with a variety of digital services. A free plan is available for use on a single screen; Basic, Premium, and Enterprise plans come with the Yodeck Player and more features.



Raspberry Pi's largest partner for digital signage

Screenly

screenly.io

Screenly went from a side project into powering over 10,000 screens. One of Raspberry Pi's earliest partners, Screenly has been around since 2012. Its paid platform is ScreenlyOS, which runs on Raspberry Pi hardware (**rpimag.co/screenly-install**). You can also pick up a player from the website for $219. There is a free, community-maintained open-source edition called Anthias for those who want a self-hosted option (**anthias.screenly.io**).



NoviSign

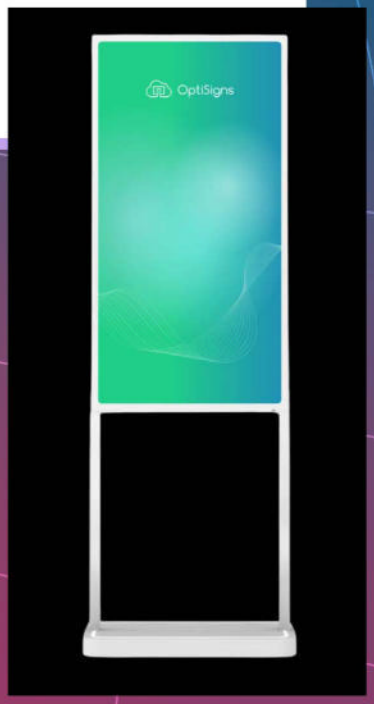
novisign.com

NoviSign is hardware-agnostic and has just announced official Raspberry Pi support, bringing its drag-and-drop content editor and cloud-based CMS to Raspberry Pi-powered players. NoviSign runs on over 50,000 screens across five continents, and clients include Disney, Ikea, Dole, Home Depot, NASA, Nokia, Ferrari, and Papa John's Pizza. Take a look at its Digital Menu for Pizza Shops to see an example of its high-impact work (**rpimag.co/novisign-pizza**).

OptiSigns

optisigns.com

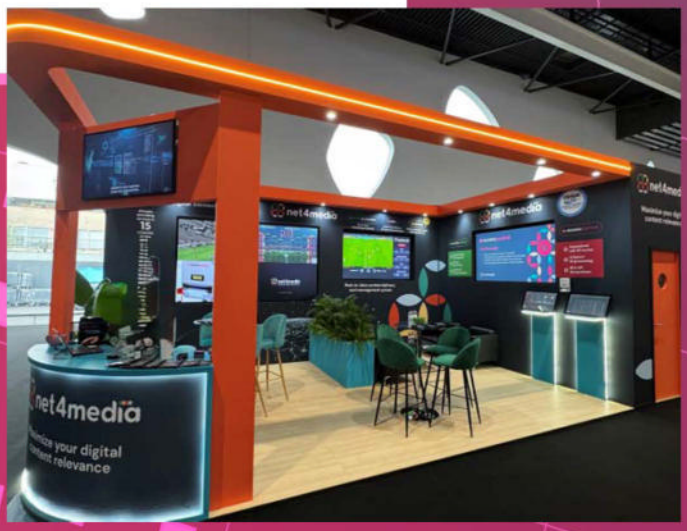
A US-based provider, OptiSigns is hardware-agnostic and has a strong following among Raspberry Pi users. It offers an accessible CMS alongside dedicated kiosk and interactive-display features. It currently serves over 30,000 companies and its OptiKiosk hardware is particularly interesting for event spaces, retail environments, and digital signage. Pricing runs from $10 per month. Take a look at its OptiSigns + Raspberry Pi page (**rpimag.co/optisigns-raspberrypi**).



Net4Media

net4media.net

Based in Europe, Net4Media predominantly supports the betting shop and gaming industry and is powering over 75,000 displays with 40,000 Raspberry Pi media players. Net4Media has a particular focus on safeguarding data and holds ISO/IEC 27001 certification, the globally recognised standard for information security management systems.





Info-Beamer

info-beamer.com

The German founder of this project-turned-business is a long-time friend of Raspberry Pi. Info-beamer offers a highly advanced, fully programmable signage solution capable of smooth Full HD animation, available as standalone software or SaaS. It is proud of its content player software which offers unique features including dual display playback and HEVC videos up to 4K on the Raspberry Pi 4 and Raspberry Pi 5 thanks to using the latest standard Linux APIs. Take a look at Info-Beamer's Pi Wall case study (**rpimag.co/info-beamer-wall**).

Cenareo

cenareo.com

France-based Cenareo powers 35,000 screens deployed worldwide with 90% of them running on Raspberry Pi. Alongside its SaaS business whose clients include L'Oreal, Nespresso, and Saint-Gobain, Cenareo recently acquired a French signage company focused on manufacturers, adding major industrial customers such as Michelin. It offers an energy-saving feature that allows customers to centrally switch off all their digital signage displays at night. Capable of creating advanced signage solutions, Cenareo recently produced digital signage that reacts to real-time weather data to trigger at specific climate conditions (**rpimag.co/cenareo-weather**).

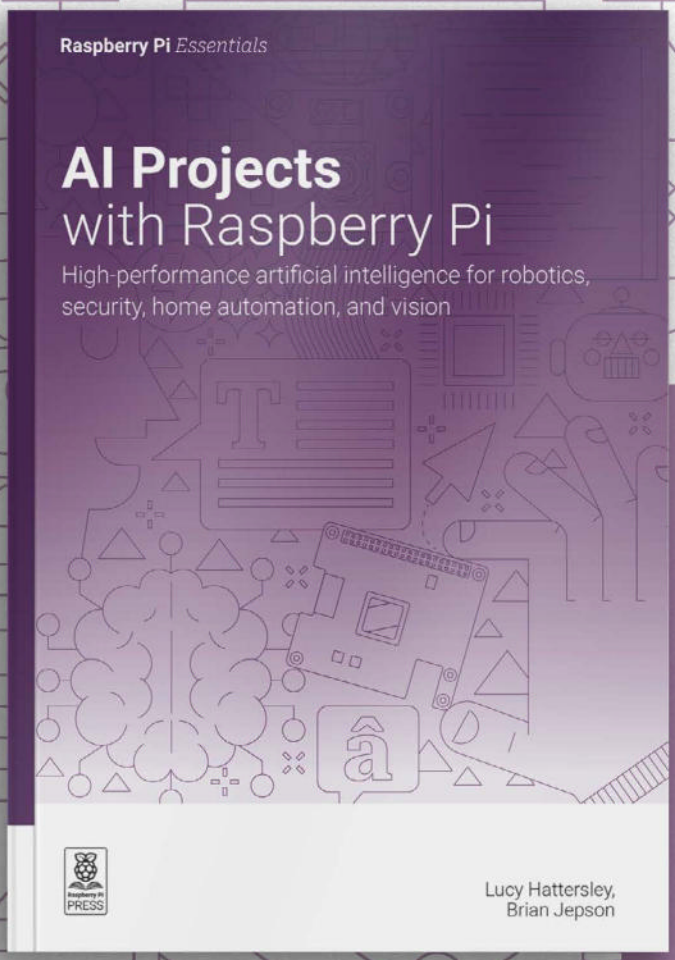
Cenareo recently produced digital signage that reacts to real-time weather data

Fugo

fugo.ai

Fugo is a UK-based company offering a cloud digital signage platform built for ease of use, with Raspberry Pi support at the heart of its player line-up. It specialises in organisations and in-house signage. See its page on Raspberry Pi integration for more information (**rpimag.co/fugo-pi**).





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**

Touch-enabled flight information display

Keep track of planes you can see from your current location with this touch-enabled flight information screen



Maker
Nik Rawlinson
Writer and artist who much prefers travelling by train than by plane and, if he could see them from his house, would be tracking them instead.

nikrawlinson.com

DOWNLOAD THE FULL CODE:
rpimag.co/infoboard-code

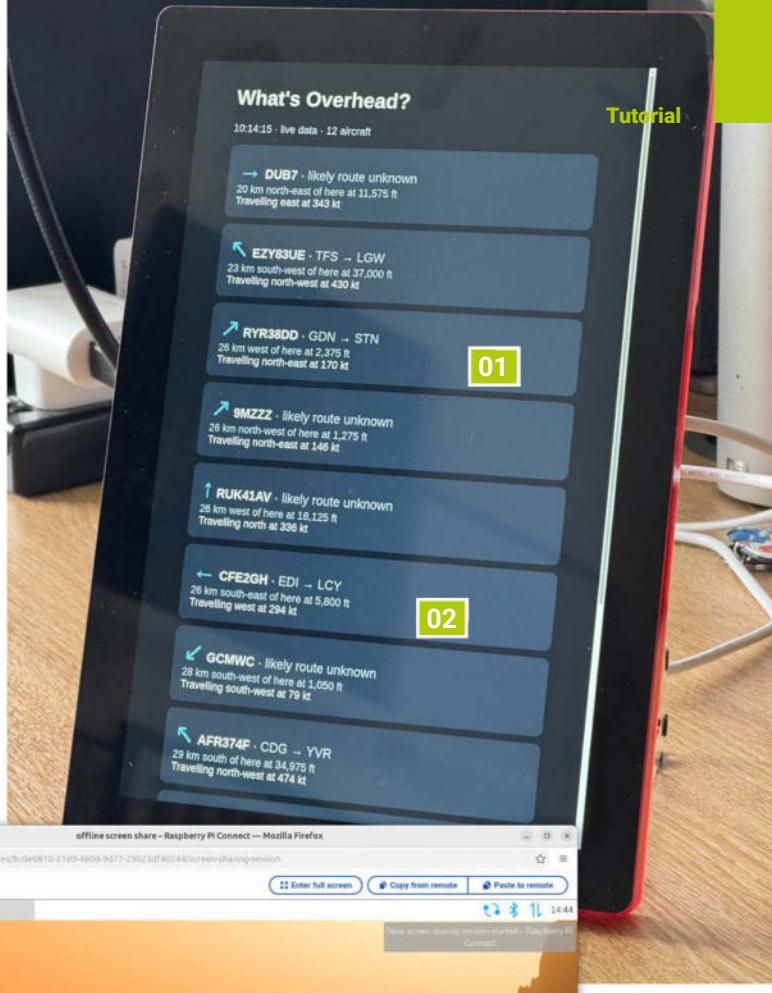
We're fascinated by the planes that fly over our house. Fortunately, we never need to wonder where they're headed, as we can whip out our phone and look them up on FlightRadar24 (**flightradar24.com**). Much of the data on which FlightRadar24 relies is available outside of the app, with several served through various APIs that allow you to use them for personal, non-commercial purposes. We're going to use two of them here – the OpenSky Network (**opensky-network.org**) and adsbdbd (**adsbdb.com**) – to build a live flight information board which displays regularly refreshed information about the planes passing by.

A live flight information board

We're using the 10-inch Raspberry Pi Touch Display 2 and Raspberry Pi 5 for this project, but the principles involved can be easily adapted for use on alternative displays and Raspberry Pi computers. If you don't have a touch-sensitive display, you can also click the overview boxes to pull up more granular information about each flight.

Attach Raspberry Pi

We're using the 10-inch Touch Display 2, which is exclusive to Raspberry Pi 5 and Raspberry Pi Compute Modules. The 5- and 7-inch variants of Touch Display 2 are also compatible with Raspberry Pi 4. Start by connecting your Raspberry Pi to the back of Touch Display 2 using the bundled screws. We would also recommend buying a stand or frame for the screen to keep it secure when in use. We've paired ours with a Pibow Frame in Coupe Red from Pimoroni.



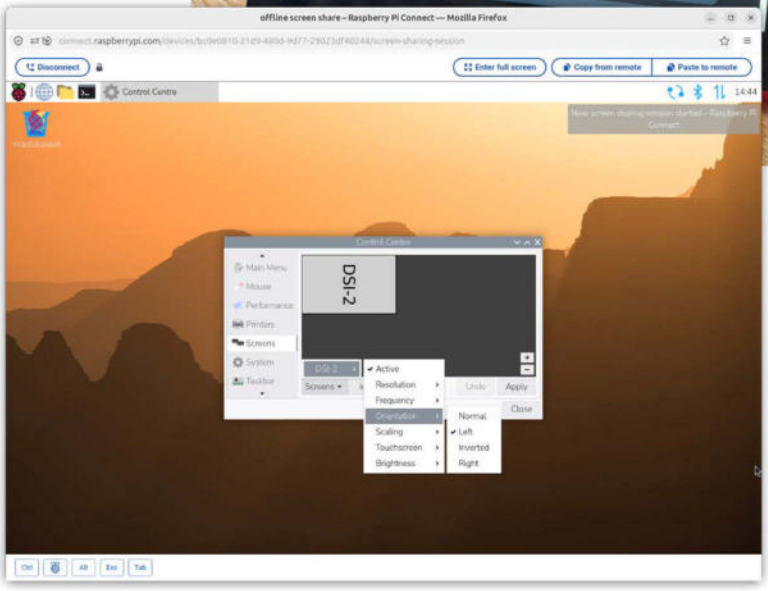
01. Each block gives a summary of nearby planes, including the direction in which we'd need to look to see it, and which way it's travelling

02. The information refreshes every minute. Here, it's showing live data, but it's also cached so the board still shows the most recent data it's downloaded

YOU'LL NEED

- 10-inch Raspberry Pi Touch Display 2 – **rpimag.co/td210**
- Pibow Frame for 10-inch Raspberry Pi Touch Display 2 – **rpimag.co/pibow-td210**
- Optional: Raspberry Pi Keyboard and Hub – **rpimag.co/keyboard**
- Optional: Raspberry Pi Mouse – **rpimag.co/mouse**

By default, Touch Display 2 outputs its contents in portrait orientation, which is great for reading web pages or typing documents. It's also perfect for our information screen as it displays more entries before you need to scroll. However, many chassis, including the Pibow Frame, give you the choice of portrait or landscape orientation. If you opt for the latter, rotate the display's on-screen graphics to match by opening Control Centre from the Raspberry Pi OS menu's Preferences section, then clicking Screens in the sidebar. Click Screens in the main pane, hover over your screen, hover over Orientation, and select the orientation you require (**Figure 1**).



▲ **Figure 1:** If you mount Touch Display 2 in landscape orientation, make sure you rotate its output through Control Centre

Set up nginx and PHP

Our touch-enabled flight information board will be rendered as a web page running on a local server. There are several web servers to choose from, but nginx is one of the quickest and easiest to set up. Open the Terminal by clicking its icon in the Raspberry Pi OS quick launcher, and type

```
$ sudo apt install -y nginx
```

When it's finished installing, open your web browser and visit **http://localhost/**. You should see a 'Welcome to nginx!' page, which indicates that the server is up and running.

Our information panel's web page will be pulled together by PHP. This isn't installed by default alongside nginx, so it will need to be added manually. Return to the Terminal and type the following to install the necessary PHP modules:

```
$ sudo apt install -y php-fpm php-mysql php-
curl php-gd php-mbstring php-xml php-zip
```

Now check which version of PHP you've installed by typing:

```
$ ls /run/php
```

Make a note of the response that matches **phpx.x-fpm.sock**, where x.x is the version number. Ours was **php8.4-fpm.sock**.

Edit nginx's configuration

Type the following command to open the config file:

```
$ sudo nano /etc/nginx/sites-enabled/default
```

Then, within the 'server' block, look for the line '#Add index.php to the list if you are using PHP'. Key down to the line below and before the closing semicolon, add a space followed by:

```
index.php
```

Key down to the '# pass PHP scripts to FastCGI server' block and uncomment (remove the #) from the start of the following lines:

```
location ~ \.php$ {
include snippets/fastcgi-php.conf;
fastcgi_pass unix:/run/php/php7.4-fpm.sock;
}
```

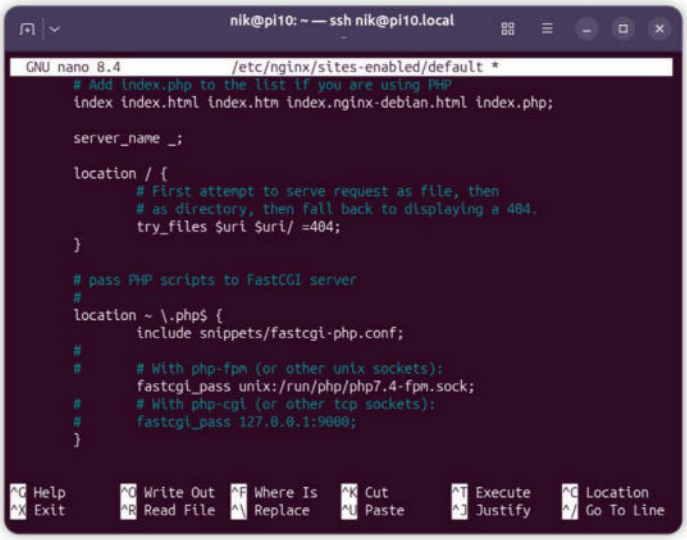
You can see each of these changes implemented in **Figure 2**.

Update your PHP setting

You're probably going to be running PHP version 8.4 or later, like us, so you need to update the line

```
fastcgi_pass unix:/run/php/php7.4-fpm.sock;
```

to reflect this. Change the 7.4 on this line to match the version that you made a note of earlier. Now save the edited file by



▲ **Figure 2:** Update nginx's default settings so it knows how to handle web pages built using PHP

pressing **CTRL+O** followed by **Y**, then quit nano by pressing **CTRL+X**. Restart the nginx web server and PHP by typing the following two commands, pressing **ENTER** after each one. Again, remember to update the specified version of PHP to match the edition installed on your Raspberry Pi. This will make sure Raspberry Pi OS is running the latest configuration of each before going any further.

```
$ sudo systemctl restart nginx
$ sudo systemctl restart php8.4-fpm
```

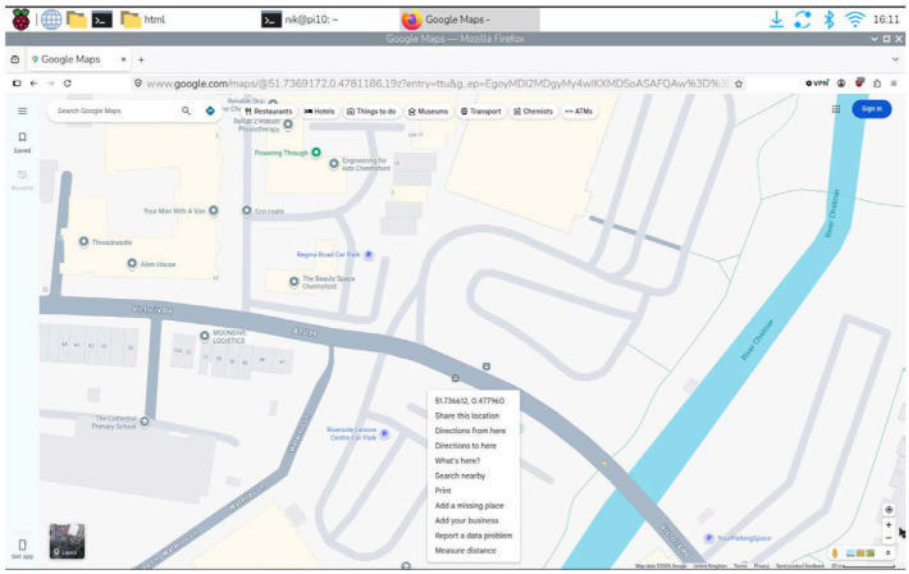
Set server data directory permissions

Right now, you can't write to the folder from which nginx reads web pages, so you won't be able to save your flight information board files there. To fix this, type the following, remembering to change 'nik' for your username:

```
$ sudo chown -R nik: /var/www/html
```

This changes the owner of the **/var/www/html** folder, where nginx looks for locally hosted web pages, to yourself (or the user 'nik' in this case). Previously, it was owned by root, which is why you wouldn't have been able to write to it.

Download the info board code (**rpimag.co/infoboard-code**), then open File Manager and copy **index.php** to **/var/www/html**. You can type in this folder address directly if you enlarge the File Manager window to reveal the address box above the file list. Now right-click **index.php** and select Geany Programmer's Editor from the menu to open it for editing. You need to tell it the coordinates for your location. You can find these by right-clicking on your address at **maps.google.com** (**Figure 3**).



▲ **Figure 3:** Right-click your address in Google Maps and the latitude and longitude coordinates of your location will appear on the top line of the menu

Get ready for your flight (watching)

Change the values for `HOME_LAT` and `HOME_LON` to match the latitude and longitude coordinates you sourced from the top line of the context menu in Google Maps. Then, optionally change the value beside `RADIUS_KM` to specify how far away from you a plane can be from that location to appear on the information board. The `40` we've specified here, for 40km (around 25 miles) is good for Chelmsford, but if you live near an airport you might want to reduce it to avoid overload. Conversely, if you're in the middle of nowhere and rarely see a plane, increase it.

Copy the **start-kiosk.sh** file into your home folder (find this by clicking Home Folder in the File Manager sidebar). Right-click it and click Properties, then switch to the Permissions tab and change the menu beside Execute to Anyone. Now, return to Terminal and type the following:

```
$ crontab -e
```

Key down to the bottom of the file and, on a new line, type:

```
@reboot /home/nik/start-kiosk.sh &
```

Again, swap out 'nik' for your username. Press **CTRL+X** to quit, then press **Y** and **ENTER** to save the modified file and overwrite the existing crontab configuration (**Figure 4**).

Close all open windows, saving files if necessary, and restart your Raspberry Pi. A few seconds after it finishes rebooting, it'll launch Chromium in kiosk mode without a UI or address bar, and load the locally hosted web page that pulls together the data for your information board. Tap any of the displayed flights to pull up more detailed info, including the airline, route, altitude, speed, and more, where known. The data refreshes once a minute, and caches locally in case your Raspberry Pi loses its network connection.

Tap any of the displayed flights to pull up more detailed info, including the airline, route, altitude, speed, and more

Go manual

We've set up our information board to launch full-screen at startup, which makes it difficult to use our Raspberry Pi for anything else. If you'd rather invoke it manually, type the following at the command prompt, then visit **localhost:8001** in your browser:

```
$ pgrep -f "php -S" || php -S 0.0.0.0:8001
-t /var/www/html &
```

start-kiosk.sh

> Language: **Bash**

DOWNLOAD THE FULL CODE:

rpimag.co/infoboard-code

```
001.  #!/bin/bash
002.  # Wait for the Wayland/X desktop session to settle
003.  sleep 5
004.
005.  # Ensure PHP server is running
006.  pgrep -f "php -S" || php -S 0.0.0.0:8001 -t /var/www/html &
007.
008.  # Wait 2 seconds for the server to bind
009.  sleep 2
010.
011.  # Set the display variable (required for background autostart
      scripts)
012.  export DISPLAY=:0
013.
014.  # Launch Chromium in Kiosk mode
015.  chromium --kiosk --noerrdialogs --disable-infobars --no-first-run
      http://localhost:8001
```

Missing information

Some flights limit the amount of data that they make available, so you will occasionally see entries for which origin and destination information is unavailable. If an aeroplane strays outside of your defined radius (40km by default from your location), it will be marked as 'no longer visible' when you tap its summary entry for more details.

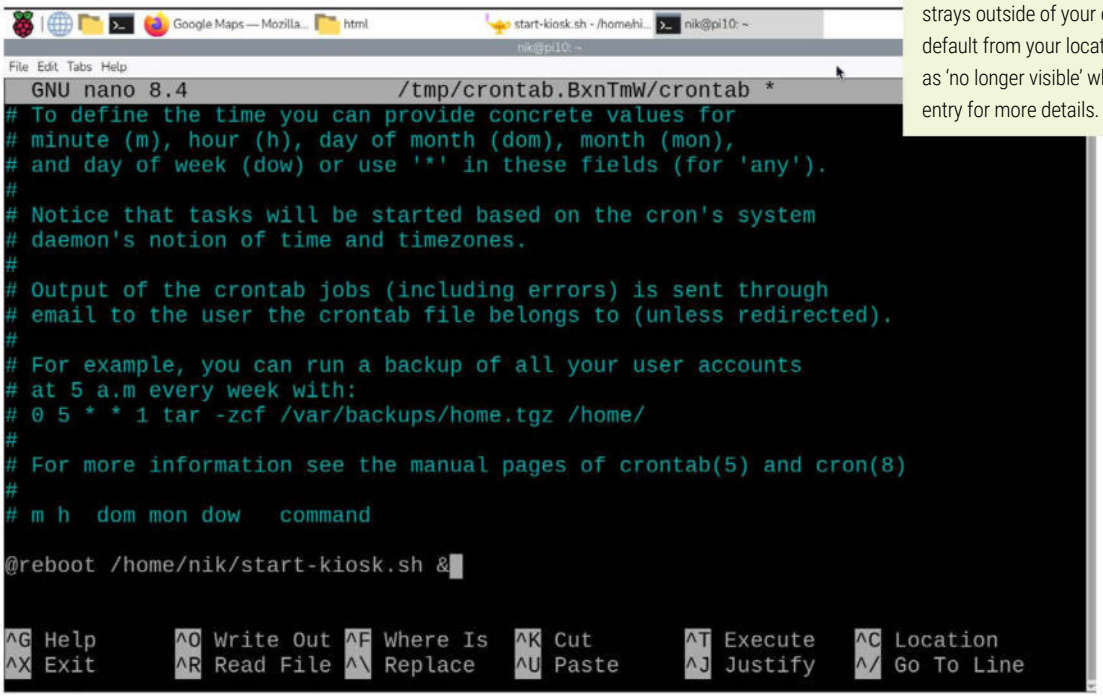


Figure 4: Set up your information board to launch automatically when your Raspberry Pi reboots

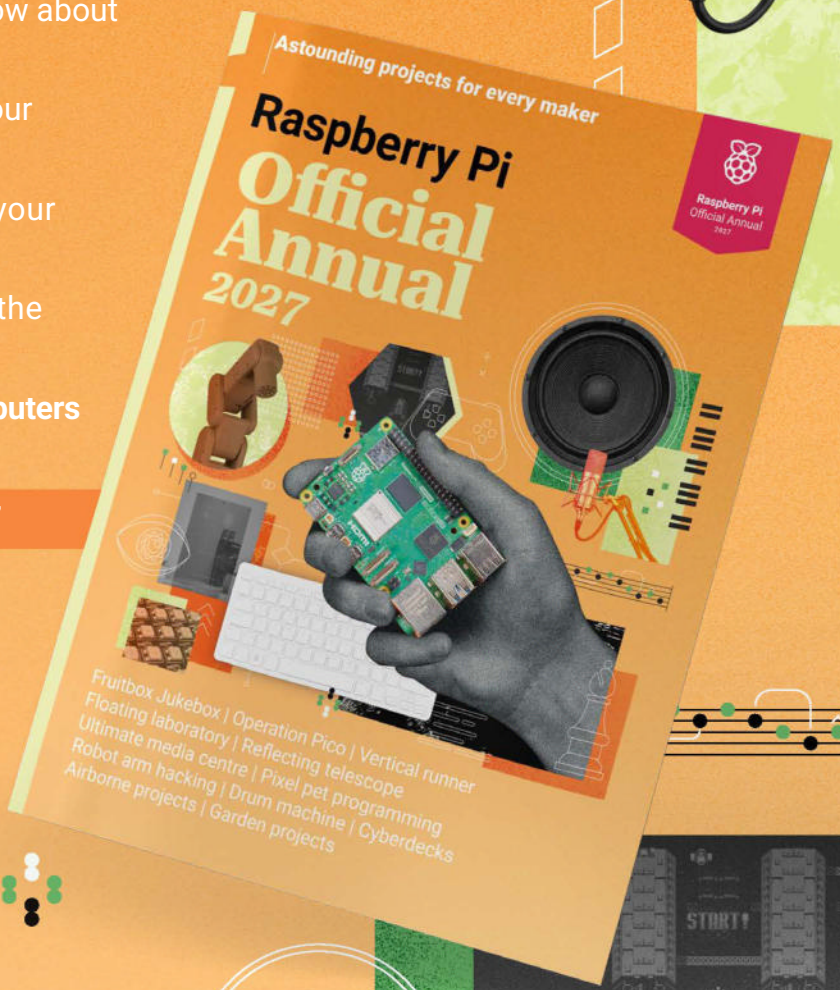
Raspberry Pi Official Annual 2027

200 pages of Raspberry Pi

- **A get started guide** covering every Raspberry Pi
- **Everything you need** to know about Raspberry Pi 500+
- **Troubleshooting tips** for your latest build
- **Inspiring projects** to fuel your next project idea
- **Tutorials on how to build** the ultimate media centre
- **Pushing Raspberry Pi computers** to extreme limits

rpimag.co/annual27











Touch Display 2 writing machine

Stay focused by building a compact, text-based writing machine that keeps your attention on the job at hand



Maker
Nik Rawlinson
Writer and artist who remembers a world when computers were all entirely text based and staying focused was easy.

nikrawlinson.com

YOU'LL NEED

- Raspberry Pi Touch Display 2 – 7-inch (**rpimag.co/touchdisplay2**)
- Pibow Frame for Raspberry Pi Touch Display 2 – 7-inch (**rpimag.co/pibow-frame**)
- Raspberry Pi Keyboard and Hub (**rpimag.co/keyboard**)

How many previous-generation Raspberry Pi boards do you have at your disposal? We've had a count-up and, allowing for one or two that we might have overlooked, we've settled on 16. Most are in active use, but one Raspberry Pi 3B+, attached to a first-generation 7-inch Raspberry Pi Touch Display, has been gathering dust for a couple of years. This felt like the perfect spec for hosting a low-powered, text-based word processor, reminiscent of the distraction-free machines we remember from the early 1990s.

