Go winter stargazing **with Raspberry Pi telescopes**

Build a submarine **with a camera system**

Real-time **object detection with Ultralytics YOLO**

Raspberry Pi

Official Magazine
#162 | February 2026

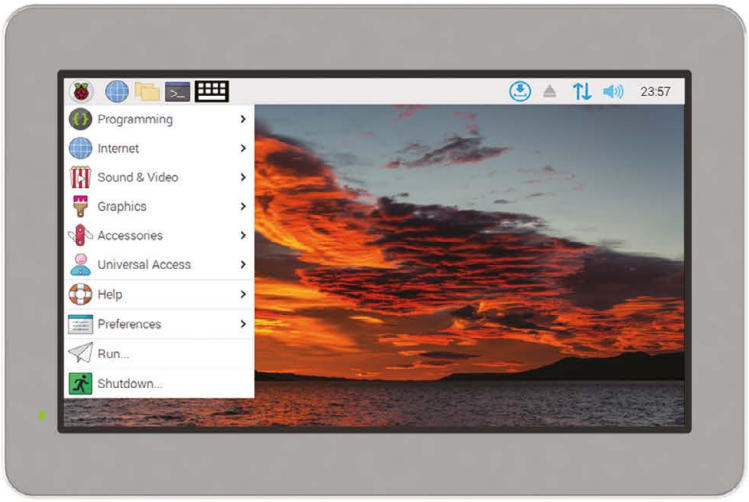
PROGRAM A ROBOT ARM

Pick up and place objects with Python code

02 **£7.99**
ISSN 2977-4403 (Print)
9 772977 440004

Industrial Raspberry Pi

ComfilePi



UL LISTED

Powered by Raspberry Pi

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

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

Visit www.comfiletech.com

COMFILE TECHNOLOGY

Welcome to Raspberry Pi Official Magazine

Editor
Lucy Hattersley
Due to the powers of magazine time travel, Lucy has been mostly doing Christmas things this month. Thankfully, normality will return in January. Happy new year everyone.

rpimag.co

In this issue, we've got both space and robots. Robot arms are where code meets reality. The robots for Raspberry Pi range from extremely cheap beginner builds up to engineering arms that start to feel at home in an industrial environment. In a world of increasing automation, it's fantastic to get an overview of how robotics works under the hood.

Rosie takes us on a tour of stargazing projects. These enable you to use Raspberry Pi's High Quality Camera system with a telescope. You can use code to automate tracking of the night sky. Use Raspberry Pi to record and decipher what's out there.

Whether you're picking things up for the first time or gazing into the vast cosmos, you'll find something thrilling to build this month.

Lucy Hattersley – Editor

PCBWay

FULL-SERVICE ELECTRONICS MANUFACTURER

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

CUSTOMIZED TECHNICAL PARTS & PCB MANUFACTURED

WHY CHOOSE PCBWAY?

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

Hassle-free ordering

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

On-time shipping

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

Quality assurance

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

Customer support

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

Scan the QR code for more details!

CONTACT US:

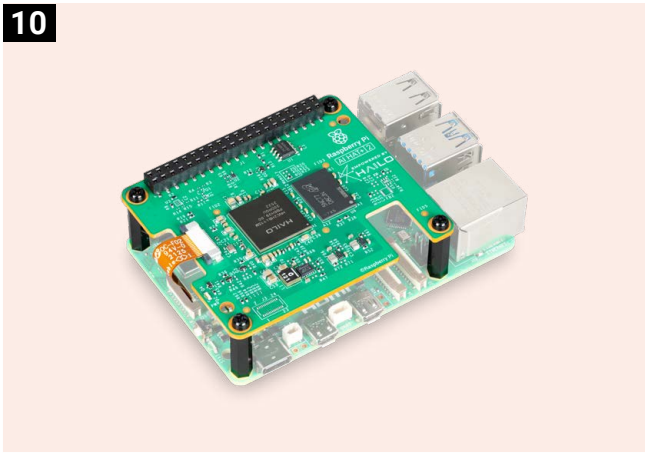
www.pcbway.com

service@pcbway.com

Contents

World of Raspberry Pi

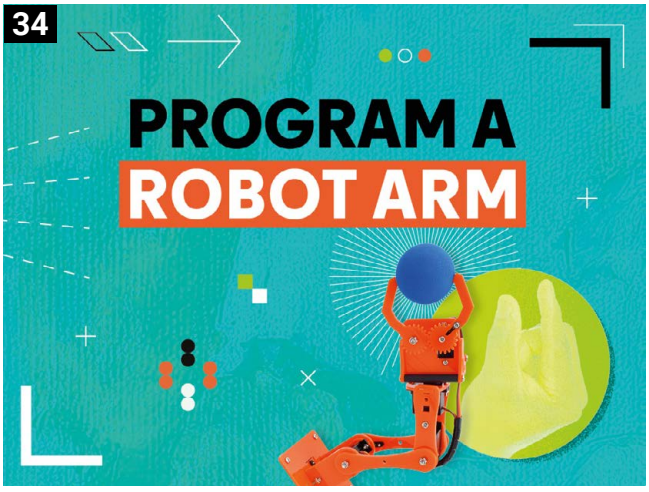
Project Showcase

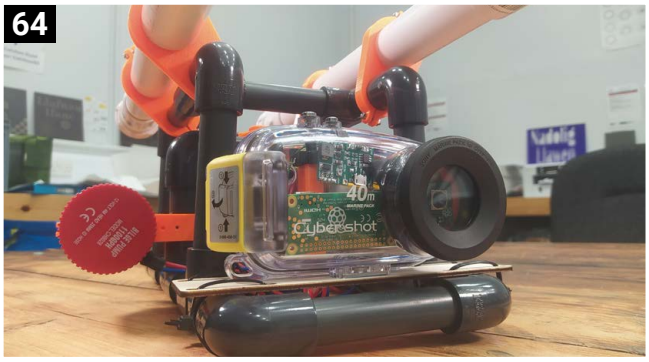


Raspberry Pi (ISSN No: 2977-4403, USPS No: 25-672) is published 12 times per year by Raspberry Pi Ltd, and distributed in the USA by Asendia USA, 701 Ashland Ave, Folcroft PA. Application to Mail at Periodicals Postage. Prices is pending at Philadelphia, PA and additional mailing offices. POSTMASTER: send address changes to Raspberry Pi, 701 Ashland Ave, Folcroft, PA. 19032.

Top Projects

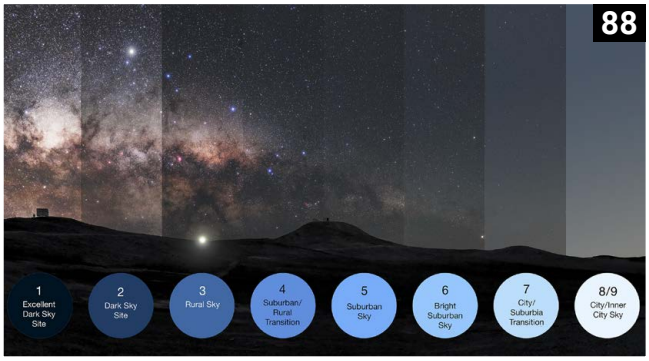
Feature





Tutorials

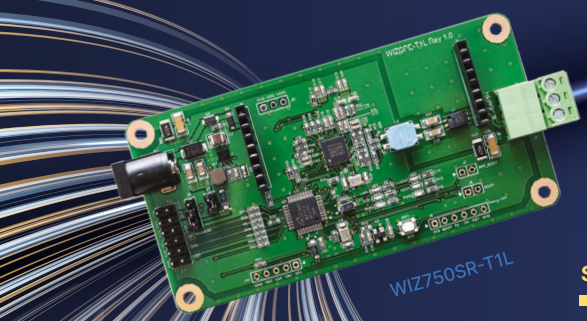
Feature



WIZnet

1 km Ethernet over a single pair

Simplify installations and lower TCO



Single Pair Ethernet → 1,000m

100m ← Traditional Ethernet

IEEE 802.3cg (10BASE-T1L) compliant
Data + power over one pair
Built for industrial environments

WIZ750SR-T1L

Single Pair Ethernet Solution

Get Started

SPE / 10BASE-T1L at a Glance

- Standard: IEEE 802.3cg (10BASE-T1L)
- Cabling: Single-pair (2 conductors); PoDL options for combined power + data
- Reach/Speed: Up to 1 km at 10 Mbps (ideal for industrial IIoT)
- Stack: Integrates with standard IP networking (TCP/UDP, IPv4)

Why Better than RS-485 / Traditional Ethernet

Versus **RS-485**

- Delivers 10 Mbps communication up to 1 km distance
- Native IP networking
- Leverage standard IT tools and monitoring
- TCP/IP, UDP, IPv4/v6 protocol support

Versus **10/100Base-T**

- Up to 1 km vs 100m distance
- Single-pair cabling for lighter, lower-cost runs
- Simpler installation with fewer connection points
- Lower infrastructure costs

SPE Market Outlook

- SPE market projected to reach $5.78 billion by 2029
- 11.1% CAGR growth (2025-2029)
- Key driver for IIoT and Industry 4.0 connectivity
- Enabling technology for 27 billion IoT devices by 2025

Why WIZ750SR-T1L

Serial-to-Ethernet protocol converter: IP-enable your serial devices fast

Industrial grade temp range, link/TCP status pins, separate Data/Debug UART

Easy configuration: AT commands, PC Config Tool, Web UI, CLI

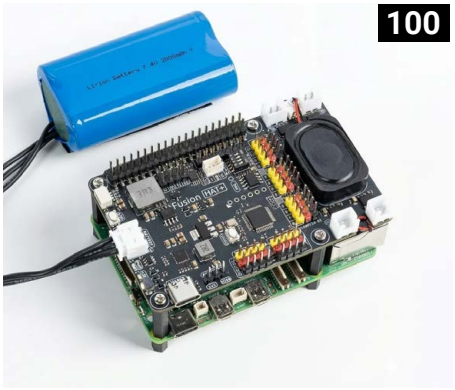
Applications

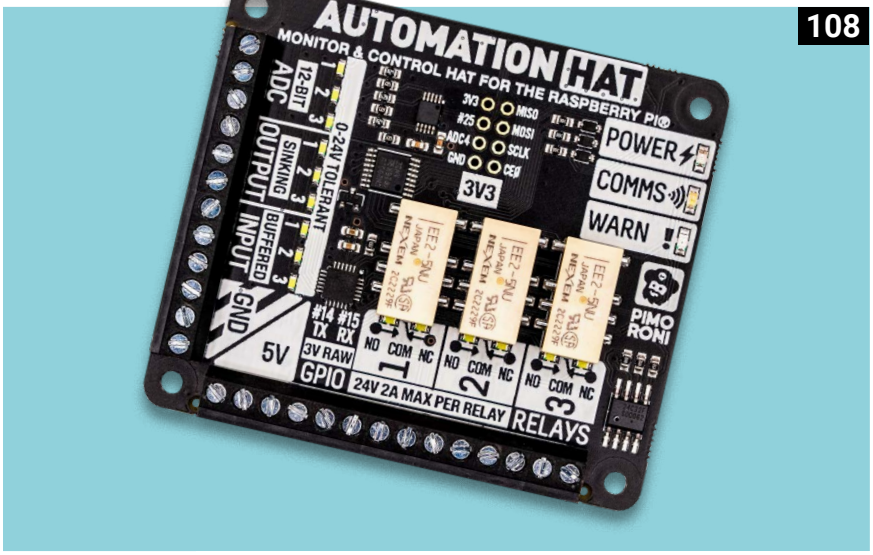
Process/Factory Automation → Sensors, actuators, valve positioners, RTUs

Building /Infrastructure → BMS, energy /environmental monitoring, long-reach edge nodes

Legacy Digitalization → Reuse existing single-pair wiring for retrofit projects

Notes: Distance and performance depend on cable type, topology, and EMI/EMC environment. PoDL functionality requires compatible IEEE 802.3cg equipment.





Reviews

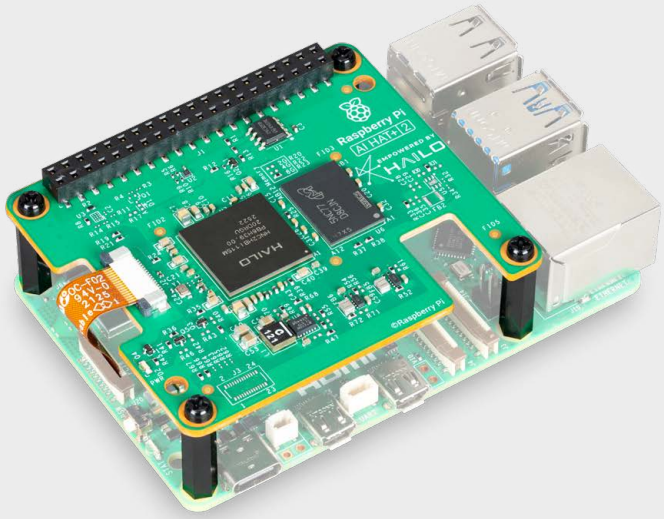
Raspberry Pi Community

Competition

AI HAT+ 2 released

Run LLMs and VLMs locally on Raspberry Pi.

By **Lucy Hattersley**



▲ AI HAT+ 2 connects to Raspberry Pi 5 and features a Hailo-10H AI accelerator and 8GB of on-board RAM

Raspberry Pi has announced a new AI accelerator board for Raspberry Pi, the AI HAT+ 2 (rpimag.co/aihat2).

Available to buy now for $130/£125, AI HAT+ 2 features a new Hailo-10H AI accelerator and 8GB of on-board RAM.

Raspberry Pi AI HAT+ 2 is designed to bring generative AI capability to Raspberry Pi 5.

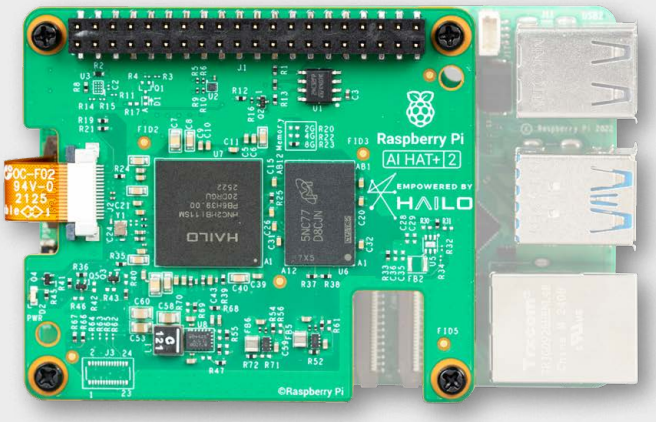
Thanks to its 8GB of dedicated on-board RAM, AI HAT+ 2 is ideal for running large language models (LLMs) and vision-language models (VLMs).

As with its predecessor, these models run locally and securely without the need for cloud computing and an internet connection. This maintains security and promotes resilience as the model can keep running in an edge environment (such as a factory floor) without networking.

The 8GB RAM on board AI HAT+ 2 leaves the host Raspberry Pi 5 free to handle other tasks.

This ensures “generative AI workloads run smoothly on Raspberry Pi 5,” says Naush Patuck, Software Engineering

- Run large language models and vision-language models offline and integrate generative AI functionality with your hardware setup



Manager at Raspberry Pi. "AI HAT+ 2 operates reliably and with low latency, maintaining the privacy, security, and cost-efficiency of cloud-free AI computing that we introduced with the original AI HAT+."

Large language models

For many people the marquee feature of AI HAT+ 2 will be the ability to run large language models (LLMs) locally. This enables you to run generative pretrained transformers, such as chatbots, locally on Raspberry Pi 5.

There are five LLM models available at launch including DeepSeek, Llama, and Qwen (see 'Example applications'). Users can also train custom vision models or fine-tune generative AI models to suit their application, such as speech to text, translation, or visual scene analysis.

"For vision-based models – such as Yolo-based object recognition, pose estimation, and scene segmentation – the AI HAT+ 2's computer vision performance is broadly equivalent to that of its 26-TOPS predecessor, thanks to the on-board RAM," explains Patuck. "It also benefits from the same tight integration with our camera software stack (libcamera, rpicam-apps, and Picamera2) as the original AI HAT+.

"For users already working with the AI HAT+ software, transitioning to the AI HAT+ 2 is mostly seamless and transparent," says Patuck.

Vision-language models

The most popular examples of generative AI models are LLMs like ChatGPT and Claude, and text-to-image/video models like Stable Diffusion and DALL-E.

> *There are five LLM models available at launch including DeepSeek, Llama, and Qwen*

More recently, VLMs have gained a higher profile. These combine the capabilities of vision models and LLMs. AI HAT+ 2 will bridge Raspberry Pi Camera hardware and large language models, enabling you to ask Raspberry Pi directly what is in the image, using an LLM.

Train your model

Raspberry Pi makers can fine-tune AI models for specific use cases using the Hailo Dataflow Compiler (visit **rpimag.co/hailodataflow** for details).

Hailo's GitHub repo features examples, demos, and frameworks for vision- and GenAI-based applications, such as VLMs, voice assistants, and speech recognition (**rpimag.co/hailogit**).

You can also find documentation, tutorials, and downloads for the Dataflow Compiler and the hailo-ollama server in Hailo's Developer Zone (sign up for free at **rpimag.co/hailodev**).

Example applications

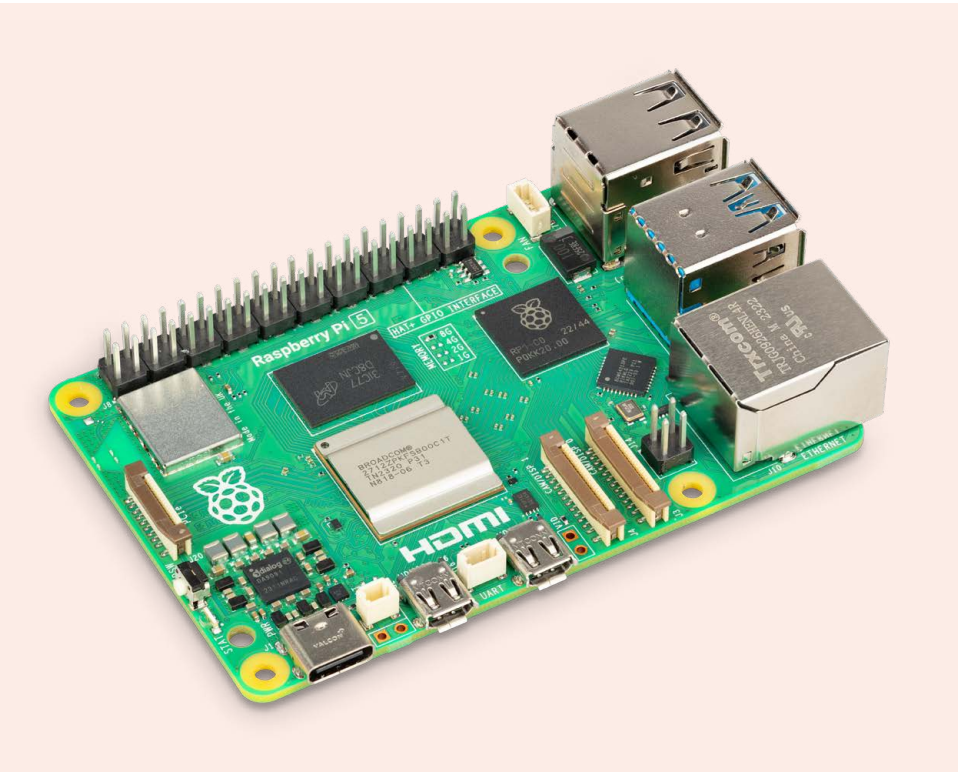
The following LLMs will be available to install at launch:

Model	Parameters/ size
DeepSeek-R1-Distill	1.5 billion
Llama3.2	1 billion
Qwen2.5-Coder	1.5 billion
Qwen2.5-Instruct	1.5 billion
Qwen2	1.5 billion

More (and larger) models are being readied for updates, and should be available to install soon after launch.