Although you can't use it to access the internet directly (too many distractions), it does connect to our local network, allowing us to offload our writing without resorting to SD-card swapping. And don't worry if your Raspberry Pi and display are younger than ours. As you can see from the 'You'll need' list, you can just as easily build it with more recent hardware.

Install Raspberry Pi OS Lite

You should already be familiar with the process of setting up Raspberry Pi OS using Raspberry Pi Imager. However, we'd bet that on almost every occasion you've opted for the GUI-based release. In this instance, that's more than you need, so instead opt for Raspberry Pi OS Lite, which you'll find in the 'Raspberry Pi OS (other)' submenu. When setting it up through Imager, make sure you enable SSH and set it to allow logging in with username and password. You'll use SSH to remotely administer the Writing Machine and download your documents. Make a note of the hostname that you enter (we've chosen 'writer').

Connect remotely

Boot your Raspberry Pi and, from your regular computer, connect remotely via SSH using the formula at the Windows Command Prompt, or macOS or Linux Terminal:

```
$ ssh username@hostname.local
```

Enter your password when prompted. The Touch Display might not be correctly oriented, depending on the way it's mounted in your stand. If you need to rotate it, type

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

to open the configuration file. Use the right arrow key to move to the end of the only line in the file and add

```
video=DSI-1:800x480@60,rotate=[rotation-value]
```

where `[rotation-value]` is 90, 180, or 270, depending on the number of degrees by which you need to rotate the output.

Press **CTRL+X** to quit nano and Y to save your file, then reboot Raspberry Pi with:

```
$ sudo shutdown -r
```

Create your folders

The Writing Machine will be used for text documents and journal entries, which we'll keep separate by saving them to different folders. Reconnect via SSH, then create your writing directory with:

```
$ mkdir -p ~/Writing
```

To save our writing folder from getting out of hand, we need a way to archive documents we've finished with. Do that by creating an **Archive** subfolder:

```
$ mkdir -p ~/Writing/Archive
```

Create a directory for journal entries with:

```
$ mkdir -p ~/Journal
```

Finally, create a hidden folder for the Writing Machine script that you'll create later:

```
$ mkdir -p ~/.local/bin
```

Touch Display 2 documentation

The screen rotation command we've used here works with our first-generation Touch Display. If you have a Touch Display 2, refer to the documentation at **rpimag.co/touchdisplay-orientation**.

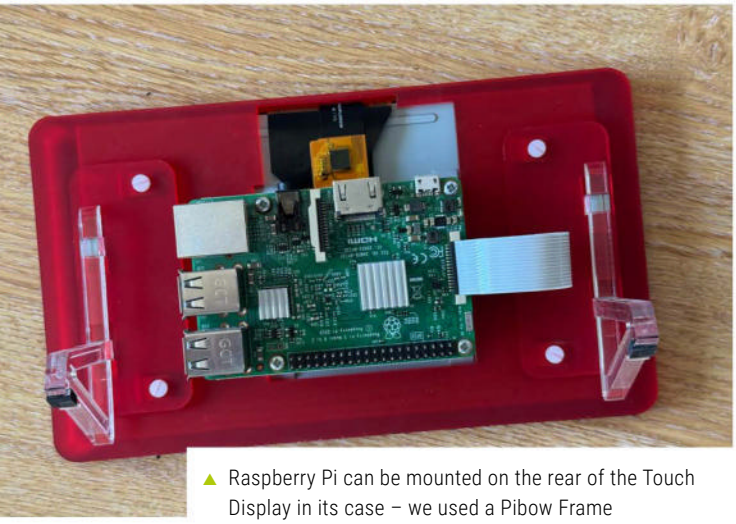


▲ Journal entries are automatically generated using the current date in yyyy-mm-dd format. Whatever you're writing, you'll do so using nano, which saves your work as a .md Markdown file

The Writing Machine will be used for text documents and journal entries

Configure nano

By default, nano, which we'll use for editing documents, wraps text without any consideration for whether or not it's in the middle of a word. This is ugly and can be hard to read. So, implement soft wrapping at spaces ('blanks' in nano parlance), tell it to automatically indent new lines by the same number of spaces or tabs as the previous paragraph, and set it to remember the cursor position in the most recent 200 files you've worked with so it can automatically return there the next time you open the file.



▲ Raspberry Pi can be mounted on the rear of the Touch Display in its case – we used a Pibow Frame

Edit nano's config file

To implement the settings we mentioned in the previous step, enter this command to open the configuration file:

```
$ nano ~/.nanorc
```

Add the following commands to the config file, each of which should be on a separate line:

```
set softwrap
set atblanks
set autoindent
set positionlog
set tabsize 4
```

Press **CTRL+X** to quit nano and **Y** to save your file.

Install Whiptail

To save ourselves the job of styling and configuring a hand-coded menu, we'll use the Whiptail tool. This produces the kind of text-based menus that you'll be familiar with if you've opened the raspi-config tool to adjust your Raspberry Pi base configuration. To do so, type:

```
$ sudo apt install whiptail
```

Press **Y** if prompted to continue with the installation.

When Whiptail has finished installing, create a new file for your writing machine script:

```
$ nano ~/.local/bin/writing-machine.sh
```

Paste in the code from GitHub (**rpimag.co/writing-machine**, full listing overleaf), then press **CTRL+X** and **Y** to quit and save.

Set the Writing Machine to appear at boot

Make your writing machine script executable by typing:

```
$ chmod +x ~/.local/bin/writing-machine.sh
```

You now need to set it to launch as soon as anyone logs in to the writing machine, which you can do by editing your hidden profile file. Enter the command:

```
$ nano ~/.profile
```

Put the following code at the end of the file:

```
if [ "$(tty)" = "/dev/tty1" ]; then
  exec ~/.local/bin/writing-machine.sh
fi
```

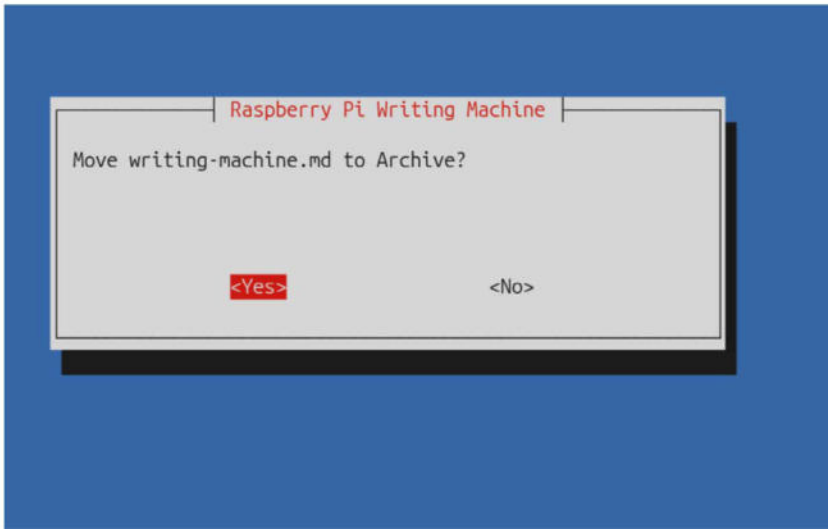
Press **CTRL+X** to quit nano and **Y** to save the file. Now reboot your Raspberry Pi by entering:

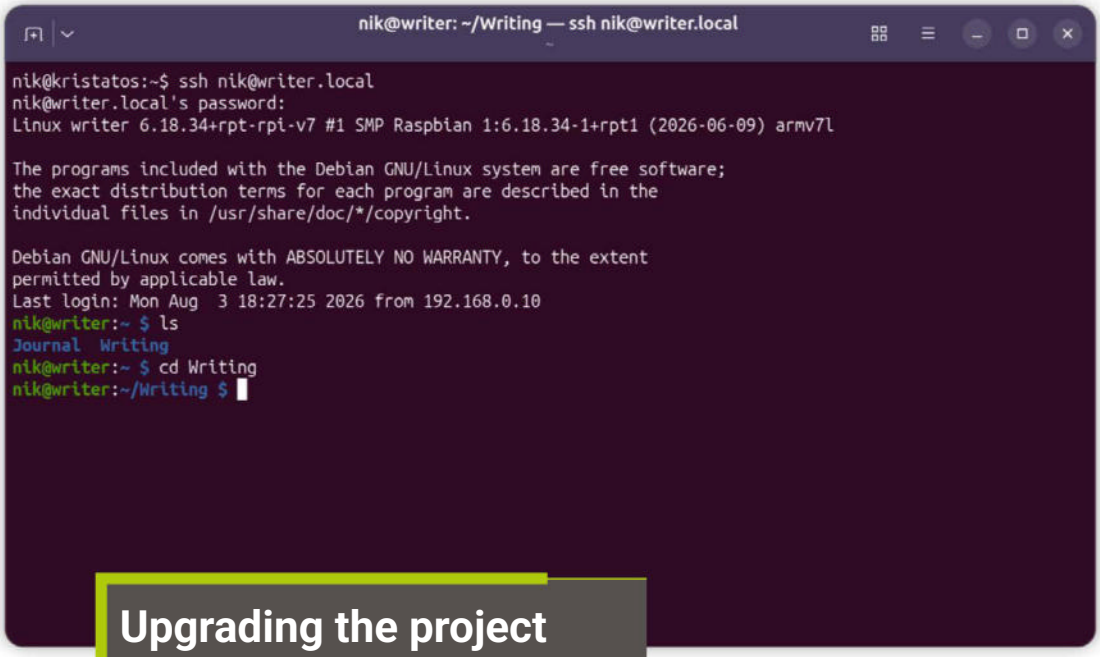
```
$ sudo shutdown -r
```

Use your writing machine

Switch from SSH to the writing machine and sign in, and you'll be dropped straight into the Writing Machine menu, with options to create a journal entry or text document, to rename or archive a document, or to power off the machine. Journal is highlighted, so press **ENTER**. You'll be taken straight into nano where, if you look at the top line, you'll see that the file name is the current date in **yyyy-mm-dd.md** format, and it's stored in the **Journal** folder. Press **CTRL+X** to quit and return to the main menu. Now key down to option 2, 'Text document', and press **ENTER**, then press **ENTER** again with New document highlighted.

▼ The option to Archive a document when we no longer need it saves the basic file management options from becoming inadequate over time





◀ Although you can't access the internet on the device itself, it still connects to your local network so you can administer it remotely via SSH

Upgrading the project

Raspberry Pi Touch Display 2 – 10-inch works vertically by default, so keeps more of your document in view at a time. It would make a great upgrade to this project, especially when paired with the affordable Raspberry Pi 5 1GB.

Writing text

Give your new document a name and press **ENTER**. Notice that this name appears in the top bar, and the document is in the **Writing** folder. Also note that the file extension is .md, for Markdown. Markdown is a character-based formatting system, in which using #, ##, ###, and so on at the start of a line styles the text on that line in heading 1, 2, or 3 style. Surrounding text by single asterisks italicises it; using double asterisks makes it bold. See **rpimag.co/md-cheat** for a comprehensive list of formatting options.

Archive a file

Enter some text in your document, then press **CTRL+X** followed by **Y** to save it and exit. Now, imagine you've finished working with it and you want to reduce the number of files that appear on the 'Open existing document' submenu within the Text document menu. This is where the **archive** folder you created earlier comes in handy. Key down to 'Archive document' and press **ENTER**, then select the document you were working with and press **ENTER**. Confirm that you want to move it to the archive by pressing **ENTER** with Yes highlighted on the following screen.

Download your documents

Switch back to your regular computer. If it's still connected to the writing machine via SSH, type `exit` and press **ENTER**. Now, type the following:

```
$ scp username@hostname.local:/home/username/
Writing/*.md .
```

(Change `hostname` and both instances of `username` as appropriate, and make sure you include the dot at the end of the line.) Press **ENTER** and enter the password for your writing machine, and it will download all of your .md files to the local folder on the machine you're using. If you only want to download one file, replace `*.md` with the name of the file you're targeting. You can also download files from the **Journal** or **archive** folders by editing the remote folder name as appropriate.

Convert your document

You need to convert your .md downloads to implement your text-based formatting by using a Markdown editor. There are plenty to choose from, including Markdown To Word online (**markdowntoword.io**). Paste the content of your file into the left-hand column (or use the upload option in the box above it) and you'll see a preview of the output in the right-hand column. Make any final edits you want to the original text, then click 'Download as Docx' to download a Word-compatible version of the file with your styling baked in.

writing-machine.sh

> Language: **Bash**

```bash
#!/usr/bin/env bash
set -u

WRITING_DIR="$HOME/Writing"
ARCHIVE_DIR="$WRITING_DIR/Archive"
JOURNAL_DIR="$HOME/Journal"
EDITOR_CMD="nano"
DEFAULT_EXT="md"
TITLE="Raspberry Pi Writing Machine"

mkdir -p "$WRITING_DIR" "$ARCHIVE_DIR" "$JOURNAL_DIR"

sanitize_name() {
  printf '%s' "$1" \
    | tr '[:upper:]' '[:lower:]' \
    | sed -E 's/[^a-z0-9._ -]+/-/g; s/[[:space:]]+/-/g; s/-+/-/g; s/^-|-$//g'
}

edit_document() {
  local file="$1"
  clear
  "$EDITOR_CMD" "$file"
}

ensure_journal_file() {
  local file="$JOURNAL_DIR/$(date +%F).md"
  if [ ! -f "$file" ]; then
    {
      echo "# Journal $(date +%F)"
      echo
      echo "Created: $(date '+%Y-%m-%d %H:%M')"
      echo
    } > "$file"
  fi
  printf '%s\n' "$file"
}

journal_mode() {
  local file
  file=$(ensure_journal_file)
  edit_document "$file"
}

}

list_docs() {
  find "$WRITING_DIR" -maxdepth 1 -type f \( -name '*.md' -o -name '*.txt' \) -printf '%f\n' | sort
}

new_document() {
  local name slug path
  name=$(whiptail --title "$TITLE" --inputbox "New document name:" 10 70 3>&1 1>&2 2>&3) || return
  [ -z "$name" ] && return
  slug=$(sanitize_name "$name")
  [ -z "$slug" ] && slug="untitled-$(date +%Y%m%d-%H%M%S)"
  path="$WRITING_DIR/${slug}.${DEFAULT_EXT}"

  if [ -e "$path" ]; then
    whiptail --title "$TITLE" --msgbox "That document already exists." 10 60
    return
  fi

  {
    echo "# $name"
    echo
    echo "Created: $(date '+%Y-%m-%d %H:%M')"
    echo
  } > "$path"

  edit_document "$path"
}

choose_document() {
  local files menu_items=() choice
  mapfile -t files < <(list_docs)

  if [ ${#files[@]} -eq 0 ]; then
    whiptail --title "$TITLE" --msgbox "No text documents yet." 10 60
    return 1
```

DOWNLOAD THE FULL CODE:

rpimag.co/writing-machine

```
    fi

    for f in "${files[@]}"; do
      menu_items+=("$f" "Document")
    done

    choice=$(whiptail --title "$TITLE" --menu "Choose a document:" 22 78 12 "${menu_items[@]}" 3>&1 1>&2 2>&3) || return 1
    [ -n "$choice" ] && printf '%s\n' "$choice"
}

open_document() {
    local doc
    doc=$(choose_document) || return
    edit_document "$WRITING_DIR/$doc"
}

rename_document() {
    local doc newname slug ext newpath
    doc=$(choose_document) || return
    newname=$(whiptail --title "$TITLE" --inputbox "New name for $doc:" 10 72 "${doc%.*}" 3>&1 1>&2 2>&3) || return
    [ -z "$newname" ] && return
    slug=$(sanitize_name "$newname")
    [ -z "$slug" ] && return
    ext="${doc##*.}"
    newpath="$WRITING_DIR/${slug}.${ext}"

    if [ -e "$newpath" ]; then
      whiptail --title "$TITLE" --msgbox "A document with that name already exists." 10 60
      return
    fi

    mv "$WRITING_DIR/$doc" "$newpath"
    whiptail --title "$TITLE" --msgbox "Renamed to $(basename "$newpath")" 10 60
}

archive_document() {
    local doc
    doc=$(choose_document) || return
    whiptail --title "$TITLE" --yesno "Move $doc to Archive?" 10 60 || return
    mv "$WRITING_DIR/$doc" "$ARCHIVE_DIR/"
}

main_menu() {
    while true; do
      local choice
      choice=$(whiptail \
        --title "$TITLE" \
        --menu "Choose an option" 18 72 10 \
        "1" "Journal" \
        "2" "Text document" \
        "3" "Rename document" \
        "4" "Archive document" \
        "5" "Power off" \
        3>&1 1>&2 2>&3) || continue

      case "$choice" in
        1) journal_mode ;;
        2)
          subchoice=$(whiptail \
            --title "$TITLE" \
            --menu "Text document" 16 72 8 \
            "1" "New document" \
            "2" "Open existing document" \
            "3" "Back" \
            3>&1 1>&2 2>&3) || continue
          case "$subchoice" in
            1) new_document ;;
            2) open_document ;;
            3) ;;
          esac
          ;;
        3) rename_document ;;
        4) archive_document ;;
        5) sudo poweroff ;;
      esac
    done
}

main_menu
```

Run Ultralytics YOLO on Raspberry Pi with OpenVINO

Optimize YOLO models to run efficiently on Raspberry Pi hardware



Maker

Naush Patuck

Naush is a software engineering manager at Raspberry Pi who specialises in AI applications. Friends of ours from Intel and Ultralytics – Alexander Nesterov, Dmitriy Pastushenkov, Francesco Mattioli, and Nuvola Ladi – helped with this tutorial.

rpimag.co/naush

YOU'LL NEED

- Raspberry Pi 5 **rpimag.co/raspberrypi5**
- OpenVINO **rpimag.co/openvino**

This guide focuses on deploying Ultralytics YOLO computer vision models on Raspberry Pi with OpenVINO (rpimag.co/openvino).

OpenVINO is an open source toolkit for optimising and deploying AI inference. It's developed by Intel but supports Arm64 CPUs with arm64-v8a and higher, which includes our Raspberry Pi models.

The fundamental idea behind OpenVINO is to optimise inferencing models to match specific hardware. It takes a model exported from a framework, such as PyTorch, converts it into an OpenVINO intermediate format, and then compiles it specifically for the CPU it is going to run on – in our instance, the Arm Cortex-A76 used in our Raspberry Pi 5.

In this tutorial, we will cover how the runtime is installed, how models become deployment artefacts, how compilation and caching affect startup, and how builds become repeatable.

The standard OpenVINO deployment practices still apply. Convert the model when load latency matters, compile for the target device, cache compiled artefacts, and package the runtime explicitly.

Raspberry Pi remains the concrete edge target

This gives Raspberry Pi a clear role. It is not an exception to OpenVINO, but rather a small Linux Arm64 target to which the standard OpenVINO deployment model can be applied directly.

System setup at a glance

The recommended first path is as follows:

- Use Raspberry Pi OS (64-bit) on Raspberry Pi 4 or Raspberry Pi 5
- Create a clean Python virtual environment
- Install OpenVINO from PyPI
- Verify that OpenVINO sees the CPU device
- Install the Ultralytics Python package
- Prefer OpenVINO IR for deployment when startup time matters
- Enable model caching for services that restart often
- Move to source builds or cross-builds when you need control over OpenVINO runtime packaging, C++ integration, or repeatable release artefacts

Raspberry Pi is the target, OpenVINO is the deployment layer

Raspberry Pi is a common target for practical computer vision. It can sit next to a camera, inside a prototype, near a machine, on a lab bench, or in a small service that needs to run without a workstation nearby.

OpenVINO gives Raspberry Pi boards a standard deployment shape. At the time of writing, the current Raspberry Pi OS 64-bit image is a Debian Trixie–based system for Raspberry Pi 3 Model B and newer, including Raspberry Pi 4 and Raspberry Pi 5. OpenVINO 2026.2.0 publishes Linux Arm64 wheels on PyPI for CPython 3.10, 3.11, 3.12, and 3.13. The official installation flow follows the standard Python pattern: create a virtual environment, install the package, and check the runtime devices.

Raspberry Pi 4 and Raspberry Pi 5 have different performance envelopes and should not be treated as the same target. Raspberry Pi 4 features a quad-core Cortex-A72 processor at 1.8GHz and can be configured with up to 8GB of LPDDR4 memory. Raspberry Pi 5 moves to a quad-core Cortex-A76 processor at 2.4GHz and has memory options up to 16GB.

For deployment planning, Raspberry Pi 4 fits prototypes and light services, while Raspberry Pi 5 gives more room for sustained camera workloads and application logic around inference.

The graph in **Figure 1** demonstrates that OpenVINO delivers strong performance compared to other frameworks on Raspberry Pi when deploying the YOLO26 model.

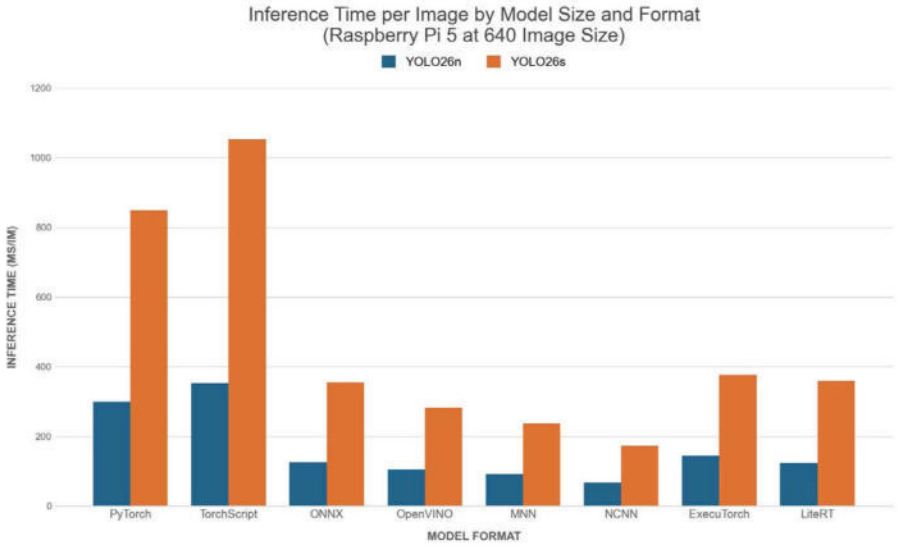
DOWNLOAD THE FULL CODE:

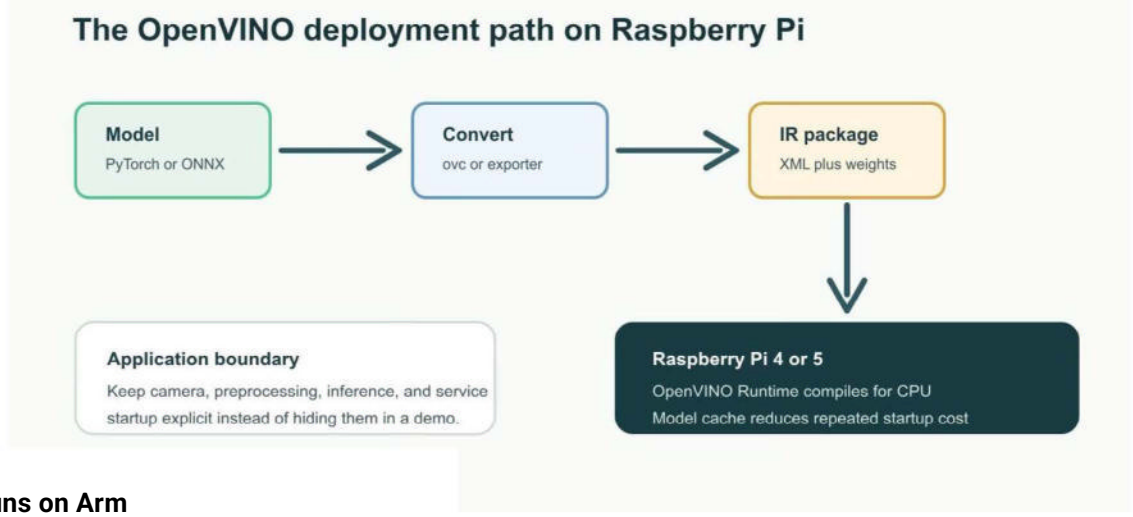
rpimag.co/github

The practical question is no longer ‘Can OpenVINO run on Raspberry Pi?’, but rather, ‘Which OpenVINO deployment path gives this edge application the right runtime, startup, and maintenance model?’

Board	Hardware profile	Best OpenVINO fit	Considerations
Raspberry Pi 4	Cortex-A72 at 1.8GHz, up to 8GB RAM	Prototypes, single-camera pipelines, lightweight services	Use a 64-bit OS and keep the first pipeline modest
Raspberry Pi 5	Cortex-A76 at 2.4GHz, up to 16GB RAM	Sustained camera workloads, richer edge applications, repeatable packaging	Treat cooling and power as part of the design

Figure 1: Raspberry Pi Comparison Chart (graph presented as provided by Ultralytics, **rpimag.co/pi-yolo-benchmarks**)





▲ **Figure 2:** How OpenVINO runs on Arm

How OpenVINO runs on Arm

OpenVINO follows the same programming model on Raspberry Pi as other OpenVINO deployments (see **Figure 2**). The application uses openvino.Core, discovers devices through available_devices, reads or converts models, and compiles a model for a device such as the CPU.

This consistency matters because it means the application does not have to become Raspberry Pi–specific. If it later moves between Raspberry Pi, an x86 laptop, an Intel GPU machine, or another edge system, the top-level OpenVINO pattern remains the same: create a runtime, choose or query devices, compile a model, and run inference.

When an application uses the CPU on Raspberry Pi, inference goes through a layered runtime path. The application does not call Arm Compute Library or KleidiAI directly; OpenVINO Runtime loads the model, prepares it for compilation, and passes it to the CPU plug-in for the target CPU. The CPU plug-in then selects an executor path for each supported operation based on operation type, shape, precision, layout, and available build features.

This distinction is important for both performance and troubleshooting. Seeing the CPU in available_devices means the OpenVINO CPU plug-in is available – it does not mean every operation uses the same low-level library. A convolution, a fully connected layer, a transpose, and a fallback operation may use different executor paths inside the same compiled model.

There are two details that deserve special attention on Arm platforms:

1. Precision requires explicit expectations. The OpenVINO CPU documentation says that Arm platforms execute quantised models in simulation mode, meaning the graph – including quantisation operations – runs in floating-point precision. Quantisation can still be useful when the same model workflow targets other devices, but an int8 export should not be presented as a guaranteed Raspberry Pi speed path.
2. Model caching is a deployment feature, not only a benchmarking option. Compilation can include target-specific work. If you enable cache_dir, OpenVINO can cache compiled artefacts and reuse them later. On a small edge device that starts as a service, reboots after power loss, or restarts after updates, the first few seconds of startup matter.

The computer vision model

Ultralytics YOLO (**rpimag.co/yolo26**) is a family of real-time computer vision models that can analyse images and video in a single pass to quickly identify and understand visual information. Ultralytics YOLO models are used because they balance speed, accuracy, and deployment flexibility, making them especially well suited for real-time and edge AI applications where latency, efficiency, and reliability matter.

Ultralytics YOLO offers a unified workflow, strong documentation, simple Python and CLI interfaces, broad export support, and compatibility with many deployment targets, from cloud systems to embedded devices like Raspberry Pi.

YOLO models support key computer vision tasks, including object detection, instance segmentation, image classification, pose estimation, oriented bounding box detection, and tracking. When used with Raspberry Pi and OpenVINO, Ultralytics YOLO models can be optimised for efficient on-device inference: Raspberry Pi provides a compact, affordable edge platform, OpenVINO helps convert and optimise models for faster execution on supported hardware, and Ultralytics simplifies the export and deployment process.

Compiling or exporting models into a target-specific format is essential, as edge devices have limited compute, memory, and power; optimisation steps such as graph conversion, precision adjustment, and quantisation help the model run faster, use fewer resources, and perform reliably on the target device.

Ultralytics YOLO models are available under the AGPL-3.0 licence by default, supporting open collaboration and transparency. Developers using Ultralytics YOLO must either open source the entire project under AGPL-3.0 or obtain an Ultralytics Enterprise License for proprietary, internal, commercial, R&D, or edge deployments. Learn more on the Ultralytics licensing page (**rpimag.co/ultralytics-license**).

Quick start with OpenVINO Runtime and the Ultralytics Python package

The most practical first setup keeps the OpenVINO environment explicit. Open Terminal and enter:

```
  $ sudo apt update
  $ sudo apt install -y python3-venv python3-pip
git libglib2.0-0 libgl1
  $ python3 -m venv ~/venvs/ov-rpi
  $ source ~/venvs/ov-rpi/bin/activate
  $ python -m pip install --upgrade pip
  $ python -m pip install openvino onnx
  $ python -c "from openvino import Core;
print(Core().available_devices)"
```

If the output includes `['CPU']`, OpenVINO is installed, and the CPU plug-in is visible to the runtime.

The next decision is model format. OpenVINO can read ONNX models directly, which is convenient during development. For deployment, converting to OpenVINO IR is preferable when load latency matters.

The ovc application can be used to convert ONNX models:

```
  $ ovc <your_model.onnx> --output_model
<your_model_ir>
```

NOTE

You can take a look at the options for OpenVINO with ovc.

`$ ovc --help`

Install the Ultralytics Python package that provides all the tools and interface to download and operate Ultralytics YOLO models.

```
  $ python -m pip install ultralytics
```

In **openvino_1.py**, YOLO26 gives the runtime path a visible workload. For the first OpenVINO run, export the smallest reference model without extra precision options:

```
  from ultralytics import YOLO
  model = YOLO("yolo26n.pt")
  model.export(format="openvino", imgsz=640)
```

This keeps the first result easy to validate. Ultralytics exposes `half`, `int8`, `dynamic`, `nms`, and other export arguments for OpenVINO, but those belong after the plain model path is verified on the board.

Run the code with:

```
  $ python openvino_1.py
```

If run correctly, the output should contain:

```
  OpenVINO: export success ✓ 2.9s, saved as
'yolo26n_openvino_model/' (9.6 MB)
```

Use OpenVINO Runtime directly when the application needs explicit control, as shown in **openvino_2.py**:

```
  import openvino as ov
  import openvino.properties as props
  core = ov.Core()
  print("Available devices:",
        core.available_devices)
  core.set_property({props.cache_dir:
                     "./ov_cache"})
  model = core.read_model(
          "yolo26n_openvino_model/yolo26n.xml")
  compiled_model = core.compile_model(
                   model, "CPU")
  print("Compiled for:",
        compiled_model.get_property(
        "EXECUTION_DEVICES"))
```

The important part is the deployment pattern: discover devices, enable caching, read the model, compile for CPU, and keep the inference service explicit.

Compiling or exporting models into a target-specific format is essential

Three OpenVINO deployment paths

Start with PyPI for the first working deployment. It lets you verify the model, preprocessing, camera path, and service logic on the board before changing OpenVINO itself. As the deployment matures, the requirement becomes repeatable for rebuilds, packaging, transfers, verifications, and updates without creating a board-specific setup.

The PyPI path is the application path. Use it when your product is a Python application and the OpenVINO package can be treated like any other dependency. For more details, take a look at the OpenVINO PyPI installation documentation: **rpimag.co/install-openvino**.

The native source-build path is the runtime-control path. It makes sense when OpenVINO itself is part of what you are validating: C++ integration, custom build options, local wheel generation, or debugging a platform-specific runtime issue. It's straightforward, but it uses Raspberry Pi CPU time and memory. See the OpenVINO documentation on building OpenVINO for Raspberry Pi: **rpimag.co/openvino-rpi**.

The cross-build path is the release-engineering path. Use it when you need Linux AArch64 OpenVINO artefacts for Raspberry Pi, but do not want every build to happen on the board itself. It's suitable for CI, reproducible releases, and desktop-based contributors. See the OpenVINO cross-compilation guide: **rpimag.co/openvino-cc**.



▲ OpenVINO is used to optimise AI models for different hardware platforms

Details that make deployment reliable

The visible part of edge AI is the model. The part that determines whether a deployment remains reliable is often the runtime and packaging work around the model.

Check the architecture first. If `uname -m` does not return `aarch64`, the default assumptions in this article do not apply.

Check the Python version. OpenVINO 2026.2.0 was released in June 2026. It requires Python 3.10 or newer, and its Linux Arm64 wheels are published for CPython 3.10, 3.11, 3.12, and 3.13 with the `manylinux_2_35_aarch64` tag. Check the current PyPI wheel list before choosing a Python version for a fresh image. In practice, wheel install failures on Raspberry Pi are often about user-space age or Python version rather than OpenVINO not supporting the board.

Keep startup work out of the hot path. OpenVINO can load ONNX models directly, but IR plus model caching is often better for services. Convert the model once, package the converted model, and let the application startup focus on loading and compiling for the target device.

Be explicit about the device. On Raspberry Pi, use `CPU` when the CPU is the target, and use `AUTO` for deliberate multi-device portability. For a fixed Raspberry Pi deployment, explicit CPU selection makes behaviour easier to verify.

Treat power and cooling as runtime dependencies. Sustained inference, camera capture, storage, networking, and logging can all run at the same time on a small board. Raspberry Pi 5 especially should be designed with cooling and power headroom in mind.

Build the application around the OpenVINO deployment model

Go forth and build

Raspberry Pi remains the concrete edge target: it's small, familiar, and easy to place near sensors. YOLO26 remains the concrete model example: it's visual, recognisable, and easy to export. The technical focus is OpenVINO itself.

The same runtime model, conversion flow, CPU selection, caching behaviour, and packaging choices can support a Raspberry Pi deployment without turning it into a platform-specific exception. Build the application around the OpenVINO deployment model: clean environment, converted model, explicit device, persistent cache, and a packaging path that matches the release process.

▼ A checklist prior to deploying on Raspberry Pi hardware

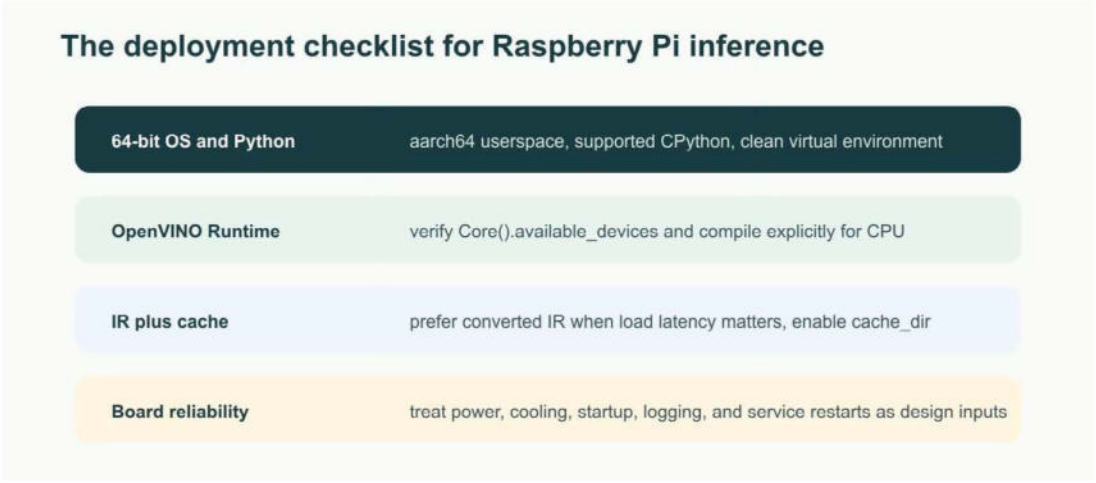
Contribute to OpenVINO and Ultralytics

OpenVINO and Ultralytics are open-source projects, and contributions from the community directly improve the ecosystem for everyone, including Raspberry Pi users. If this article helped you deploy a model on a Raspberry Pi, please consider contributing to OpenVINO and Ultralytics, whether that's by fixing a bug you found, improving the documentation, or implementing a missing operation for the Arm backend.

There are several ways to get involved:

- **Pick a good first issue:** the 'Good first issues' board tracks tasks designed for newcomers; Arm-specific issues are labelled with 'arm': **rpimag.co/openvino-first**.
- **Report issues from real deployments:** if you encounter a problem with OpenVINO, open an issue: **rpimag.co/openvino-issues**.
- **Join the community:** OpenVINO's GitHub Discussions page is the place to go for design questions and feature proposals (**rpimag.co/openvino-discuss**). Curious about Ultralytics? Discover their licensing options to bring computer vision solutions to your projects. Visit their GitHub repository (**rpimag.co/ultralytics-git**) and join the community (**community.ultralytics.com**).

The CPU plug-in source that runs on Raspberry Pi lives in **src/plugins/intel_cpu**. Every improvement there – a new JIT emitter, a performance fix, better test coverage for Arm, etc. – directly benefits the Raspberry Pi deployment path described in this article.



Custard Cream Camera

A Raspberry Pi-powered instant printing camera that looks a bit like a custard cream biscuit and responds to spoken commands to transform your pictures



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

robmiles.com

QUICK TIP

Any Raspberry Pi Camera Module can be used.

Instant cameras are a popular way to enjoy photography. But they usually have plastic lenses, are expensive to feed, and produce tiny pictures. The Custard Cream Camera uses a Raspberry Pi High Quality Camera module fitted with a glass closed-circuit television (CCTV) zoom lens and a dye-sublimation printer to create vibrant 6 × 4-inch prints that don't break the bank.

The Custard Cream Camera also uses the latest AI-based photo editing technology to turn photos into cartoons, line drawings, carvings, or even a Japanese print. The camera can listen to your spoken commands and send them off with the image to an AI-powered image editor. In this article we'll discover how the camera is built and how the software inside captures images, responds to user input, converts speech to text, and accesses the AI service. If you fancy building a camera the source code and hardware designs are here: **rpimag.co/custardcream**. You can see pictures taken by, and processed using the camera at **custardcream.cam**.



▶ **Figure 1:** Running the connecting cables outside the box and back in again saves space and makes the design a lot more flexible

Figure 1 shows the Custard Cream Camera. In the bottom box is a Canon Selphy printer mechanism. The middle box houses a printer battery and a Raspberry Pi HQ camera with a zoom lens. The top box contains a Raspberry Pi 4, an uninterruptible power supply (UPS) module, and a 5-inch touchscreen.

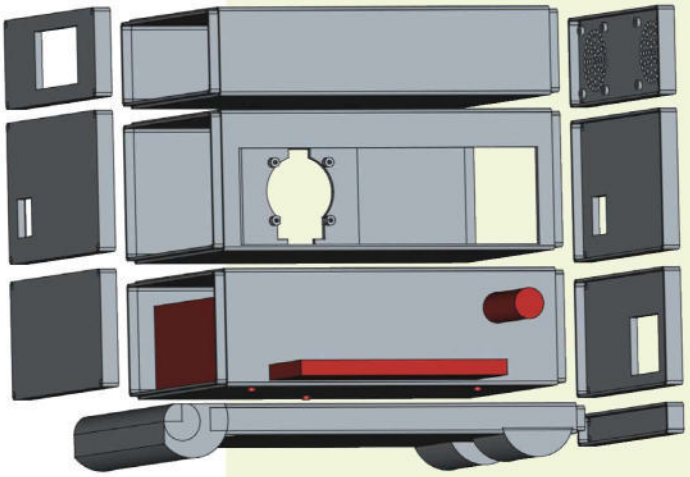
Boxing clever

Figure 2 shows how the camera is constructed. It is made up of four tubes which are held together by M3 bolts. The base provides a handle and feet on which the camera stands. All the boxes have friction-fit caps for each end. Some of the caps have holes in them for cable connections. The battery for the printer is fitted into the middle section. The red element in the figure is not printed; it is a cutout in the shape of a Selphy printer that is cut from the printer box to provide the mounting holes, the hole for the power supply, and the slots for the paper.



Figure 3 shows how the box segments were printed. The tree support worked very well, even printing mounting pillars on the inside of the box.

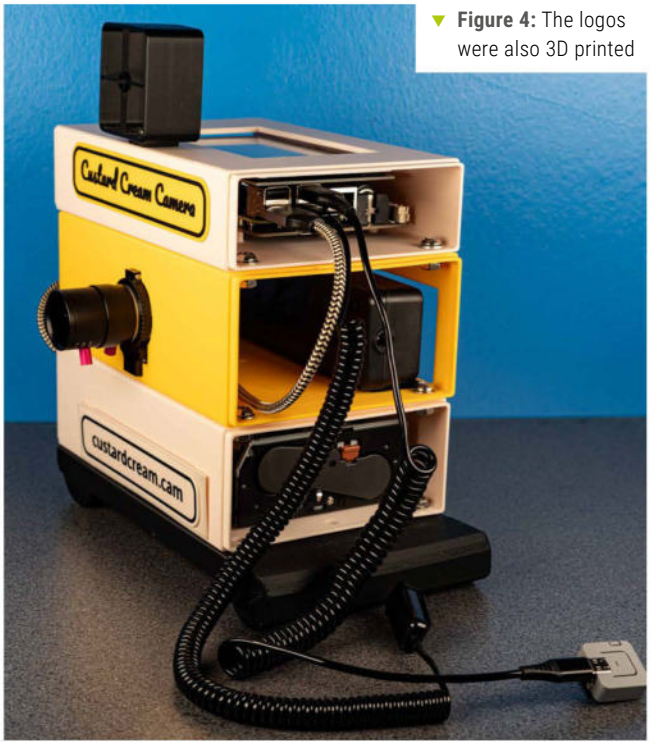
Figure 4 shows how the components are fitted into the case. The removable ends provide access to the printer to change the ribbon cartridge. The curly cables connect a USB microphone and an M5Stack Atom device which can be used as a remote control for the camera shutter. Now we know how the camera fits together, let's look at what makes it work.



◀ **Figure 3:** The support came away cleanly from the box

▲ **Figure 2:** You can omit the bottom box if you don't want to use a printer

Uses the latest AI-based photo editing technology



▼ **Figure 4:** The logos were also 3D printed

YOU'LL NEED

- Raspberry Pi 4
- Rechargeable battery pack or uninterruptible power supply - **rpimag.co/upsx703**
- Raspberry Pi Camera Module
- Canon Selphy CP400 printer - **rpimag.co/cp400**
- B CP2L Battery + Charger Adapter
- 16 GB microSD card
- USB cable for the printer
- USB microphone for voice commands
- Case and grip - **rpimag.co/custardcream**

DOWNLOAD THE FULL CODE:

rpimag.co/custardcream

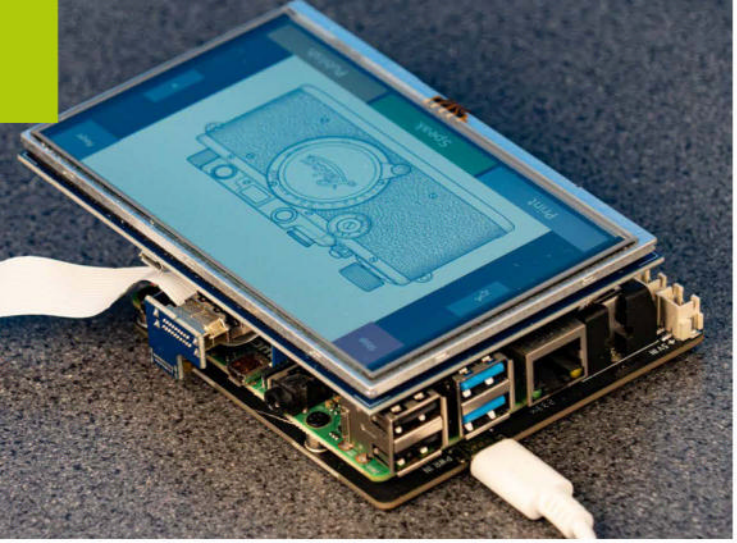


Figure 5: The device is powered via the power input on the UPS board



Figure 6: The viewfinder is shown at a lower resolution than the captured picture

Brains of the operation

Figure 5 shows the brains of the camera. Raspberry Pi 4 is plugged into an uninterruptible power supply (UPS) base, and the 5-inch LCD panel is plugged into Raspberry Pi. The panel is connected to Raspberry Pi via an HDMI jumper. This makes for a very compact unit which could be used as the basis of a much smaller camera.

The software that controls the camera is written in Python. It is constructed as a set of cooperating software objects.

```
def main():
    custard_cream_camera = CustardCreamCamera()
    custard_cream_camera.run()
```

The code above shows the `main` function, which is called to start the camera running. It creates an instance of the `CustardCreamCamera` class and then calls the `run` method on that instance. The `run` method repeatedly updates the screen, drawing the preview output of the camera if the viewfinder is being used, or displaying other menus and waiting for user input. When the software starts running, it first displays the camera viewfinder.

Taking a view

Figure 6 shows the camera viewfinder display for the Custard Cream Camera. The display is repeatedly updated with the view from the camera so that the framing, focus, and exposure can be adjusted and previewed. The Picamera2 library is used to control the camera. The `camera` device is created and configured when the `CustardCreamCamera` instance is created. The statement below creates a `Picamera2` instance and sets the `picam2` member of the `CustardCreamCamera` class to refer to it. The program will send commands to the camera by calling methods on `picam2`.

```
self.picam2 = Picamera2()
```

QUICK TIP

An HDMI screen is preferable to one driven via Raspberry Pi's GPIO header because it needs to be updated quickly when the camera viewfinder is being displayed.

You can think of `picam2` as an assistant who knows a lot about taking pictures. You give the assistant commands to tell it what to do. Now that we have a camera, the next thing to do is configure it. Just like an assistant would need to know how to make your coffee, the camera needs to be told the resolution of the pictures to capture, along with their format and any transformations that need to be applied. This is done using Python dictionaries which hold named setting values. To see how this works, look at the code below, which creates the preview configuration for the camera viewfinder. This is not the complete call, but it allows us to understand how the process works.

```
self.preview_config =
self.picam2.create_preview_configuration(
   main={"size": (640,480), "format": "BGR888"},
   ...
)
```

The `picam2` object we created has a method called `create_preview_configuration` that builds and returns a configuration object. This is a bit like your assistant giving you a form to fill in that describes your perfect cup of coffee. You fill in the form and hand it back to them. In the case of the camera, you 'fill in the form' by setting arguments to the call that

QUICK TIP

If you are just getting started with the camera, there are built-in configurations with which you can start.

describe the image format you want. The main argument sets the size of the image (in this case, 640 × 480 pixels) and the format (in this case Blue, Green, and Red 8-bit values). There are other arguments that can be provided to specify transformations and controls to be applied to the image. Once we have a configuration object, we can use it to configure the camera:

```
self.picam2.configure(self.preview_config)
```

The statement above sets the camera up with the preview configuration. The software also creates a `still_config` configuration for use when a picture is taken. This produces a much higher-quality image. We will use that later. Now that we have our camera created and configured, the next thing to do is start getting images from it.

Frame grabbing

The software asks the camera to fetch the image in the background. This lets the software respond quickly to user input, rather than being stuck waiting for camera data. An event is set by the camera to indicate when an image frame is ready. The event is created in the constructor of the `CustardCreamCamera` class:

```
self.frame_ready = Event()
```

The statement above creates an event called `frame_ready`. Let's see how the software uses it in the function that requests the next frame from the camera:

```
def request_next_frame(self):
    self.pending_job =
self.picam2.capture_request(
        wait=False,
        signal_function=lambda job:
self.frame_ready.set()
    )
```

The `request_next_frame` function calls the `capture_request` method and passes it two arguments: `wait` and `signal_function`. The value of `wait` is provided as `False` so the `capture_request` function will return immediately, rather than making the calling function wait until the capture is complete. The `signal_function` value is supplied as a lambda function (a function with no name) that contains a single statement that sets the `frame_ready` event. The `capture_request` function returns a reference to an object that describes this request. This reference is stored in a variable called `pending_job`.

QUICK TIP

All the configuration options for the camera are set in the **settings.json** file.

If this seems a bit confusing, let's step back and consider what we are trying to do. We don't want the program to stop while the camera fetches an image, just like we don't want to stand and wait while our assistant makes us a cup of coffee. So, we tell the camera to work in the background by setting `wait` to `False`. But now we need two things: a way of getting hold of the picture when it has been fetched, and a way that the camera can tell our program when it has finished fetching the frame. We store the request in `pending_job` (so we can use it later) and we set `signal_function` to code that sets the `frame_ready` event, so that we can monitor the event and process a frame when it arrives. The `frame_ready` event is checked regularly to see if it has been set. If the event is set, the software fetches the image from the request, draws it on the viewfinder, and asks for another frame.

```
if self.frame_ready.is_set():
    self.frame_ready.clear()
    request = self.picam2.wait(self.pending_job)
    frame = request.make_array("main")
    request.release()

    self.finder = Image.fromarray(frame, "RGB")
    self.request_next_frame()
```

The code above checks the `frame_ready` event. If the event is set, it clears the event and then asks the camera for the request matching the `pending_job` request that we stored when we created the request. The method that is called to get the request is called `wait`, but it will return instantly because the frame is ready (that's what triggered the event). The code then gets the frame out of the request, releases the request, sets the finder to the new image, and then requests the next frame. This ensures that the viewfinder is continuously updated.

Variables vs events – fight!

Programs use variables to remember things. They put values into them and get them out later. However, variables are not designed for communication between concurrently executing processes. One process may read a variable while another is in the middle of changing it, which can lead to unpredictable behaviour. An event provides a safe way for one process to signal another that something has happened. The Python runtime system makes sure that only one process at a time can interact with an event.

Happy snapping

The Custard Cream Camera provides a variety of different shutter triggers. You can press the 'Click' button on the screen, use a Bluetooth camera remote, a remote serial device, or a wired button. When the shutter is triggered, the software must select a different camera configuration to capture a high-quality still image to be saved. Below, you can see the code that does this.

```
self.picam2.switch_mode_and_capture_file(
  self.still_config,
  str(self.save_file_name))
```

The camera variable `picam2` (which represents the camera we are using) contains a method called `switch_mode_and_capture_file`. This method is given a configuration and a file name when it is called. It takes a picture using the supplied configuration and then stores it as a file with the specified name. Then the camera flips back to the original mode. The `save_file_name` string is constructed from the date and time when the picture was taken. Now let's look at how the camera user interface works.

On the menu

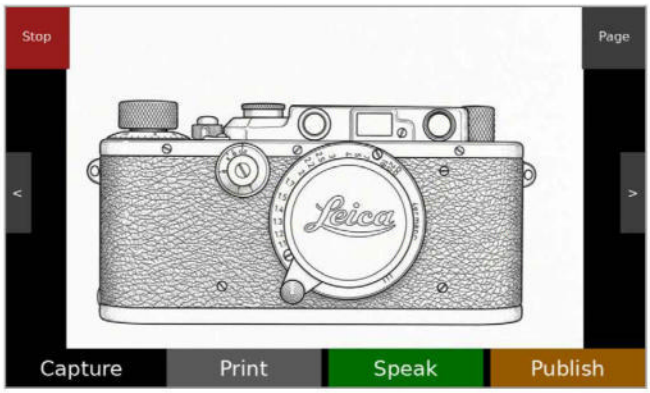
If you look back to **Figure 6**, you'll notice that as well as the viewfinder image, the screen contains several buttons. There are buttons you use to lighten and darken the image by changing the Exposure Value (EV), along with Click and Play buttons. Pressing Click takes a picture. Pressing Play switches to the Play page.

Figure 7 shows the Play page where users view pictures they have taken and the results of AI processing. There are several buttons on this page. Pressing Capture returns you to the viewfinder; Print starts a print of the image. If you press and hold Speak, you can give spoken instructions for an AI edit. If you press Publish, you open a page from which you can publish pictures to Flickr, Bluesky, and the Custard Cream Server. There are also buttons to move forward and back through pictures, display a Page of pictures, and Stop the program and return to the Raspberry Pi desktop. Let's look at how the buttons work.

Button pusher

The buttons are generated within the camera software. The `Button` class is defined in the **widgets.py** file in the **lib/display** folder. Each displayed button is represented by an instance of the `Button` class. The `Button` instance contains two Python functions. One is called when the button is pressed and the other when the button is released.

We can see how this works by looking at the implementation of the Stop button in the top left-hand corner of the Play screen shown in **Figure 7**. This button stops the camera software and



▲ **Figure 7:** The camera has turned the original picture into a line drawing

returns to the desktop. To create a button, we need to specify where the button is, how big it is, the text, font and colour to be used to display it and the functions to be called when the button is pressed and released. Each of these values is an argument to the `Button` constructor:

```
 Button( 0, 0, # position
         side_button_width, side_button_height,
# size
         "Stop", # text
         self.small_font, # font
         colours.BUTTON_TEXT, # text colour
         colours.DANGER, # background colour
         None,  # called when released
         self.quit_app # called when pressed
         ),
```

The comments (marked with a #) on the end of each line of the code above tell you what each argument is used for. References to all the buttons on a page are held in a `Menu` object which is given user input events. When the user causes an input event by left-clicking on the mouse or pressing the touchscreen, the program works through all the buttons in the `Menu`, giving each in turn the coordinates of the event. If a button decides that it has been clicked, the button function (in the above case `quit_app`) is called. A similar process is followed when a click is released so that buttons can react to being released. This is used by the Speak button so that the camera knows when the user has finished speaking. Let's have a look at how that works.

QUICK TIP

There is no handler for the Stop button being released because the program has stopped by then.