1GB Raspberry Pi 5

New model introduced to offset price rise of LPDDR4. By **Lucy Hattersley**



▼ The new Raspberry Pi 5 1GB variant enables users to access Raspberry Pi 5 power at a low cost

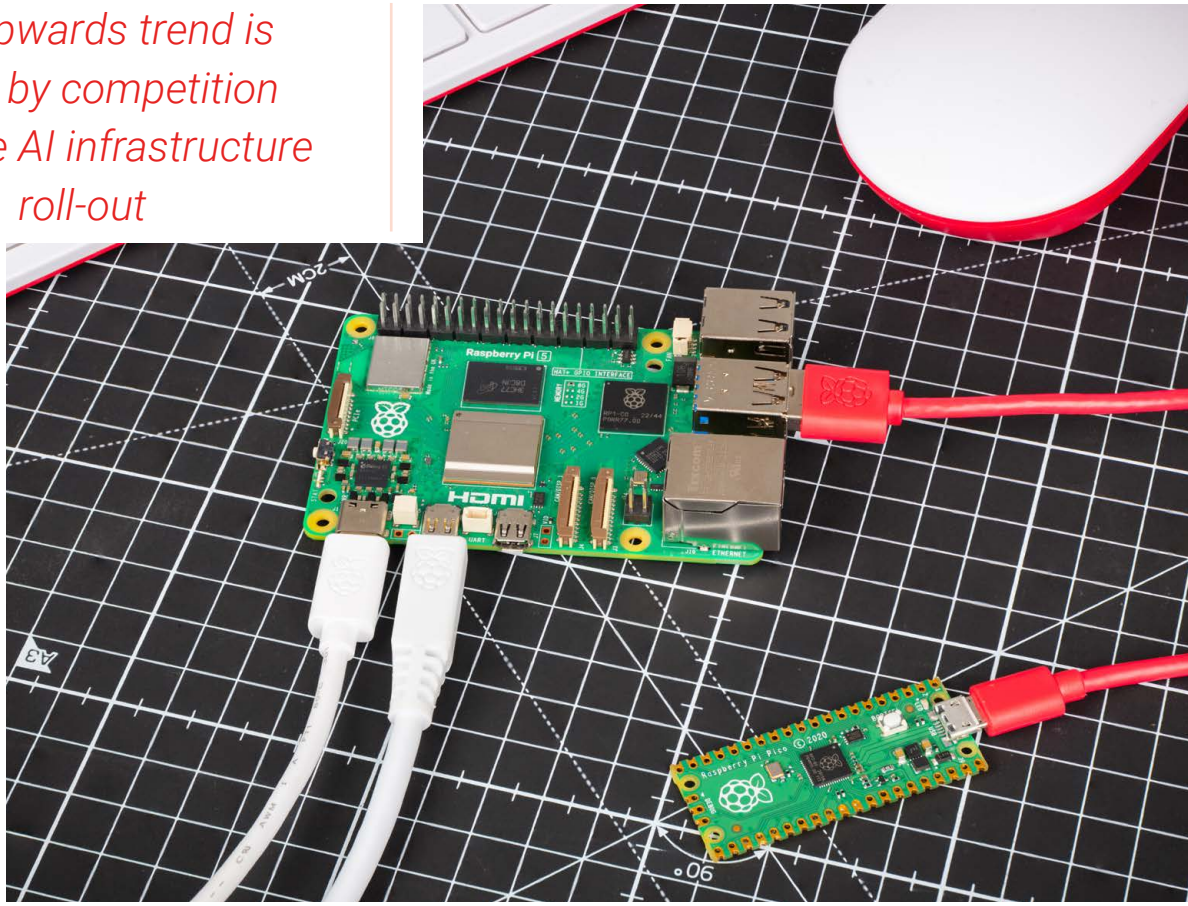
Raspberry Pi introduced a new model of Raspberry Pi 5 on 1 Dec 2025. The 1GB model is priced at $45 to maintain the low price amid rapid increases in the global price of DRAM (dynamic random-access memory).

"Our mission is to put high-performance, low-cost, general-purpose computers in the hands of people all over the world," wrote Eben Upton, CEO and co-founder of Raspberry Pi.

The original Raspberry Pi 1 with 256MB RAM launched in 2012 at just $35, and Raspberry Pi currently sells the Raspberry Pi 4 with 1GB RAM at the same price point. Meanwhile, it has introduced new computing products at lower price points, from the $10 Raspberry Pi Zero to the $4 Raspberry Pi Pico.

In 2025, there was an unprecedented rise in the cost of LPDDR4 (Low-Power Double Data Rate 4) memory (**rpimag.co/memcost**). "The pricing for PC RAM has gotten so ridiculous that a pack of 64GB DDR5 memory can now easily exceed $500, more than a Sony PlayStation 5," reports PCMag (**rpimag.co/pcmagram**).

The upwards trend is driven by competition from the AI infrastructure roll-out



▶ Raspberry Pi 5 and Raspberry Pi Pico offer low-cost computing options

Short memory

To offset the price rises, Raspberry Pi has also announced price increases to some Raspberry Pi 4 and Raspberry Pi 5 products. These largely mirror the increases announced in October 2025 for the Compute Module products. This "will help us to secure memory supplies as we navigate an increasingly constrained market in 2026." explains Upton.

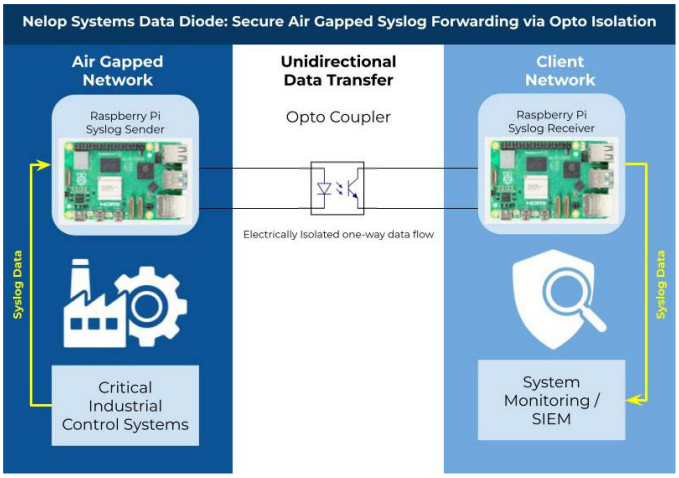
The upwards trend is "driven by competition from the AI infrastructure roll-out," explains Upton. Adding that it "is painful but ultimately temporary. We remain committed to driving down the cost of computing and look forward to unwinding these price increases once it abates."

Price changes

Product	Density	Old price	New price
Raspberry Pi 4	4GB	$55	$60
Raspberry Pi 4	8GB	$75	$85
Raspberry Pi 5	1GB	-	$45
Raspberry Pi 5	2GB	$50	$55
Raspberry Pi 5	4GB	$60	$70
Raspberry Pi 5	8GB	$80	$95
Raspberry Pi 5	16GB	$120	$145

Data Diode

One-way, secure data transfer without a traditional network connection. **Rob Zwetsloot** sees the light



▲ A diagram of how the system works

Maker
Neil Chandler
An IT veteran who worked in investment banking, defence, and government. His core focus has always been Linux and HPC (High-Performance Computing).

rpimag.co/datadiode

A diode is a simple electronic component that only allows current to travel in one direction in a circuit. It's something you learn about very early on in electronics, and the basic concept of it having 'one-way travel' should clue you into what this project is about.

"We have two isolated systems, and we need to get data from one side to the other with no possibility of a return signal," Neil Chandler, who runs Nelop Systems, tells us. "They can't be network-connected, and everything needs to stay secure. With this setup, we can move system logs from one system into another, and because the channel is one-way, nothing can leak back. This is especially useful for air-gapped networks."

Neil describes Nelop Systems as a company that helps "organisations solve awkward technical problems; for example, keeping old or vulnerable systems running securely so they remain compliant". So while a client's old industrial machinery software was upgraded as much as possible, it still had security issues meaning it would need to be isolated.

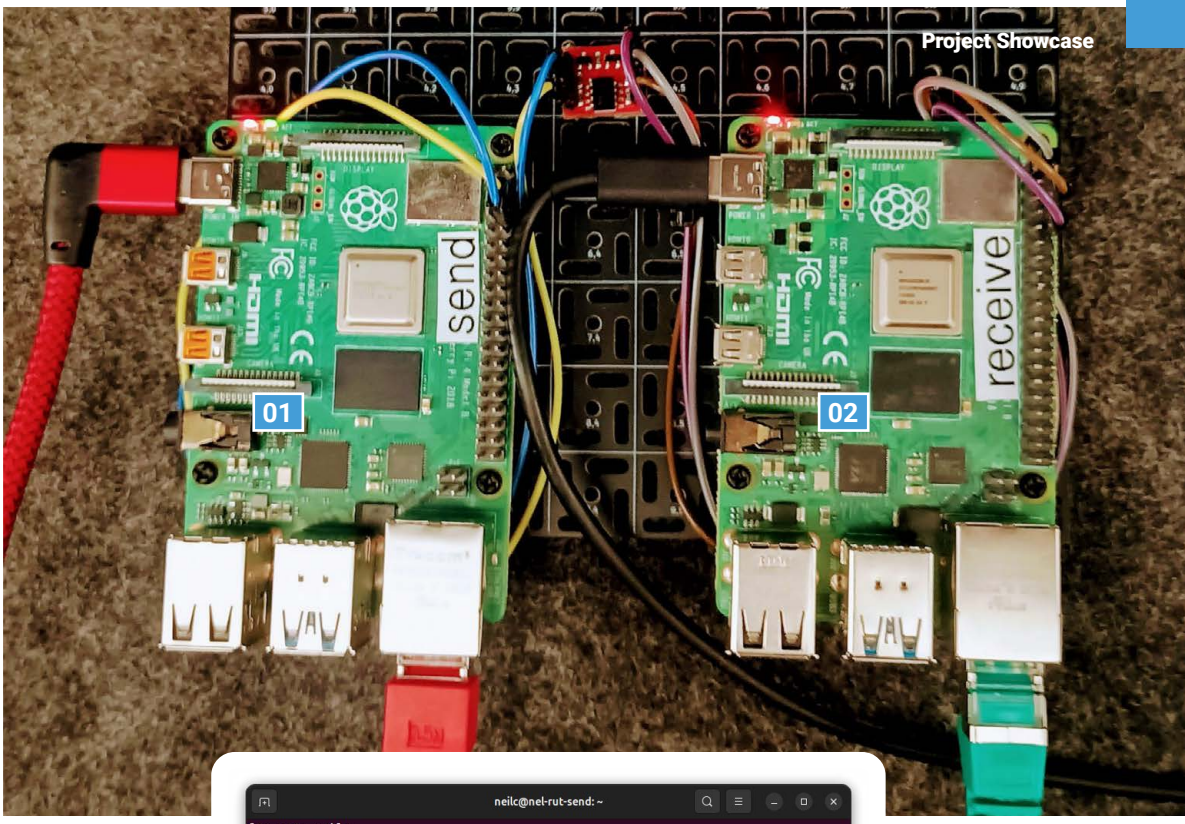
"That created a problem because they had to physically visit each machine every day to extract data," Neil explains. "After a chance conversation about the issue, I started working on this project so they could get the data out in real time."

Neil was inspired by his experience with audio electronics and opto-isolators to create this air-gapped data diode.

Future proofed

The process uses two Raspberry Pis for the data transfer: "one Raspberry Pi acts as the sender. A Python script reads the system's syslog messages, processes each one, and sends it out of GPIO 14 (TXD)," Neil says. "An opto-isolator sits between GPIO 14 on the sender and GPIO 15 on the receiver.

> *Because the channel is one-way, nothing can leak back*



01. The Raspberry Pi boards are not connected via wires – data is transferred over light signals

02. The system only uses a couple of GPIO pins on each Raspberry Pi

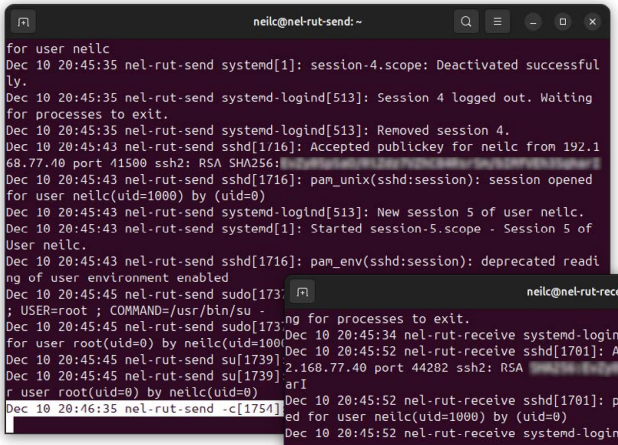
"The second Raspberry Pi is the receiver. A Python script listens on GPIO 15 (RXD), processes anything that comes in, and then generates a tagged syslog message. That Raspberry Pi is set up to forward syslog messages into an internal system where they get handled normally. Both Raspberry Pi boards are locked down. No Wi-Fi, no Bluetooth, and we've added alerts for any tampering."

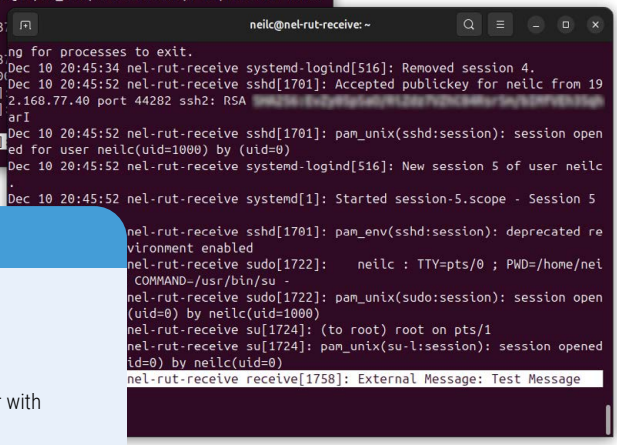
Using this method allows for the system to have more future proofing than a wired alternative – and using Raspberry Pi meant the whole system was easy to develop, cheap, and easy to maintain if there are any hardware issues.

Working limitations

The system is in place and working as expected, with reliable data transfer. More units are being planned for the company – and it's also going to be used by the security team in the long run.

"There's a bit of latency across the whole chain, so I wouldn't use this method for anything that needs millisecond accuracy," Neil mentions. "There are also throughput limits, though with testing I'm sure the baud rate could be increased."





◀ Data being sent via the sending Raspberry Pi

▼ Data being received by the second Raspberry Pi

Quick FACTS

- The client was having to visit machines physically
- Neil was previously unfamiliar with the GPIO…
- … Leading to trial and error before using the UART pins
- The baud rate was slowed to counter data corruption
- Using Raspberry Pi helped with rapid prototyping

Friends

Precarious-seeming graphical goings-on courtesy of Raspberry Pi impress **Rosie Hattersley**

▲ Each 2m-tall frame balances on a single narrow edge, making it prone to toppling over

Makers
Kevin Rathbone and Antoine Espinasseau
Kevin Rathbone's expertise in pharmaceutical automation and robotics led to him working with kinetic sculptor Antoine Espinasseau.

robotae.com
antoineespinasseau.com

We are somewhat envious of Dr Kevin Rathbone's postdoctoral studies: they involved designing and building wheeled and flying autonomous machines. His Sheffield University-based PhD in robotics and early AI involved a crash course in engineering and led to Kevin setting up businesses in mechatronics and automating pharmaceutical hardware. He now specialises in robotics-based projects and for the past few years has been involved in the Raspberry Pi 3B+ 'Friends' art project, which demonstrates various artistic balancing acts.

Artistic endeavours

In 2021 Kevin was approached by Belgian sculptor (and graduate from Versailles prestigious National Superior School of Architecture) Antoine Espinasseau who was keen to create a self-balancing sculpture using a reaction wheel to balance a pendulum. Crucially, he wanted this balance wheel at the top, adding a fair amount of technical challenge to his kinetic sculpture plans. Antoine had spotted a similar trick used as the basis of a camera-stabilising gimbal on Kevin's Robotae website. An exploratory phone call later and they were both keen to get started. The plan was for several two-metre-tall tubular carbon fibre frames to be attached to reaction wheels and each fitted with a Raspberry Pi 3B+, an accelerometer, and gyroscope. A screen-printed image of a classical sculpture would be displayed on each frame. The delicately balanced fragile rice paper prints would be in contrast to the hefty stone plaster and bronze from which the depicted sculptures had been hewn. Raspberry Pi would check on the location and angle of each frame every ten seconds using the gyroscope and accelerometer, and trigger the reaction wheel as needed. The finely balanced frames would of course be prone to movement – even a person wafting past close by could easily destabilise them.

Having learned to code on a Commodore VIC-20, Kevin became interested in Raspberry Pi

Constant readjustments

Having learned to code on a Commodore VIC-20, Kevin became interested in Raspberry Pi as a convenient platform for early prototypes and rapidly developing test rigs. "It's very easy to interface with hardware and log data or add a GUI." Having tried other single-board computers, Kevin soon found Raspberry Pi's reliability and large community of fellow makers and developers just what he

01. Images of sculptures printed on rice paper waft in the breeze, but never quite fall over

02. Raspberry Pi 3B+, a reaction wheel, and IMU provide controlled movement

Warning!
Electrical Safety
Please be careful when working with electrical projects around the home. Especially if they involve mains electricity.
rpimag.co/electricalsafety

needed when it came to starting out with code to kick-start new projects or control add-on hardware. Antoine designed the reaction wheels and the basic structure, preloading the carbon fibre tubes to keep them apart and increase the resonant frequency. This meant the control loop would have a sufficiently high bandwidth and response time to ensure the frame was balanced. The setup also includes a servo-driven geared motor and an IMU. A PI (proportional-integral) controller determines the torque needed to move the frame towards its target angle.

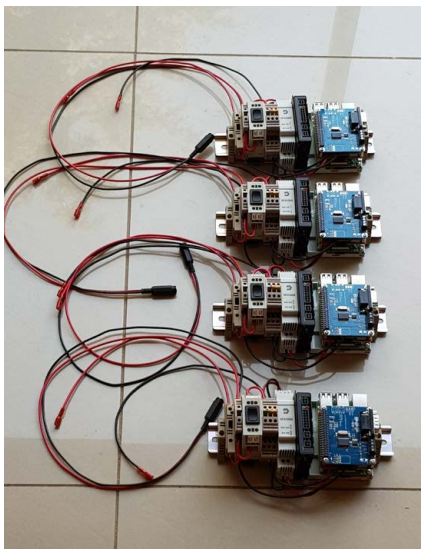
Given both men's commitment to other projects, it was a relatively fast six months between Antoine and Kevin's first chat and the initial prototype in which Kevin was able to demonstrate the carbon-fibre frame balancing "for a few seconds". Adjustments to the centre of mass (which was originally too close to the hub axis) fixed issues relating to it disturbing the control, while Kevin added a complementary filter to adjust for the fairly common accelerometer drift. Another two years would elapse before Friends saw its first outing at Antoine's own studio in Brussels, with further private views and boutique exhibitions in 2025, most recently at Faugères in southern France and the gallery of architecture in Paris.

▶ The ancient cave was a popular but rather chilly installation setting

There's a video of the kinetic sculptures at **rpimag.co/friendsvid**.

Quick FACTS

- Each carbon-fibre display frame cost around €2,000
- There was a two-year waitlist for the zero backlash Harmonic Drive gear mechanism
- The biggest challenge was identifying every possible cause of imbalance
- Extreme cold at one installation site in a cave almost broke it
- A Raspberry Pi USB power hub saved the day



▶ Each carbon-fibre display frame has its own Raspberry Pi 3B+, IMU, servo motor, and reaction wheel

SmartCoop: controlling chickens with Java

Reduce the effort to keep chickens safe, fed and watered on a hobby farm. By **Frank Delporte**

Maker
Dave Duncanson
Maker Dave is an ex-Royal Australian Airforce (RAAF) electronics technician and embedded software developer.

smartcoop.tech

Owning a small flock of chickens means you have to open and close the main door, collect the eggs, and make sure there is enough water and food. Given that most of this needs to be done daily, if you want to get away for more than a day or two, you need to arrange for someone to perform these tasks. One of the key design goals of this project was to ensure that the system was robust enough that its creator Dave could get away for up to a week, without having anyone physically attend the system, but also prevent the local foxes from getting to the chickens.

Dave started working on SmartCoop over ten years ago, and the current version contains the fourth generation of his custom-created PCB. With this new design, he could bump the system to use a Raspberry Pi Zero 2 W. The full system contains a wide variety of automated doors, light sensors, manual push buttons, water tank measurements, feeders, etc.

On the software side, an MQTT broker distributes the data and a Java application, based on Pi4J, uses live weather data from an API and measurements from the sensors to open and close the gates, keep track of the feeding, etc.

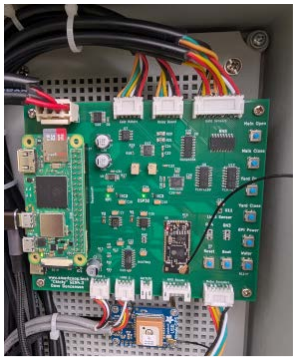
The project not only evolved because the technology changed, but also got influenced by nature! Dave was struggling with a fox that loved to hunt the chickens and became smart enough to know at which time the gates opened automatically. Because of this, the system was adapted to use the expected dawn and dusk time, but actually open and close the gate taking the light sensor measurement into account.

Another problem was caused by... the teenager. As anyone with kids will confirm, teenagers tend to forget a lot of important things, like closing the gate of the coop.

01. The main gate is opened and closed automatically

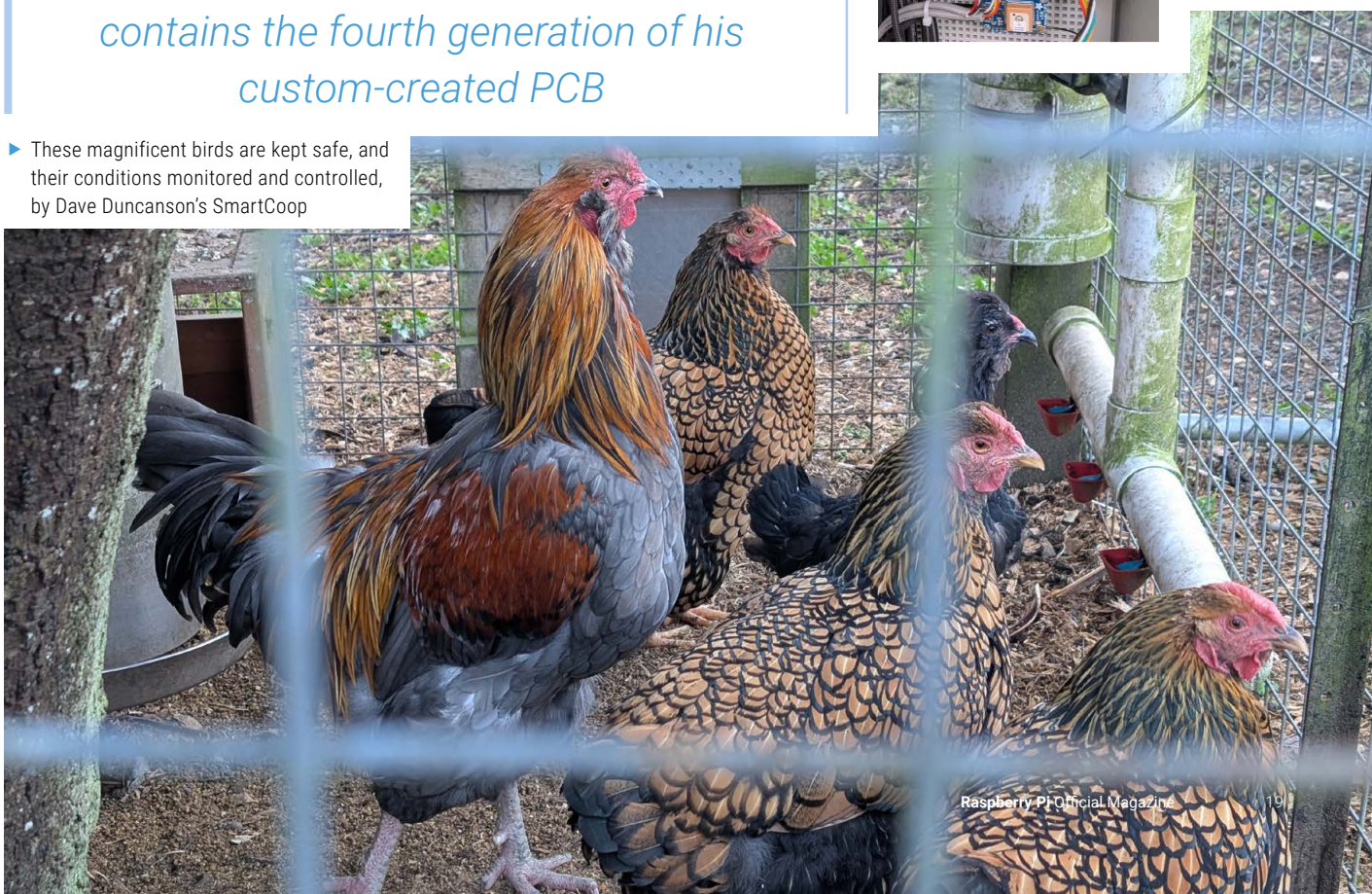
02. Sensors measure things like water and food levels

▼ Raspberry Pi Zero 2 W is mounted on a custom PCB with ports connected to multiple sensors



Dave started working on SmartCoop over ten years ago, and the current version contains the fourth generation of his custom-created PCB

▶ These magnificent birds are kept safe, and their conditions monitored and controlled, by Dave Duncanson's SmartCoop



The SmartCoop monitors this gate and the status of the food and water supply, and can inform a configurable number of people when something is wrong.

As a plan for the future, a UHF RFID reader could be added to the system, combined with an RFID ring for each of the chickens, to monitor if they are all inside at night. By adding such a reader to each of the laying boxes, it would even be possible to keep track of the most (or least) productive chickens.

Raspberry Pi + ESP32

Around 80% of the core functionality is handled by Raspberry Pi Zero 2 W via a Java application which uses the Pi4J library to control the GPIO pins and interact with I2C devices, stores data in an H2 database, and provides GPS and NTP functionality, event scheduling, and a template-based web interface.

The remaining work is done by the ESP32. Its initial key role was just to power Raspberry Pi on and off at preset configurable times to conserve battery via RTC interrupts. But its task got extended and it now also checks the door positions and motor encoders. Because a lot of existing Arduino examples also work on the ESP32, these were used to understand how some components are controlled, before porting the code to Java and Pi4J.

Chicken town

Dave is the first to admit that this solution is most likely over-engineered and therefore not cost-effective. But it's the perfect solution to fully automate his chicken coop. He has no plans to turn this into a commercial product; he shares both the software and hardware on Bitbucket.

Quick FACTS

- Raspberry Pi schedules the opening and closing times for the main door, based on preconfigured dawn and dusk times.
- The weather forecast is fetched from an API daily, and the doors stay closed when there is a lot of rain expected.
- At dawn/dusk, Raspberry Pi monitors the light level and notifies the ESP32 when the door can be opened or closed.
- A preconfigured timeout is set for the door opening/closing time, so if there is a sensor failure, the motor will be powered off.
- With a proximity sensor, Raspberry Pi can detect when the door is closed, and stores this into the H2 database.

▲ The chicken coop door opens and closes, controlled by a daylight sensor

▼ Not all wildlife in Australia wants to kill you; some animals, like this wallaby, just want to steal your chicken feed

▼ The feed level is monitored by sensing the weight of the container

Chicken run

1. During the day, the chickens are free to roam outside.

2. Several sensors, like this water level sensor, monitor multiple circumstances to keep the chickens safe with enough water and food.

3. When it gets dark, or intruders are detected, the gates close to keep the chickens inside.

SUBSCRIBE TODAY FOR JUST £10

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

SUBSCRIBER BENEFITS

Free delivery
Get it fast and for free

Exclusive offers
Great gifts, offers, and discounts

Great savings
Save up to 37% compared to stores

SUBSCRIBE FOR £10

Free Pico 2 / Pico 2 W

3 issues of Raspberry Pi Official Magazine

£10 (UK only)

SUBSCRIBE FOR 6 MONTHS

Free Pico 2 / Pico 2 W

6 issues of Raspberry Pi Official Magazine

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

Subscribe by phone: 01293 312193

Subscribe online: rpimag.co/subscribe

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

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

Solder fume extractor

By Arnov Sharma

rpimag.co/Fume-extractor

We made the switch to lead-free solder years ago, and while we don't have any first-hand evidence that it's better for our health than leaded solder, we trust the experts who say that breathing in lead fumes is bad for us and we should try to do less of it. If you're still using solder with lead in it, do give unleaded a try and do yourself a favour. It's not quite as easy to work with, but that's a small price to pay for the potential health and environmental benefits.

Unleaded solder obviously isn't made of rosewater and unicorn kisses though, so whatever you're using when you're sticking electronic components on to PCBs, you'll need some sort of fume extraction method. Prolific maker Arnov Sharma has come up with this fume extractor, built on a Raspberry Pi Pico and assembled out of 3D printed parts that he's designed himself.

At the core of this device is a repurposed CPU fan, a Raspberry Pi Pico, and a custom PCB driver board. Arnov also designed his own PCB to host the switches, and another one to hold the LEDs in the light – because you need good lighting when you're soldering components together. The body is all made from 3D printed PLA, including the diffuser that sits over the LEDs, and there's room for a replaceable filter for the fumes, so you're actually removing toxins from the air and not just cycling air around.

▲ This device has adjustable fan speeds and light levels, courtesy of Raspberry Pi Pico

Programming station

By Wake-Of-Chaos

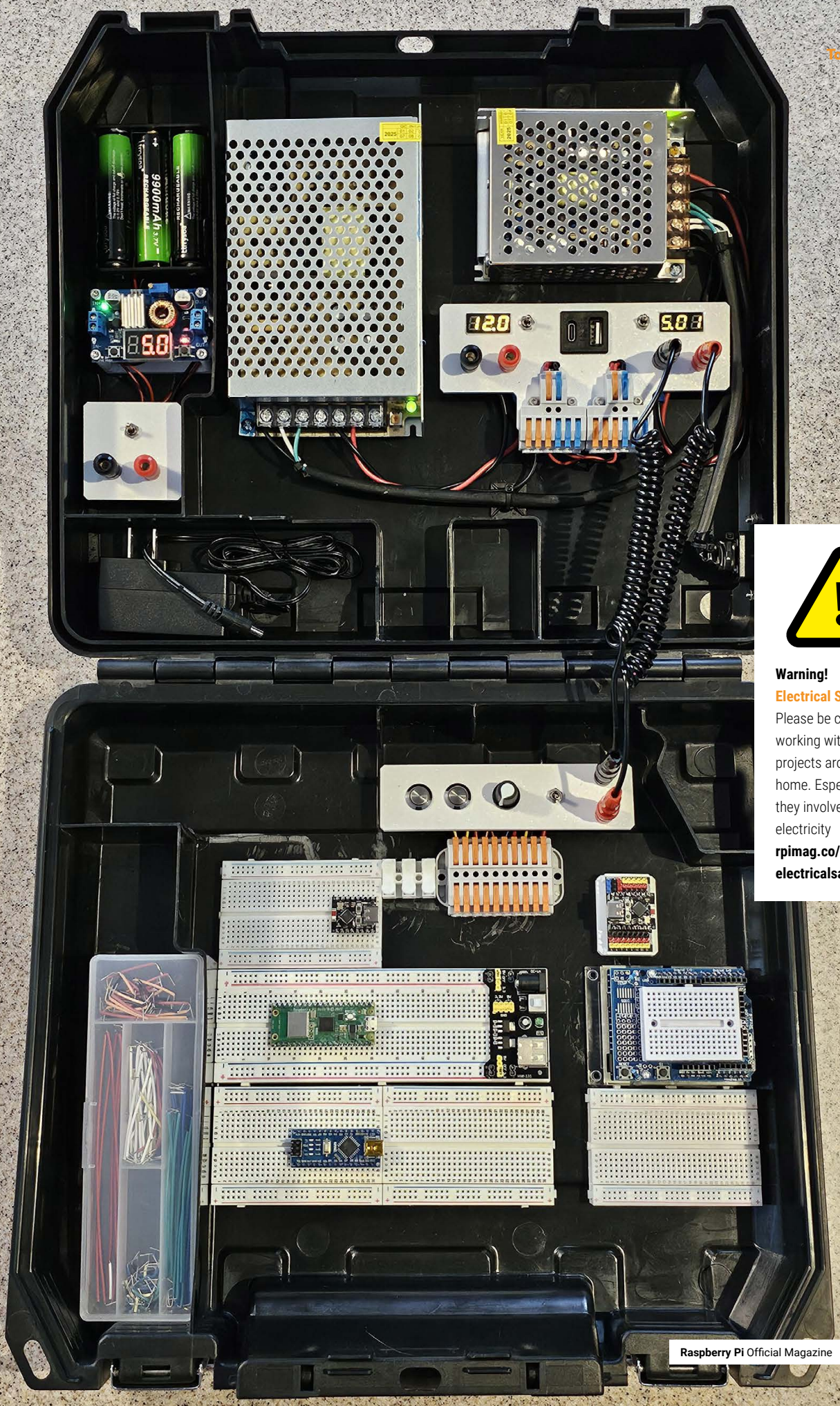
rpimag.co/Programming-station

The more ideas we have, the more stuff we buy to test those ideas. So far, so good. But when shiny new toys arrive and we don't use them immediately, they begin the process of transmutation into junk. A useful component that never gets used is junk. A tool for a job that you never do is junk. And a component or tool that you can't find instantly becomes junk – it's just taking up space, somewhere, with the even worse second-order effect that it makes your workspace smaller and less useful, and thus makes everything else harder to use. Oh no!

Fighting against this cycle of doom is Reddit user and electronics tinkerer Wake-Of-Chaos. They've been learning electronics with microcontrollers, and decided that there must be a better way than keeping all your components in the bottom of a shoebox. And so they've come up with this portable system that integrates breadboards, storage, and power supply.

The shell is the unused case of a power tool, with some of the ribs cut out to make space. There's a 5V and 12V power supply, and a battery power supply. Switching options include two momentary push-button switches, a toggle switch, and a 10kΩ potentiometer, with breadboards and everything else held in place with 3D printed brackets. Just be careful when you're taking it through airline security...

▶ Spot the microcontroller: we can see an Arduino Uno, Arduino Nano, a pair of ESP32 boards, and a Raspberry Pi Pico, all waiting to be played with



Warning!

Electrical Safety

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

rpimag.co/electricalsafety

Cyberdeck

By Personalitysphere

rpimag.co/Cyberdeck

We see plenty of cyberdeck builds comprising a keyboard, Raspberry Pi 5, and a Pelican case, and for good reason: if you're living in a post-apocalyptic age, and somehow the Wi-Fi is still working, a moisture- and dust-proof case is your best bet for keeping a computer safe amid the ruins of our former civilisation. But it's not all that original, and we're always looking for fresh ways to take a computer on the move.

Enter this beautiful build by Personalitysphere. It uses a Raspberry Pi 5 8GB and, if you look closely, there's a part of the case marked 'GPIO' that gives a clue where the GPIO pins are. Other treats are the Jaeger connectors (those industrial-looking three-pin connectors on the machine's top-left corner), LEMO connectors on the back (these are used to get USB signals in and out – they're the industrial answer to the conundrum of always needing two attempts to get a USB to plug in the right way round), and the trackball instead of a mouse or trackpad.

- We can't imagine the looks you'd get using this machine on your daily commute, but apparently the maker of this wonderful machine does just that

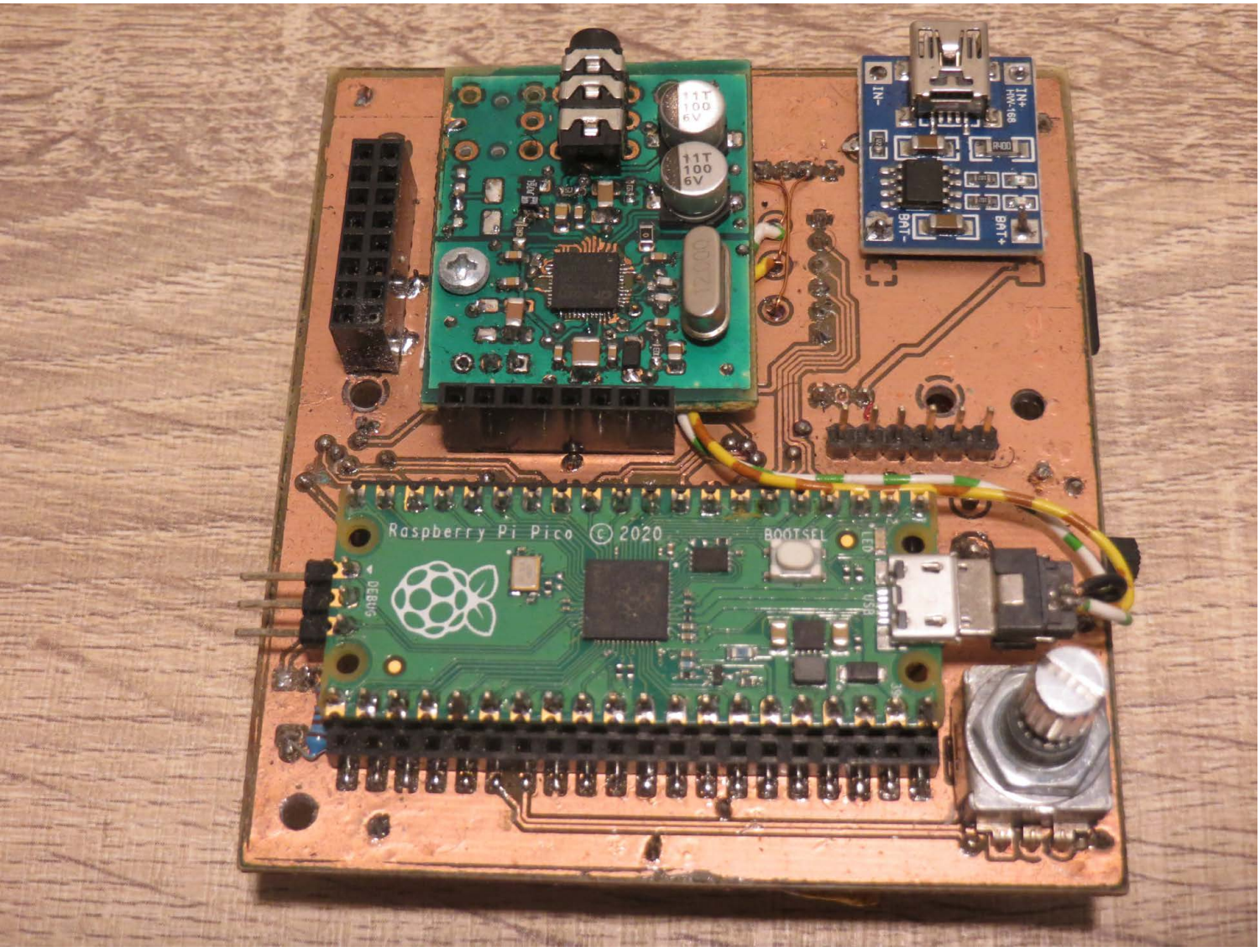
Kuensa

By Tinashe

rpimag.co/Kuensa

There are certain subject areas that seem to draw in computer people like moths to light bulbs. Astronomy is one; as is photography. But the biggest area on the Venn diagram of interests, at least as far as we can tell, is music. Notes are just oscillations, waves with predictable and mathematical definitions. Music was all around us in physics, and we humans have come along and decided that certain notes in certain contexts sound jazzy, or sad, or upbeat. It's amazing really. And this mathematical nature of music makes it perfect for computerisation – after all, computers are famously very good at mathematics.

This ongoing project, called Kuensa, is, according to its maker, "a small portable music sequencer which can be connected to USB MIDI and brought out whenever inspiration hits on the go." It uses a Raspberry Pi Pico to handle the computing, and a Dream SAM2695 module to handle the audio. What we like here is that in order to put the two together, the maker has had to craft not one but two custom PCBs. If these had come out of a factory they'd be much smaller, but as they're home-made (or rather, home-designed), they're a lot bigger. In fact, if you take apart any bit of audio hardware from the eighties, you'll probably find PCBs that look similar lurking within. Technology that was state-of-the-art just a few years ago (yes the eighties were 'just a few years ago' as far as we're concerned) is within the grasp of anyone with an internet connection and the imagination to use it today. We'll be keeping a close eye on this project.



▲ We like the look of where this device is going

3D print

Breaking things purposefully in the pursuit of perfection

rpimag.co/itkacher

Eugene Tkachenko tried to build an unbreakable RC car. And the best way to do that, as any engineer knows, is to build an RC car, abuse it until it breaks, then fix the part that breaks so it doesn't break any more. Repeat that process and with every iteration your car trends toward indestructibility.

In testing, the car achieved a battery life of half an hour, and a top speed of 35km/h (22mph). It struggled a bit when the snow came, thanks to an opening on one of the covers, but with 3D printing it was an easy job for Eugene to design a replacement part that kept the snow out. In fact, the failures, followed by the fast improvements, really highlight what's good and bad about 3D printing. The bad part, at least with the filament that most people have access to, is that plastic is susceptible to deformation at high temperatures: twice the motor overheated and deformed one of the drive gears, which was fixed with the addition of air vents, a fan and a heat sink.

So that's the downside, but the obvious advantage is that once a design flaw is identified, it's a matter how hours to fix it and go again. We're not sure whether the car actually is indestructible, but we know that when it stops working it's because of an empty battery, rather than a broken part.

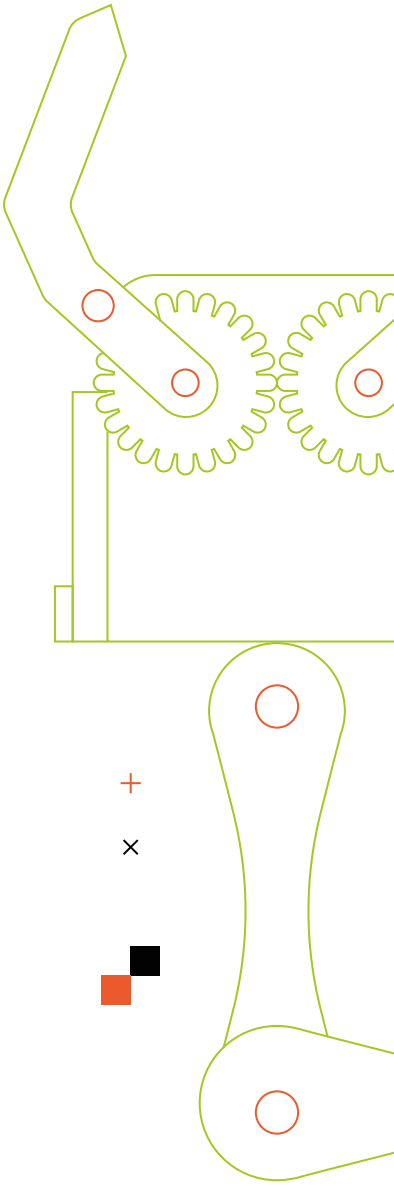
PROGRAM A ROBOT ARM

By **Phil King**

Back in the 1940s, mechanical manipulators were created for the safe handling of radioactive materials. From the US science labs of that time to the modern day, robotic arms have been put to good use. They are able to operate in settings that are hazardous or inaccessible to humans, such as on Mars or inside the Fukushima Daiichi Nuclear Power Plant. They are also used in factories to move things, paint things, and assemble items. For example, Raspberry Pi computers are made by robots in the Sony Pencoed factory in Wales. Robot arms are prized for their ability to reliably perform repetitive actions. With a kit and a Raspberry Pi, you can experiment with your own robotic arm at home, and scale up to using an industrial arm in a work environment.

CHOOSE AN ARM

With so many robotics arms to choose from, there are quite a few key factors to consider when buying one



There are numerous robotic arms available for use with a Raspberry Pi computer or Pico. Most are supplied prebuilt, while others are in the form of a kit to put together. They vary widely in cost, so it's worth exploring the tech specs and details to find the best arm for what you need. Here are a few key factors to consider...

- **Degrees of freedom (DOF):** Based on the number of moving joints your robotic arm has, this is key for how manoeuvrable it will be in 3D space. Note that the figure typically includes a swivelling base and a gripper for picking up objects. Some compact arms have 4 or 5 DOF, while others may offer 6 or 7 DOF for a wider range of motion and flexibility.

- **Servo motors:** The quality of the servo motors used in the moving joints of a robotic arm is very important, as it determines the accuracy of movement and general durability. While many inexpensive arms use lightweight (e.g. 9g) hobbyist micro servos, higher-spec arms will feature more robust and powerful servos (or stepper motors in some cases).