The listening camera

Figure 8 shows what happens when you press and hold the Speak button. The camera begins recording sound and converting it to text. The speech

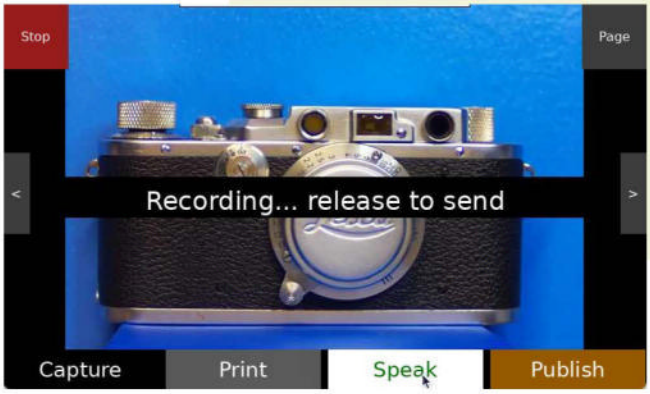


Figure 8: Recording continues as long as the Speak button is held down

conversion happens dynamically; the camera updates the message on the screen as it recognises more words. It uses the Vosk speech recognition toolkit (**alphacephei.com/vosk**), which works well on a Raspberry Pi 4. The Vosk software receives incoming audio and continuously updates the recognised text as more audio is received. The speech decoder runs in the background in the same way as the camera fetches images while the program is running. Speech decoding is implemented as a module which is bound to the Speak button. The decoder is managed by a wrapper class which is part of the Custard Cream Camera software. Using a wrapper class makes it easy to switch between decoders. You can find the decoders in the transcription folder in the code for the application. When the Speak button is pressed, the `start_voice_prompt` method is called:

```
 def start_voice_prompt(self):
     try:
         self.transcriber.start(
on_partial=self._on_voice_partial)
     except Exception as e:
         self.show_result(self.status_frame(
"Mic error"), hold_seconds=2)
         return
     self.voice_recording = True
     self.voice_partial_text = None
     self.screen.draw(self.status_frame(
"Recording... release to send"))
```

The method first calls the `start` method on the transcriber which was created when the program started. If this attempt fails, the `start` method will throw an exception which is caught and displayed for the user for two seconds. If the `start` method succeeds the program sets the `voice_recording` flag to `True`, the `voice_partial_text` to `None` (we haven't received any yet) and draws a status message showing that voice recording is now active (as seen in **Figure 8**).

When the transcriber is started, the `on_partial` argument is set to a method that is called when the voice recogniser has some new data. In the camera, this method is called `_on_voice_partial`. The job of this method is to update the partial display on the screen.

```
def _on_voice_partial(self, text):
    self.voice_partial_text = text
    self.voice_partial_ready.set()
```

When `_on_voice_partial` is called, it is passed the latest text from the decoder. It copies this text into `voice_partial_text` and sets an event to tell the software that new partial text is available for display. The `run` loop of the camera checks this event and updates the display when it is set:

```
if self.voice_partial_ready.is_set():
    self.voice_partial_ready.clear()
    dirty = True
```

Above is the code that runs when the partial text is to be displayed. It clears the `voice_partial_ready` event and then sets the display `dirty` flag to `True`. This will cause the display to update, drawing the new partial text. When the user releases the Speak button, the recognised text is displayed and can be edited before being sent to the AI.

Figure 9 shows how a message is edited. The keyboard is made up of `Button` objects which add the appropriate key to the message string when pressed. When the command string has been entered, it is passed over to the image processing, so let's look at that next.

QUICK TIP

The camera will only ever listen to your voice while the Speak button is held down.



Figure 9: Commands can be long and complex. The AI will even generate cartoon strips if you ask it to

Going Nano bananas

The camera can send images to the Google Nano Banana service for processing. The image is sent to the service endpoint, along with text commands. The service will reply with an updated image. This is a paid-for service. You sign up at **aistudio.google.com**, buy some credit, and obtain an application programming interface (API) key that authenticates your requests. The key is stored inside an environment variable on Raspberry Pi and used to create a `Client` object that will perform transformations. The software for the camera contains a `CustardCreamClient` class which manages interaction with the AI.

```
 def __init__(self,
      api_key_env="GOOGLE_API_KEY",
      edit_model="gemini-2.5-flash-image",
timeout_seconds=30):
    api_key = os.environ.get(api_key_env)
    self.client = genai.Client(api_key=api_key,
       http_options=types.HttpOptions(
timeout=timeout_seconds * 1000))
```

The `__init__` method above is part of the constructor for the `CustardCreamClient` instance which manages the connection to the Nano Banana service. It creates a member of the class called `client`. The `client` provides a handle to a service in the same way as `picam2` provides a handle to the camera. When the `client` is created, we set a `timeout` value so that failed requests don't cause the program to get stuck. When we want to process an image, we send the `client` the bytes of the image along with a string of text describing the edit to be performed:

```
 def edit_image(self, image_bytes, prompt_text,
mime_type="image/jpeg"):
   response = self.client.models.generate_
content(
       model=self.edit_model,
       contents=[
           prompt_text,
           types.Part.from_bytes(data=image_
bytes, mime_type=mime_type),
       ],
  )
```

The response from the call of `generate_content` contains status information and a new image which is stored in the camera and then displayed. Each use of the service costs a few pence.

You can get detailed reports of your usage, including the success rate of your requests. In **Figure 10** you can see one of the reports showing the amount of use and the number of failed requests. The total cost of the activity was around £8. The camera will work perfectly well without using AI support, but it is great fun to play with, and people love getting printed copies of their modified portraits.

Playing with pictures

The camera is a complex build, but it is definitely worth the effort. The author has had great fun taking it to parties, inviting people to suggest image modifications, and then printing the results. It also incorporates techniques that you might want to put into other things you build, particularly voice response. The repository for the project contains detailed setup instructions to make it as easy as possible for you to get started. You may be wondering how the camera publishes pictures, and what the 'Custard Cream Server' does. You can find out more in a future article.

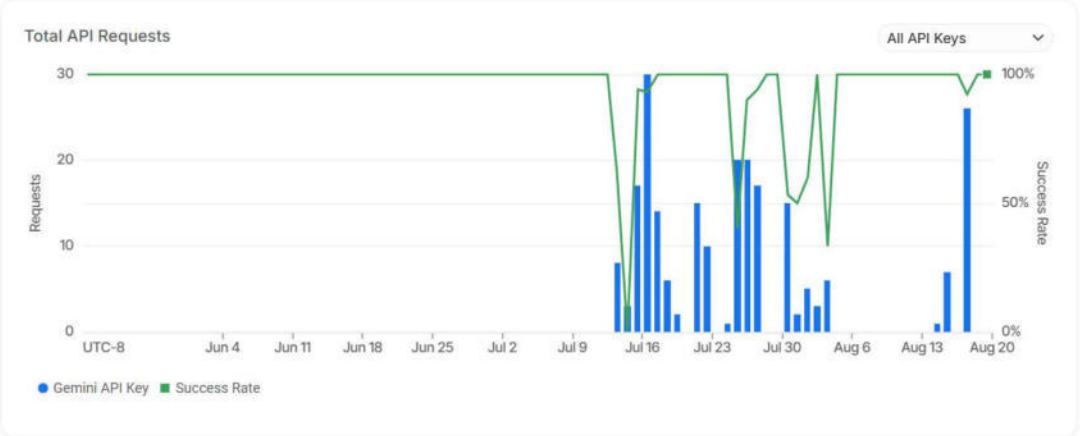


Figure 10: The camera got a lot of use in mid-July at the Electromagnetic Field Camp

QUICK TIP

To prevent inappropriate use, Nano Banana has built-in guard-rails.

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

Upcycle a cheap colour printer

Create a colour printing solution for your home using the smallest Raspberry Pi computer



Maker

Rob Miles

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

robmiles.com

QUICK TIP

The server will also work with many other printers that have USB connections.

Figure 1 shows a print of the printer printing a print of the printer printing a print. If you've ever fancied making your own high-quality colour prints but baulked at the price of a new printer, this article will show you how to use Raspberry Pi to turn an old device into something that can be readily used from your home computers and mobile phones. You can find full instructions for creating and configuring a print server, along with the printer monitoring application and 3D printable case designs, at **rpimag.co/PrinterServer**.



▶ **Figure 1:** The print server is on the right of the picture



Figure 2: The used ribbon retains the images that were printed

YOU'LL NEED

- A cheap colour printer. The project was tested with a Canon Selphy CP400 model
- Raspberry Pi (from Raspberry Pi Zero W upwards)
- microSD card – at least 4GB
- USB cable. A USB-A to USB-B cable will work with most Raspberry Pi models.
- Power supply

Pictures to dye for

Home colour printing has been with us for many years, the most popular devices being inkjet printers that form images by squirting highly expensive ink. But they are not the only way to print pictures. A 'dye-sublimation' printer evaporates coloured wax ink from a plastic film onto paper. The results are bright, detailed, and especially long-lasting.

QUICK TIP

If you happen to have a 'retro' digital camera from the early 2000s, you may be able to connect it directly to an old Canon Selphy and make prints. Look for 'PictBridge' in the camera play menu.

Figure 2 shows the ink ribbon inside a Canon Selphy cartridge. It is good for 36 prints and costs around £12 with paper. You can send used ribbons back to Canon for recycling. Canon launched its 'Selphy' range of dye-sublimation printers over 20 years ago. The first ones were plugged into computers or directly connected to a camera. Later versions added screens, memory card sockets, and wireless connectivity. You can still buy brand new Selphy printers today. They use the same print mechanism as the original devices, with ink cartridges and paper still readily available.

QUICK TIP

You can get battery packs that make the printers totally portable.

Figure 3 shows the internals of a Selphy printer. It is easy to remove them from the case and install in another device if you want to add a printer to something you are building.

CUPS runneth over

The key to making everything work is CUPS (the Common UNIX Printing System). This is the standard printing system used by Linux and many Unix-like operating systems. Clients use the Internet Printing Protocol (IPP) to communicate with local and network printers, allowing applications to print without printer-specific drivers. CUPS also supports printer sharing, print queues, driver management, and technologies such as AirPrint.

Turn an old device into something that can be used

Figure 3: The printer makes four passes over the picture for yellow, magenta, and cyan, and a clear coating



You make a printer available for use over a network by plugging it into a computer running a CUPS printer server. Raspberry Pi is perfect for this role. Even one of the lowest-powered Raspberry Pi models, the original Raspberry Pi Zero W, can host a printer and provide a low-cost way of putting a printer on your home network. All you have to do is install and configure CUPS and a couple of other packages.

Gutenprinting

To get a printer to work, you don't just need CUPS; you also need something to drive the printer. You can think of CUPS as the 'front office' of the printing process. It provides services, negotiates with customers, and accepts print jobs. However, the printing process also needs something that talks to the printer to turn jobs into printed output. On Raspberry Pi we can use Gutenprint.

Gutenprint is a collection of printer drivers that connects to CUPS, translating generic print instructions into the printer-specific commands needed by devices such as the Canon Selphy CP400. You might think that all we have to do is install Gutenprint and our printer will just work. Most of the time this is true, but unfortunately for the author's chosen printer this is not the case. For reasons beyond the understanding of the author, the most recent version of Gutenprint is broken for the Selphy CP400. To make the printer work, he had to download and build an earlier version of the Gutenprint framework. It takes a while, but full instructions are provided in the repository for this project. Now that we know what we have to do, let's go through the steps to actually do it.

QUICK TIP

Gutenprint is named in honour of Johannes Gutenberg, inventor of the movable-type printing press.

Printer admin power

The starting point for the project is a Raspberry Pi device running the latest version of the operating system. The author installed Debian Trixie on his Raspberry Pi Zero W using the Raspberry Pi Imager to build the image. He then connected to a console running on that machine. If you want to administer printers (and you do), you need to add your user to the `lpadmin` group on the machine. You do this with the following terminal command:

```
$ sudo usermod -aG lpadmin $USER
```

You only need to issue this command once. You will have to log out and log back in again (or reboot the machine) for the change to take effect. Now you can create and manage printers.

Up for the CUPS

You might find that CUPS is already installed on Raspberry Pi. But to make sure it is available, you can use the following commands:

```
$ sudo apt update
$ sudo apt install cups
```

When the above commands have completed, you have CUPS on your machine. Now you need to get it going:

```
$ sudo systemctl enable cups
$ sudo systemctl start cups
```

The commands above enable CUPS and start it. Once CUPS has been enabled and started, it will run automatically each time the machine reboots. You can check if CUPS is running by issuing the following command:

```
$ systemctl status cups
```

If CUPS is running, you will get a description of what the `cups` service is up to (which will be not much at the moment). Now we can install Gutenprint.

Getting Gutenprint

Gutenprint contains a huge number of drivers for different printers which are all downloaded as part of the package. We install it in the same way we installed CUPS.

```
$ sudo apt install printer-driver-gutenprint
```

This install might take a while, but when it has finished, we can install our printer.

The first step in the installation process is to power up your printer and plug it into a USB port on Raspberry Pi. If you are using a Raspberry Pi Zero, you can use a USB OTG (On-The-Go) cable to connect the printer directly to the single USB port on Raspberry Pi. Now we can ask our Raspberry Pi to tell us what printers it can see:

```
$ lpinfo -v
```

This command will produce quite a bit of output, particularly if you already have printers on your local network. The line you are looking for is something like this:

```
direct gutenprint53+usb://canon-cp400/NONE_UNKNOWN
```

This line indicates that a Canon CP400 printer has been detected on USB. If you have plugged a different printer into your Raspberry Pi, you will see a different name. Now we know the printer exists, we next need to find the precise name of the driver for it. We do this by searching through all the available printers to find one with the name of the printer we are adding:

```
$ lpinfo -m | grep -i canon-cp400
```

The above command gets a list of all the printers and filters out everything but the `canon-cp400` (the name we saw in the previous response). This will produce a single line displaying the driver information for the printer. If you are adding a different printer, change `canon-cp400` in the above command to the name of your printer.

```
gutenprint.5.3://canon-cp400/expert Canon SELPHY
CP400 - CUPS+Gutenprint v5.3.3
```

With all the info we need to create the printer, let's do that:

```
$ sudo lpadmin \
    -p SELPHY_CP400 \
    -E \
    -v "gutenprint53+usb://canon-cp400/NONE_
UNKNOWN" \
    -m "gutenprint.5.3://canon-cp400/expert"
```

The command above is spread over several lines. Information from the previous two status commands has been inserted into it. If you are using a different printer, you will have to edit this command appropriately. Note that the `-p` option sets the name that the printer will have. This will end up being the name of the printer when it is made visible on the network. Now that we have our printer, the next thing to do is enable it:

```
$ sudo cupsenable SELPHY_CP400
$ sudo cupsaccept SELPHY_CP400
```

These two commands start the printer and tell CUPS to accept jobs for it. If you have given your printer a different name, use that instead. The final thing that we need to do is make the printer the default printer for the device:

```
$ sudo lpadmin -d SELPHY_CP400
```

Now any print job which is not addressed to a specific printer on Raspberry Pi will be set to the Canon Selphy. If all you want is to print things from Raspberry Pi, you can stop now. But if you want to use your Raspberry Pi from other devices, the final thing that you need to do is make the shared printer visible on your local network.

Make the shared printer visible on your local network

Getting noticed

The names of websites – such as the world-famous **www.robmiles.com** – are stored in globally accessible databases that map their 'logical' (i.e. human-centric) names into physical (i.e. network-centric) addresses. If we want devices on a local network to be able to find printers from their logical names (for example, SELPHY_CP400) we need a local service to do this. Avahi is an open-source implementation of protocols that allows a server to advertise services. We need to download Avahi and get it running on our Raspberry Pi. We do this in the same way as we did with CUPS.

```
$ sudo apt update
$ sudo apt install avahi-daemon avahi-utils
$ sudo systemctl enable avahi-daemon
$ sudo systemctl start avahi-daemon
```

The above instructions download, install, and start Avahi. Now we can share our printer:

```
$ sudo lpadmin -p SELPHY_CP400 -o printer-is-
shared=true
```

Replace `SELPHY_CP400` if you have given your printer a different name. We can check the printer status to make sure that it is shared:

```
$ lpstat -l -p SELPHY_CP400
```

This will give you quite a lot of output. Look for `printer-is-shared=true` in the text. The final thing we do is allow CUPS to publish shared printers:

```
$ sudo cupsctl --share-printers
$ sudo systemctl restart cups
$ sudo systemctl restart avahi-daemon
```



Figure 4: The lights are controlled by the GPIO 21 pin (D21)

Those statements turn on sharing and restart the services. At this point you should be able to browse for your printer on an iPhone or a PC. If you want to check that your printer is being shared, you can use the following command:

```
$ avahi-browse -rt _ipp._tcp
```

This will produce a lot of output, which is quite hard to understand, but it is nice to know that it is there.

Monitoring CUPS

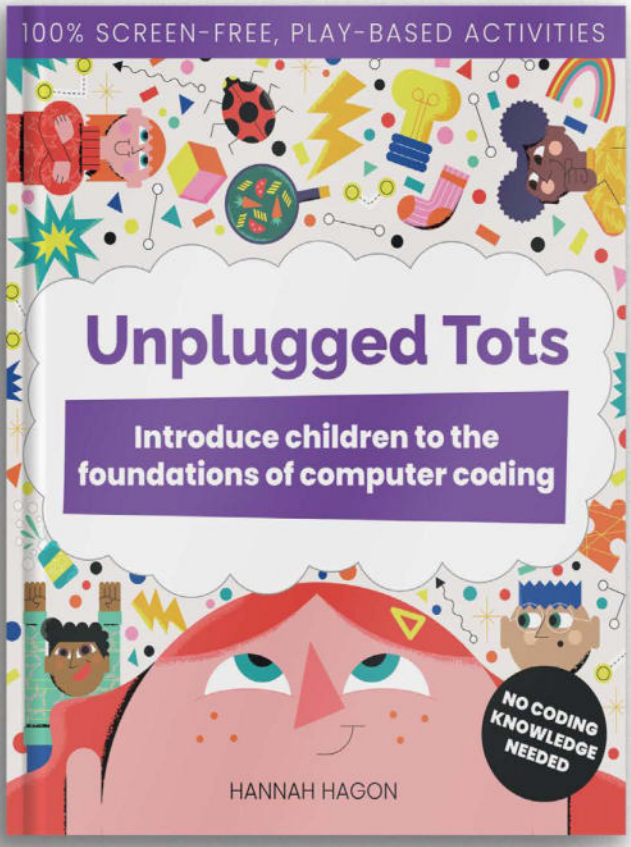
The CUPS framework is designed for high-end professional print services, supporting large numbers of printers and print jobs. It is highly configurable and has a rich management interface which is a bit of a pain to learn (in the author's experience). There are commands to stop and start the printer and recover the printer state if it runs out of paper or ink.

If you make the mistake of trying to print without having a paper cartridge in the printer (an easy mistake to make with the Selphy), CUPS will sometimes decide that the printer is broken. CUPS will then refuse to print anything until it has been told that the problem has been fixed and the printer queue can be restarted. This means opening a terminal or browsing a web page and giving a few commands, which rapidly gets tedious. The author put up with this for about a day before creating a program to monitor the printer status and reset it when things go wrong.

The printer monitor is a Python program which regularly issues printer status commands and reads the responses. If the printer becomes unavailable, the monitor will try to restart it. When the printer is restarted, any existing print jobs are discarded (so that retries don't create lots of copies that you don't want). The monitor program starts automatically when Raspberry Pi boots up.

The monitor can drive indicators to show the printer status. It can use LEDs or NeoPixels and shows when the printer is available, busy, or needs attention (**Figure 4**). There are full monitor installation instructions in the repository for this project. The program is written in Python and uses the pycups library that allows a Python program to interact with CUPS printers. The author has found the Raspberry Pi and printer combination a useful addition to his home network. It's nice to be able to make physical copies of photos.

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



rpimag.co/unpluggedtots

Part 02

Doll's house dec and tech

Textiles and tech: yin and yang



Maker

Nicola King

Nicola is a freelance writer and sub-editor. She's spent most of this month in front of a fan at her desk, crafting, playing with tech, and dreaming of rain.

@holtonhandmade

QUICK TIP

A modern style of décor is much easier to pull off in a doll's house than a period style. However, if you want a challenge...

In this second instalment of our doll's house project, we're going to craft a couple more items and of course, add some tech... because that's what this magazine is about. Here are just a few ideas to use as a jumping-off point, but what anyone does with their doll's house in terms of decoration, furnishing, and technical touches is limited only by their imagination.

First of all, let's consider some textile accents which will soften the look of the house and add some realism.

Curtain raising

So, our windows are pretty small, as they are squares each measuring 5cm by 5cm in size and the size of the curtains you make obviously depends on the size of your windows. We cut our cotton fabric into two rectangles, each measuring 7 cm by 8 cm for each window. That takes into account seams around each side, including a deeper seam at the top where we'll be threading our dowel through, and also gives some 'gathering' for each curtain.

Once cut, use an iron to fold over and press your three seams on the sides and bottom (0.5cm, or whatever your preference), ready for you to sew them into place by hand or with a sewing machine. We folded the fabric over more generously for the top seam, but adjust this based on the size of dowel you are using.

Let's consider some textile accents which will soften the look of the house and add some realism



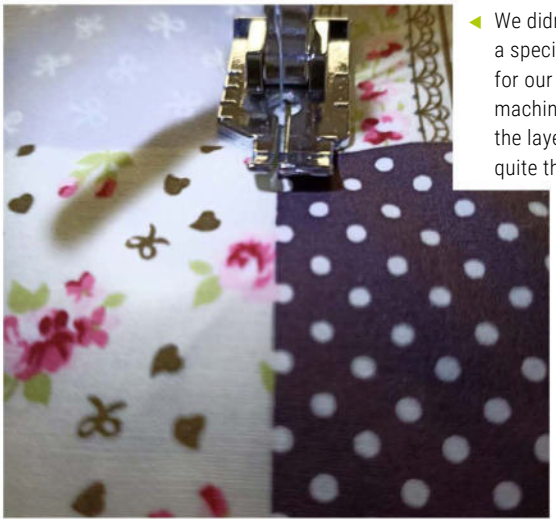
▲ It's coming together. We've added curtains, lighting, and some more furniture. You can see how this is an ongoing project

YOU'LL NEED

- Your doll's house
- Cotton or lace fabric (for curtains)
- Hardwood dowel (which you can cut to size – we used size 0.48 × 91.4cm)
- Small hacksaw
- Screw eyes
- Cotton fabric, wadding, and sewing thread for quilt
- Sewing needle or a sewing machine
- Tape measure or ruler
- Pencil
- Pins
- Iron and ironing board
- Fabric scissors or a rotary cutter
- Embellishments for quilt (optional)
- Pimoroni Tiny FX (**rpimag.co/tinyfx**)
- LED Dots for Tiny FX
- Infrared Receiver Stick + Remote Kit
- Battery pack

To fix the curtain pole, we found the easiest way was to use two 'screw eyes' (aka eye screws) which have loops to push the dowel through and are easy to attach to the wall. Pop an appropriate length of dowel through the top channel that you've created in the fabric (you can use tweezers to help pull it through if necessary) and your curtain is ready to hang. We then put a couple of stitches in each curtain to hold our gathers.

You can obviously do other things too in relation to framing a window. If you have some lace fabric, make some net (sheer) curtains, line your curtains, use a gauze to make a diaphanous type of curtain, or you can even make a Roman blind. We found that the more drape the fabric has, the better the curtains will hang.



◀ We didn't need a special foot for our sewing machine as the layers were quite thin

Warning!

Gluing and cutting

If you glue the seams, make sure your room is ventilated to get rid of any fumes. Also take the usual safety precautions (wearing gloves etc.) when cutting your dowel with a small hacksaw.

QUICK TIP

Try looking online to maybe find some second-hand doll's house furniture that you can upcycle and refresh.



Here is a small selection of vintage quilt pieces that we picked up in a second-hand store. Any of these could be made into a doll's house quilt

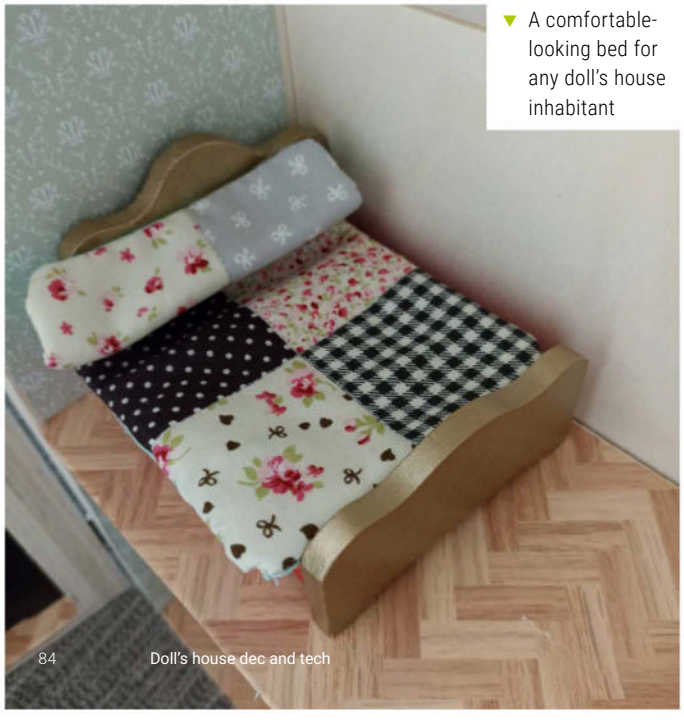
QUICK TIP

If the curtain fabric you choose already has a finished edge, use it – that's one less edge to sew.

Cover up

Now, if you have lots of time, you could make an ornate patchwork quilt for a doll's house bed, but we are just going to use two fabrics (one for the front and one for the back), and some wadding. The fabric we've chosen is actually faux patchwork, which makes it easy to cut and easy to quilt. The bed we're making the quilt for is obviously tiny, so your measurements will differ from ours, but we used fabric 15cm by 10cm, and bear in mind the finished item will be smaller than this once seams are sewn.

We used the lines on our fabric to quilt and attach the wadding to our main fabric, and then with right sides together, we sewed the backing to the main piece, leaving a small gap for turning it the right way round. We then hand-sewed the gap shut. You could include lace around the edges when you sew it together, or other embellishments should you wish.



A comfortable-looking bed for any doll's house inhabitant

Taking things further

We've only really scratched the surface here in terms of what you can do to bring a doll's house to life, with some crafting and some tech know-how. There are plenty more ways of adding your own unique personality, a couple of which we'll touch on in the final tutorial. The possibilities are literally endless. So, consider if you will:

1. Indulging in some woodwork. Our house has no doors, which is a bit remiss and, some would argue, a security risk. If your stash harbours a small piece of wood and some hinges, have a go at any wooden additions that the house would benefit from. Woodwork on a small scale may be fiddly, but it's hugely gratifying. Mini wood dowels could be added for window sills, for example. Remember too, that whatever you want to make, there's usually someone on YouTube who has already done it.

2. Using some off-cuts – whether that's spare carpet, wallpaper, curtain material, use whatever you have left over from your own home décor projects on your mini house.

3. Making some artwork for the walls… scaled down, of course! Downsize some images and put them in tiny frames, or use a fine-tipped brush to paint your own work of art. How about a mini portrait of your dog/cat/other pet?

4. Creating an outdoor space or garden with some real miniature plants to tend.

5. Adding seasonal decorations as required, such as a miniature Christmas tree or some tiny sewn Christmas stockings hanging from the mantelpiece.



We attached our dowel curtain pole to the wall with two screw eyes that fitted it perfectly

QUICK TIP

If you don't want to sew at all, you can just glue fabric seams instead.

Tiny lights

Let's now take a look at some tiny technical enrichment. In the previous part, we added some audio effects – for a barking dog and telephone ringing. This time, we thought we'd illuminate the room interiors with some lighting.

You could use a Raspberry Pi Pico and LEDs for this, but we opted again for a Pimoroni Tiny FX due to its dinky size. It's powered by the same RP2040 chip as Pico, so is easy to program in MicroPython. You can also easily connect up to six tiny LED 'dots' to it for your lights.

With the addition of an infrared remote (**rpimag.co/irremotekit**), you can even make them remote-controlled for a mini smart home! As in our 'Wearable beaded tech' tutorial (issue 168, **rpimag.co/168**), we plugged the remote's IR sensor stick into the Tiny FX's Sensor port (marked 'S'), then uploaded the Aye-Arr MicroPython library (**rpimag.co/ayearr**) to Tiny FX's libs folder via Thonny. We then used the **control_channels.py** code example – shown overleaf – from the GitHub repo (**rpimag.co/tinyirchannels**) to control the individual LEDs using the numbers on the remote. You can even alter their brightness by holding a number button.