- **Build quality:** While some inexpensive arms are built from plastic or acrylic pieces, others have metal parts that should prove far more robust and durable in the long term. Some arms even cover the mechanical joints with plastic or other materials to protect them.

- **Payload:** This is the maximum amount of weight the arm can lift. Some higher-spec arms can handle up to 2kg. Note that for higher loads, the base of the arm will need to be well secured to a desk or table, typically using a G-clamp – standard rubber suckers aren't going to cut it!

- **Built-in brain:** Some arms come with a Raspberry Pi 4 or 5 built-in, while others need to be connected to your own Raspberry Pi, typically via a HAT or breakout board (sometimes supplied, or you can buy one). Some, like the MeArm Kit V3, can even work with a Raspberry Pi Pico microcontroller.

- **Software support:** If you're buying a higher-spec arm, it should be accompanied by detailed documentation and code examples. You'll typically be able to program the arm's movements using Python or a block-based language such as Blockly, or via a web-based interface that enables you to record and playback movements. More advanced users may want to look for compatibility with ROS (Robot Operating System) to develop more complex applications such as those requiring the use of inverse kinematics and/or AI.

Some arms come with a Raspberry Pi 4 or 5 built-in

MeArm Maker Kit V3

MeArm / The Pi Hut | £47 / $63 | mearm.com / thepihut.com

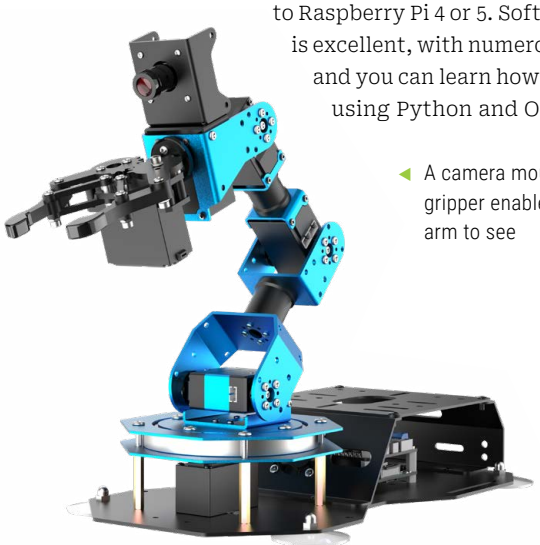
Ideal for beginners, this inexpensive arm with 4 DOF comes in kit form and takes around an hour to build from laser-cut acrylic arm pieces. The four servos are connected via cables to sockets in a base PCB which you can then connect to the GPIO pins on a Raspberry Pi Pico (as in our tutorial) – or via a servo driver HAT (not supplied) 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 in a GitHub repo (**rpimag.co/mearmgit**).

This entry-level kit is great for starting to experiment with robotics

ArmPi FPV AI Vision

HiWonder | £224 / $300 | hiwonder.com

One of the smartest robot arms around, this sturdy metal model with 6 DOF is equipped with a wide-angle HD camera (above its gripper) for AI applications using computer vision – it can recognise objects by shape and colour to track and sort them. A breakout board lets you connect it to Raspberry Pi 4 or 5. Software support is excellent, with numerous examples, and you can learn how to program it using Python and OpenCV.



A camera mounted above the gripper enables this smart arm to see

OTHER OPTIONS

Here are some more robot arms to consider…

- Adeept 5-DOF Robotic Arm Kit (**rpimag.co/adeept5dof**, £52/ $70) – low-cost, includes a gamepad controller kit.
- Freenove Robot Arm Kit (**rpimag.co/freenovearm**, £112/ $150) – with stepper motors, plus a pen holder for drawing.
- Adeept PiCar Pro V2 Robotics Kit (**rpimag.co/adeeptarmcar**, £123/ $165) – 4 DOF arm mounted on a robot car.
- Adeept RaspTank (**rpimag.co/adeepttank**, £82/ $110) – 4 DOF arm on the back of a tracked vehicle!
- MyPalletizer 260 Pi (**rpimag.co/mypalletizer**, £488/ $649) – compact 4 DOF arm, built-in Raspberry Pi 4.
- MechArm Pi (**rpimag.co/mecharm**, £600/ $799) – compact 6 DOF covered arm with built-in Raspberry Pi 4; 1kg payload.
- Niryo Ned2 (**rpimag.co/ned2**, £3492/ $4674) – Raspberry Pi 4-based high-spec arm designed for the educational market.

Alternatively, you could even 3D-print the pieces for a robot arm, such as the GrippyBot (**rpimag.co/grippybot**), and add your own servos.

MyCobot 280 Pi

Elephant Robotics | £600 / $799 | elephantrobotics.com

This is an impressive arm with large high-efficiency servos, all enclosed in protective plastic casings. There's even a built-in Raspberry Pi 4 in the base, so it's all ready to use out of the box – although you'll need to buy a separate gripper. With 6 DOF, a working rotation arc of 330° and radius 280mm, and numerous optional accessories to attach, it's a versatile arm that can be programmed with Python, Blockly, or ROS.

▼ The MyCobot 280 Pi has a Raspberry Pi 4 built into the base

CONTROL METHODS

Once you have your robotic arm, there are different ways of controlling it. Some arms can be used with a game pad for manual control of the arm. That's good fun, but the next step is to program the arm's movements.

Whatever programming language is used for this, the simplest method is direct joint control: setting the angles of each joint in the arm by sending PWM signals to their servos. This makes it difficult to know what position the arm will end up in, however.

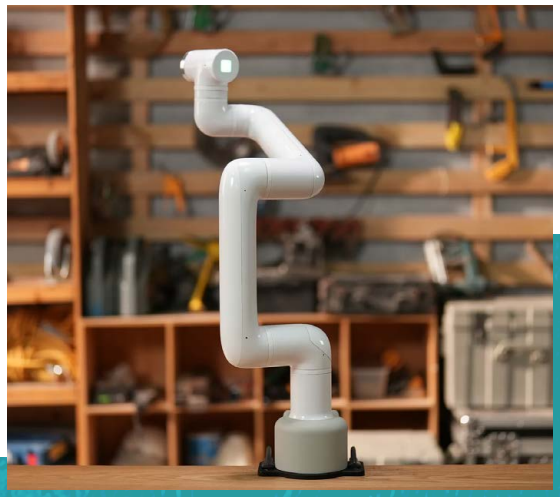
The alternative is a system that uses x, y, z Cartesian coordinates to move the end of the arm from one position in 3D space to another. This requires inverse kinematics, a mathematical process that works out how to set the joint angles to reach that position. We explored kinematics in our tutorials on CDP Studio in issues 141 (**rpimag.co/141**) and 142 (**rpimag.co/142**).

An even simpler system is one that lets you manipulate the arm manually and record a sequence of positions that you can then play back. This is what we're using for our tutorial later on in this feature.

MyCobot Pro 630

Elephant Robotics | £5261 / $6999 | elephantrobotics.com

If you really want to push the boat out, you can spend thousands on a higher-spec robot arm. Built around a Raspberry Pi 4, the Pro 630 is a heavy-duty model with a long reach, boasting a working radius of 630mm and high-precision movement. It's also pretty strong, able to lift a payload of up to 2kg – although you'll want to clamp the base securely, or use vacuum suction cups. It's programmable in numerous languages and could even be used for commercial automation applications.



◀ The MyCobot Pro 630 is a heavy-duty arm with a working radius of 630mm

EXAMPLE PROJECTS

Take some inspiration from these Raspberry Pi robot arm projects

01 Robot Arm Clock

rpimag.co/robotarmclock

When his clock stopped working, maker Hendrik Ohrens came up with an ingenious, if unorthodox, solution: building a robotic arm to physically move the clock hands to the correct time every minute!

After exploring the possibilities of using inverse kinematics and computer vision, he came up with a far simpler solution. Using Python code running on a Raspberry Pi 3B+, he trained the arm manually for each tiny movement of the minute hand required for a complete rotation.

Once trained, the Raspberry Pi relays the precise positional instructions to a connected Arduino board equipped with a shield to control the robot arm's four servos. Instructions, code, and files for the arm's 3D-printed parts can be found in this GitHub repo: **rpimag.co/armclockgit**.

02 Robot Dog with Arm

rpimag.co/robotarmdog

This one may freak you out a bit, especially seeing it in action on video, but the technical ingenuity is impressive. Created by Zipeng Fu, Xuxin Cheng, and Deepak Pathak from Carnegie Mellon University, it's an experimental design to show how controlling limbs centrally can transform the functionality of quadrupedal robots.

A Raspberry Pi 4 powers the neural network of this robotic canine companion, controlling all the joints: twelve for the legs and six for the arm on its back. It'll even bend its front legs to crouch down and enable the arm to reach an object to pick up from the ground.

In addition, the arm is equipped with a camera for AI computer vision that enables the dog to track a human and wipe a whiteboard clean.

03 Machine-Learning Prosthetic Arm

rpimag.co/mlparm

YouTuber James Bruton is a keen Iron Man cosplayer, and his invention would likely impress Tony Stark: an experimental backpack-mounted prosthetic arm that responds autonomously to the wearer's movements.

Equipped with three heavy-duty servos, the intelligent arm moves naturally based on the data sent to a Raspberry Pi Zero from IMU sensors on the wearer's other limbs and head, so it automatically adjusts to the user's body posture and limb movements. For instance, if James starts walking on the spot, the prosthetic arm swings the opposite way to his left arm as he strides along, and moves forward as he raises his left leg.

To train it, he even created a motion-capture suit from 3D-printed parts to gather all the data from his own body motions: arms, legs, and head. Clever stuff.

04 Raspberry Turk

raspberryturk.com

Joey Meyer's chess-playing robot features a robotic arm that moves the pieces around the board. Built from Acrobotics components with Dynamixel servos and some 3D printed parts, the arm uses an electromagnet to pick up a piece and then release it.

One challenge was making the arm movements precise enough to pick up pieces reliably. Due to inaccuracies in measurements, the simple maths equation model used would break down on occasion. Joey solved the issue by collecting a dataset of arm movements to see where the model was having problems.

Using the Stockfish chess engine on Raspberry Pi, the Turk plays a pretty mean game, too. To evaluate the positions of the pieces, a top-mounted Raspberry Pi Camera Module captures a view of the board which is then perspective-transformed using OpenCV.

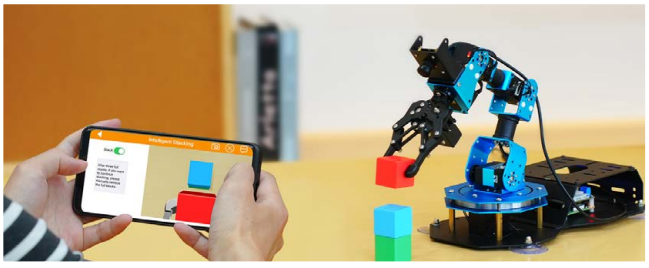
The arm uses an electromagnet to pick up a piece and then release it

STRENGTH TO YOUR ARM

Programmable robotic arms are perfect for performing repetitive actions, which is why they're so useful in industrial processes. Typical uses include picking up and moving objects, as well as sorting and stacking them. Robots are also used to spray-paint things like car parts; similarly, get your Raspberry Pi arm to hold a pen and it can be used to draw images.

By equipping your arm with a camera (as on the ArmPi FPV) you can make it smart with computer vision to recognise and track objects. You could also add IR (infrared) or ultrasonic distance sensors so it can avoid unwanted collisions.

Taking it much further, advanced makers could even have a go at building a giant industrial-style arm like Jeremy Fielding's Jarvis 2.0: see the video at **rpimag.co/jarvisrobot** for inspiration.



05 FarmBot

farm.bot

While not of a traditional jointed design, the robotic arm in this agricultural project moves around on rails and a gantry cross-slide using accurate stepper motors – in similar fashion to a CNC machine or 3D printer. This means it can move freely above a soil bed of up to 4.5m² (or 18m² for the XL version).

Equippable with a range of tools, the end of the arm can move up and down to plant seeds, remove weeds, sense moisture levels, and water the plants. Since the project is fully open source, you could even design your own tools to use.

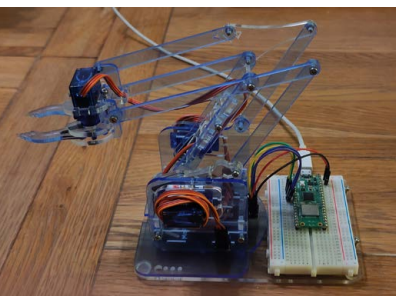
The brain of the FarmBot is a Raspberry Pi 3 connected to a custom Farmduino microcontroller board. It delivers a web interface to monitor and control the system.

PROGRAM A ROBOT ARM

Control an arm with Raspberry Pi Pico and get it to pick up and move an object

YOU'LL NEED:

- MeArm Maker Kit V3
- Raspberry Pi Pico W (or 2 W) with pin headers soldered
- Solderless breadboard
- Jumper wires
- Servo tester (optional)



▲ You can control the MeArm Maker Kit V3 with a Raspberry Pi Pico W

With a Raspberry Pi Pico W or 2 W with pin headers soldered and an entry-level kit, this simple setup offers an inexpensive way to start experimenting with a robot arm. It's based on Danny Staple's guide in issue 131 (**rpimag.co/131**), so check that out if you need more details on how it all works with the MicroPython code running on Pico.

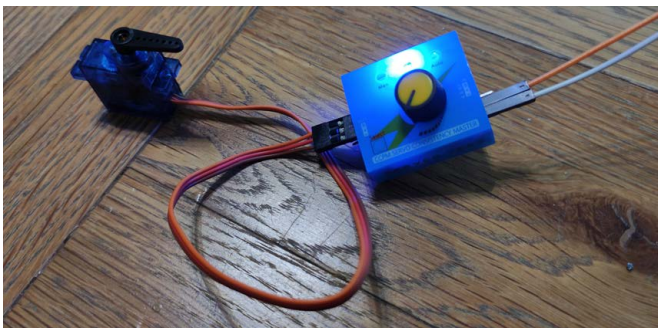
01 Calibrate the servos

Before building the arm, you need to calibrate each of the four servos to ensure that when in the neutral position (mid-point of its arc of movement), its plastic horn is oriented correctly, as shown in the official instructions (**rpimag.co/mearmbuildpdf**). You can either do this with Pico using the **calibrate.py** program in the GitHub repo (**rpimag.co/picomearm**) or with an inexpensive servo tester, e.g. **rpimag.co/servotester**. Secure each horn with the smaller screw.

02 Build the arm

The MeArm Maker Kit V3 takes around an hour to assemble as you snap out the acrylic parts from the sheets and snap-fit or screw them together, adding the servos for the joints.

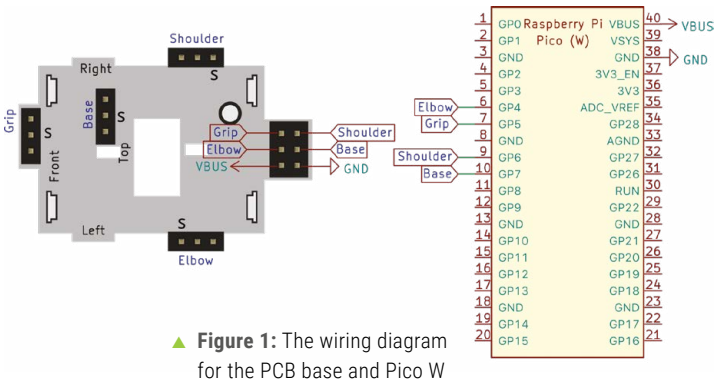
Step-by-step diagrams are provided in the official instructions (**rpimag.co/mearmbuildpdf**), although you may find the video (**rpimag.co/mearmbuildyt**) easier to follow.



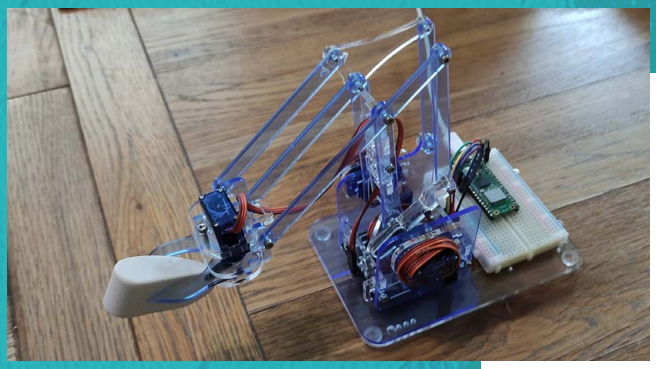
▲ Calibrate the servos by setting them to the neutral position and placing the horn in the desired position

03 Connect servos

If you haven't done so already while building the arm, connect the servo cables to the correct ports on the base PCB. Then connect the base's six-pin header to Raspberry Pi Pico: for the four servos' signal inputs, power, and ground (see **Figure 1**). Note that the GPIO pins used here for the signal outputs are on different PWM channels (Pico has 16 in all) so that they won't interfere with each other.



▲ **Figure 1:** The wiring diagram for the PCB base and Pico W



▲ You can record a sequence to pick up an object and move it

04 Prepare the code

Download the code from the GitHub repo at **rpimag.co/picomearm**. Then transfer the code files to Pico in Thonny: In the top of the Files pane, browse to where you downloaded the files on your computer, select them, right-click, then choose 'Upload to /'.

You'll also need to install the microdot library for the web server for the arm controls interface. Go to Tools > Manage Packages and search for it, then install it.

05 Connect Pico W to Wi-Fi

To deliver the web interface for the arm controls, Pico W needs to be connected to your wireless network. To do so, create a **secrets.py** file in Thonny and add your Wi-Fi credentials to it:

```
SSID = "<network name>"
PSK = "<network password>"
```

Replace `<network name>` and `<network password>` with the correct ones for your wireless network.

06 Manual control

To get it to run properly, we found that we needed to make an edit to line 5 of **web_arm.py**, changing `microdot_asyncio` to `microdot`. You can then run it in Thonny and the Shell area at the bottom will show the IP address (the first one listed) of the web server. Visit that in a web browser to find a control interface with four sliders: for Base, Shoulder, Elbow, and Grip. By moving these, you can control the arm's four joints manually.

07 Record sequence

The web interface also enables you to store steps for the arm's movements to record a sequence that can be played back. Try getting the arm to pick up a small object (e.g. an eraser) and move it to another position. Note that it's better to click Add Step after every movement – in particular, for the gripper closing to hold the object, to make sure it's grasped it before moving upwards.

08 Play it back

With the steps recorded, you can play back the arm movement sequence by clicking Run Steps. You may need to rerecord it a few times, after clicking Clear Steps, to get it to pick up the object reliably and move it to the desired position. Clicking Save Steps will save the sequence as a JSON file which can be loaded again later with Load Steps.

SERVO ANGLES

The code works by setting the servos to different angles to control the four joints of the arm. To try this out manually, you can run the **mearm.py** code from the GitHub repo, then type terminal commands to change the angle of each servo. For example, you can enter `arm.base.set_angle(-30)` to move the arm base to -30 degrees. For the other joint servos, use `grip`, `elbow`, or `shoulder` instead of `base`. You can find the movement limits by starting from 0 and increasing by 5 degrees at a time until the joint doesn't move any further (to avoid possible damage, don't leave a joint past its limits; go back a step by reducing the angle by 5°).

▼ Move the sliders on the web interface to control the arm's four joints

Part 04

Make a push-button music box

Use two or more tactile push buttons to play different sound samples

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

philkingeditor.com

YOU'LL NEED

- 1× solderless breadboard
- 2× push buttons
- 3× pin-to-socket jumper wires
- 2× pin-to-pin jumper wires
- Headphones or speaker

In this tutorial, we'll use several push buttons to make a GPIO music box that triggers different sounds when we press different buttons. For this, we'll make use of GPIO Zero's `Button` class again, as well as using the Python dictionary structure to assign sounds to buttons.

DOWNLOAD THE FULL CODE:
rpimag.co/gpiobookgit

Get some sounds

Before we start building our GPIO music box circuit, we'll need to prepare some sound samples for it to play. The easiest way to do this is to install the samples from Sonic Pi, a live coding environment that lets you create music with code that you write. Helpfully, Sonic Pi includes a large collection of sound samples that it installs in **/usr/share/sonic-pi/samples**. You can install the samples with `sudo apt install sonic-pi-samples`.