As usual, you can make the program run automatically on bootup by saving it as **main.py**. We opted to power the Tiny FX using the Starter Kit's 3× AAA battery pack, but you could use a USB power bank or suitable LiPo battery.

We used sticky tack to adhere the LED dots and other electronics to ceilings and walls, but for a neater finish, you may well want to conceal them somehow (see the 'Hiding your wires and electronics' box). If you need longer wires for the LEDs, you could use expansion boards for Tiny FX (**rpimag.co/tinyfxexp**) or just manually splice extra sections of wire.

Next time, in our final instalment, we'll look at how to add more unique touches to our doll's house.

Hiding your wires and electronics

There's no question that doll's houses benefit tremendously from miniature electronics such as microcontroller boards, LEDs, speakers, and battery power packs. However, the downside of incorporating this tech is that we somehow need to hide the various wires and other accoutrements so they are far less obvious, and that's harder to do in a doll's house than in your actual house.

One way to do this would be to install a false ceiling or false walls, so that you can use the gap between the actual wall and false wall to hide your wires. While this is a very tidy approach, it is time-consuming and obviously cuts into the space in your doll's house.

Another option would be to run thin wires across a ceiling and then cover them with textured wallpaper (which helps to mask the bumps), or hide them behind, or run wires through, large pieces of furniture (as we've tried to do). Or, run them over the floor and then put in a 'hard wood' floor covering (some people have used ice-cream/popsicle sticks) over them.

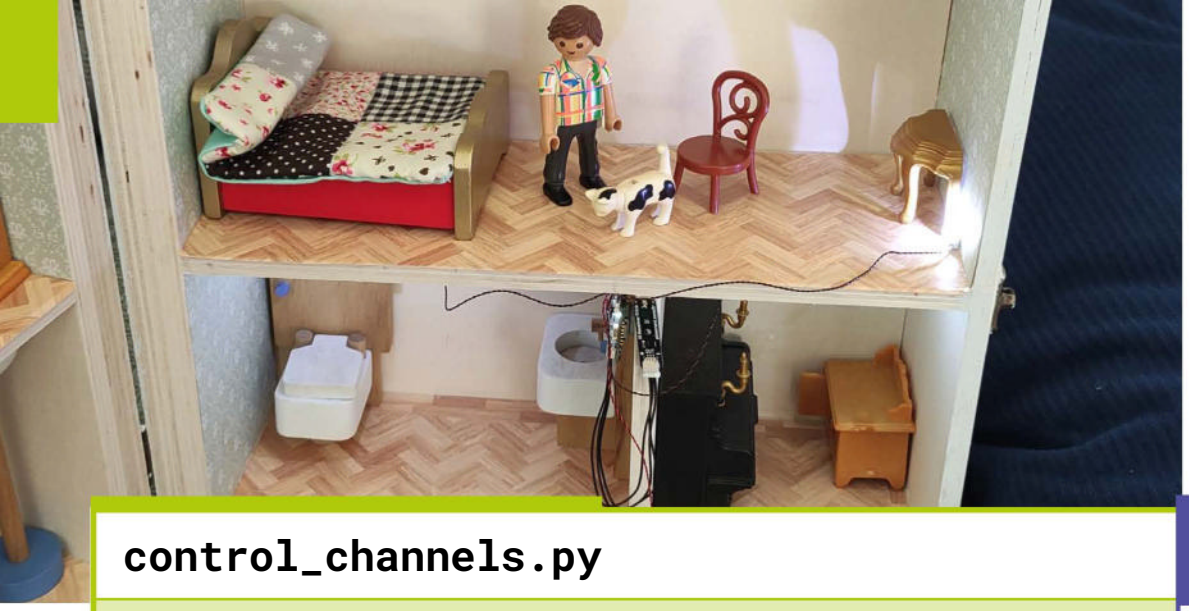
An idea that we really like is running the wires along the bottom edges of rooms, or running them through small holes drilled into corners of floors, and then adding some kind of skirting board to cover them. You can also cut V-shaped channel grooves in floors so that the wires are able to sit flush within them, and then you could cover them with carpet, rugs, etc.

QUICK TIP

If you can find some beads that fit your dowel, you can load one on at each end to give an authentic curtain pole feel.



The curtains are essentially two rectangles fitting the space. You can iron in a pleat if you want



We used tiny LED dots to light rooms and even added a remote control

control_channels.py

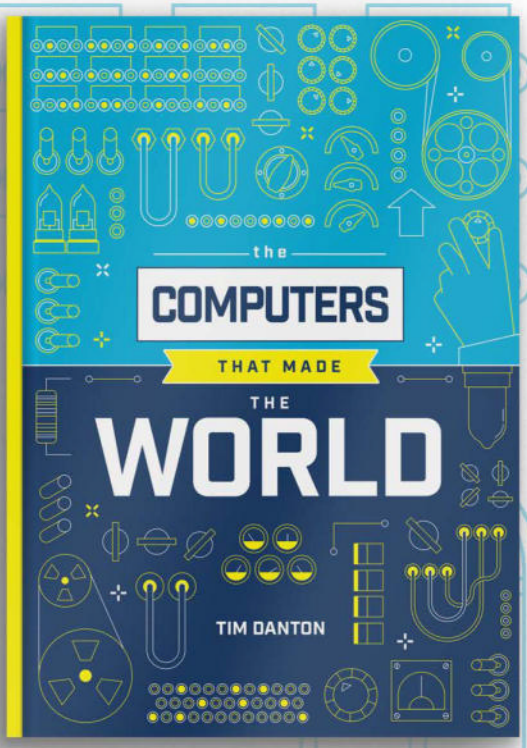
DOWNLOAD THE FULL CODE:

rpimag.co/github

> Language: **MicroPython**

```
001. from aye_arr.nec import NECRemoteReceiver
002. from aye_arr.nec.remotes import PimoroniRemote
003. from tiny_fx import TinyFX
004.
005. from picofx import MonoPlayer
006. from picofx.mono import StaticFX
007.
008. BRIGHTNESS = 1.0 # Initial brightness
009. BRIGHTNESS_STEP = 0.02 # For fading
010.
011. tiny = TinyFX()
012. player = MonoPlayer(tiny.outputs)
013.
014. # Create static LED effects
015. player.effects = [
016.     StaticFX(BRIGHTNESS),
017.     StaticFX(BRIGHTNESS),
018.     StaticFX(BRIGHTNESS),
019.     StaticFX(BRIGHTNESS),
020.     StaticFX(BRIGHTNESS),
021.     StaticFX(BRIGHTNESS),
022. ]
023.
024. # Toggle specified channel
025. def toggle_mono(channel):
026.     fx = player.effects[channel - 1]
027.     if fx is not None:
028.         fx.brightness = 0.0 if fx.brightness >
     0.0 else BRIGHTNESS
029.     print(f"Toggle Channel #{channel},
     Brightness: {fx.brightness:.2f}")
030.
031. # Fade specified channel
032. def fade_mono(channel):
033.     fx = player.effects[channel - 1]
034.     if fx is not None:
035.         fx.brightness = (fx.brightness +
     BRIGHTNESS_STEP) % 1.0
036.     print(f"Fade Channel #{channel},
     Brightness: {fx.brightness:.2f}")
037.
038. # Set up remote button functions
039. remote = PimoroniRemote()
040. remote.bind("1_RED", on_press=None, on_short=(
     toggle_mono, 1), on_repeat=(fade_mono, 1))
041. remote.bind("2_GREEN", on_press=None, on_short=(
     toggle_mono, 2), on_repeat=(fade_mono, 2))
042. remote.bind("3_BLUE", on_press=None, on_short=(
     toggle_mono, 3), on_repeat=(fade_mono, 3))
043. remote.bind("4_CYAN", on_press=None, on_short=(
     toggle_mono, 4), on_repeat=(fade_mono, 4))
044. remote.bind("5_MAGENTA", on_press=None, on_short=(
     toggle_mono, 5), on_repeat=(fade_mono, 5))
045. remote.bind("6_YELLOW", on_press=None, on_short=(
     toggle_mono, 6), on_repeat=(fade_mono, 6))
046.
047. # Set up receiver on sensor pin
048. receiver = NECRemoteReceiver(TinyFX.SENSOR_PIN,
     1, 0)
049. receiver.bind(remote)
050.
051.
052. try:
053.     receiver.start()
054.     player.start() # Start effects running
055.
056.     # Loop until effect stops or "Boot" pressed
057.     while not tiny.boot_pressed():
058.         # Decode IR pulses received
059.         receiver.decode()
060.
061. finally:
062.     receiver.stop()
063.     player.stop()
064.     tiny.shutdown()
```

The Computers that Made the World



How twelve pioneering computers built between 1939 and 1950 helped shape modern technology

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

Part 11

Build a GPIO Zero robot

Control DC motors with GPIO Zero and build a robot



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

DOWNLOAD THE FULL CODE:
rpimag.co/gpiobookgit

Raspberry Pi robotics is a popular pastime; you can scratch the surface with a basic robot that moves in a predetermined pattern, but you can also go much deeper and build fully autonomous bots. As with many other components, GPIO Zero makes it much simpler to build your own robots especially with its `Motor` and `Robot` classes. We'll take a quick look at those before showing you how you can extend GPIO Zero to build bots with components that aren't supported out-of-the-box.

Connect DC motors

Before we get into this tutorial's build, let's look at what GPIO Zero supports without modification. To enable the Raspberry Pi to control DC motors, you'll need some kind of motor driver board; H-bridge controllers like Texas Instruments' DRV8833 work quite well.

You must never connect motors directly to Raspberry Pi's GPIO pins, as this is guaranteed to destroy your Raspberry Pi unless you are lucky enough to have forgotten to wire it up to power. In a typical configuration, you'd connect a motor driver board to the appropriate GPIO pins and connect the motor's two wires to terminals provided on the driver board. The CamJam EduKit 3 is a fantastic all-in-one kit that includes a driver board, motors, a battery box, a distance sensor, and a line-following sensor. You reuse its packaging to build the robot, and best of all, the example code is based on GPIO Zero! For more information, see **rpimag.co/edukit3**.

Never connect motors directly to Raspberry Pi's GPIO pins

Run a motor

GPIO Zero includes a `Motor` class for running bidirectional motors connected via an H-bridge motor driver circuit. Here's an example of how you'd move a motor forward and backward with GPIO 8 connected to an H-bridge's forward input, and GPIO 7 connected to its backward input:

```
from gpiozero import Motor
from time import sleep
motor = Motor(forward=8, backward=7)
while True:
    motor.forward()
    sleep(5)
    motor.backward()
    sleep(5)
```

At the top, we import the `Motor` class from GPIO Zero, along with `sleep` from `time`. We then set the `motor` variable to a `Motor` class on the GPIO pins connected to our motor (GPIO 8 and 7). When we call `motor.forward`, the motor moves forward. Within the brackets, we can add a speed between 0 and 1 (the default). Similarly, `motor.backward` moves it backward, while `motor.stop` will stop a running motor.

Move a robot

While you can control your motors individually using the `Motor` class, GPIO Zero also includes the `Robot` class for controlling a two-wheeled robot. Here's how you could control a two-wheeled robot with the motor for the left wheel connected to GPIO 8 and 7, and the motor for the right connected to GPIO 10 and 9:

```
from gpiozero import Robot
from time import sleep
robot = Robot(left=(8, 7), right=(10, 9))
for i in range(4):
    robot.forward()
    sleep(1)
    robot.right()
    sleep(0.2)
```

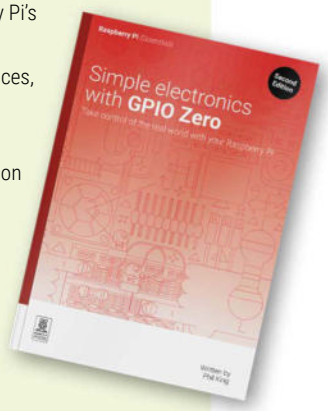
YOU'LL NEED

- 1× 3D-printed or off-the-shelf robot chassis and wheels, or some craft materials
- 1× HC-SR04 ultrasonic sensor
- 2× resistors: 1× 510Ω, 1 × 1kΩ
- 2× 28BYJ-48 stepper motors & ULN2003A driver boards
- 1× half-size solderless breadboard
- Mobile power bank
- Various jumper wires

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



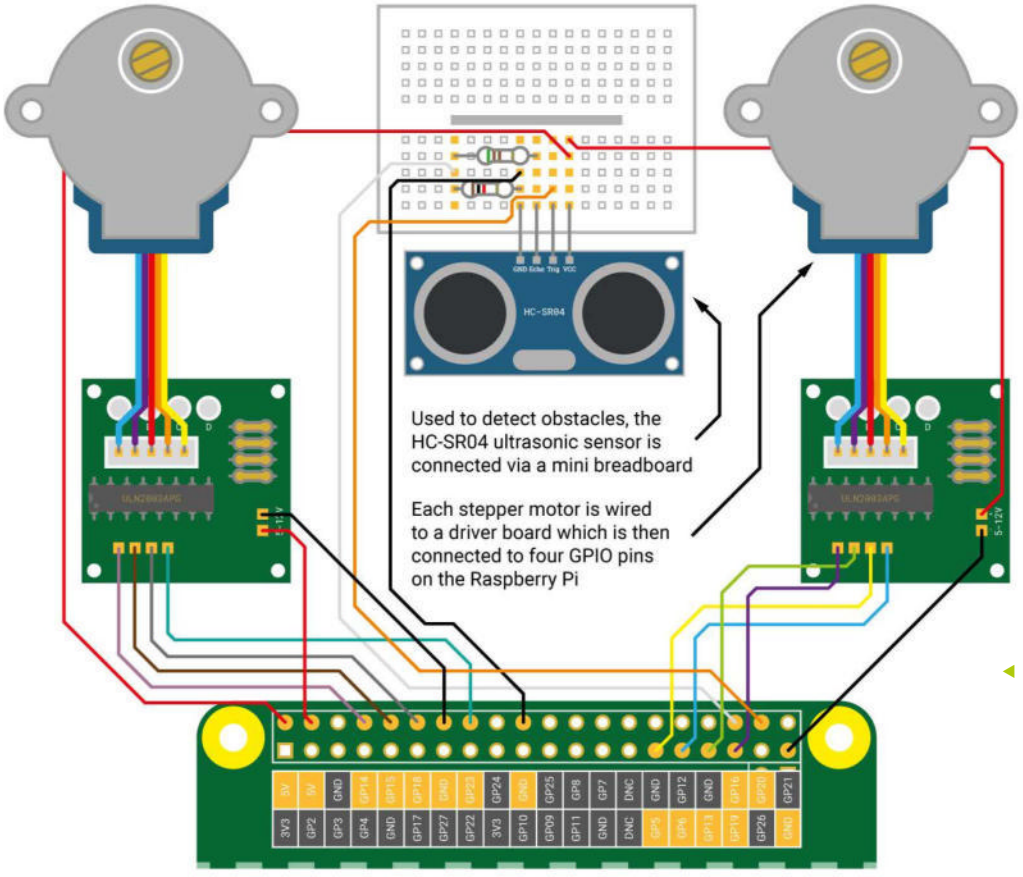


Figure 1: Wiring the ZeroBot

At the top, we import the `Robot` class from GPIO Zero, along with `sleep` from the `time` library. We then set the `robot` object to a `Robot` that has the GPIO pins set for the left and right motors. We can then run various commands to control it, including telling it to spin left or right. In this example, we're using a `for` loop with forward and turn right directions to make it drive around in a square pattern; adjust the `sleep` values to determine the square size. Try altering the directions to make different patterns.

NOTE

To save connecting your robot's Raspberry Pi to a keyboard and display when you need to modify code, SSH into it from another computer, tablet, or smartphone that's connected to the same wireless network.

Build a ZeroBot

Next, we'll show you how to build a ZeroBot based on a Raspberry Pi Zero W and two stepper motors. The 28BYJ-48 is a cheap but versatile stepper motor that can normally be bought with a ULN2003A driver board for under £4. Stepper motors can be programmed to move in discrete steps, rather than just turned on/off like other motors. They also have a lot more torque, so if you need to build a bot that can drive over obstacles, or even cross from a bare floor to a carpet without flipping over, you'll be pleased with stepper motors. They may not be as fast as some other motors, but they will get your robot from point A to point B.

Using your Raspberry Pi Zero W, you'll be able to control the speed and positioning of the motors very accurately. To cause the motor to rotate, you provide a sequence of 'high' and 'low' levels to each of the four inputs. The direction can then be reversed by reversing the sequence. In the case of the 28BYJ-48, there are four- step and eight-step sequences. The four-step is faster, but the torque is lower.

Power down your Raspberry Pi, disconnect it from power, and wire everything up as shown in **Figure 1**. Each motor has a connector block at the end of its coloured wires that slots into the white header on the ULN2003A. The GPIO pins controlling that motor connect to the four input pins below the IC, while the 5V power and ground connections go to the bottom two pins on the right.

Eyes to see

We'll give our ZeroBot some simple 'eyes' that allow it to detect obstacles, courtesy of the HC-SR04 ultrasonic sensor. This has four pins, including ground (GND) and 5V supply (VCC). Using Python, you can tell the Raspberry Pi to send an input signal to the Trigger Pulse Input (TRIG) by setting a GPIO pin's output to HIGH. This will cause the sensor to send out an ultrasonic pulse to bounce off nearby objects. The sensor detects these reflections, measures the time between the trigger and returned pulse, and then sets a 5V signal on the Echo Pulse Output (ECHO) pin. Python code can measure the time between output and return pulses. Connect the HC-SR04 as shown. Its ECHO output is rated at 5V, which will damage the Raspberry Pi. To reduce this to 3.3V, use two resistors to create a simple voltage divider circuit, as shown in the **Figure 1** diagram (with the 510Ω one connected to ECHO and the 1kΩ connected to GND).

Once you have all your components connected, you can test the code on a bench before building the full robot. Point the 'eyes' away from you and run the code shown next. The red LEDs on the ULN2003 board should flash and both motors should start turning. Our example has the bot move in a square. Check that the motors behave accordingly then rerun the code, but this time place your hand a couple of centimetres in front of the HC-SR04 and check that everything stops.

> *We'll give our ZeroBot some simple 'eyes' that allow it to detect obstacles*

At the time of this writing, GPIO Zero didn't include a class for stepper motors, but some community members have built their own. One of our favourites is Glenn Fabia's `Stepper` class, which you can find at **rpimag.co/gzstepper**. Download the **gpiostepper.py** file and put it in the same directory as the code you're about to run. Create a new program, enter the following code, and save it as **zerobot.py**.

```
from gpiozero import DistanceSensor
from gpiostepper import Stepper

sensor = DistanceSensor(echo=16, trigger=20)

# Steps per revolution and gear ratio
# from the 28BYJ-48 datasheet.
num_steps = 32
gear_ratio = 64
geared_steps = num_steps * gear_ratio

step_motor_l = Stepper(motor_pins=[14, 15, 18,
23],number_of_steps=num_steps)
step_motor_r = Stepper(motor_pins=[19, 13, 5,
6],number_of_steps=num_steps)

def move(direction='F', ctr=geared_steps):
    step_dir = 1 # 1 for fwd, -1 for back
    if direction == 'B':
        step_dir = -step_dir

    while ctr > 0:
        if sensor.distance > .1: # move if no
                                 obstacles
            # F=fwd, B=back, L=left, R=right
            if direction in ['L', 'B', 'F']:
                step_motor_l.step(step_dir)
                             # Left wheel only
            if direction in ['R', 'B', 'F']:
                step_motor_r.step(-step_dir)
                            # Right wheel only
            ctr -= 1

for i in range(4): # Draw a right-handed square
    move("F", geared_steps*2) # move forward
                              two revolutions
    move("R", geared_steps)   # Turn right 1/2
                              revolution
```

At the top of the program, we import the `DistanceSensor` class from `gpiozero` and the `Stepper` class from `gpiostepper`.

Have a look at the **gpiostepper.py** source to see how GPIO Zero devices are implemented. Next, we define the number of steps per revolution, taking into account the gear ratio inside the stepper motor. After that, we instantiate a left and right motor, and define a function called `move`, which moves the robot. You pass in a direction (F for forward, B for backward, L to turn left, and R to turn right) and number of steps. We set the step direction (`step_dir`) as `+1` when moving forward, and `-1` when moving backward. Next, we count down from the number of steps passed in, and so long as nothing is in the way of the distance sensor, we move forward. Because the wheels are oriented opposite each other, we reverse direction for the right-hand motor.

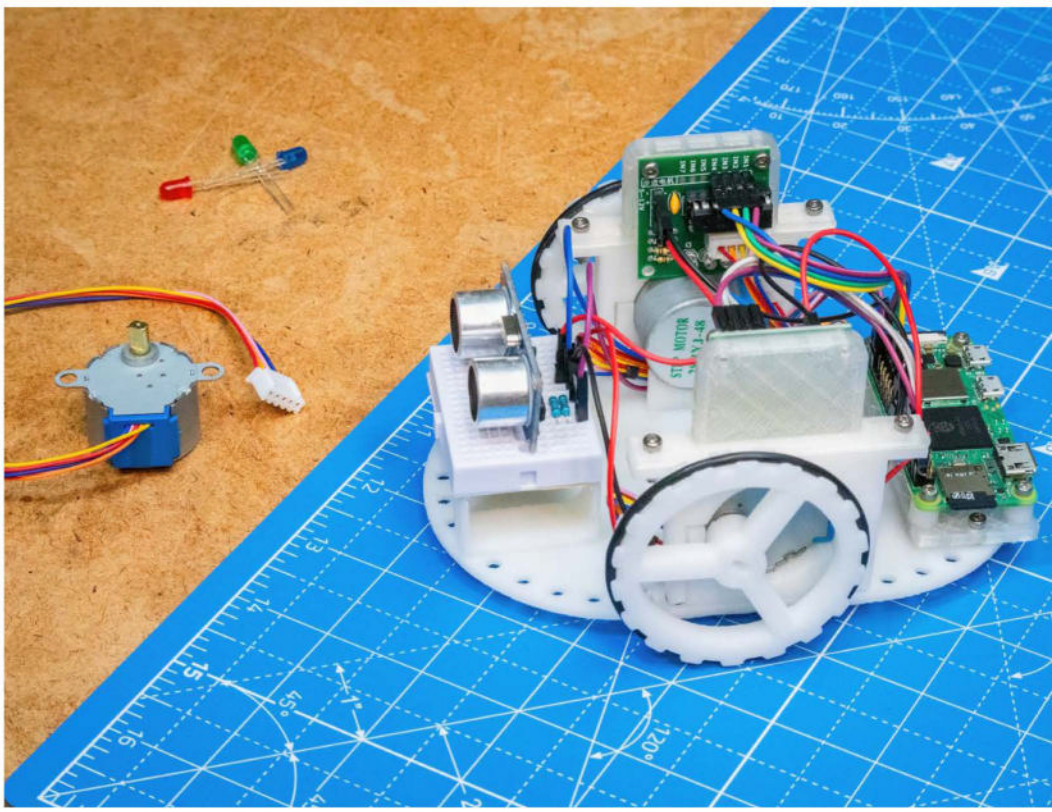
Back in the main body of the code, we run a loop four times. Each time through, we move forward through two full revolutions of the wheels, turn right one-half revolution, and then that's it. Your ZeroBot should have traced out a square with its movements!

Give it a body

Now it's time to give the bot a body. If you have access to a 3D printer, you can print the parts for the ZeroBot. This design fits together easily, although you do need to glue the chassis end-caps in place. Alternatively, you could construct a similar design using reasonably thick cardboard for the wheels and part of a plastic bottle as the main tubular chassis. Use more cardboard for the end-caps.

Put your mobile power bank at the bottom of the chassis tube, then attach the motors to the end-caps with screws. Next, place the ULN2003A boards on top of the power bank, and then sit the breadboard with the HC-SR04 'eyes' on top. Finally, slot your Raspberry Pi Zero W in at the back. All nice and cosy, and ready to roll!

If you have access to a 3D printer, you can print the parts for the ZeroBot



▼ The diminutive ZeroBot features a Raspberry Pi Zero, two stepper motors, and a 3D-printed chassis







- ***Get game design tips and tricks from the masters***
- *Download and play game examples inspired by classics*
- ***Learn how to code your own games with Pygame Zero***
- *Explore the code listings and find out how they work*

Code the Classics Volume II not only tells the stories of some of the seminal videogames of the 1980s, but shows you how to create your own games inspired by them using Python and Pygame Zero, following examples co-programmed by Andrew Gillett and Raspberry Pi founder Eben Upton.

Available now: **rpimag.co/store**

A RASPBERRY PI PICO HALLOWEEN

Petrifying Pico projects that are sure to fright and delight

By **Rotten Rob Zwetsloot**



Welcome, foolish mortals, to another year of haunted hacks from *Raspberry Pi Official Magazine*. These brutal builds are sure to send a chill down your spine and 'liven up' even the putrid parties of the undead.

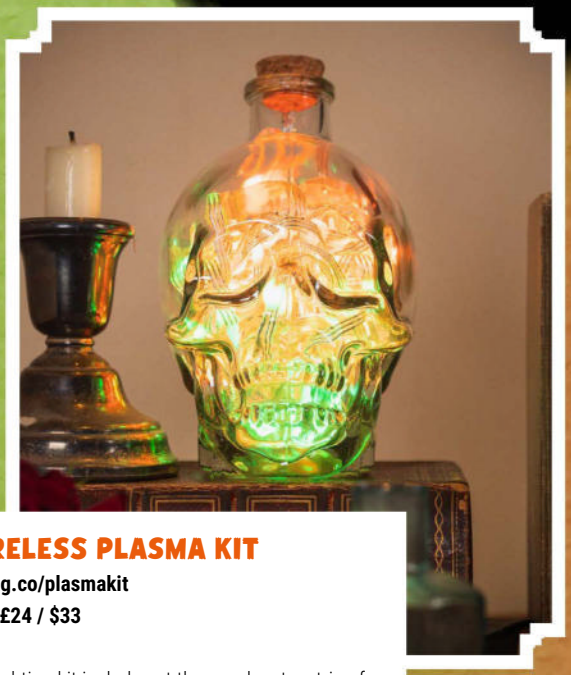
This year we're focusing on the tiny Raspberry Pi Pico as the dark driver of your foul festivities – don't be fooled by its small stature, though; it too can produce supernatural spooks to scare your gruesome guests. So start practising your programming sorcery – the witching hour awaits...

SIMPLE SPOOKY PICO HACKS

Upgrade some cheap decorations with lurid lighting

At this time of year, you're sure to find a lot of Halloween tat – excuse us – icky items of terror in your local supermarket or hardware store. Some of them may even be very cool (and very reasonably priced), but could use something... a little extra.

While there are plenty of projects with servos and sensors and HaaS (Halloween-as-a-Service, natch) over the next few pages, the quickest way to spruce up these fiendish finds is simply with some lighting upgrades from our friends at Pimoroni.



PICO UNICORN PACK

rpimag.co/picounipack
£22 / $30

An LED strip with buttons that is barely bigger than a Raspberry Pi Pico, these programmable lights can be easily slipped inside decorations and programmed with greens, oranges, and purples to give off a Halloween vibe. You need to provide a Pico yourself, but programming is very easy.



WIRELESS PLASMA KIT

rpimag.co/plasmakit
From £24 / $33

This lighting kit includes at the very least a strip of programmable LEDs and a Plasma 2350 W (basically a custom Raspberry Pi Pico 2 W designed to control LEDs: **rpimag.co/plasma2350**), perfect for creating a custom lighting scheme for any Halloween prop. Pimoroni also sells a version that comes with a skull bottle to put the lights in – Dan Aykroyd not included – as a quick Halloween decoration in itself.

DASTARDLY DIY DECORATIONS

Putrid props to 'liven up' your home... wait, did we already use 'liven up'?

PICO PUMPKINS

rpimag.co/picopumpkins

Not dissimilar to what we mentioned on the previous page about upgrading some decorations with simple lighting, although the lights in these jack-o'-lanterns are much more advanced with LED matrices used to create eyes. You'll also need to carve the pumpkin yourself, which means you don't have to do a traditional spooky face or even normal eyes to go in them.



▲ This project was created by the GurgleApps kids, who we've featured many times in this mag



▲ Having LEDs light up the mouth as well is an extra good touch, and saves on candles

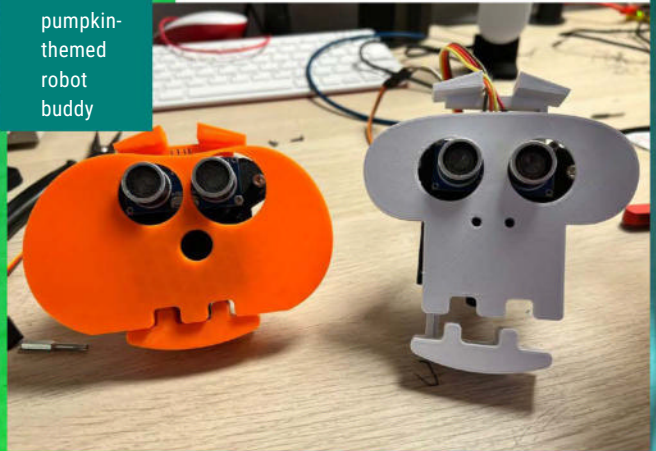


▶ The cartoon skull may not be super spooky, but it is frightfully fun

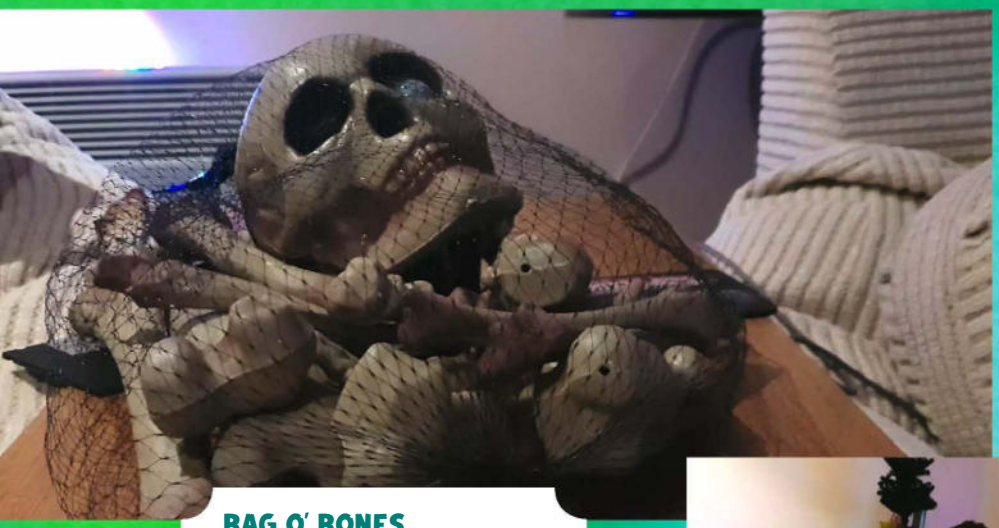
SPOOKY SCARY SKELETON

rpimag.co/scaryskelly

Kevin McAleer makes a lot of great robots over at **kevsrobots.com** and he created this 'bare bones' robot (get it?) for Halloween one year. It's made up of 3D printed parts and has ultrasonic sensors, eye-shaped sensors usually used to detect distance, to sense when someone is in front of it. When it detects movement, it starts moving its mouth and eyebrows.



▼ It also has a pumpkin-themed robot buddy



◀ It's a very descriptive project name
▼ Use it to guard your Halloween snacks from unsuspecting thieves



BAG O' BONES

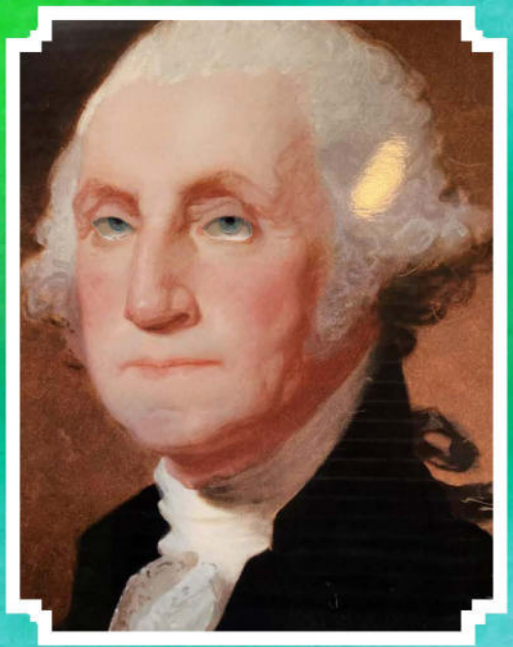
rpimag.co/bagobones

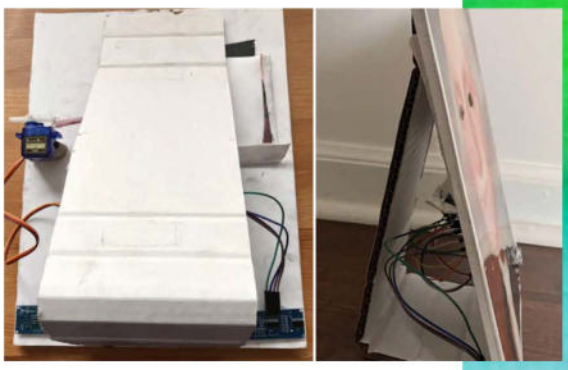
The bag o' bones is a decoration that maker 314Reactor found in a Sainsbury's supermarket. He upgraded it with a PIR motion sensor, various motors, and even a speaker so that any unsuspecting passers-by would be scared sockless by the suddenly writhing bones that begins to play Toccata and Fugue in D minor – which most folks will know as the cliché spooky organ music.

POSSESSED PORTRAIT

rpimag.co/picoportait

We cannot tell a lie; this portrait of George Washington following you around the room with his eyes is very creepy. Usually for a project like this, folks would make use of computer vision to track faces; however, this one uses ultrasonic sensors, much like the skull robot. The servos actually just move a bit of card around behind the picture – very low-tech but very effective.





▲ We love when you can recycle your delivery boxes to make up your project
◀ We didn't know ol' Georgie Dubs had such striking blue eyes, but he truly did

CURSED COSTUME CREATIONS

Well, we don't have printers on this list so they're not **THAT** cursed, we suppose

VORACIOUS VACUUM

rpimag.co/poltergust

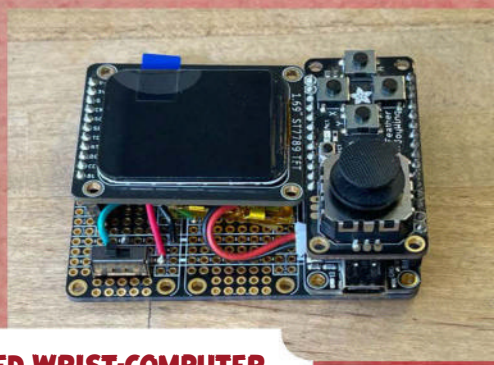
The Poltergust is a ghost-hunting contraption from the Luigi's Mansion series, modelled after your average household hoover. Here we've recreated the Poltergust G-00 specifically from Luigi's Mansion 3, and added Pico-powered lights that glow and cycle to give it a more ethereal/accurate vibe. Technically it can have moving parts too, but you'll need to make extra code modifications yourself for that.





▲ A mixture of 3D printed parts is used to get the right look as the orientation of the canister is upside down compared to real hoovers

◀ Our own Rob modelling his creation, complete with appropriate Green Plumber costume

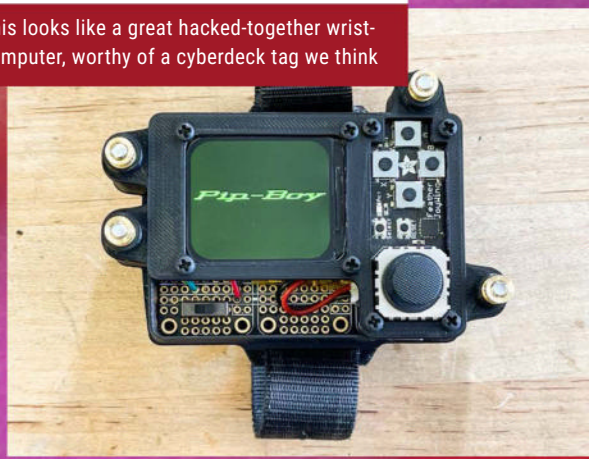


▶ It's mostly circuit boards enclosed in 3D printed parts, but it works

WRETCHED WRIST-COMPUTER

rpimag.co/pipboy2040

Modelled after the wrist-computers in the video game (and now TV series) Fallout, this very DIY device has buttons, a joystick, and even a little screen. The code provided on Adafruit is currently just a little slide-show, but you can easily program it to show little interactive interfaces or play simple animations. It probably won't tell you about any nearby radioactivity, though.

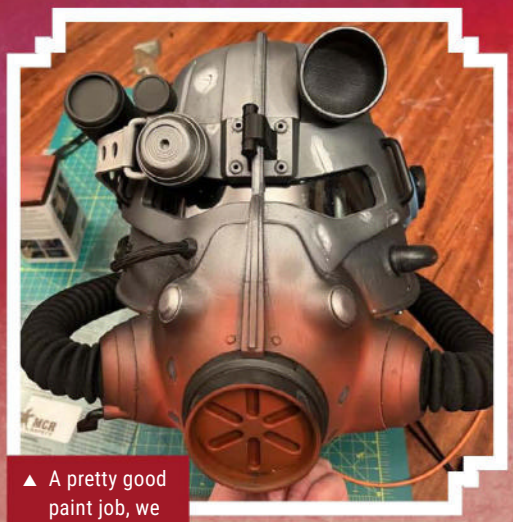


▼ This looks like a great hacked-together wrist-computer, worthy of a cyberdeck tag we think

HORRIFYING HELMET

rpimag.co/t45helmet

Another Fallout-inspired build, this meticulously 3D printed helmet for maker Grimdaria's 11-year-old not only includes lights but also a little HUD, cooling fans for inside the helmet, headphones to hear what's going on outside, and ways to control all the various lights on the helmet. It's probably more comfortable than even what the stunt actors have to wear on set…

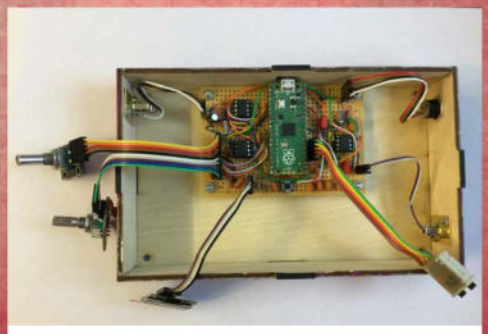


▲ A pretty good paint job, we have to say



▼ All the electronics fit inside the helmet

▼ The circuitry doesn't take up a huge amount of the box

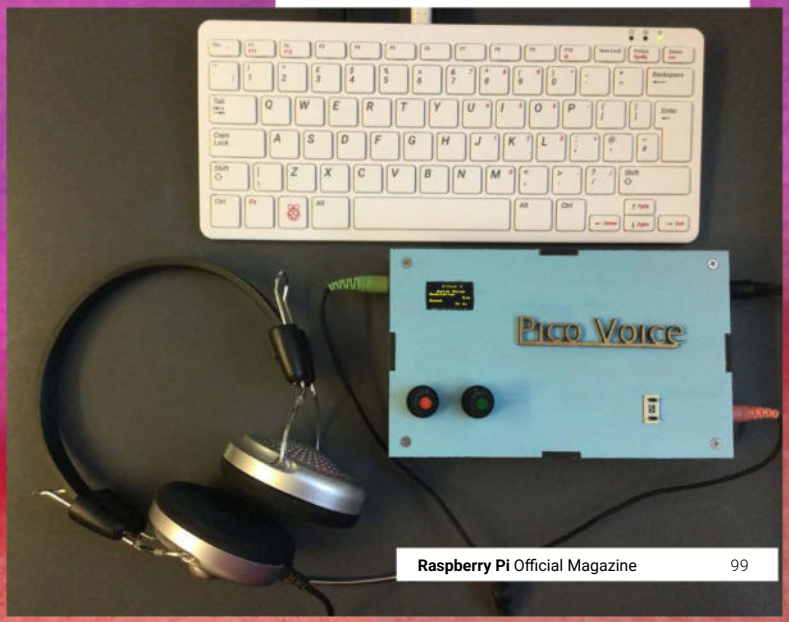


VICIOUS VOICE-CHANGER

rpimag.co/106

As *Scream* has shown us, asking someone if they like scary movies in an altered voice can be quite unsettling. This series of tutorials from way back in issue 106 goes over not only creating a Pico-powered voice changer, but one you can modify the settings of on the fly to get the exact right weird voice you need in whatever costume you're wearing.

▼ It's quite a large build; however, this one is designed to be used at your desk and can easily be miniaturised

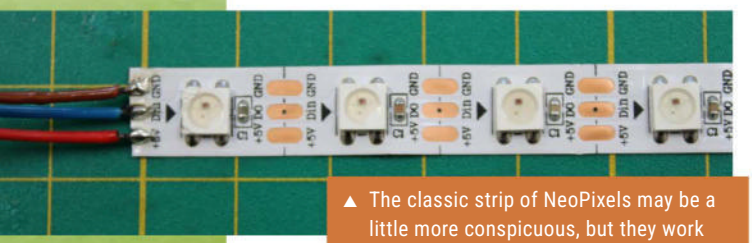


HACK YOUR OWN LIGHT DISPLAY

The pun there is that we mean hack as in computing, but you can hack at something with an axe too. That's scary, right?

In issue 159 (rpimag.co/159), we started a two-part tutorial on how to create your own Christmas lights with a Raspberry Pi Pico and NeoPixel LEDs. Apologies for bringing up the spectre of Christmas during this spooky time – if you need to, please think about ghosts and then return to reading – but such light displays don't have to be limited to December.

Using the same principles as these two tutorials, you can easily create light displays fit for Halloween. Raspberry Pi Pico is well suited for powering NeoPixels – even better than a standard Raspberry Pi, we'd say – and with a Pico W or Pico 2 W, you can control them remotely too. There's also the benefit of hooking up sensors to the system; who wouldn't like lights that react to movement or sound during the height of trick-or-treating?



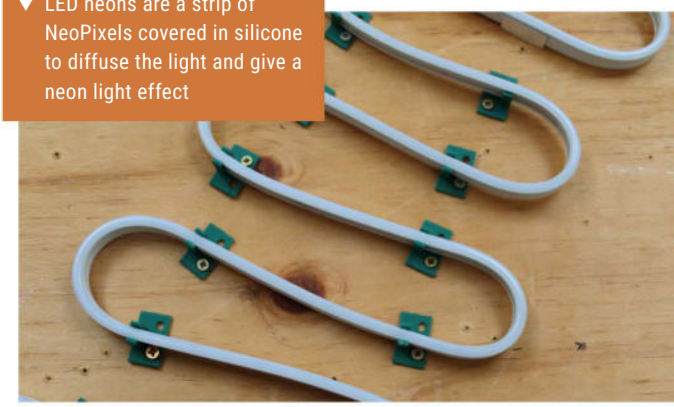
▲ The classic strip of NeoPixels may be a little more conspicuous, but they work well under cabinets, eaves, or gutters

CREEPY COLOUR VALUES

When programming your Halloween light displays, here are some RGB values you can use:

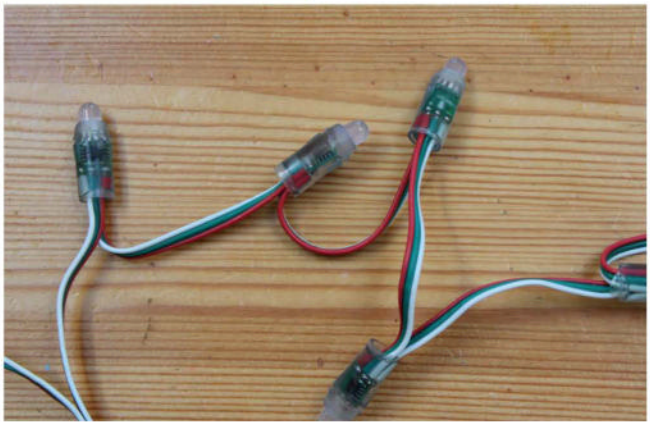
Ominous Orange	255,129,0
Putrid Purple	100,0,255
Ghoulish Green	17,209,46
Repulsive Red	170,0,0
Bone-chilling Black	0,0,0

▼ LED neons are a strip of NeoPixels covered in silicone to diffuse the light and give a neon light effect





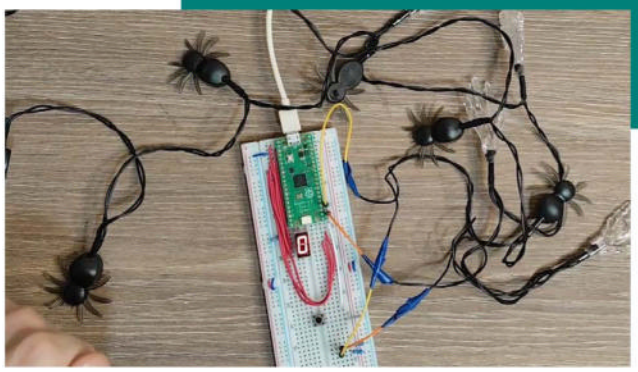
- Create your own colourful light display, controlled with a Raspberry Pi Pico
- These strings are a more familiar style of lights that can be wrapped around trees – autumnal ones with browning leaves in this case



Hacking themed lights

As well as controlling NeoPixel LEDs with various colour configurations, you can just hack some store-bought themed lights! Computing Mongoose gave it a go with some skeleton hands and spiders (**rpimag.co/hackedlights**) by – carefully, while turned off – snipping the lights off the battery pack they came with. He combined the two lights and then made a simple program to control when each set lit up in specific sequences. You can't control any colours with something like this, but you can still make neat displays timed to music or even motion.

Who wouldn't like lights that react to movement or sound?



ONLY THE **BEST**

Robots

By **Phil King**

The ultimate physical computing project is to get a robot moving, whether it's manoeuvred manually from a remote computer or controller, or – even better – you program it to move autonomously.

The most popular type of Raspberry Pi robot is still the wheeled vehicle. There are countless kits available from different manufacturers, ranging greatly in price and specifications. The number of wheels also varies, from two to six.

Some robots don't even need wheels to move about, instead crawling around on four or more legs. It can look a little creepy, but it's ideal for traversing tough terrain, especially if the robot has an inertial measurement unit (IMU) to help it self-balance.

Then there are robotic arms for manipulating objects, often featuring a camera for AI vision. An arm's degrees of freedom (DOF) depend on how many joints it has.

This time we take a look at an inexpensive arm, three vehicles (one with an arm on its back), and a couple of quadrupeds. Let's get moving...

PiCar-B

Adeept | £85 / $115 | adeept.com

While billed as a 'MARS Rover', this four-wheeler lacks the suspension system of the six-wheeled 4tronix MARS Rover (still one of our faves), so isn't quite so adept at handling tough terrain. Still, it has a proper steering mechanism for the front wheels, chunky tyres, and the metal chassis is an upgrade on the original acrylic model.

Like most Raspberry Pi robot cars, it features an ultrasonic sensor for obstacle avoidance and a (three-channel) line-follower mounted at the front. More unusual is the inclusion of a light level sensor, along with a mini OLED display to show info such as battery level and its IP address – handy for SSHing in from another computer. There are also several RGB NeoPixels for indicating its status and providing cool lighting.

The addition of a camera on a motorised pan-tilt mount enables real-time video transmission to a web interface and computer vision functionality, while there's a mic for speech recognition. You also get an IR remote control.



▲ Control this robot car via the web interface or its IR remote

Verdict

A capable car, programmable in Python.

WAVEGO Pro 12-DOF Bionic Dog-Like Robot

Waveshare | From £179 / $243 | waveshare.com

▼ Cheaper than a real dog and it won't bite you as it has no head!

Verdict

This robot dog isn't the prettiest, but it's smart.

Robotic pets are becoming more popular, but we're not sure this one would make a cute canine companion. Unlike the more obviously dog-like SunFounder PiDog, this pre-assembled robot lacks a head. It does still have four legs and – unlike the famous Boston Dynamics Spot – a 'tail', although it turns out the latter is actually part of the housing for its ultra-wide-angle camera.

Still, it can do some clever tricks. Its 12 high-torque servos (three per leg) give it 12 degrees of freedom, enabling it to move more flexibly than some quadruped robots. Low-level control of these is handled by a built-in ESP32 microcontroller, leaving Raspberry Pi 4 or 5 to handle the more advanced stuff.

Using the camera view, it can perform OpenCV-based functions like facial recognition, colour tracking, and motion detection. Thanks to an embedded 9-axis motion tracker, the robot can also self-balance on rough terrain or even a moving surface. Program it in Python or use the web interface for direct control.

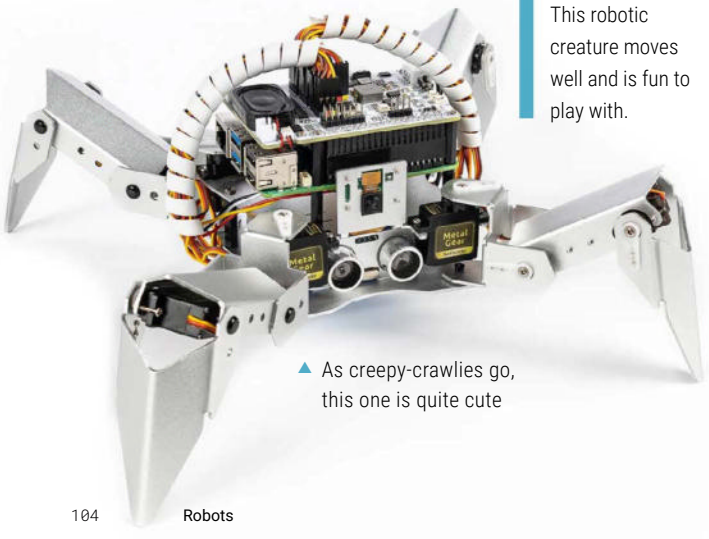


PiCrawler

SunFounder | £119 / $140 | sunfounder.com

Verdict

This robotic creature moves well and is fun to play with.



▲ As creepy-crawlies go, this one is quite cute

If you prefer your robotic pets to be a bit more creepy-crawly, you'll probably appreciate this one – although if you want more legs, you could opt for something like HiWonder's SpiderPi or ROSpider (both with six).

As its name suggests, the PiCrawler can crawl along the floor with its four limbs, using a total of 12 servos, and even wave a leg at you. That's once you've assembled the kit, which takes around three or four hours. Its Robot HAT works with nearly any Raspberry Pi model, including Zero 2 W.

Like other SunFounder robots we've tried, it features a mic for voice control, a speaker to talk to you, a 5MP camera for computer vision, and an ultrasonic sensor for obstacle detection. So you can do quite a lot of fun AI stuff with it, like getting it to follow your face or an object, or dance to music. It can be programmed in Python or the block-based EzBlock.

TurboPi

HiWonder | From £74 / $99 | hiwonder.com

This HiWonder kit for Raspberry Pi 4 or 5 enables you to build a standard-looking four-wheeled robot car with an aluminium chassis, except it comes with special Mecanum wheels. These make it far more manoeuvrable – including moving sideways, diagonally, and spinning on the spot without turning like a standard car.

Another interesting addition is the pan-tilt mechanism housing a camera that can be used for AI vision, enabling object and face detection. The car also features four-channel line-follower and ultrasonic sensors for navigation and obstacle avoidance.

While the Standard kit's default firmware uses Python and OpenCV, the Advanced and Ultimate versions work with the ROS 2 framework for more sophisticated AI features such as multimodal large models, the OpenClaw agent, and a voice interaction box. The Ultimate kit also adds a gripper arm.

Verdict

A robust robot car kit with some interesting additions.



▼ Mecanum wheels give this robot car lots of manoeuvrability

LanderPi

HiWonder | From £317 / $430 | hiwonder.com

Want a robot car to play with? And a robotic arm? LanderPi has both in the one package, the arm mounted on top of the four-wheeled vehicle's chassis. It's rather pricey, but that reflects the all-round high spec of the components and feature set.

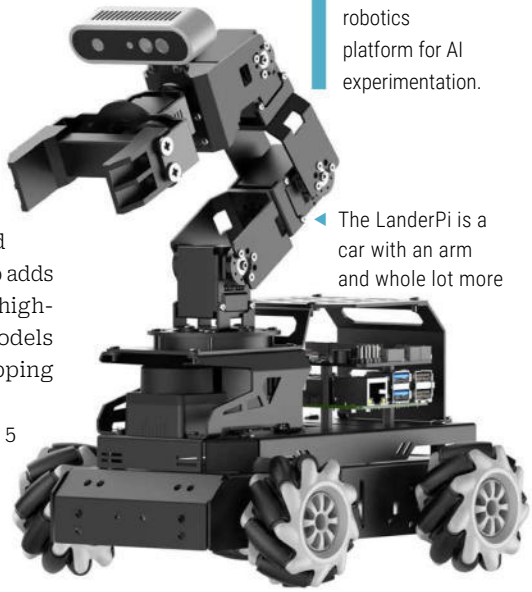
You get a choice of three chassis: one with Mecanum wheels for maximum manoeuvrability, one with an Ackermann front steering mechanism (as used in standard road vehicles), and a tank chassis with caterpillar track and shock-absorbing suspension.

The arm has six degrees of freedom with high-torque servos and a camera – monocular in the Starter Kit, upgraded to a 3D depth camera in the Standard and Advanced kits. The Advanced kit also adds a WonderEcho Pro AI voice box for high-quality speech recognition. All models have a lidar system for room mapping and navigation.

Compatible with Raspberry Pi 5 and ROS 2, the LanderPi is a high-quality, feature-packed robot with superior AI capabilities.

Verdict

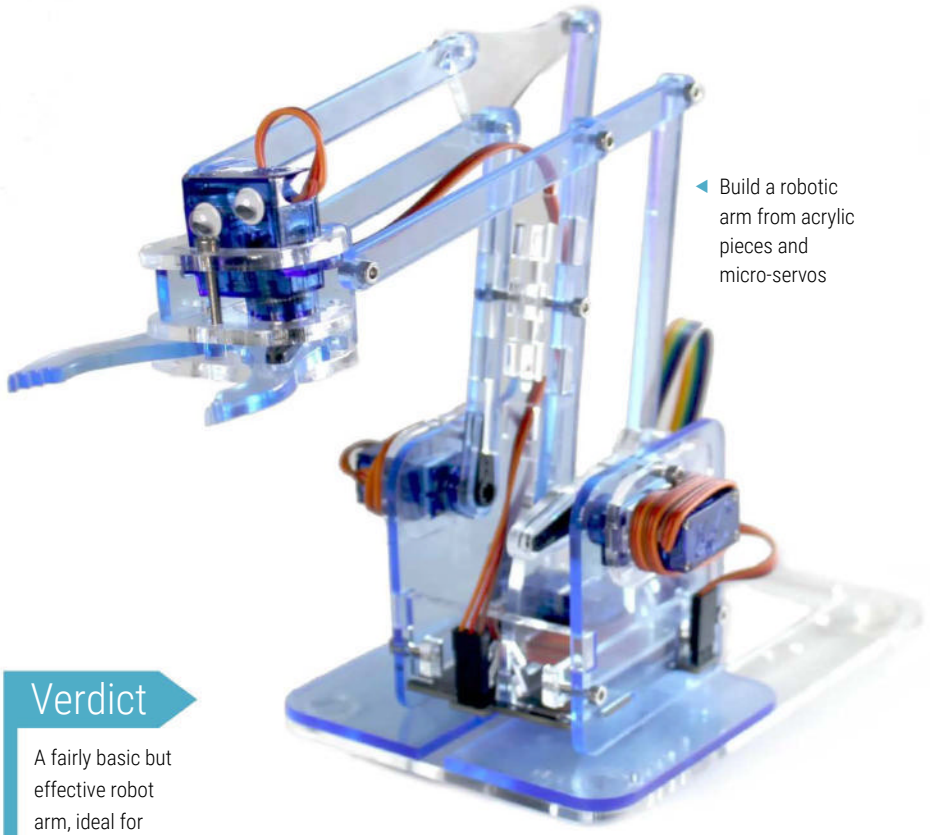
An impressive robotics platform for AI experimentation.