Play a drum

We'll now create a simple Python program to play a drum sample repeatedly, to check everything is working. Create a new file with the following code, using the nano editor (or Thonny IDE), then save it as **drumbeat.py**:

```
import pygame.mixer
from pygame.mixer import Sound

pygame.mixer.init()
samples = "/usr/share/sonic-pi/samples/"
drum = Sound(f"{samples}/drum_bass_soft.flac")
while True:
    drum.play()
```

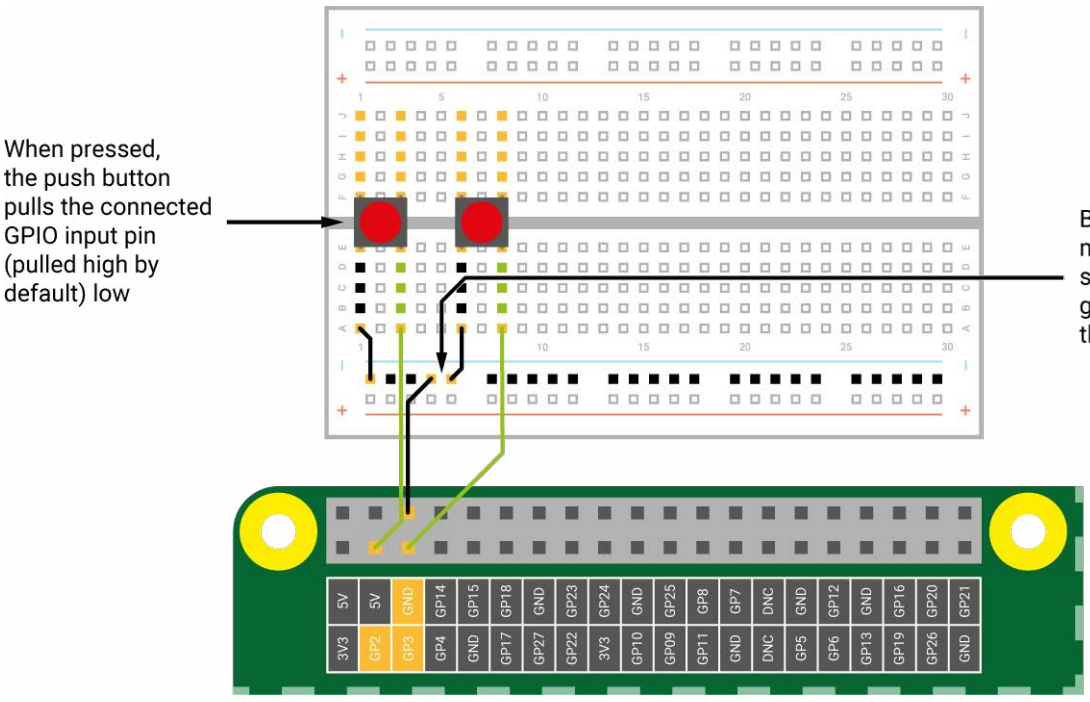


Figure 1: Wiring up two buttons

Before we start building our GPIO music box circuit, we'll need to prepare some sound samples for it to play

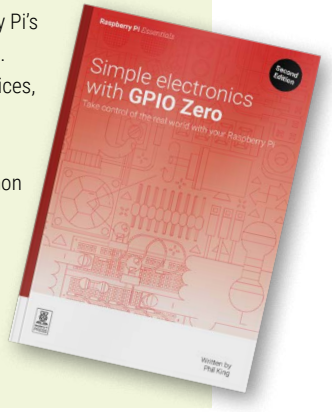
At the start of the program, we import Pygame's `mixer` module as well as its `Sound` class, which enables multichannel sound playback in Python. Next, we add a line to initialise the Pygame mixer: `pygame.mixer.init()`. We then create a variable to hold the path to where the samples reside, use Python's f-string notation to embed that variable in a string and combine it with the sample's file name (**drum_bass_soft.flac**), and create a `Sound` object for it.

Finally, we add a `while True:` loop to repeatedly play the drum sound. Run the program and listen to it play. When you get tired of the drumming, press **CTRL+C** to stop the program.

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



Wire up a button

As usual, you must turn your Raspberry Pi off while wiring up a circuit on the breadboard. First, we'll add a single button. As before, place the button so it straddles the central groove of the breadboard. One leg is connected to GPIO 2, and the other to the common ground rail on the breadboard, which in turn is wired to a GND pin.

We will now make a sound play whenever the button is pressed. Open a new file with nano (or Thonny IDE), enter the code below, and save it as **play_drum.py**.

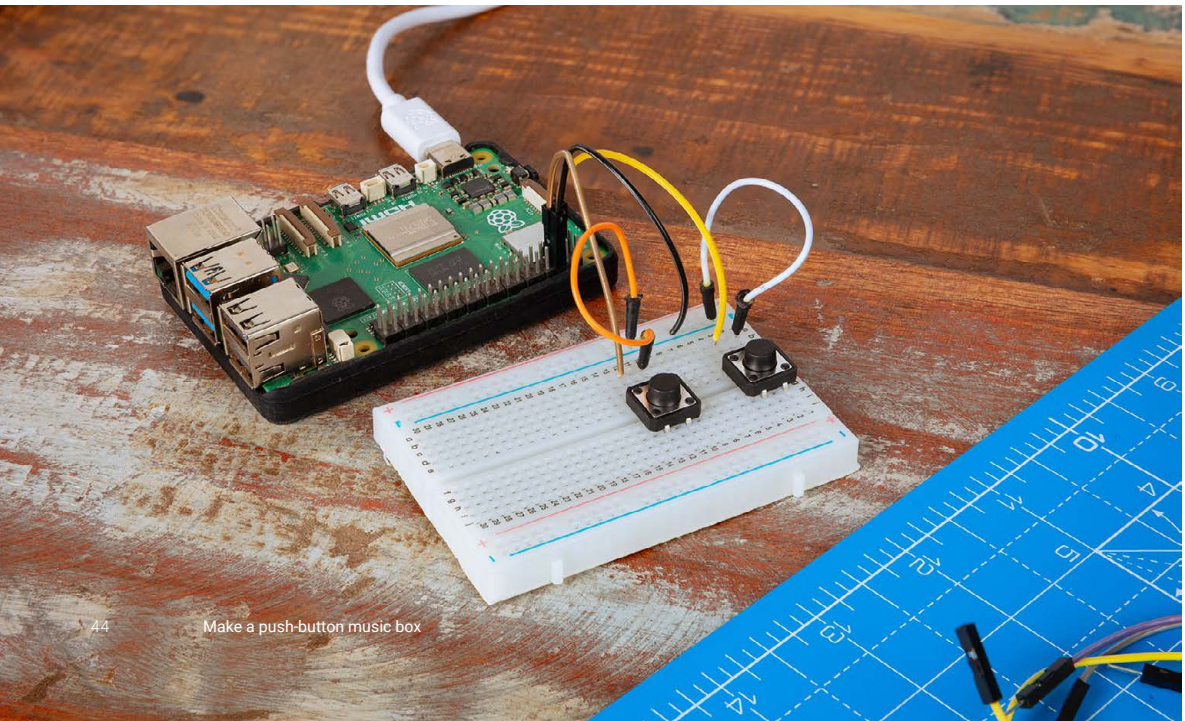
```
from gpiozero import Button
import pygame.mixer
from pygame.mixer import Sound
from signal import pause

pygame.mixer.init()
button = Button(2)
samples = "/usr/share/sonic-pi/samples/"
drum = Sound(f"{samples}/drum_bass_soft.flac")

button.when_pressed = drum.play
pause()
```

The way we've structured the program makes it easy to add extra buttons and assign them to sounds

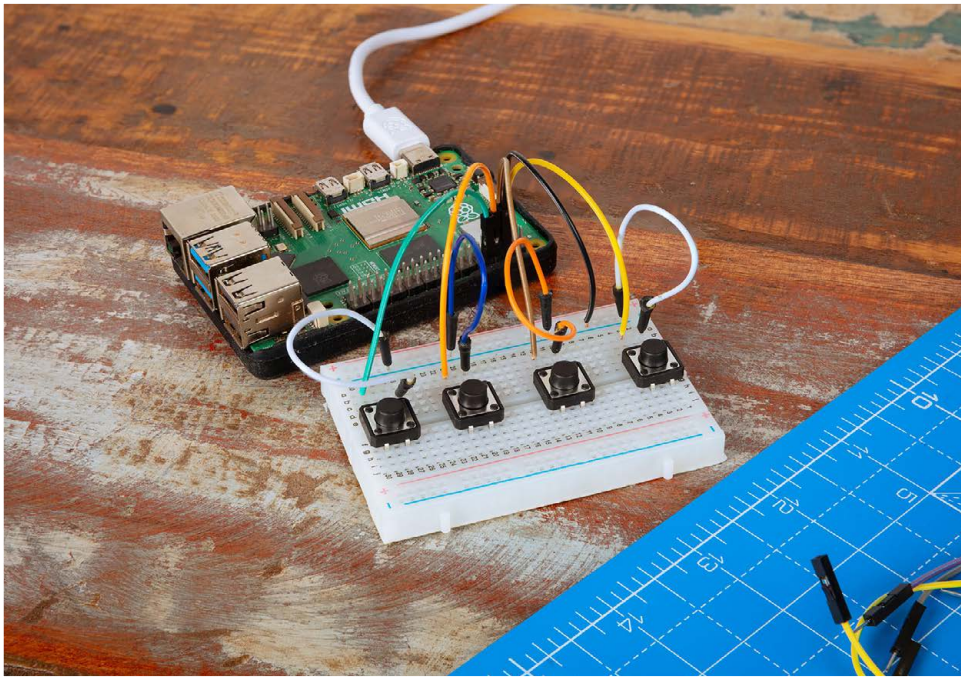
At the start of the program, we also import the `Button` class from GPIO Zero, and the `pause` class from the `signal` library. We set the `button` variable to GPIO 2, with `button = Button(2)`. We then tell the sound to play when the button is pressed:

```
button.when_pressed = drum.play
```

Finally, we add `pause()` at the end so that the program will continue to wait for the button to be pressed. Run the program and every time you press the button, the drum sound should play.



▼ Two buttons on a breadboard



▶ Extra buttons can easily be added to the circuit to play more sounds assigned in the Python code

Add a second button

Add a second button to the circuit — it should now look like the diagram in **Figure 1**. Place it on the breadboard, and wire it up to GPIO 3 and the common ground rail.

Create a new file, type in the code shown below, and save your new program as **drum_board.py**.

```
from gpiozero import Button
import pygame.mixer
from pygame.mixer import Sound
from signal import pause

pygame.mixer.init()
samples = "/usr/share/sonic-pi/samples/"
sound_pins = {
    2: "drum_bass_soft.flac",
    3: "drum_cowbell.flac",
}

buttons = [Button(pin) for pin in sound_pins]
for button in buttons:
    file = sound_pins[button.pin.number]
    sound = Sound(f"{samples}/{file}")
    button.when_pressed = sound.play

pause()
```

As with the previous example, we import the libraries we need and initialise the mixer and define a variable called `samples` that contains the path to the directory that holds our samples. We then define a dictionary (`sound_pins`) that maps GPIO numbers (2 and 3) to file names.

Next, we create a list of `Button` objects (`buttons`) for each pin number in the `sound_pins` dictionary. Finally, we create a `for` loop that looks up each button in the dictionary, constructs a new `Sound` object, and configures the button's `when_pressed` event to play the related sound. Run the program and press each button to hear a different sound.

Add more buttons

The way we have structured the program makes it easy to add extra buttons and assign them to sound samples. Just connect each button to a GPIO number pin (not any other type) and the ground rail, as before. Then add the GPIO pin numbers and sound file names to the dictionary, as in the following example. You can see all the sound samples in the directory by running the command `ls /usr/share/sonic-pi/samples/`.

```
sound_pins = {
    2: "drum_bass_soft.flac",
    3: "drum_cowbell.flac",
    4: "tabla_ghe1.flac",
   14: "vinyl_scratch.flac",
}
```

Part 07

Conquer the command line: predictable networking

Give the Raspberry Pi a permanent network address of its own

Maker

Richard Smedley

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

about.me/RichardSmedley

Raspberry Pi OS takes care of automatically connecting in most situations, but sometimes you need to override automatic configurations to ensure a consistent network setting for your Raspberry Pi project: Raspberry Pi OS has the tools, and we'll show you the essentials you need to stay connected.

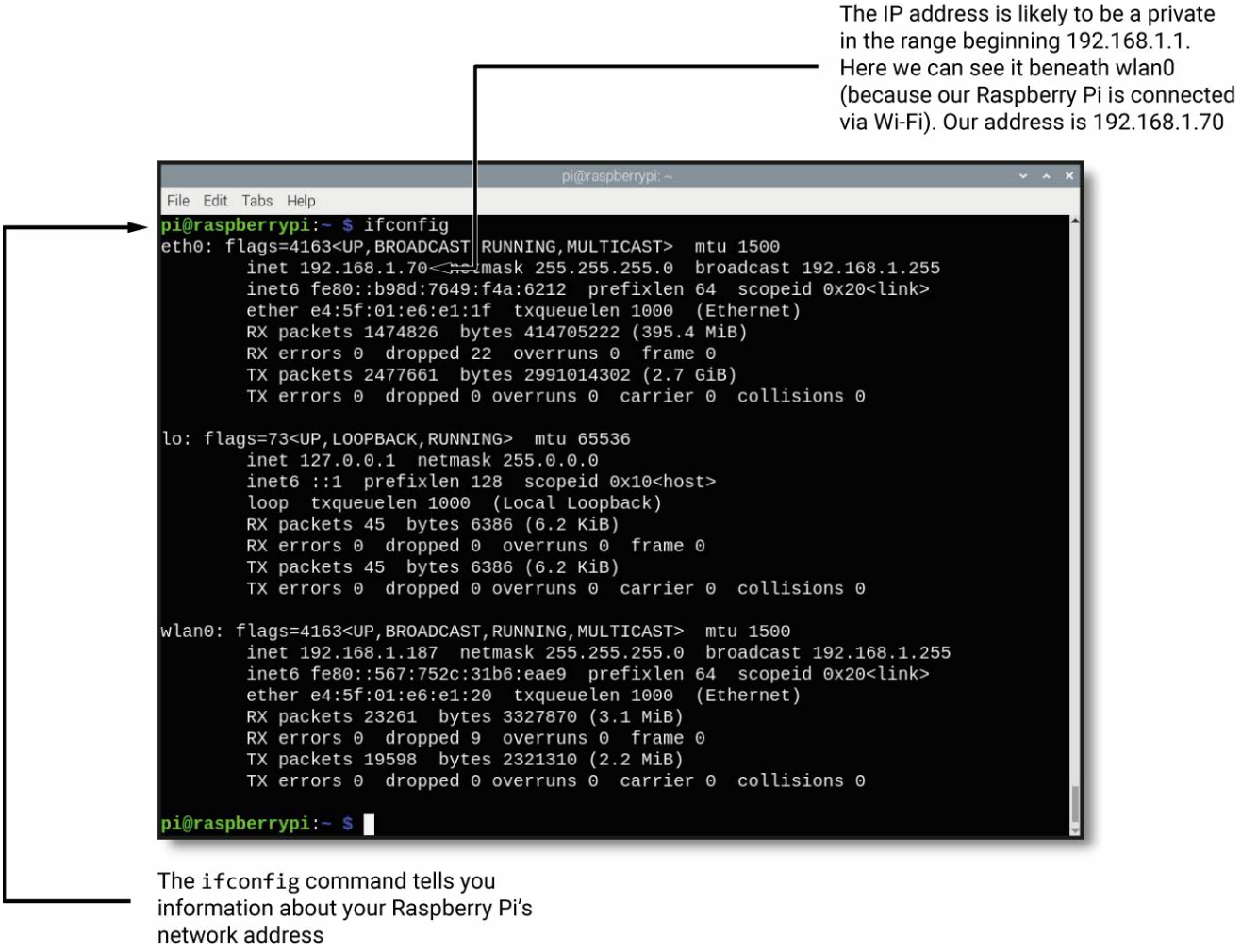
Plug an Ethernet cable from your router to your Raspberry Pi (or connect via Wi-Fi) and, automatically, Raspberry Pi OS knows where it is on the network, and can talk to the outside world.

All of this is thanks to DHCP – Dynamic Host Configuration Protocol – which provides network configuration for every device connected into a network. Typically, this comes in the form of an IPv4 (Internet Protocol version 4) address, a pair of four numbers separated by a period. For example: 192.168.1.100

The first two sets of numbers, 192.168, mark the start of a private range. These are the numbers for all devices in your house, ranging from 192.168.0.0 to 192.168.255.255.

Check your Raspberry Pi's current connection with the `ifconfig` command. This should show, amongst others, a line like `inet 192.168.1.100` (with your own numbers). This will be below the `eth0` section if you are connected via Ethernet, or under the `wlan0` section if you're connected via Wi-Fi.

A faster way to get your IP address is to run the command `hostname -I`.



You may want your Raspberry Pi to boot up with the same IP address each time

Configure your Raspberry Pi with a static IP address

Usually when you connect a Raspberry Pi to a local area network (LAN), it is automatically assigned an IP address. This address may change the next time you reconnect to the network.

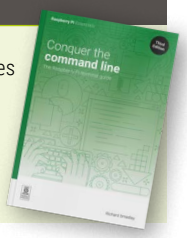
Sometimes, however, you may want your Raspberry Pi to boot up with the same IP address each time. This can be useful if you are making a small self-contained network, or building a standalone project such as a robot. Here's how to do it.

Raspberry Pi OS uses a system called NetworkManager to manage network connections. You can use the nmcli tool to configure these connections. Run the command `sudo nmcli dev status` to see a list of connections. You'll see output similar to this:

▲ Examining your network configuration

Conquer the Command Line 3rd Edition out now

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




```
DEVICE     TYPE       STATE         CONNECTION
eth0       ethernet   connected     Wired
                                    connection 1
lo         loopback   connected
                      (externally)  lo
wlan0      wifi       connected     preconfigured
p2p-dev…   wifi-p2p   disconnected  --
```

You can determine the name of the connection to edit by checking its type. In the preceding example, we have one Ethernet, one Wi-Fi, and one loopback (for the localhost, which we'll discuss later). You don't want to mess with the loopback settings, and you can ignore `wifi-p2p`. Suppose you want to set the Ethernet connection to a static IP. You could type the following (the `\` symbol allows us to break a single command across multiple lines):

```
$ sudo nmcli con mod "Wired connection 1" \
  ipv4.addresses 192.168.1.253/24 \
  ipv4.gateway 192.168.1.1 \
  ipv4.method manual
```

You should replace 192.168.1.253 with the IP address your Raspberry Pi currently has (or the one you want it to have), and 192.168.1.1 with your network gateway (usually your router) IP address. If your network is configured with anything other than a 24-bit network prefix (in this case, 192.168.1.x), you'll need to change the /24 as appropriate.

> *Your Raspberry Pi will now boot up with the specified IP address every time*

Your Raspberry Pi will now boot up with the specified IP address every time; we didn't use 192.168.1.1 as this is reserved for the router. You can of course use any address you like, but in the configuration above, the range must be between 192.168.1.2 and 192.168.1.254, and may depend on your router and other devices connected to the network.

Reboot with `sudo shutdown -r now`. Log in and type `hostname -I`. You should then see the IP address you set.

Normally, you don't want your computer set to use a static IP address. You can change the network configuration back by running the following command, and then rebooting again:

```
$ sudo nmcli con mod "Wired connection 1" \
  ipv4.addresses "" ipv4.gateway ""  \
  ipv4.method auto
```

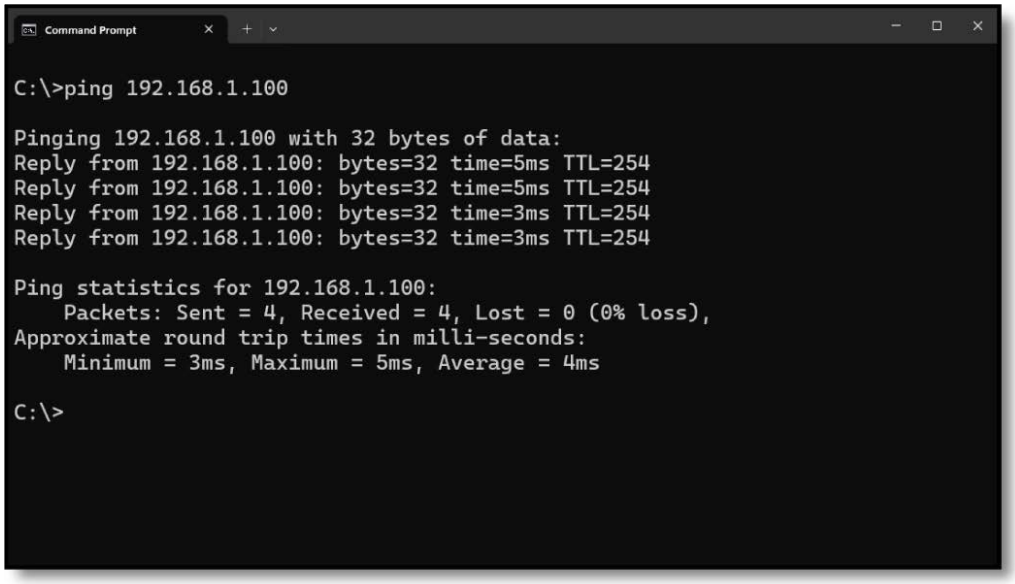


Figure 1: Use ping from another computer to detect if your Raspberry Pi is responding to network requests

If you don't want to reboot between making changes, you can use the command `sudo nmcli dev reapply eth0` to immediately apply the changes (note that we use the device name rather than the connection name here).

Ping!

Ping is the most basic tool in the network testing armoury, but it is one which is often called upon. It sends an ICMP (Internet Control Message Protocol) ECHO_REQUEST to a device on the network. ICMP is built into every connected device and used for diagnostics and error messages: a ping will produce a reply from the pinged machine, which tells you it is on, and connected, and that the network is working between you and it. Information about packets lost, and time taken, also helps with fault diagnosis.

A successful `ping localhost` from Raspberry Pi tells you not just that the local loopback interface is working, but that localhost resolves to 127.0.0.1, the local loopback address. Name resolution is the cause of many computing problems. Try to ping your Raspberry Pi from another machine on your local network: `ping 192.168.1.100` (**Figure 1**) – you'll need to use your actual IP address, though. If you're doing this from a Windows machine, ping defaults to four attempts; from another system (another Raspberry Pi, a Mac, or Ubuntu or other GNU/Linux), it will carry on until you stop it with **CTRL+C**, unless you set a number of ECHO_REQUEST sends with `-c` like so (this will attempt to ping Raspberry Pi's web server):

```
$ ping -c 5 raspberrypi.com
```

IPv6

The four-digit IP address style we use (such as 192.168.1.100) is IPv4. The newer standard, IPv6, is becoming more common. These are longer 128-bit addresses represented in hexadecimal (for example, fd51:42f8:caae:d92e::1).

Unusual tools: marine epoxy

The emergency tool that you never knew you needed

Maker

Dr Andrew Lewis

Andrew is a specialist maker and fabricator, and is the owner of the Andrew Lewis Workshop.

lewiswork.etsy.com

Some toolbox items have a dual use. They can solve problems in the everyday world, but they can also serve as a literally life-saving tool in an emergency. Just like Gordon Freeman's crowbar in Half-Life, marine epoxy is one of those tools.

When you're in the middle of an emergency, you're unlikely to care too much about making a visually appealing repair

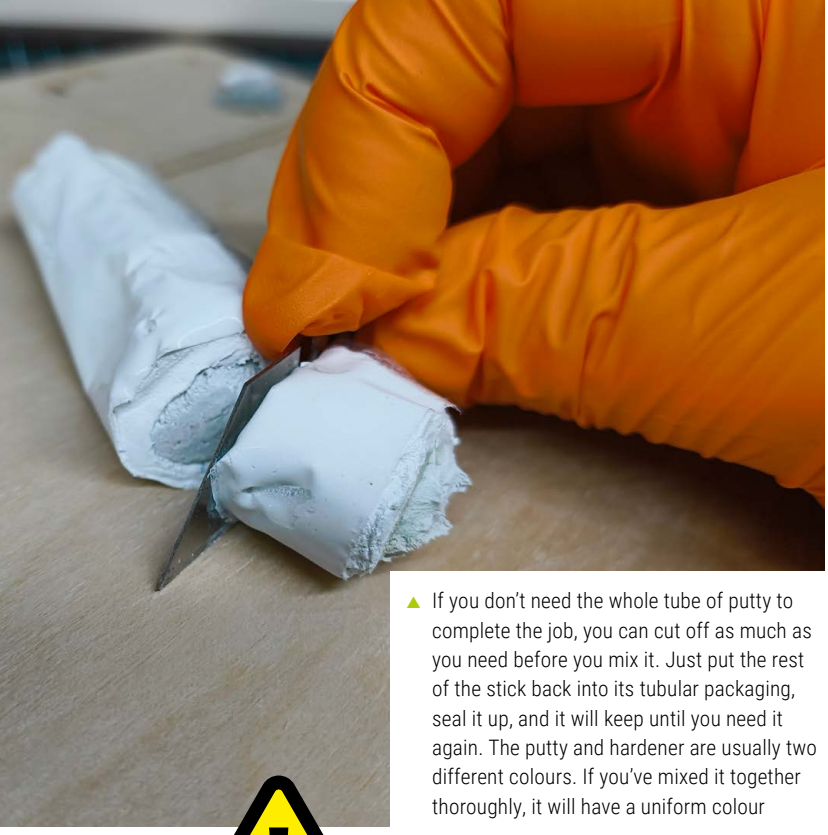
Mix it up

You are probably familiar with epoxy and other two-pack resins, where you mix resin with a hardener to create a durable thermosetting polymer. If you fall onto the right social media channels, it's impossible to scroll more than a couple of inches without running into live-edge wood and resin tables or an immaculate and perfectly levelled garage floor. Resin putty is less trendy in media circles than its liquid cousin, but it is a ubiquitous tool in garages and workshops. Putty is used to fill holes and disguise battered old panel work on all types of vehicle. Isopon, J-B Weld, and other manufacturers all produce a range of putty products designed to fill up and weld together the imperfections of workshop life. Some products are designed for filling small dents, while others are for bridging large gaps, sealing fish tanks, or can be used where a high resistance to heat or chemical solvents is important.

Marine epoxy putty is quite possibly the most useful of these putties. It's generally more expensive than other formulations, but it has a standout quality that makes it much more valuable in an emergency than regular putty. Unlike most other glues and resins, marine epoxy putty can be used underwater, and on

QUICK TIP

Applying constant pressure with a weight or clamp will improve the surface contact and consequently strengthen the bond with the putty. For unusual shapes, a bag of ball bearings or even sand can create a deformable weight to hold pieces in place.



▲ If you don't need the whole tube of putty to complete the job, you can cut off as much as you need before you mix it. Just put the rest of the stick back into its tubular packaging, seal it up, and it will keep until you need it again. The putty and hardener are usually two different colours. If you've mixed it together thoroughly, it will have a uniform colour

Warning!

Epoxy resin

Dispose of carefully

When you're working with resin, always wear the right safety gear for the job and make sure that you read the safety documentation. If you are using it on an engine, please follow the manufacturer's guidance and only as a temporary fix.

rpimag.co/epoxysafe

wet surfaces. If you've ever been on a boat that's been damaged below the water line, or have suddenly needed to patch a fuel tank in an emergency, marine epoxy becomes a standard item that you will keep in your toolkit from that day forward. It's an incredibly versatile product that is resistant to most chemicals (including petrol and diesel), and comes in a tube with a putty stick containing unmixed hardener inside. Putting on some protective gloves and squishing the stick with your fingers mixes the putty and hardener together. When the putty is properly mixed, you'll have about 20 minutes to use it before it starts to set.

Although marine epoxy will work on wet or dirty surfaces, you'll get better results if you can clean up the surface first to remove any grease, rust, or dirt. Roughing the surface with coarse sandpaper will help anchor the resin if you are working with a very smooth surface. Obviously, if your boat is currently sinking or you're working underwater for some reason, then you might have limits on how much you can do. A stiff brush will help clean a surface and remove any loose debris, and that will do as a bare minimum if you're in difficult circumstances. If you're in a more comfortable position where you can use alcohol wipes and detergents to dislodge grease, that will increase your chances of success even more.

When you're in the middle of an emergency, you're unlikely to care too much about making a visually appealing repair, and that's OK. Once marine epoxy has set hard, it can be sanded, drilled, tapped, sawn, painted, and shaped to suit your needs. Any excess putty can be wiped away with acetone or isopropyl alcohol before it sets hard.

▲ This visually enticing forbidden marshmallow is marine epoxy putty. It's extremely toxic, but in an emergency it can save you from complete disaster. The two colours differentiate the putty and the hardener. When mixed properly and allowed to cure, they make a solid resin that can rival the strength of metal

▲ Don't be gentle with the putty when you're using it. Push the putty hard against the surface that you want to stick it to

Part 02

Object design with Python in FreeCAD

Use your programming skills to make printable 3D designs

Maker

Rob Miles

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

robmiles.com

In the previous article, we got started running Python inside the FreeCAD application. In this instalment, we are going to create more complex shapes, starting with a desk tidy and moving on to a Raspberry Pi Pico box. You can find all the sample programs in the GitHub repository for this series: **rpimag.co/pythonfreecad**.

Figure 1 shows a simple desk tidy box you might like to make. If someone in the physical universe asked you to make a box like this, you'd get five flat panels and fit them together to make a box. This is not how you design a box in the 3D modelling universe. It is much easier to make a solid block and then cut another block out of it to form the required shape. Let's start by importing the FreeCAD libraries and making a new document:

```
import FreeCAD as App
import FreeCADGui as Gui
import Part

doc = App.newDocument("Desk Tidy")
```

Now we tell the program the size of the box that we want to make.

```
width = 100
depth = 50
height = 50
wall_thickness = 3
```

FreeCAD places the bottom left-hand corner of the box at the origin

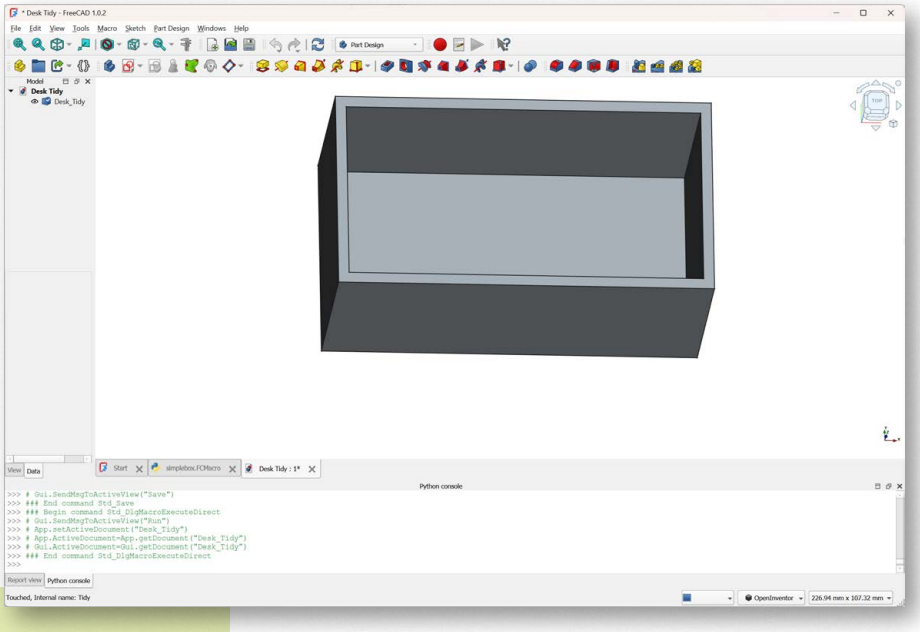


Figure 1: The box shape looks strange because it is being displayed isometrically with no perspective, so things further away are the same size as those close

These statements create the variables and assign values that determine the size of the box we are making. We want a box which is 100mm wide, 50mm deep and 50mm high with walls which are 3mm thick. These are the outer dimensions of the box, so the next thing we need to do is work out the inner dimensions for the hole in the box.

```
  inner_width = width -
(2*wall_thickness)
  inner_depth = depth - (2*wall_thickness)
  inner_height = height - wall_thickness
  outer_box = Part.makeBox(width, depth, height)
```

QUICK TIP

If we wanted a box to contain items of a particular size we'd need to allow for the wall thickness in our outer dimensions.

The last statement creates the outer box. FreeCAD places the bottom left-hand corner of the box at the origin – coordinate (0,0,0). Now let's make the cutout which will form the hole in the box.

```
  inner_box = Part.makeBox(inner_width,
inner_depth, inner_height)
```

That statement makes the inner box, which is smaller than the outer one. We are going to make the hole in the box by cutting the inner box from the outer one. Because the inner box is at the same position as the outer one, we wouldn't cut a hole; instead we'd cut a corner off the box. We fix this by translating the inner box up and inward by the thickness of the walls.

```
  move = App.Vector(wall_thickness,
wall_thickness,wall_thickness)
```

The statement above creates a vector called `move` which describes how the inner box is to move. It must be moved in and up by the thickness of the walls of the box. Now, let's move the inner box:

```
  inner_box.translate(move)
```

Now that we have the inner box, we can make the desk tidy object by cutting the inner box from the outer one:

```
  tidy = outer_box.cut(inner_box)
```

The last thing we need to do is add the box to the FreeCAD

document so that it can be viewed and exported:

```
  tidy_obj = doc.addObject("Part::Feature", "Desk
Tidy")
  tidy_obj.Shape = tidy

  Gui.ActiveDocument.ActiveView.fitAll()
```

We now have a box design that we can export and print. We can change the values in the size variables to make different-sized boxes.

```
  def makeOpenBox(width,depth,height,
wall_thickness):
      inner_width = width - (2*wall_thickness)
      inner_depth = depth - (2*wall_thickness)
      inner_height = height - wall_thickness
      outer_box = Part.makeBox(width, depth,
height)
      inner_box = Part.makeBox(inner_width,
inner_depth, inner_height)
      move = App.Vector(wall_thickness,
wall_thickness,wall_thickness)
      inner_box.translate(move)
      return outer_box.cut(inner_box)
```

We can make our life easier by creating a function which will make a box of a specified size. The function above returns the box which has been created.

```
  deskTidy = makeOpenBox(width=100, depth=100,
height=15, wall_thickness=3)
```

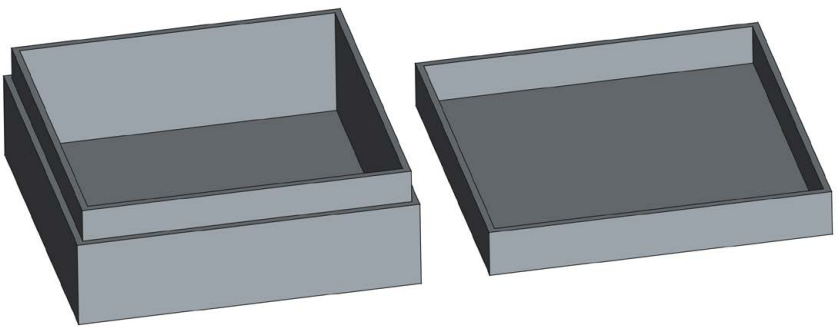
The above statement would make a square tray. We can turn our desk tidy into a box by printing a lid for it. We could make a lid which fits completely over the box, but we can make things more interesting by making a lid which fits over the box and is flush with the sides.

Putting a lid on things

The lid of the box is easy to make; it is just an open box with sides which are half the width of the box sides. But the box body now needs to have the top part cut so that the lid fits onto it. There are lots of different ways of doing this; one way is to create a "cutter" which is used to cut off the rim around the top of the box.

Figure 2 shows the 'cutter' object in the position on the box where the cut needs to be performed. The cutter is the 'square ring' around the top of the box. We can make a cutter from a block with a hole cut in it:

```
  # Create a block
  rim_cut = Part.makeBox(width,depth,lid_height)
  # Now cut a hole in the rim cutter – make a box
which forms the hole
  rim_hole = Part.makeBox(width-wall_thickness,
depth-wall_thickness,lid_height)
  # Move the rim hole cutter into the right
position
  rim_hole.translate(App.Vector
(wall_thickness/2.0,wall_thickness/2.0,0))
  # Cut a hole in the cutter
  rim_cut = rim_cut.cut(rim_hole)
  # Move the cutter to the right position
  rim_cut.translate(App.Vector(0,0,
box_height-lid_height))
  # Cut the rim off the box
  box=box.cut(rim_cut)
```



◀ The sides of the lid are thinner so that the lid fits flush with the box sides

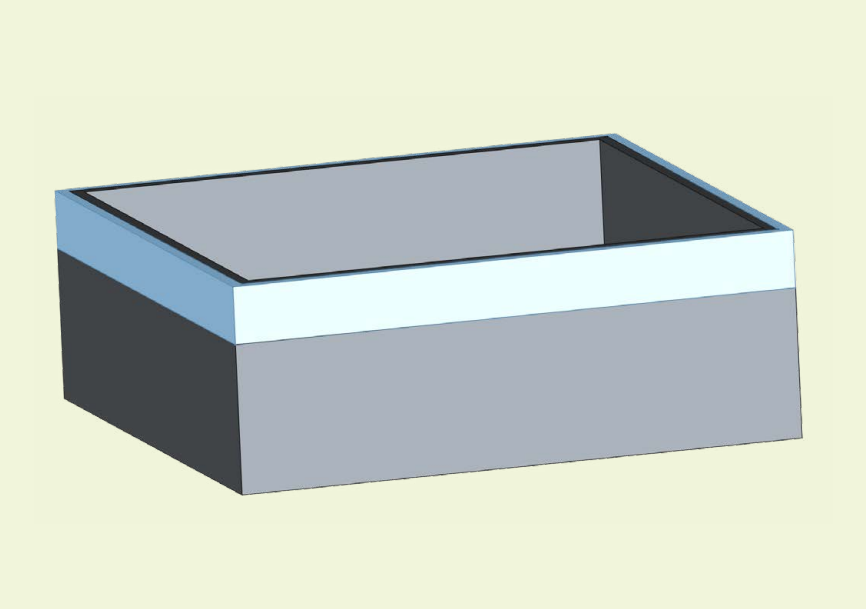


Figure 2: The cutter must be placed in the correct position to cut off the rim

Those statements create the outer cutter block, then create a block the size of the hole to be cut. The hole shape is then moved into position and used to make the hole in the cutter. Then the cutter is moved up to the required position to make the cut on the box. Finally, the cut is performed.

QUICK TIP

When you start creating complex shapes using code, you frequently find yourself creating objects that will be used to cut shapes from others.

These statements can be used in a method we could call `makeBoxAndLid` that creates a box with a lid.

```
  def makeBoxAndLid(width,depth,box_height,
lid_height,wall_thickness):

      # Allow for the thickness of the lid
      box_height=box_height-wall_thickness

      # Make the box
      box = makeOpenBox(width,depth,
box_height,wall_thickness)

      # Create the outer cutting ring
      rim_cut = Part.makeBox(width,depth,
lid_height)
      rim_hole = Part.makeBox(
width-wall_thickness,depth-wall_thickness,
lid_height)
      rim_hole.translate(App.Vector(
wall_thickness/2.0, wall_thickness/2.0,0))
      rim_cut = rim_cut.cut(rim_hole)
      rim_cut.translate(App.Vector(0,0,
box_height-lid_height))
      box=box.cut(rim_cut)

      lid = makeOpenBox(width,depth,
lid_height+wall_thickness, wall_thickness/2.0)

      # Move the lid where we can see it more
easily
      lid.translate(App.Vector(width+10,0,0))

      return (box,lid)
```

The method returns a tuple which contains the box and the lid shapes.

```
  box,lid=makeBoxAndLid(width=100, depth=100,
height=15, wall_thickness=3)
```

The statement above would set the variables box and lid to the components of the box.

Make a Pico project box

Now that we know how to make boxes, let's make a properly useful one. **Figure 3** shows the dimensions of a Raspberry Pi Pico device. We can use these to make a design for a Pico project box. The board dimensions of Pico are 51mm long and 21mm wide. The mounting holes are 11.4mm apart and are inset 2mm from the top and the bottom of the board. We can express this information as follows:

```
board_width=21.0
board_depth=51.0
x_hole_inset=4.8
y_hole_inset=2.0
x_spacing=11.4
y_spacing=47
```

The statements specify the size of the board, the offset of the bottom-left mounting hole, and the x and y spacing of the mounting holes. Now we need to set some preferences:

```
wall_thickness=2.0
margin = 2.0
pillar_radius=2.0
pillar_height=4
hole_radius=0.9
```

These set the thickness of the box walls, the margin around Pico, the dimensions of the pillar, and the radius of the hole in the pillar. Now we need to work out the size of the box. Remember that the values given to the `makeBoxAndLid` function specify the outside of the box, so we need to add in the margins and the wall thickness.

QUICK TIP

If you want to put Raspberry Pi Pico in a bigger box, just increase the margin value.

```
width=board_width+(2*margin)+(2*wall_thickness)
depth=board_depth+(2*margin)+(2*wall_thickness)
height=20.0
lid_height=5.0
```

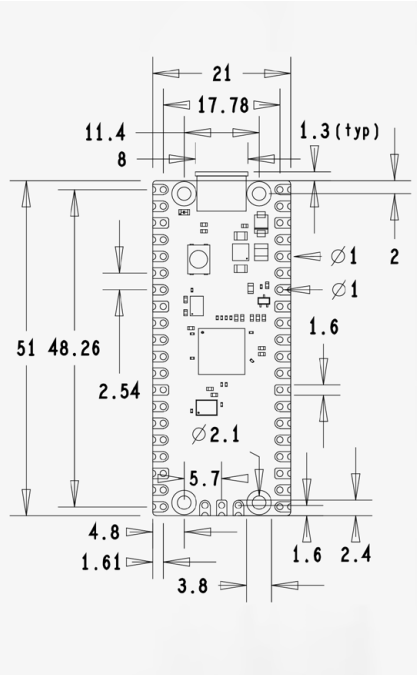


Figure 3: This diagram was obtained from the Pico datasheet

QUICK TIP

Be generous with your margins. It is much easier to assemble something if there is a bit of space around each component.

Now we can create a box for Pico:

```
box,lid=makeBoxAndLid(width,depth,height,
lid_height,wall_thickness)
```

The statement above creates a box. The variable `box` refers to the box that has been created. We now need to add pillars that will support Pico. Let's start by making a pillar.

```
pillar = Part.makeCylinder(pillar_radius,
pillar_height)
hole = Part.makeCylinder(hole_radius,
pillar_height)
pillar=pillar.cut(hole)
```

Those statements create a pillar as a cylinder with a hole cut in it for the screw that holds Pico in place. Now that we have a pillar, we need to move it into position. Let's work out a few coordinates:

```
x=wall_thickness+margin+x_hole_inset
y=wall_thickness+margin+y_hole_inset
z=wall_thickness
```

The statements above create the x, y, and z coordinates that describe where we want the pillar to be. The pillar is created at the position (0,0,0). It needs to be moved in past the side of the box, the margin, and into the position on the board. We also need to lift the pillar up so that it will be placed on top of the bottom of the box. Now that we have the distances to move the pillar, we can move it:

```
pillar.translate(App.Vector(x,y,z))
```

The statement above creates a vector and then uses this to tell the `translate` method where to move the pillar. Now that we have the pillar in the right place, we can fuse it with the box:

```
box = box.fuse(pillar)
```

The statement above creates a new shape which has the pillar fused to it. If you apply the `fuse` method to a shape, it generates a new shape with the specified shape added to it. We now have one pillar, let's add the other three:

```
pillar.translate(App.Vector(x_spacing,0,0))
box = box.fuse(pillar)
pillar.translate(App.Vector(0,y_spacing,0))
box = box.fuse(pillar)
pillar.translate(App.Vector(-x_spacing,0,0))
box = box.fuse(pillar)
```

We now have a box with four pillars. The next thing to do is to cut a hole for the USB connector for Pico.

```
usb_hole_width=15.0
usb_hole_height=8.0
usb_hole_margin=3.0
```

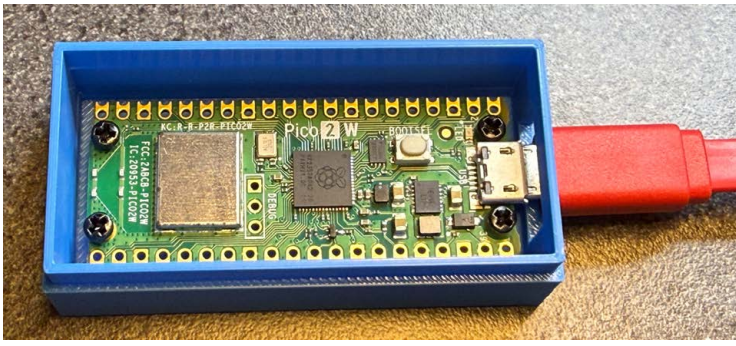
Those statements specify the position and size of the hole we want. This means that if we discover the hole is in the wrong place, we can just change these values and rerun the program. Let's use these values to create our box and cut the hole:

```
  x=(width-usb_hole_width)/2.0
  y=depth-wall_thickness
  z=wall_thickness+usb_hole_margin
  pos=App.Vector(x,y,z)
  usb_hole = Part.makeBox(usb_hole_width,
wall_thickness,usb_hole_height,pos)
  box=box.cut(usb_hole)
```

The statements above calculate the hole position, create the hole, and then cut the box. Note that the `Part.makeBox` method can accept a vector which gives the position of the box. This simplifies our code. We now have a box.

Boxing clever

Now that we know how to create boxes and pillars, we can create boxes for any circuits that we might care to create. We can add pillars for additional devices and cut extra holes as required. Next time, we will look at using vectors and extrusion to create even more complex shapes using code. You can find the box-making code, along with lots of other examples, at the repository for these articles: **rpimag.co/pythonfreecad**.



▲ The screws are M2 self-tapping, 4mm long

Pioreactor: an automated Raspberry Pi bioreactor

Let's look at building a 'Pioreactor', an amazing project for running long automated bioreactor experiments

Maker

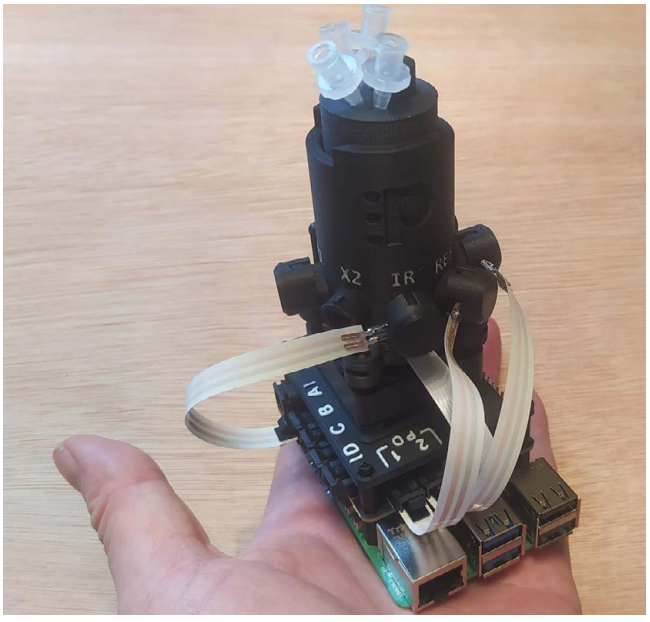
Jo Hinchliffe (aka Concretedog)

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

concretedog.blogspot.com

Whilst at the Open Hardware Summit 2025 in Edinburgh UK, we met Gerrit who was speaking about 'Growing food with electricity' and was representing AMYBO (amybo.org), an online community that is dedicated to developing sustainable protein as food sources.

The talk is excellent and available on YouTube (**rpimag.co/growingfood**). In it, you can see and hear a lot of information about Pioreactor. This project is a tiny automated bioreactor that allows complex science to take place on your desk and is powered by a Raspberry Pi. It's the go-to tool for a lot of professional, amateur and hobbyist biologists and chemists; it's also used by a fascinating research community called AMYBO.

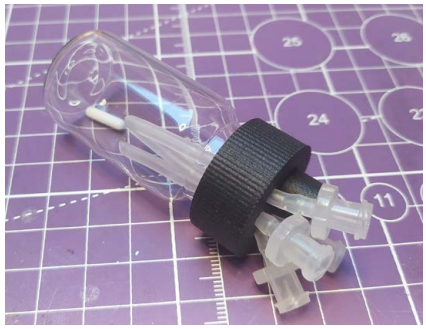


▶ A fully assembled Pioreactor

So what is a bioreactor? A bioreactor is a vessel that creates an environment for the growing of cells, microorganisms, and microbial cultures. In its simplest form it could simply be a jar, but commonly 'bioreactor' as a term describes more complex setups where the environment can be controlled and automated. Bioreactors get used in the development of pharmaceuticals, in the food sciences, medical sciences, and many other chemistry and biology adjacent sectors.

The Pioreactor is small: the working volume of our version is just 20ml (see **Figure 1**). You definitely aren't going to grow enough algae to create a decent food supply or for your fuel cell. For

experiments and research, however, it offers a wide range of environmental control out of the box. Additionally, the wider community has built all manner of add-ons. It's capable of automating experiments over long periods of time and can also log data about the experiments you schedule it to perform. You can see the BOM (Bill of Materials) on the AMYBO documentation website (**rpimag.co/pioreactorBOM**).



▲ **Figure 1:** The small 20ml glass vial allows for small samples and cultures to be grown

Warning!

Health risks

There is significant risk to health in this project. Please take care when handling microorganisms and dispose of safely. Do not eat! The section on hydrogen and oxygen is experimental and should not be attempted at home. Keep your workspace well ventilated. In-chamber electrolysis has a fire risk.

rpimag.co/pioreactorrisk

rpimag.co/hsebiosafety

Building the Pioreactor is pretty straightforward. You'll need a Pioreactor kit. Gerrit is the co-founder of LabCrafter, a company that supplies open-sourced science equipment, including the OpenFlexure microscope kits we wrote about in issue 158 (**rpimag.co/158**). It stocks the current models of Pioreactor, as well as numerous add-on accessories and expansions. The kit arrived from LabCrafter really well packed in nice sustainable recyclable packaging, ready to be built.

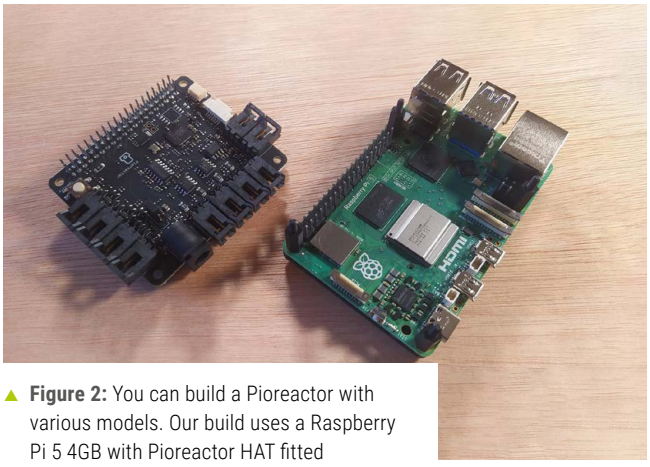
You can build a Pioreactor using any choice of Raspberry Pi model. You can use a Raspberry Pi Model A or Model B, as well as Zero models. We went with a Raspberry Pi 5 with 4GB RAM, because this will give great performance (**Figure 2**). You begin by simply attaching a base to your Raspberry Pi, some standoffs are fitted, and then the Pioreactor HAT board is fitted onto Raspberry Pi's GPIO header. The instructions are online and they are excellent. Do, however, double-check which version of Pioreactor you have, as the assembly approach has changed slightly for the recent v1.5 hardware design update.

> *Building the Pioreactor is pretty straightforward*

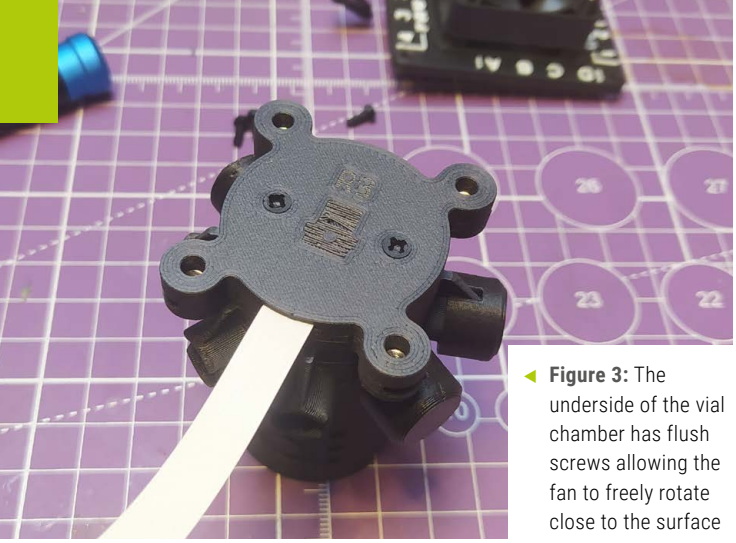
We then move on to assembling the 'wetware' section: the main chamber of the Pioreactor which will hold your glass vial that contains your experiment. The base of the vial chamber and the chamber wall need some supplied O-rings fitted, and then you insert a small heater element into the chamber. The clearances for various parts of the mechanism are quite accurate (**Figure 3**), so you need to double-check you are assembling using the correct bolts, but handily, the packaging has a nice to-scale labelled image of all the bolts to check them against.

There are numerous holes in the side of the chamber wall. The small circular holes allow later for the addition of an optical system. The included optical system consists of an infrared (IR) LED and then two photodiodes in the same 5mm LED form factor.

▼ The complete set of components to build a standard Pioreactor



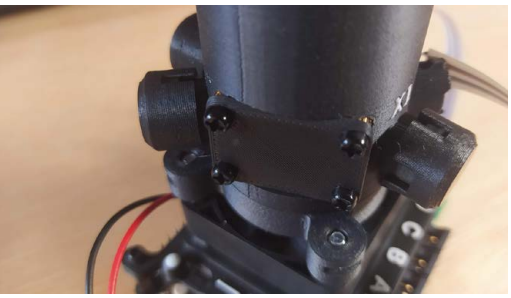
▲ **Figure 2:** You can build a Pioreactor with various models. Our build uses a Raspberry Pi 5 4GB with Pioreactor HAT fitted



◀ **Figure 3:** The underside of the vial chamber has flush screws allowing the fan to freely rotate close to the surface

These are fitted later in the build and you can automatically measure the optical density of your experiment. This is obviously a potential metric for growth, so imagine you start with a reasonably clear liquid in which something, say some yeast, is to grow; over the time of your experiment, you would expect the optical density of the liquid to increase as the IR LED is obscured more by the cloudiness of the mixture.

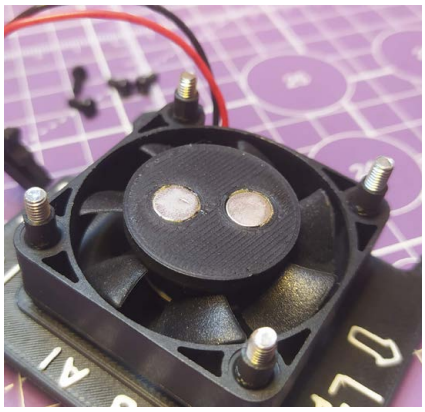
Also on the chamber wall is a small rectangular aperture with some small threaded inserts in the corner. This is a 'viewing window' (**Figure 4**) and there is a supplied blanking plate for this area. The viewing window is also designed to receive additional hardware. One option for this is to add an Adafruit AS7341 sensor, which is a pretty well-featured spectrometer. You can purchase this separately, but it will mount directly to the viewing window and there is software support for this device enabling you to directly retrieve readings from it.



▲ **Figure 4:** A window in the chamber wall allows viewing, or is a place for add-ons like the Adafruit AS7341 spectrometry board

Between the Pioreactor HAT and the vial chamber there sits an upper faceplate and this has a mount for a fan unit, and then in turn the vial chamber assembly fits on top of the fan mounting bolts. The fan, you will notice, has been retrofitted with a pair of strong magnets (**Figure 5**). This is the clue that the 'fan' isn't really used as a fan – it's actually a common approach to creating a stirring mechanism for inside the vial. Supplied with the kit is a tiny plastic covered metal stir bar which sits inside the vial. When the fan is instructed to turn, the magnets cause the stir bar to spin, allowing you to schedule the periodic agitation of your experiment.

Assembly continues with the mounting of the fan and the vial chamber to the upper faceplate and then, in turn, mounting the faceplate and attachments to the HAT and Raspberry Pi. There are rugged connections for the fan/stirrer cable, and the heater element ribbon cable is fitted at this point.



◀ **Figure 5:** The included fan unit isn't actually used as a fan; it's modified to run the magnetic stirring system

Finally, we add the IR LED and the two supplied photodiodes and cover them with neatly designed protective covers (**Figure 6**). There are spare covers and, for now, we can cover the additional vial chamber holes with them, but these additional holes are there so you can add further LEDs depending on your experiment's needs. Many bioreactor experiments will need some form of light source and so it's common to mount 5mm LEDs of the target wavelength into these mount holes.

Once you have all the hardware assembled, it's time to grab a microSD card and install the software to run your Pioreactor. Neatly, this is achieved by the Pioreactor team supplying a custom Raspberry Pi OS image. Using the latest 2.0.0 version of the official Raspberry Pi Imager application, it's easy to get it installed.

After booting Raspberry Pi Imager, you need to click the 'App Options' button on the main page.

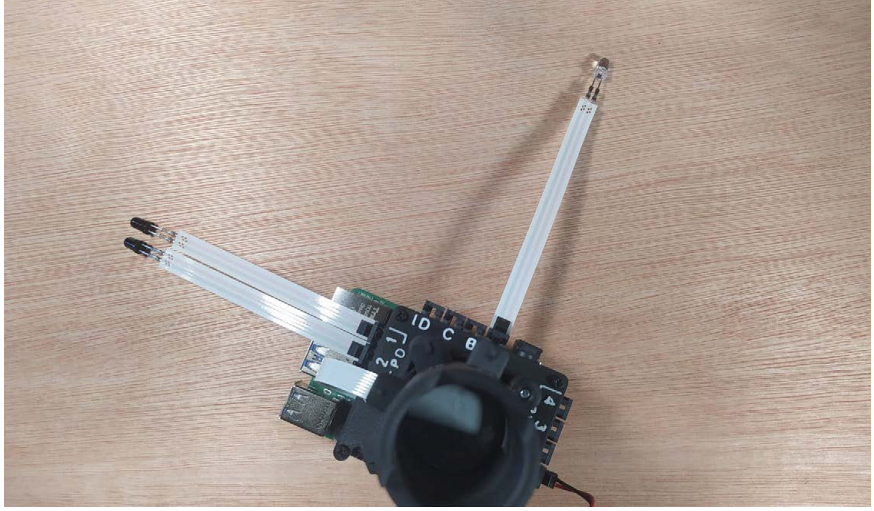
If you're using the AppImage version of the Raspberry Pi Imager tool on Ubuntu, you'll need to run Imager with root permissions. To do this, navigate to the directory where the AppImage is and then launch Imager with:

```
$ sudo ./imager_2.0.0_amd64.AppImage
```

On the App Options page, you can edit the 'content repository' tab and add the custom URL for the Pioreactor OS image. If you then reboot Imager, you should be able to select your Raspberry Pi device and then see the Pioreactor OS available as an image to install to your microSD card.

The Pioreactor instructions walk you through this process really well, but in essence Imager will prompt you to set localisation settings, and you need to add a specific username and password from the Pioreactor instructions, as well as your own Wi-Fi network credentials. Don't worry, though: you can change these down the line as needed.

With the software installed, you can now boot your Pioreactor by connecting a power supply. We made sure to use an official

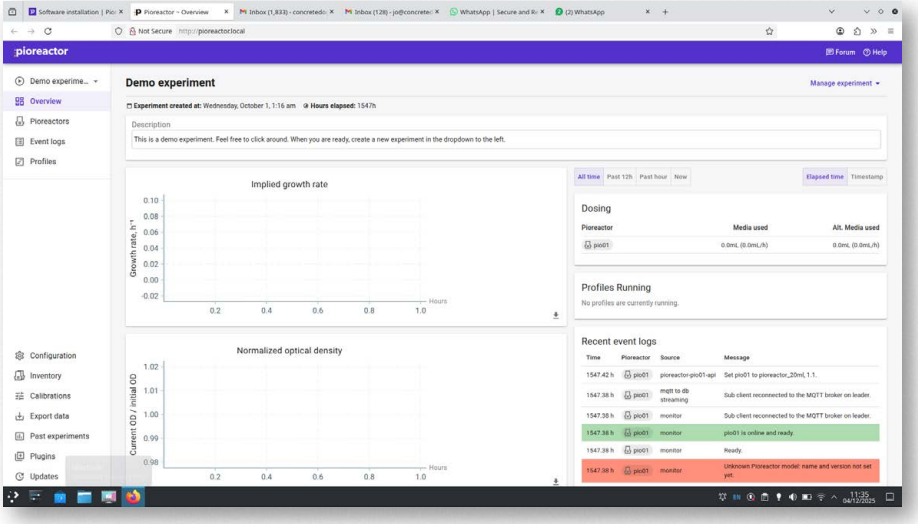


▲ **Figure 6:** Adding the IR LED and photodiodes

Raspberry Pi power supply and with all being well, we saw after a couple of minutes a blue LED blink for a short period on the Pioreactor HAT. Then on our laptop, connected to the same Wi-Fi network, we opened a web browser and navigated to **http://pioreactor.local**. A window pop-up requires us to confirm which Pioreactor version we have; after selecting this, a wonderful dashboard for our Pioreactor appears (**Figure 7**).

There's lots you can check, even without beginning to run an experiment on your Pioreactor. As a simple test, you can select the Profiles tab from the list on the left-hand side of the screen and then select the Demo Stirring Example from the drop-down list of available profiles. You should be able to work out from the description that this little community-contributed example will turn on the stirring system at a particular rpm, then increase the speed of stirring after 90 seconds and finally then stop stirring after three minutes. Before you run this, if you remove the lid from your vial chamber, you can watch the stir bar in action. Similarly, if you click the Pioreactors tab from the list on the left-hand side, you can then select your Pioreactor (you can run multiple Pioreactors from a single interface) and assign your Pioreactor as 'leader'. Then, if you click the Manage button, you will see a list of 'Activities' which you can start or stop running inside your experiment: stirring, optical density (as a culture grows, less light from the IR LED can be detected at the photodiodes), temperature control, and more.

▼ **Figure 7:** The served web interface is neat and easy to explore



AMYBO

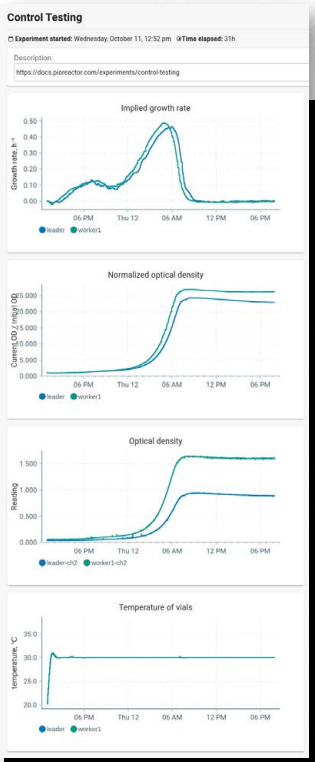
AMYBO is a global group of people working to democratise food production. They have a focus on protein fermentation and aim to give everyone the skills, information, and equipment designs to do so. They are open source and are keen to engage with anyone interested. So, whether you are a biomedical scientist or a keen maker, an engineer, or just interested in tasty protein, there's plenty to get involved with.

AMYBO at the moment is focused on developing the Pioreactor to grow hydrogen oxidising bacteria (HOB) as the first target single-cell protein. Project members have been gathering samples of fresh water that they hope contain HOB and then using this water, with added nutrients, to try and accelerate the HOB growth. As mentioned elsewhere, this involves adding an electrolysis system to the Pioreactor as well as CO_2 sparging and other additional features. It also means that AMYBO volunteers like to trek into areas to find sources of natural rivers and streams to collect potentially HOB-containing samples. It's always great when citizen science can get you outdoors.

A good first experiment is described on the AMYBO website. As written, it's an experiment to calibrate two Pioreactors to each other; however, you could run the experiment as a test for a single Pioreactor. Essentially, we are going to grow some yeast using a yeast extract peptone dextrose (YPD) broth. This YPD broth is a common growth medium used in all manner of microbiological cultivation. It's excellent food for baker's yeast! The experiment basically grows yeast in the YPD broth, stirring and warming the mixture whilst taking periodic optical density measurements to track the growth of yeast (**Figure 8**).

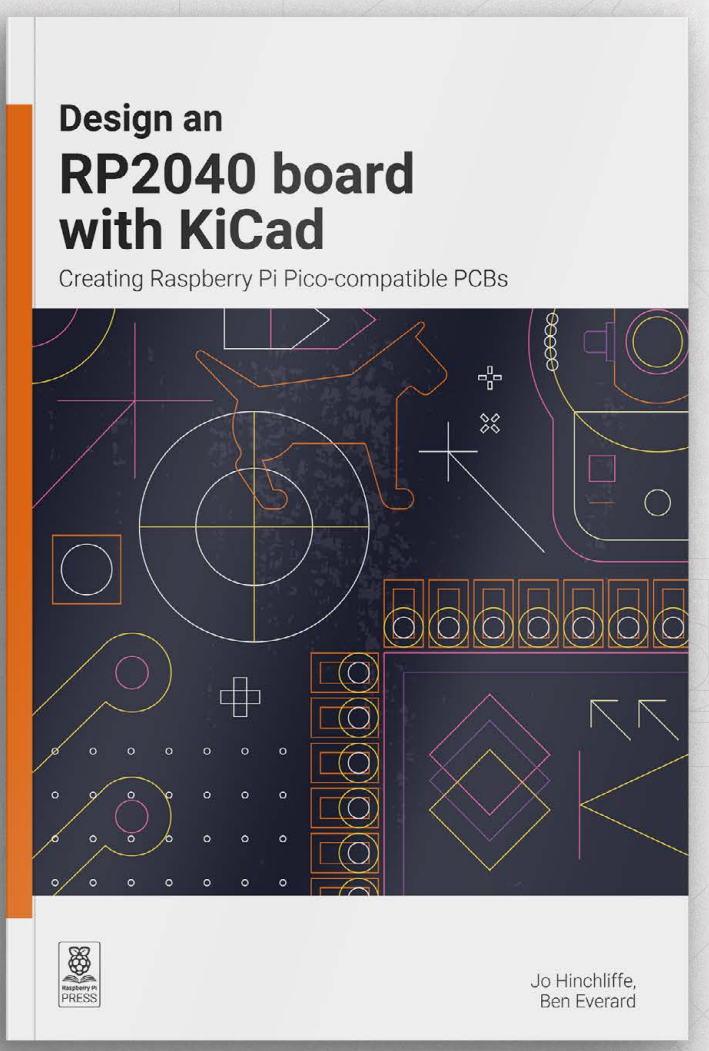
The Pioreactor is a capable device in its standalone form, but of course there are lots of add-ons and modifications available or in development in the community. As an example, **Figure 9** is an image of an expanded Pioreactor system. This system has a CO_2 system that can push CO_2 through the liquid in the chamber, which can be used to remove other volatile compounds in a sample. This process is known as 'sparging'. The CO_2 sparging system is well engineered, but made using items like the CO_2 bottle from a Sodastream device which are widely available and readily refillable. This modified Pioreactor also has peristaltic pumps added with surgical tubing and this allows for accurate dosing of additional material into a given experiment. For some of the AMYBO experiments, there is a need for hydrogen and oxygen to be present in the vial chamber; fabulously, this can be achieved by in-chamber electrolysis, all of which is being explored and developed. It's superb to see the community building and developing these complex tools for everyone.

It's been a great project to build so far, we enjoyed getting it built and commissioned, and we look forward to a lot of learning as we run some experiments and tests with our Pioreactor.



▲ **Figure 8:** An example set of results from an experiment with two Pioreactors both growing yeast, as a calibration test

▶ **Figure 9:** A Pioreactor set up with lots of additional features, CO_2 sparging, in-chamber electrolysis, and more



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

Part 02

Building a capable underwater rover

Last issue, we built a pipework chassis for a larger underwater rover. In this section, we look at creating drive control systems and a Raspberry Pi-powered underwater camera

Maker

Jo Hinchliffe

With a house and shed full of lathes, milling machines, 3D printers and more, Jo is a constant tinkerer and is passionate about making.

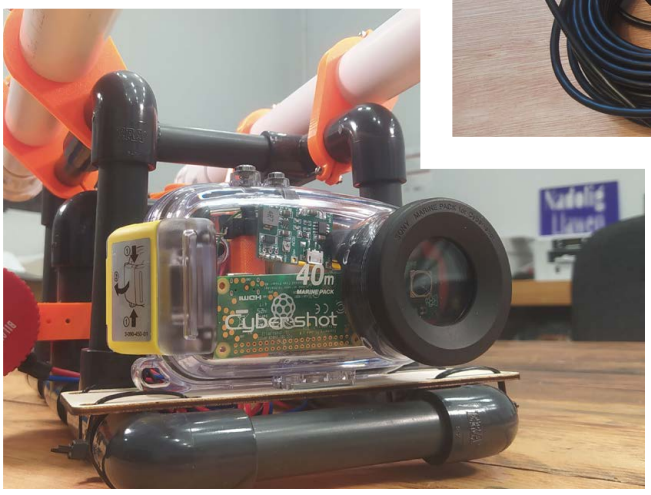
concretedog.blogspot.com

In the first part of this two-part series, we looked mainly at the chassis design of a larger and more capable underwater rover. We used FreeCAD to prototype a pipework assembly before moving to cutting and test-fitting a chassis and some additional floatation tubes. In this part, we'll look at some subsystems in development for the craft. We'll look at creating a simple drive system with a Raspberry Pi Pico, creating a prototype waterproof Raspberry Pi Zero camera system, and also look at a simple accessory to add a live visual feed from the craft.

Previously, we looked at the common approach of converting 1100 gallon per hour 12V bilge pumps into thrusters for the underwater vehicle. We briefly mentioned that we planned to use a BTS7960 motor driver board in conjunction with some buttons to create a simple drive controller. This essentially is similar in operation to the controller we built for the Tiny Open-source Underwater Vehicle (TOUV) in issue 157 (**rpimag.co/157**). However, of course, the uprated motor drivers are needed to run at the higher current draw and higher voltage.

The motor driver boards we bought claim to have a regulated 5V output so that, on paper, we could run a Raspberry Pi Pico handling the PWM signals to the boards without an extra power supply. However, on receipt of the modules, none of them actually appeared to have this regulated output, so for our prototyping we've provided power to Pico via a USB power bank.

For in-water testing, which will probably now wait till slightly warmer weather next year, we aim to emulate the type of simple control we built for the TOUV project. Namely, three buttons



Warning!
Water and electricity
This project is low-voltage (12V), but water and electricity don't mix. Make sure to test your waterproofing.
rpimag.co/waterproofing

◀ A DIY Raspberry Pi dive camera mounted to a front deck on our underwater rover

▲ **Figure 2:** An affordable 10m waterproof endoscope can make a simple live visual feed back to a phone or tablet

controlling the thrusters but with the Pico using tuned PWM values to set each thruster speed. As a starting point, this is a very usable system, but also allows for more complex control systems to be developed which might include variable speed with potentiometers or joystick input further down the line.

As an example of motor control using the BTS7960 IBT-2 board, we can wire up a Pico to the module as well as adding a momentary press-to-make switch, a pull-down resistor and, of course, our motor/bilge pump thruster. You can see the pin value connections in **Figure 1**.

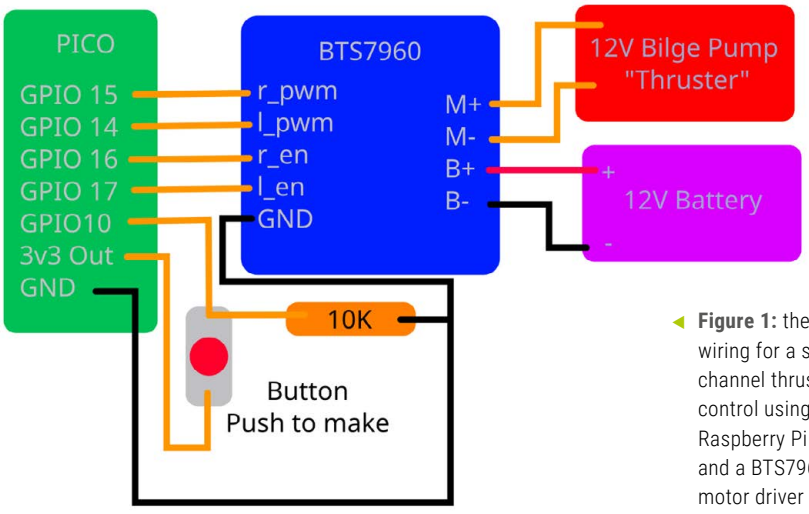
With our wiring in place, we created a simple MicroPython script using Thonny. The script is pretty straightforward and we can vary the speed by changing the `speed` variable value. We've found that it only really makes sense to run the motors in one direction on these three-thruster underwater craft, but you should see that it's perfectly possible to create bidirectional motor control should you wish to. This BTS7960 script is available in a GitHub repo: **rpimag.co/bts7960**.

One of the great things to do with this larger and hopefully more capable underwater platform is to add some form of live visual feed. We'd also like some method of capturing footage or potentially still images from on board. An easy and common way to approach a live visual feed from underwater is to run a video-carrying cable up through the tether. Indeed, this could be very possible looking at affordable wired CCTV or FPV cameras. However, another option that caught our eye was using a budget drain inspection camera, often referred to as a USB endoscope (**Figure 2**).

These are commonly available for not huge amounts online and they can be connected to your mobile phone with an app installed to allow you to view the video stream. The one we ordered was rated as waterproof and had a 10m cable, so it's certainly enough for us to begin to explore underwater, and adequate for some test missions with the platform when the summer months return.

The video feed from the endoscope can be captured, but we wanted to build something more experimental that could be used in lots of different ways using a Raspberry Pi.

One of the great things to do is add some form of live visual feed



◀ **Figure 1:** the wiring for a single channel thruster control using a Raspberry Pi Pico and a BTS7960 motor driver board

As a first experimental payload, we built an 'action camera' equivalent using a Raspberry Pi Zero and Camera Module 3. Obviously Raspberry Pi boards, such as the Zero 2 W we used for this section of the project, aren't waterproof. Please don't set one up and dunk it in water as you won't get much further use from it! So we started to think about waterproof enclosures. We'd also dug out a small action camera as part of the exploration and that prompted an interesting idea we hope others enjoy playing with.

▲ **Figure 3:** There are lots of older dive cases for early digital cameras available on second-hand marketplaces

Back in the early noughties, lots of handheld digital cameras started to hit the shelves and became much more common for people to buy and own. At the more capable and expensive end of the market, many manufacturers produced add-on dive enclosures to allow individual camera modules to be used underwater. Of course, these underwater dive enclosures were very well made and pretty expensive at the time, but often had the button layout that matched to a specific model of camera. Of course, this now means that they are very out of date and as such are very affordable on second-hand marketplaces (**Figure 3**).

We bought a couple that were branded for different models of Sony camera. Both of the camera dive cases were under £9 and one arrived with a complete spare set of seals and some silicon seal lubricant! The dive enclosures contain lots of waterproof button and dial connectors mounted through the case, which would have aligned with the camera model inside. With some 3D printing and some retrofitting, it's totally possible to mount momentary touch buttons or latching buttons so that they can be actuated by these original buttons (**Figure 4**).

▼ **Figure 4:** There are heaps of waterproof button pushers. We can use 3D printing to place switches and buttons within reach of these case features

As these dive cases are for older handheld cameras, there is lots of space inside and so it's pretty possible to add a Raspberry Pi Zero, a Camera Module, and a power system. Further down the line, we imagine this could become its own project and there's no reason we couldn't pack in camera lighting and more button controls as well as potentially adding other sensors and experiments. Even aside from the underwater rover, this could become a great project in its own right.

As a first pass at creating an underwater action camera, we designed and 3D-printed a mount for the Camera Module 3; this means that we can swap between the standard module or the NoIR version which may give better results in some situations. It's a challenge to measure the internal section of the lens area of the dive enclosure as it is slightly flexible, in an awkward spot, and is a tapered tube design. We made a couple of attempts and, as it is a relatively small part to 3D-print, we created a best guess version first, test-fitted after printing, and adjusted the CAD a couple of times for it to end up as a reasonably snug fit (**Figure 5**).

With a mount for the Camera Module 3 in place, we then created a simple post which in turn held a latching momentary push switch underneath one of the original buttons in the waterproof enclosure. Keeping things simple, we just used this inline with the positive connector of a LiPo cell and we created a system so that when the power button was pressed, Raspberry Pi Zero would boot and then run a small script to record a video file (**Figure 6**).

QUICK TIP

Before you mount electronics inside your dive enclosure, it's a good idea to place a piece of tissue paper inside and test the enclosure underwater, checking for any leaks.

▲ **Figure 5:** 3D-printed camera mount

Before any final assembly into the dive enclosure, we set up our Raspberry Pi Zero and Camera Module 3 to do some testing and to create the automated scripts. We actually set up our Zero 2 W with the Raspberry Pi HDMI monitor and a keyboard and mouse to do this and used the graphical version of Raspberry Pi OS with the desktop graphical environment. Of course, you could set the following up by SSHing into Raspberry Pi Zero from your desktop or laptop computer and, in turn, you could install lighter versions of the OS without the graphical desktop for a more lightweight responsive design. For ease of prototyping, though, a keyboard, mouse, and monitor can be easier, especially when visually checking the camera.

We connected the Camera Module 3 to Raspberry Pi Zero 2 W with the supplied ribbon cable (**Figure 7**). After doing the basic Raspberry Pi setup and booting to the desktop, we opened the terminal application and used some simple `rpicam` commands to check the camera was functional. A good start is to try:

```
$ rpicam-hello
```

This will launch a small preview window with a livestream preview from the camera. Similarly, you can check that the camera can take a still image with:

```
$ rpicam-still -o test_image.jpg
```

This will create an image file called **test_image.jpg** in your home folder.

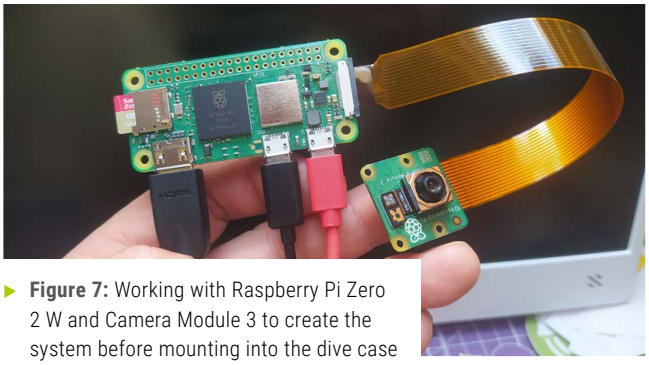
Finally, you can test and perhaps experiment with changing settings/frame rate/dimensions to create short video clips with a command like:

```
$ rpicam-vid -t 5s -o test_video.h264
```

▼ **Figure 6:** Using tiny spots of hot glue, we can create a sturdy yet reversible mount post to place a latching button under a button presser

…where the value following `-t` is the duration of the clip in seconds. There's a whole lot more you can do with the Camera Module 3: long exposure photography, intervalometer projects, and more. A great primer for it can be found here: **rpimag.co/camera-software**.

Next, we created a couple of scripts that formed the system so that when the Raspberry Pi Zero was powered up, it would boot and then record a small video clip. We wanted to be able to increment the file name of the saved video clip so that we could repeat the operation and create numerous test files without overwriting the previous recording. To do this, we created a script called **record_boot_video.sh** in our home directory. Note that both the following scripts we create are available here: **rpimag.co/autorecordvideo**.



▶ **Figure 7:** Working with Raspberry Pi Zero 2 W and Camera Module 3 to create the system before mounting into the dive case

To do this, we opened the terminal application and issued the command:

```
$ sudo nano record_boot_video.sh
```

This launches the nano text editor, into which we pasted:

```
#!/bin/bash

sleep 10 # wait 10 seconds after boot, I found
10 seconds the minimum delay that worked. Might
be safer at 15 seconds.

COUNTFILE="/home/cdog/bootcount.txt"

if [ ! -f "$COUNTFILE" ]; then
echo 1 > "$COUNTFILE"
fi
```

```
  COUNT=$(cat "$COUNTFILE")

  rpicam-vid -o "/home/cdog/video_${COUNT}.h264"
-t 5000 #this command captures a 5 second video
adapt for your needs

  NEXT=$((COUNT + 1))
  echo $NEXT > "$COUNTFILE"
```

Note that you need to change the username sections of the above script to match your username of your system; so wherever our script says 'cdog', you insert your username.

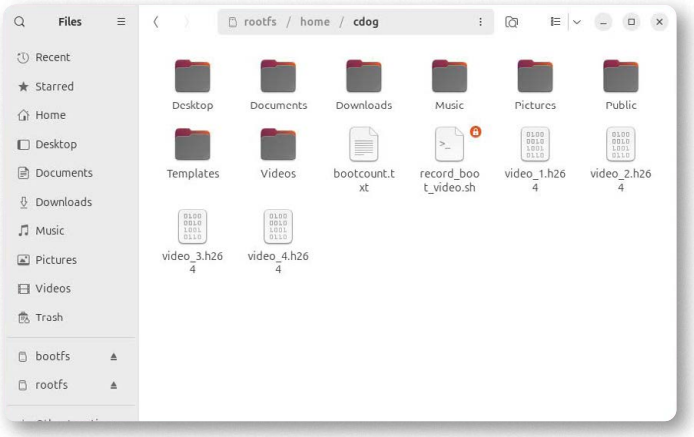
You can see at the top of this script that when it runs, it is going to check for (and in the first run, create) a text file called **bootcount.txt**. Each time this script is run, it will add 1 to the running count value inside this script. It then goes on, in the middle of the script, to record a video file, using the `rpicam-vid` command we tried earlier, but this time it will name the file 'video_' and then the number detected in **bootcount.txt** will be suffixed to the file name. This means that we can repeatedly try the system and instead of always overwriting the same single video file, we will end up with multiple uniquely named files: **video_1**, **video_2**, **video_3**, etc.

Next, we need to add the following script to the directory **/etc/systemd/system**:

```
  #sudo nano /etc/systemd/system/bootrecord.
service #add below script

  [Unit]
  Description=Record video on boot
  After=graphical.target

  [Service]
  Type=idle
  User=cdog
  ExecStart=/home/cdog/record_boot_video.sh

  [Install]
```

Once you have added the **bootrecord.service** script to systemd, you need to enable the service by running the command:

```
  $ sudo systemctl enable bootrecord.service
```



▲ Having booted the test camera system, we can see that it has captured and created numerous sequentially numbered video files

You can then, as a first time test, run the service manually. Normally the service will just run on boot, but for this first run, issue the following command:

```
  $ sudo systemctl start bootrecord.service
```

This will make the system run immediately, but the process will fail! This is because the first time the **record_boot_video.sh** script runs, it will create the **bootcount.txt** file, but we will need to change the permissions of that file to be able to write to it. To do this, we can make the **bootcount.txt** writable by any user by using the `chmod` command:

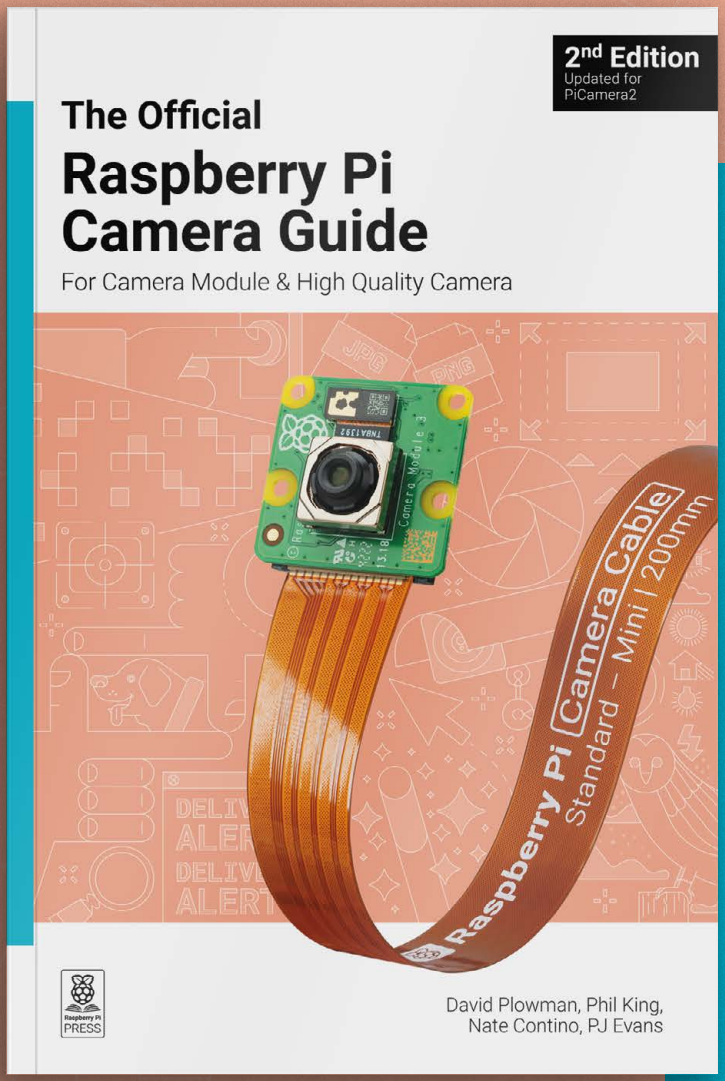
```
  $ sudo chmod 664 /home/cdog/bootcount.txt
```

Again, you need to change 'cdog' to your own username in the above command.

You should now find that whenever you reboot the system, it will boot and then automatically record a new video clip.

As the Sony camera dive case is designed as a real camera system, it's pretty adaptable in terms of ways that it can mount. On the underside of the case, there is a standard ¼-inch 20 turns per inch threaded camera mount, so we can use all manner of standard camera mounting systems for it. In the first instance, we've laser-cut a small 'for deck' section which is cable-tied to the front of our chassis and we can use a small bolt from our collection of camera mount equipment to mount the DIY underwater camera system.

We're really pleased with the whole build and can't wait to get out testing and exploring with it when the weather is a little warmer. That said, it's a great project to tinker with on your desk in winter, and we are inspired to play more with the idea of retrofitting old camera dive enclosures with fun experiments.



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

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

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

Part 01

Object detection with Ultralytics YOLO26 on Raspberry Pi

Use Docker and Ultralytics YOLO to perform object detection on images and video

Maker
Lucy Hattersley
Lucy is editor of Raspberry Pi and is classifying images of her cat into 'furball' vs 'clawball'.

rpimag.co

YOLO (You Only Look Once) is a powerful object detection model created by Ultralytics that enables you to identify content in images and video from the command line and Python. From here, you can perform classification and respond to images or videos with your code.