◀ The LanderPi is a car with an arm and whole lot more

MeArm Maker Kit V3

MeArm / The Pi Hut | £47 / $64 | mearm.com / thepihut.com



Build a robotic arm from acrylic pieces and micro-servos

There's a wide range of Raspberry Pi-based robotic arms on the market, some of which are pretty pricey due to the use of high-spec mechanical components and extra features like AI cameras. If you want to experiment with a less expensive arm, this MeArm one is ideal for beginners.

Coming in kit form, it takes around an hour to build from laser-cut acrylic arm pieces. The four servos for the joints are connected via cables to sockets in a base PCB which you can then connect to the GPIO pins on a Raspberry Pi Pico, or via a servo driver HAT to a Raspberry Pi computer. The arm design is open source, too, so you could even laser-cut the pieces yourself and get the PCB made – the files are on GitHub (**rpimag.co/mearmgit**).

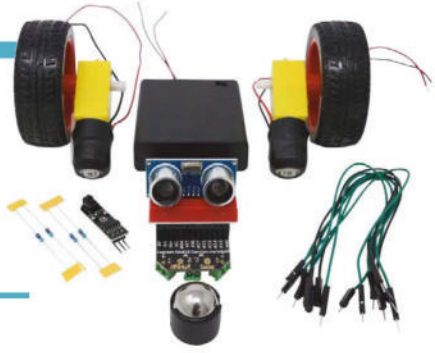
We used it with a Raspberry Pi Pico W for a tutorial in issue 162 **(rpimag.co/162)** and found it easy to control with Danny Staple's code at **rpimag.co/picomearm**, which lets you record and play back movements via a web interface.

Verdict

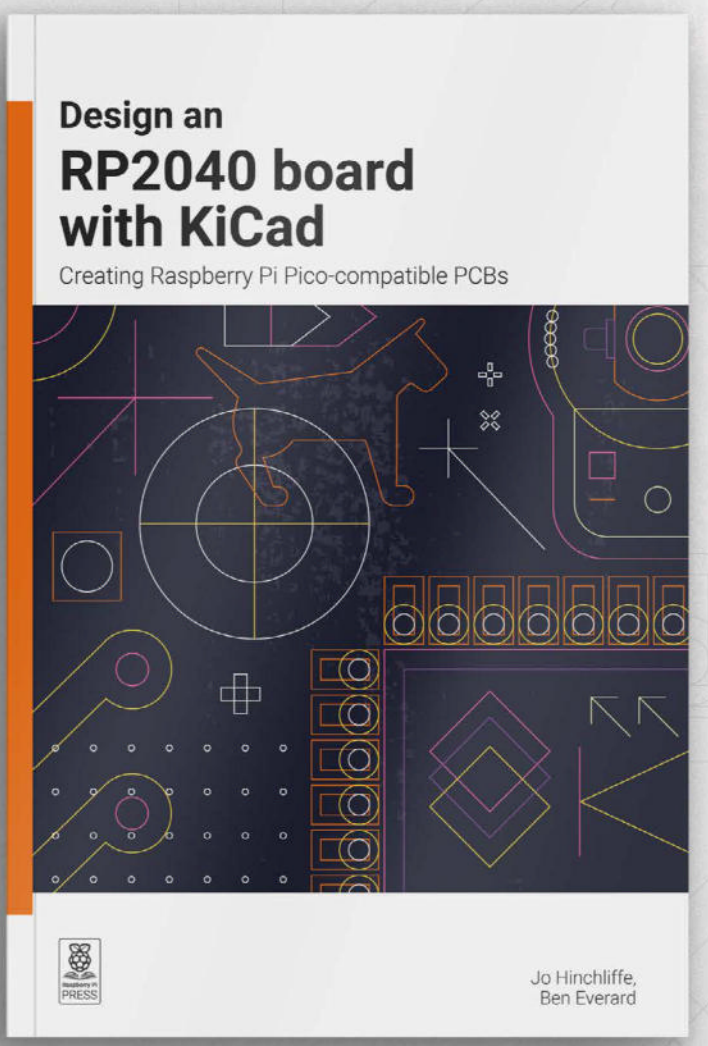
A fairly basic but effective robot arm, ideal for beginners.

CAMJAM EDUKIT #3

The Pi Hut | £20 / $27 | thepihut.com



An old favourite, this inexpensive kit lets you build a two-wheeled robot car – you can use the box as a chassis or supply your own. It works with any Raspberry Pi and the downloadable worksheets and Python code examples are easy to follow.



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



APPLY TO POWERED BY RASPBERRY PI

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

Whichever type of project you're involved with, Powered by Pi products come with a cast-iron guarantee of quality and durability. By **Rosie Hattersley**

When Raspberry Pi first arrived on the consumer technology scene back in 2012, it was firmly pitched as an entry level to real computing. It still fulfils this role, with pocket money kits getting kids started with coding and electronics, much like the Cytron EDU kits covered here. College courses focusing on Raspberry Pi challenge students to control or even create their own robots and vehicles, as per the Duckietown modules which originated as a means of intriguing freshers on a course at MIT. Thousands of hobbyists now use Raspberry Pi Zeros and Pico microcontrollers for smart home installations and camera-based observation stations. SparkFun's LoRaWAN IoT kit is just one such example.

It's only a small jump from these applications to dedicated industrial tasks in which Compute Modules and higher-end Raspberry Pi 4 and 5s work alongside environmental sensors and embedded camera modules empowered by Hailo-8 machine learning. These products may make intelligent judgements about how well a critical piece of equipment is performing, provide real-time data processing, or simply govern a more complex set of controllers. Hellbender's new Tendryl AI camera, developed alongside us here at Raspberry Pi, combines both these abilities, with sensors built into the camera itself.

The story of Raspberry Pi and what it's for is really a bit of a mix. It's for fun, for learning, for business, often all at once. It's certainly gratifying to see how many startups and businesses have used Raspberry Pi during development, eventually launching products they believe in. Compliance testing is an expensive and time-consuming process for manufacturers. We introduced Powered by Pi accreditation as a means of demonstrating the sheer breadth of products that have some form of Raspberry Pi at their core while standardising compliance regulations and testing procedures for every Raspberry Pi product at **pip.raspberrypi.com**. To become a Powered by Pi manufacturer, head to **rpimag.co/poweredbypiapply** to apply.

Bandai Jukebox Bowl-O-Rama

Japan/France | rpimag.co/jukebox-bowl



Should you be thinking ahead to the festive party season (guilty as charged), some form of entertainment device might well appeal. Jukebox Bowl-O-Rama has both commercial chops, having been built by the folks at Bandai – makers of many an arcade experience – and enough play modes to keep guests entertained. Its 12-inch subwoofer pumps out tunes from any connected smartphone and a wireless microphone to accompany your own karaoke favourites. Games include ten-pin and regulation poker, flash-o-matic, trick shot, and bowl 21, all of course controlled by the Raspberry Pi hidden by the robust wooden cabinet.

SparkFun Digi X-ON Kit for LoRaWAN (European Edition)

USA | rpimag.co/digix-on



The European edition of SparkFun's Digi X-ON Kit makes setting up a new smart home or office a breeze. The LoRaWAN-based hardware includes a Digi HX15 gateway, SparkFun's own IoT Node for LoRaWAN board, and Qwiic-based dual-action Environmental Combo board. The LoRaWAN board is based around Raspberry Pi's RP2350 microcontroller, which can be used with Raspberry Pi Pico SDK but also works with MicroPython and Arduino. LoRa non-line-of-sight communications work in both noisy environments (where signals from multiple devices might interfere) as well as over long distances. A 30-day subscription to Digi's X-ON device management platform and cellular network comes as part of the simple-to-set-up package.

Duckietown Duckiedrone DD24 and Traffic Lights

USA | rpimag.co/duckie-robots



Duckietown is a range of modular open-source kits based around the iconic rubber duck. Having started as a first class at MIT, it has proved a hit with schools and colleges, where the tactile ducks and range of layouts provide a stepping stone from creative play to electronics and coding. Kits based around robots, cars, and drones are all available, along with routes for them to navigate. The Duckiedrone DD24 is a Raspberry Pi-based quadcopter kit with a fish-eye lens tied to high-school and undergraduate curricula so students can learn about autonomous flight. Traffic Lights, meanwhile, are static robots with onboard cameras that can sense vehicles arriving, and form part of a user-controlled, Raspberry Pi-based smart city.

Hellbender Tendryl AI camera

USA | rpimag.co/tendryl

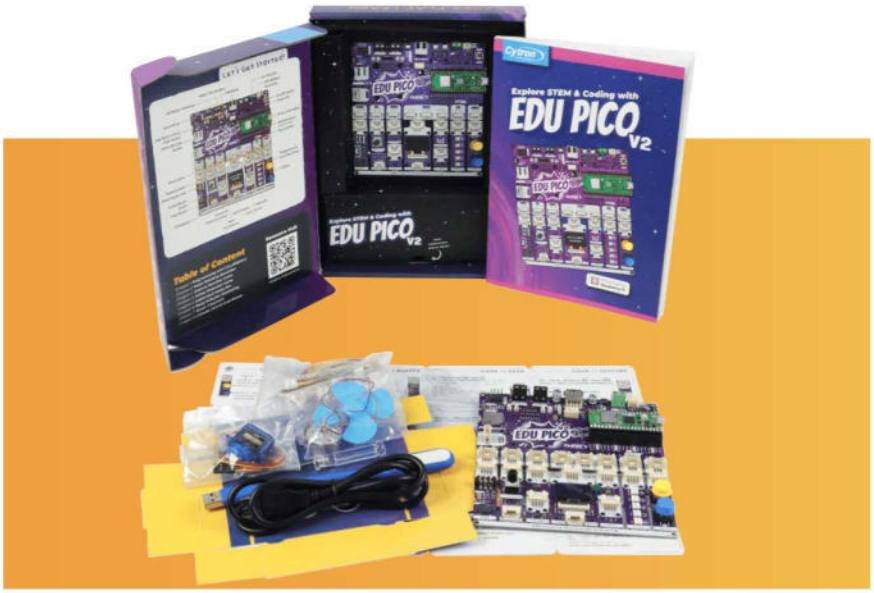


Hellbender has recently partnered with Raspberry Pi on this fully integrated edge AI camera which enables advanced computer vision tasks such as real-time object detection and recognition without needing to be connected to the cloud. With a powerful Raspberry Pi Compute Module 5 inside, the Tendryl 12MP autofocus camera uses a Hailo-8 inference accelerator capable of operating at 26 TOPS, making it ideal for smart monitoring. The camera has a built-in IMU, environmental sensor, mic, speaker, and LED for peripheral sensing and output. It features an HDMI port and is powered via USB-C; this same connection can be used for data transfers and flash updates. Power over Ethernet, Bluetooth 5.0, 2.4GHz, and 5GHz wireless networking connectivity are all included. Standard and wide-angle lens models are available.

Cytron EDU Pico 2 Innovation Kit

Malaysia | rpimag.co/cytron-edu

The EDU Pico 2 Innovation Kit provides an excellent introduction to the world of CircuitPython with nine separate but complementary projects to complete using Pico 2 W. If you or your children have mastered coding principles using Scratch, or simply want to step up from following preset instructions to writing your own text-based code, CircuitPython is a great choice. Cytron's kit provides a clear visual guide and accompanying tutorials. A range of stimulating projects using Grove components introduces you to the worlds of potentiometers, humidity, temperature, and proximity sensors as well as motor and servo controls.



Riverdi CM4 displays

Poland | rpimag.co/riverdi-cm4



Poland's Riverdi specialises in designing displays for Internet of Things and industrial uses and has a range of high-brightness 10.1-inch IPS capacitive touchscreens based around Compute Module 4. The company works with embedded engineers in industrial, medical, automotive, energy, and secure hardware sectors to provide bespoke display options featuring a range of finishes and bonding types. Riverdi favours Raspberry Pi for its open-source but flexible approach, including solid power management. As an industrial display specialist, the firm offers expert tips on making the most of your display, including how to adjust the backlight, avoid phantom touches, and ensure flawless connectivity.

10 amazing: computer vision projects

Incredible ways that Raspberry Pi can see and understand the things around it



Even before the current wave of LLM-led AI, Raspberry Pi was using neural networks and machine learning in computer vision projects to see and react to its surroundings. Here are some of the amazing ways you can see with a Raspberry Pi.

01. Avanade Intelligent Garden

Bee pollination tracking

rpimag.co/avanade

This system involves various sensors to monitor a garden, alongside an interactive camera that tracks insects and pollinating bees.

02. Seeing Wand

Point-and-see

rpimag.co/seeingwand

This was quite a crude project back in the day, but the idea was that you'd point the wand at something and it would say aloud what it was. An early Google Lens, maybe.

03. Face recognition door lock

Face ID door

rpimag.co/facelock

Phones do it, so why can't doors? Well, there are other security issues with allowing your entire house to be accessed via your face; however, it is a fun idea.

04. Raspberry Pi Lego sorter

In its place

rpimag.co/legosort

This massive contraption – made of Lego – allows you to sort your Lego through AI vision alone. How you then sort the sorter is another problem.

05. Nut and bolt inspection

Quality: controlled

rpimag.co/ai-qc

We've heard of factories using Raspberry Pi cameras and trained datasets to do quality control of their production line, and they work a lot like this.

06. Mini Raspberry Pi Wildlife Monitor

Automatic critter detection

rpimag.co/mini-wildlife

Raspberry Pi being used for wildlife monitoring is nothing new. What is new with this project, however, is the inclusion of object detection to see what beasties are on camera.

07. Formula Pi

Autonomous racing robots

formulapi.com

This old robot racing series would require teams to program their robots to follow colours on the track to best outpace their rivals.

08. Monitor plant growth

Lego and OpenCV

rpimag.co/plantgrow

Timelapse is one way to see your plants grow. However, getting analysed records and video/photos allows you to track it much more closely.

09. Open weed locator

Robot weed killer

rpimag.co/openweed

This large-scale agricultural weeder uses "entirely off-the-shelf components, very simple green-detection algorithms and entirely 3D printable parts," which sounds easy and cheap to set up.

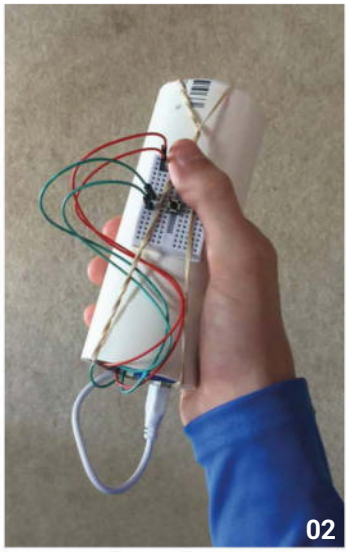
10. Pokédex

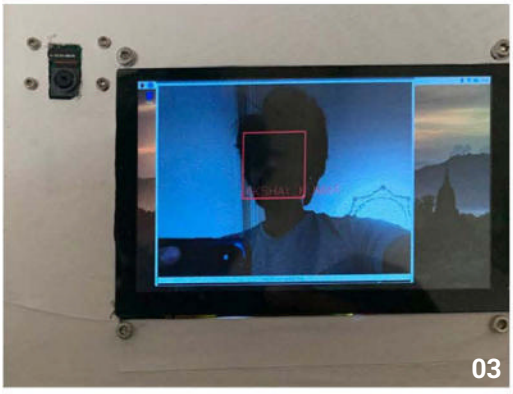
See 'em all

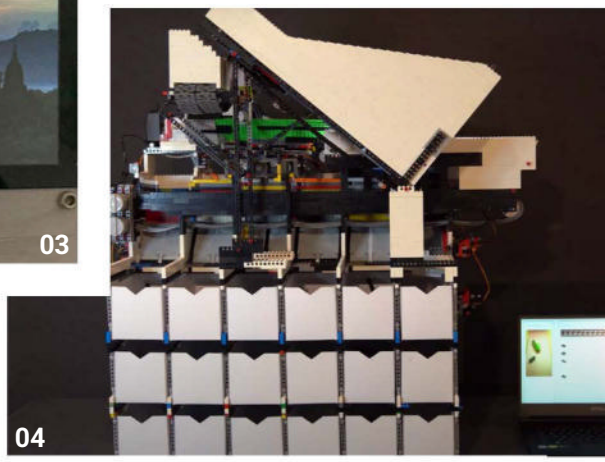
rpimag.co/pokedex

In the games, Pokémon can be identified via a device with a camera called a Pokédex which then gives you information and stats about the creature. Perfect for an AI vision project.


01

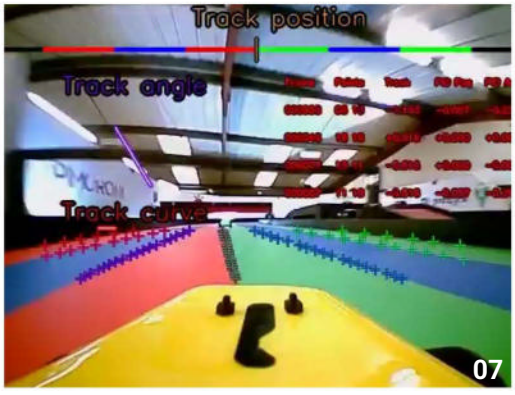

02


03


04

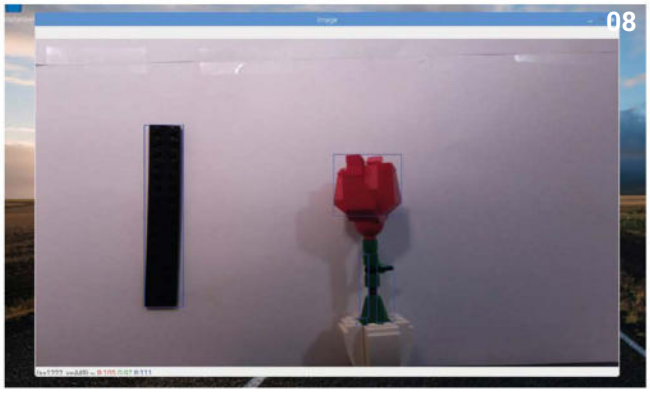

FPS: 29.95
DPS: 30.02
Last Result: good
Time stamp: 2025-02-12T22:08:32.356909
05

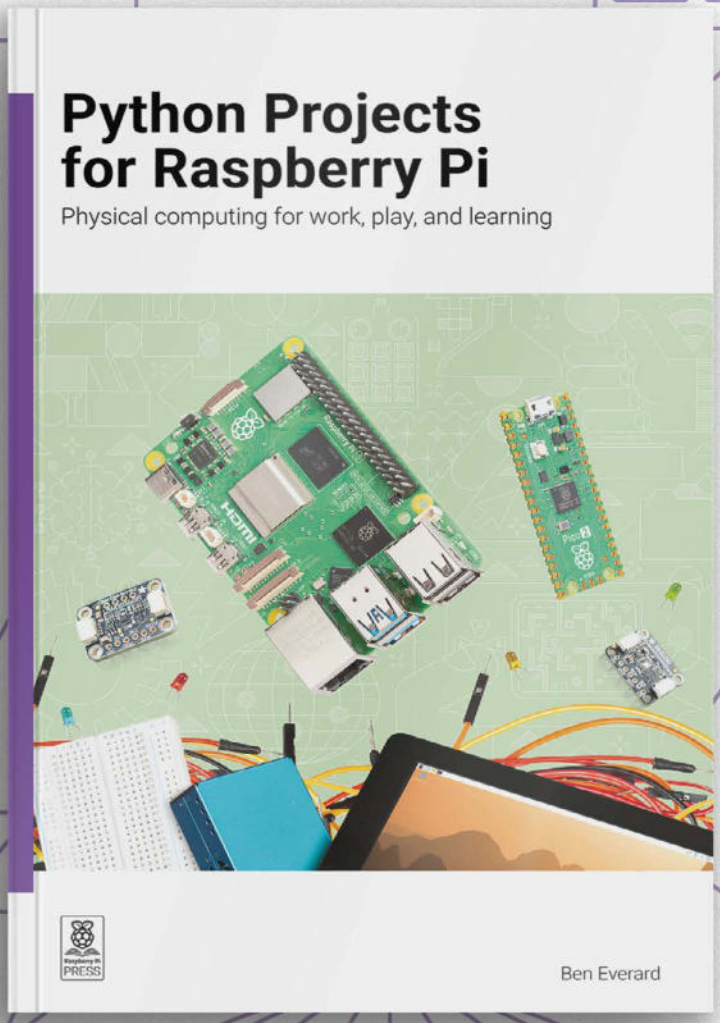

06


Track position
Track angle
Track curve
07


09


Output
Bulbasaur: 99.35% (correct)
10


08


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

Python Projects for **Raspberry Pi**

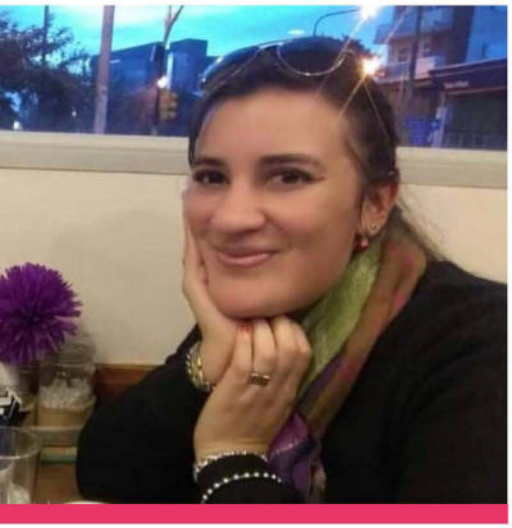
Physical computing for work, play, and learning

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**



Eliana Otrera

Bringing computing to rural schools in Argentina and running Raspberry Jams so kids can access computers and coding

Name Eliana Otrera
Occupation Teacher
Community role Educator
URL rpimag.co/raspisde

Eliana Otrera lives in Forres in the province of Santiago del Estero, Argentina. A city now, it was just a "small town with dirt roads" when she was a kid, but that didn't stop her from seeking out technology and ultimately, computing.

"That was when I decided to study Computer Science at university," Eliana tells us. "Since I did not have my own computer, I had to go to internet cafés and rent one. I'm not talking about the 1980s – this was in 2003! But none of that stopped me from continuing my education. Obviously, as I progressed through my degree, my parents were eventually able to buy me my first computer."

In 2017 she joined a programme called Conectar Igualdad (Connecting Equality) and started to bring coding and robotics to rural schools.

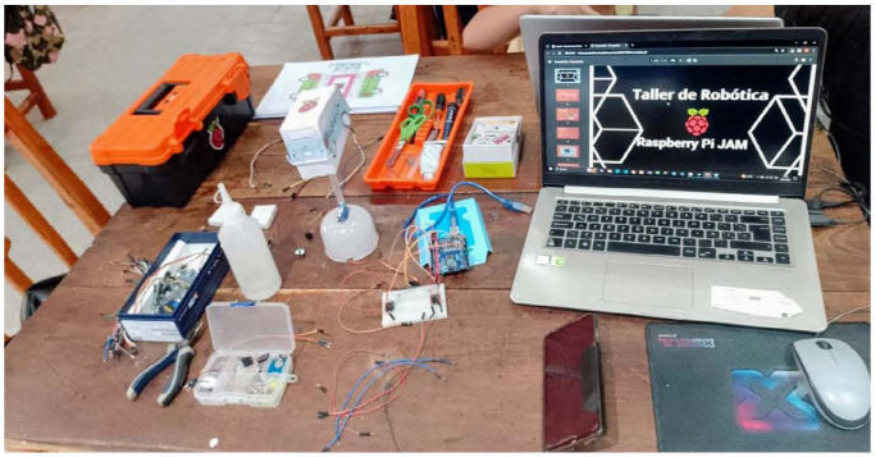
"My experience in rural schools was, without a doubt, one of the stages that most shaped my relationship with technology and education," Eliana says. "There I understood that talking about the digital divide was not simply a matter of statistics or a lack of devices. The technology gap had faces, names, and stories."

When did you first discover Raspberry Pi?

In 2017, when the Conectar Igualdad programme introduced the Pi-Top distribution, which consisted of portable computers designed for educational use. Inside them, I discovered this powerful little tool, and I used it in many projects across different schools. Since then, I have stayed connected with you, keeping myself up to date and making the most of the new features and possibilities that Raspberry Pi continues to offer.

▼ Programming in Minecraft on a Pi-Top is a classic of Raspberry Pi digital making





◀ Raspberry Pi is an important part of the computing education
▼ Eliana teaching kids about robots and robotics

How did the Raspberry Jams in Santiago del Estero come about?

The Raspberry Jams in Santiago del Estero grew out of a shared passion. In 2019, I began promoting Raspberry Pi among my professional circle and colleagues as an educational tool, while also looking for new ways to bring technology closer to students and teachers.

Some time later, together with two colleagues who share the same interests and passion for technology, we became pioneers in promoting Raspberry Pi in Santiago del Estero. On 6 March 2020, in Forres, we organised the first Raspberry Pi workshop open to the community in the province.

From that moment on, we began organising them more regularly and in different places – schools, cultural centres, and academies – both in Forres and in the surrounding areas.

Since our first workshop in 2020, we have seen growing interest from students, teachers, families, and technology enthusiasts.

What are some of the student projects you've particularly enjoyed at the Raspberry Jams?

At the introductory levels, I have come across projects such as electronic trivia games and basic robots, where the most important thing is the artistic expression they put into the finished result.

In these projects, the expression and emotion they manage to convey through their decoration are more important than any other automated function that could be added to the robot.

> *Since our first workshop in 2020, we have seen growing interest from students, teachers, families, and tech enthusiasts*

At the more advanced levels, I have seen programming projects such as programmed constructions, where students use Python and Minecraft as a foundation to create different institutional buildings automatically.

In robotics, I have been able to see the evolution of an automated greenhouse project – from an initial model to what the students are now implementing in productive agricultural projects for irrigation control, drone deployment, and crop planning.



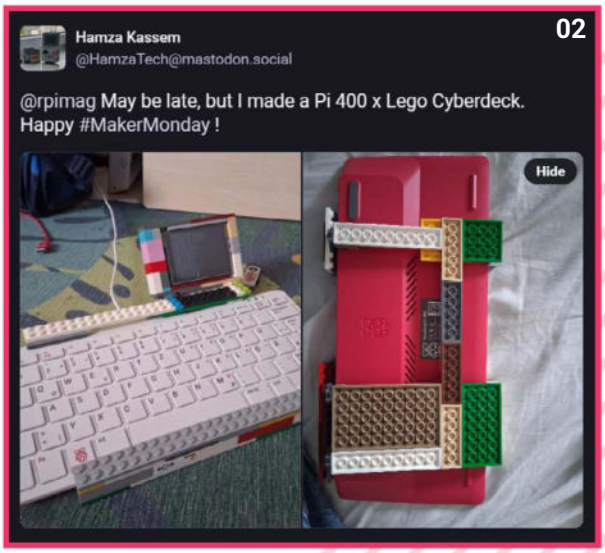
Maker Monday

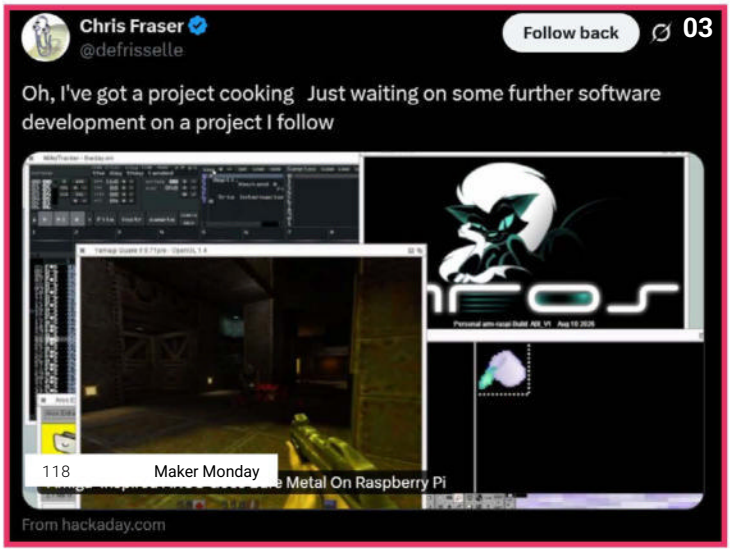
Amazing projects direct from social media!

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!







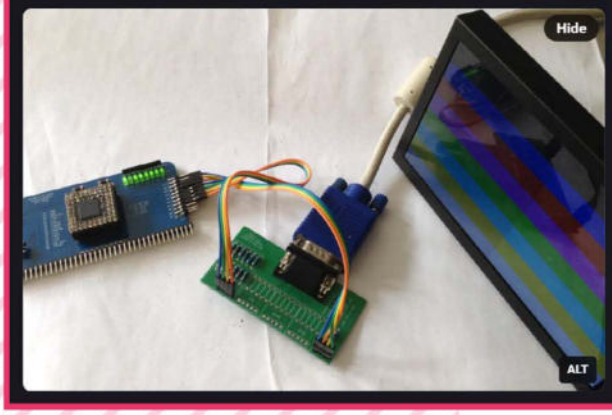
01. The quality of the final model here is incredible – **rpimag.co/170mm1**
02. It's been 14 years and we never tire of Lego cases for Raspberry Pi – **rpimag.co/170mm2**
03. Old-school Amiga fans, you might enjoy this – **rpimag.co/170mm3**
04. We don't need to ask why; sometimes you just do something for a bit of fun – **rpimag.co/170mm4**
05. Compute Module 5 robotics, you say? We like the sound of that – **rpimag.co/170mm5**
06. A PCB with some space to breathe – **rpimag.co/170mm6**
07. We're seeing a lot of CRT love recently, and it's great – **rpimag.co/170mm7**
08. It's a crocodile-shaped PCB that looks like a Nixie-tube clock. Excellent – **rpimag.co/170mm8**

04

Kevin
@emalliab@mastodon.social

@rpimag I've managed to pull together code from around the Internet to reimplement ZX Spectrum compatible graphics for an RC2014 Z80 computer using an RP2350B.

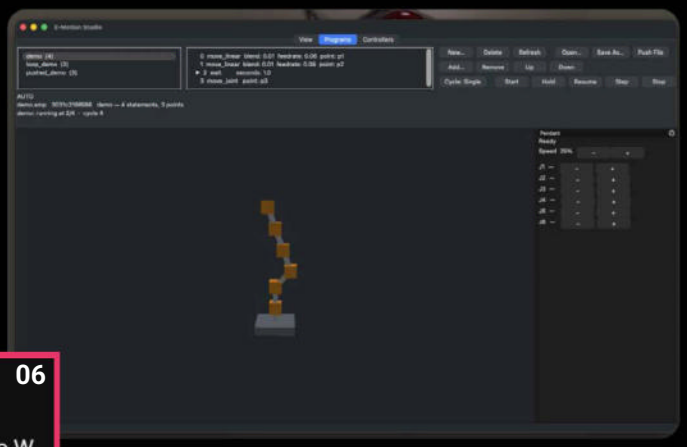
No, best not to ask why...



05

Freda Enjoyer
@FredaEnjoyer

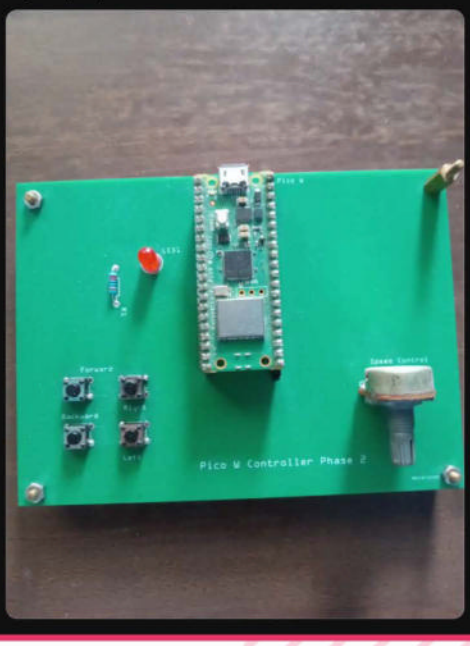
Working on a CM5-based Robot Controller and a new robotics framework:



06

linuxrobotgeek 24/08/2026

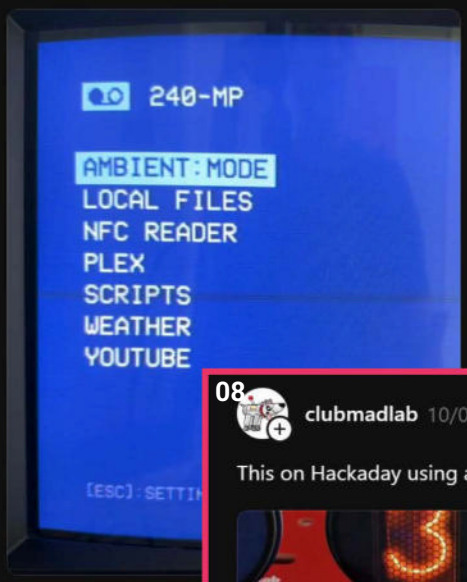
My all purpose Pico W controller that works with both the Pico and Pico W.



07

antcaccese › MakerMonday 24/08/2026

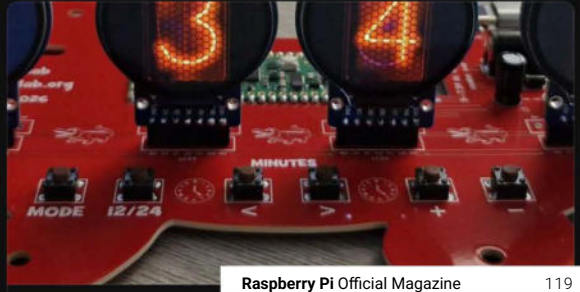
Maker Monday is one of my favorite things... i've been continuing to build ideas into my CRT focused media player project for the Raspberry Pi and over the weekend I pushed an update that includes a way to use an NFC Reader to load videos from physical tags as well as a retro 90s styled weather module (with period appropriate music 😊). Added a video below with details:



08

clubmadlab 10/08/2026

This on Hackaday using a Pico 2 W - hackaday.io/proje...



hackaday.io
Clocodile

Raspberry Jam at Wikimania 2026

Wikimedia's annual gathering gets another Raspberry Jam

This year's Wikimania was held in Paris, and once again Suyash Dwivedi put on a Raspberry Jam for the various community members in attendance. We interviewed Suyash in issue 168 (**rpimag.co/168**) about these Jams, and one of the highlights of this event was about language preservation.

"Projects such as Lingua Libre and Mozilla Common Voice have already demonstrated the importance of recording and preserving spoken languages, especially those that are under-represented online," Suyash writes on his blog post for the event. "Imagine affordable Raspberry Pi-based recording stations that could be deployed in schools, libraries, community centres, or rural areas."

You can read the full blog post (on Medium) breaking down the event here: **rpimag.co/wikijam26**.





▲ Suyash on the left with other attendees
◀ A lot of folks attended the event, with the sign for Nerd HQ just about visible on one table for a reason

Crowdfund this!

Crowdfunding campaigns to keep an eye on

WLV-01



Not wanting to "take out a mortgage on a Leica", creator Malcolm-Jamal Wilson sought to create his own custom camera with various specific functions. He landed on a custom camera built around Raspberry Pi, like many others, and is now crowdfunding a production run of them.

▶ **kck.st/4gui2le**

XGO-mini2SW



It's been a while since we've seen a new XGO robot dog, and this one comes with wheels too. It's powered by a Raspberry Pi Compute Module 5 and has various AI capabilities including autonomous learning. You can still program it as before, though.

▶ **kck.st/4cpDmMG**

Together we can make a **difference**



Give young people the opportunity to learn about technology

The Raspberry Pi Foundation enables young people to realise their full potential through the power of digital technologies, but we can't do this work without your help. Your support helps us give young people the opportunities they need in today's world. Together we can offer thousands more young people across the globe the chance to learn to create with digital technologies.

Generous donations from organisations and individuals who share our mission make our work possible.

DONATE NOW

Donate today to make a difference: **rpf.io/donate**

Raspberry Pi Foundation

Your Letters

Where are the Zeros?

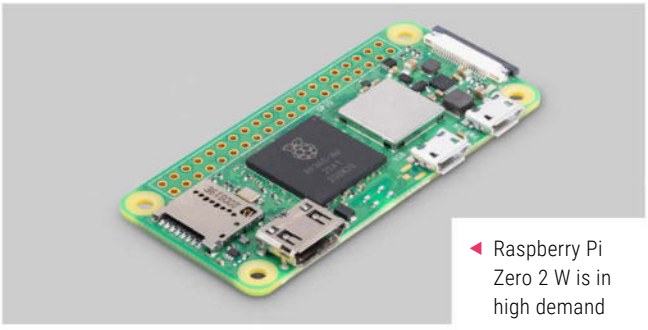
The seasons are changing, the leaves are starting to fall off the trees, and I'm painfully reminded that the timelapse camera that I planned to set up to capture the changing seasons in my garden never happened. Why? I went to order a Raspberry Pi Zero 2 W to control a Camera Module, and send back images via a website – but there were no Raspberry Pi Zero 2 Ws to be had when I first planned it, and there weren't any available when I remembered about it the month after, and then... I just put it to the back of my mind. So my question is: when will Raspberry Pi Zero 2 W be back in stock?

Cameron, via email

To answer your question, here's Eben Upton, Raspberry Pi chief executive: "We're seeing incredibly high demand for Zero 2 W, and are ramping up production of the RP3A0 SoC which powers it. We'll do our best to allocate some supply to ARs for single-unit sales, and expect to be back in a robust in-stock position across channel in Q4."

Q4 here refers to the calendar year, so there should be Raspberry Pi Zero 2 W computers back on the shelves sometime between October and the end of December.

I personally blame El Niño for anything out of the ordinary, but it's probably down to something more prosaic, like the capacity of every single chip-manufacturing fab trying to catch up with demand for AI chips to generate silly but pleasant images...



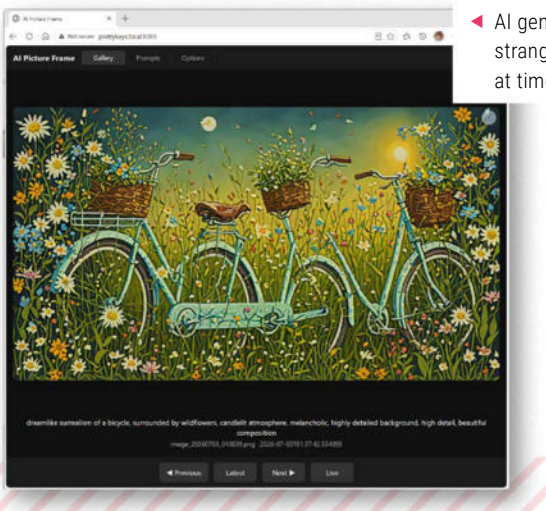
◀ Raspberry Pi Zero 2 W is in high demand

AI images

I've struggled to find a use case for AI in my working life, but in my other life as a tech tinkerer it's great. Fun stuff like the AI picture frame [which we included in RPOM169] doesn't have any commercial value that I can see, but it does teach you how it all works, and where else are we going to get images of bicycles that don't work?

Brian, via email

There are loads of real-life applications for AI that are actually useful; it's just that the people who shout loudest about AI are shouting about the simplest things, like "summarise my meeting for me", that in an ideal world wouldn't need to be done. The point of meetings is to generate simple, actionable points; if you need AI to simplify those points, then the meeting wasn't doing its job in the first place. Maybe you have a manager who loves to have meetings for meetings' sake; in that case AI might be useful for you. And if you need AI to help you generate ideas, maybe you shouldn't be working in a field that requires imaginative thought. For us, it's enormous fun to play with, and play is the best way to learn anything.



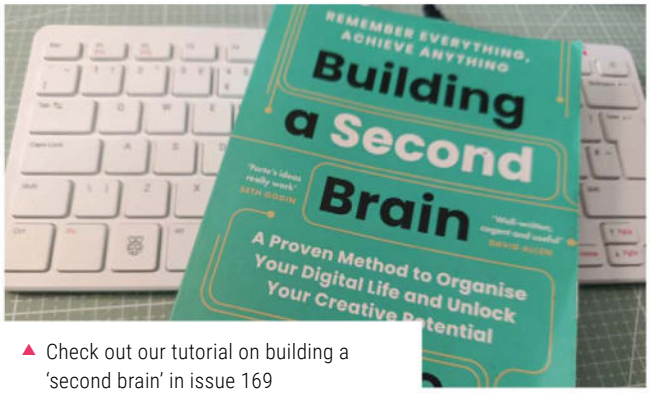
◀ AI generates strange images at times

Second brain

I was intrigued by the tutorial on building a second brain last issue. I've been doing something roughly similar – I have a few devices, a couple of Raspberry Pi computers, and my phone, all signed in to the same Google document, so wherever I am I can make a quick note and know that I'll be able to read it wherever I am. It's convenient, and better than using notes on paper because they inevitably get lost. But in terms of privacy, it's terrible: I'm moving data from a machine in my garage to a machine in my living room, via Silicon Valley. So I'm dusting my Raspberry Pi 4 off and putting a second brain on it. Hopefully it will work better than my first brain.

Sam, via email

There's always value to hosting things yourself. The cost is a little bit more time and effort; the payoff is knowing that your data stays on machines that you control. As the saying goes, there is no cloud, there is only someone else's computer. If you have a Raspberry Pi device gathering dust or languishing in a drawer, there's no better use you can put it to than safeguarding your privacy.



▲ Check out our tutorial on building a 'second brain' in issue 169

Contact us!

- @rpimagazine
- @rpimag
- rpimag.co/facebook
- magazine@raspberrypi.com
- forums.raspberrypi.com

USA SPECIAL!

6 ISSUES FOR $43

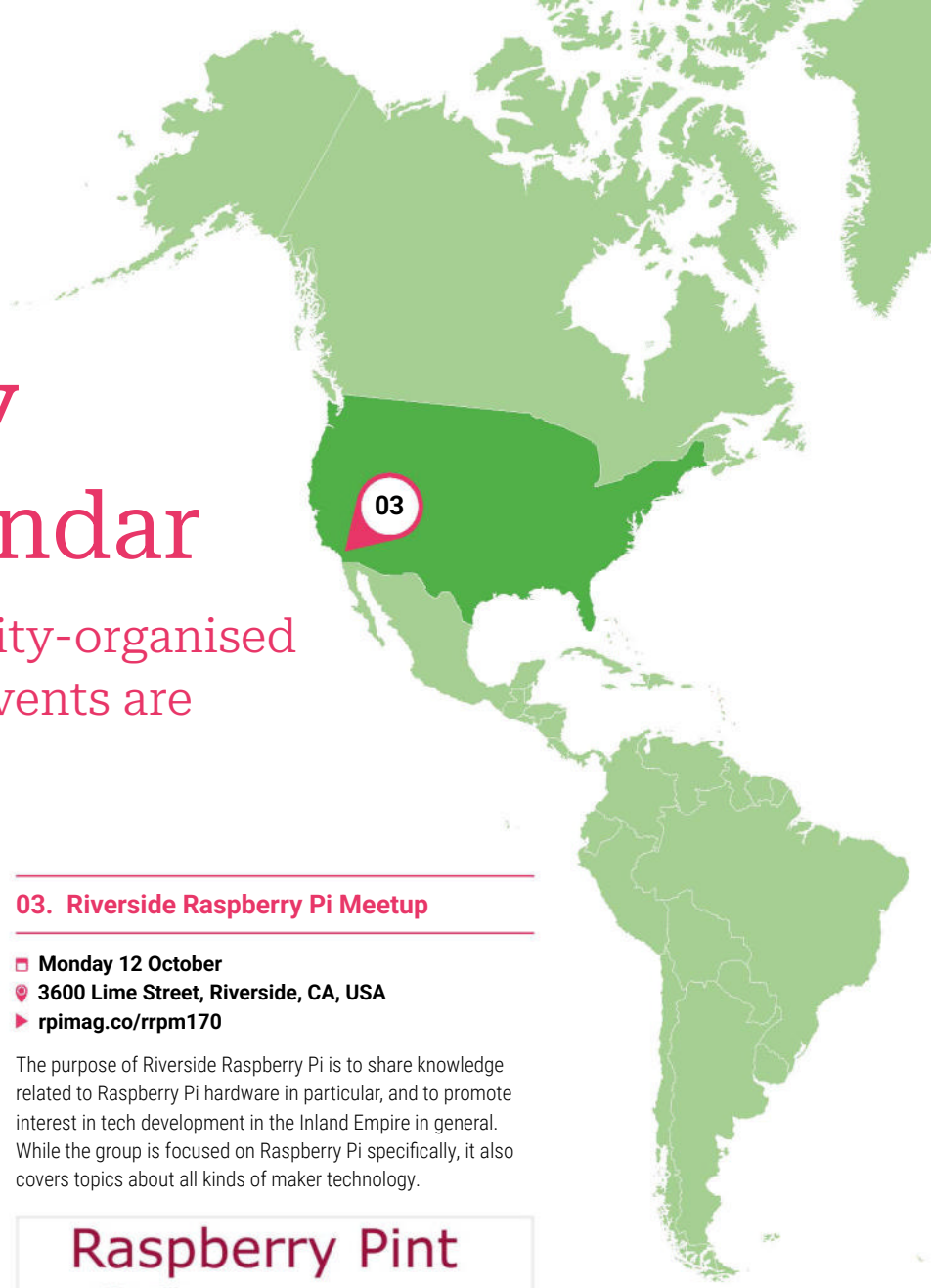


Subscribe online:

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. Wirral Code Club (Birkenhead)

Tuesday 6 October
Birkenhead Central Library, Birkenhead, UK
rpimag.co/wcc170

Wirral Code Club is a series of free volunteer-led sessions allowing kids aged 8–13 to learn how to code. Your young coders can choose to follow Scratch, Python, micro:bit, and HTML & CSS project pathways with hands-on projects from the Raspberry Pi Foundation.



02. Pi JAM Mangaluru

Saturday 10 October
Mannagudda, Mangalore, India
rpimag.co/pjm170

Whether you're just discovering Raspberry Pi or already a passionate fan, the organisers invite you to join them for an exciting event to explore digital computing in a whole new way. Learn how Raspberry Pi can bridge the gap between an idea and a physical product.

03. Riverside Raspberry Pi Meetup

Monday 12 October
3600 Lime Street, Riverside, CA, USA
rpimag.co/rrpm170

The purpose of Riverside Raspberry Pi is to share knowledge related to Raspberry Pi hardware in particular, and to promote interest in tech development in the Inland Empire in general. While the group is focused on Raspberry Pi specifically, it also covers topics about all kinds of maker technology.



04. October Raspberry Pint

Tuesday 27 October
Online
rpimag.co/pint170

Raspberry Pint is a community of makers who meet to share builds and knowledge. Most of the presentations are about building personal or professional projects with Raspberry Pi and other maker technology. They also welcome presentations about skills and techniques such as AI coding, PCB design, software development, 3D printing, soldering, etc.



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

05. Maker Faire Rome

- **Friday 23 October to Sunday 25 October**
- **Gazometro Ostiense, Rome, Italy**
- **rpimag.co/mfrome26**

The team are proud to partner with Raspberry Pi Approved Reseller Melopero to be at Maker Faire Rome again in 2026. Come and meet members of the Raspberry Pi team, learn about the latest products, and share what you've made with Raspberry Pi technology.



Official Raspberry Pi Event

Win a Raspberry Pi Development Kit for Compute Module 5

This comprehensive kit includes a Compute Module 5 and Compute Module 5 IO Board, together with all the essential accessories to kick-start your product design.



Head here to enter:

rpimag.co/win

Learn more:

rpimag.co/cm5devkit

Terms & Conditions

Competition opens on **23 September 2026** and closes on **22 October 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!

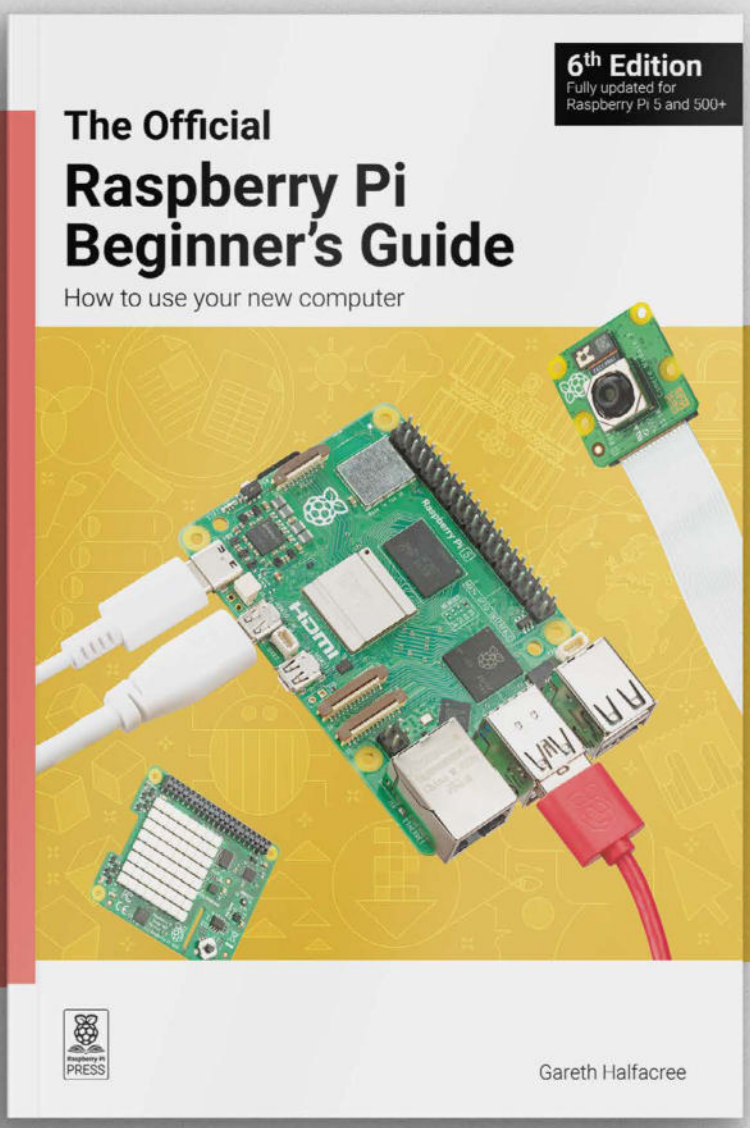


Email Charlie Milligan at: **charlotte.milligan@raspberrypi.com**

Learn coding

Discover how computers work

Build amazing things!



rpimag.co/beginnersguide

Official Magazine
#171

Operating Systems

There's more to life than Raspberry Pi OS

On sale 22 Oct

PLUS!

InkyCal V3 reviewed

Build an Apollo computer

A camera-enabled web server



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, Tom Dewey, Rosemary Hattersley, Nicola King, Phil King, Rob Miles, Naush Patuck, Nik Rawlinson

Publishing

Publishing Director
Brian Jepson
brian.jepson@raspberrypi.com

Director of Communications
Helen Lynn

CEO
Eben Upton

Distribution

Seymour Distribution Ltd
2 East Poultry Ave,
London EC1A 9PT
+44 (0)207 429 4000

Subscriptions

Unit 6 The Enterprise Centre
Kelvin Lane, Manor Royal,
Crawley, West Sussex, RH10 9PE
+44 (0)1293 312193
rpimag.co/subscribe
rpipress@subscriptionhelpline.co.uk

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)



Freedom of information

Democratising computing involves free documentation, such as free issues of this very magazine

I was catching up with an old university friend of mine recently who, nearing age 40 much like myself, had found himself back at university working on a PhD involving AI regulation. "Oh, I probably should have told you this," he says as we chat. "I moved completely over to a Raspberry Pi six months ago and haven't looked back."

He proudly showed me his Raspberry Pi 5 and fold-out Bluetooth keyboard, a setup he plugs into the monitors in the labs (the same labs I probably used with my own Linux laptops about 20 years ago). He's not actually using his Raspberry Pi for AI tasks though, just for work.

Page turners

As you'd expect, he's been doing a lot of reading as part of his studies, along with a lot of reading for pleasure. He's been signing up for various library services and apps such as Libby (hello to the readers currently reading this from the Libby app) and just enjoying all that he can get from them, from banned books translated into English to esoteric history books.

"I heard about a book that is a detailed history of this city from the 1800s to early 1900s," he explains. "It turns out the only copy was in the university library, so I decided to go read it!"

Apparently it was very dry, but the ability to go and find it is an important thing to preserve.

> *Having access to previous issues means all the tutorials can be used by everyone in perpetuity*

Free magazines

That's why we like to have the magazines available for free. Having access to all the previous issues means all the various projects and tutorials from over the years can be used by everyone in perpetuity, inspiring folks to come – even if some of the project ideas may have been far surpassed by technology. They're also all available on the Bookshelf app on Raspberry Pi OS, so my friend can easily access the PDFs from there when studying too.

Without this information being freely available, using a Raspberry Pi wouldn't be possible for everyone. That's why we continue to write and update stuff to make sure everyone can.

Rob Zwetsloot – Author
Rob confidently misidentified a Raspberry Pi 5 recently; to be fair, he couldn't see the side of the board that would otherwise have a 3.5mm jack.

rpimag.co

HiPi.io

HighPi 5S

The new case from the HiPi team



Reliable case for complex projects:

- Built for Raspberry Pi 5
- Rapid tool-free assembly and disassembly
- Large internal volume for HATs even with the active cooler installed
- Camera cable slot to connect to external cameras
- Power button and external status LED
- Passive & active cooling options
- Secure microSD cover
- VESA mount support

Customization options for volume orders:

- Output ports molded to your exact specs
- Logo printing for a branded finish

Available at these great Pi stores:

PiShop.us PiShop.ca ThePiHut Pi-Shop.ch Welectron.

Contact your favorite Pi store if it's not listed here

PiKVM

Remote control redefined

Manage your servers or PCs remotely!

Powered by Raspberry Pi

PiKVM V4 Mini

Small, cost-effective, and powerful!

- Power consumption in idle mode: just 2.67 Watts!
- Transfer your mouse and keyboard actions
- Access to all configuration settings like UEFI/BIOS
- Capture video signal up to 1920x1200@60 Hz
- Take full control of a remote PC's power

PiKVM V4 Plus

The most feature-rich edition

- More connectivity
- Extra storage via internal USB3.0
- Upgraded powering options
- More physical security features
- Extra HDMI output
- Advanced cooling solution



A cost-effective solution for data-centers, IT departments or remote machines!

Available at the main Raspberry Pi resellers

core ELECTRONICS · PiShop.ca · RPishop.cz · Welectron. · elektorstore · PIMORONI

Pi-Shop.ch · The PiHut · PiShop.us · CloudFree · KUBII · Cytron Marketplace

ROBU.IN Your ideas, Our Parts · NETTOP COMPUTERS & ELECTRONICS · netsket mall · DIREKTRONIK · RaspberryPi.dk

HiPi.io

No reseller in your country?
Check shop.hipi.io (import fees might apply).

List of official resellers by country: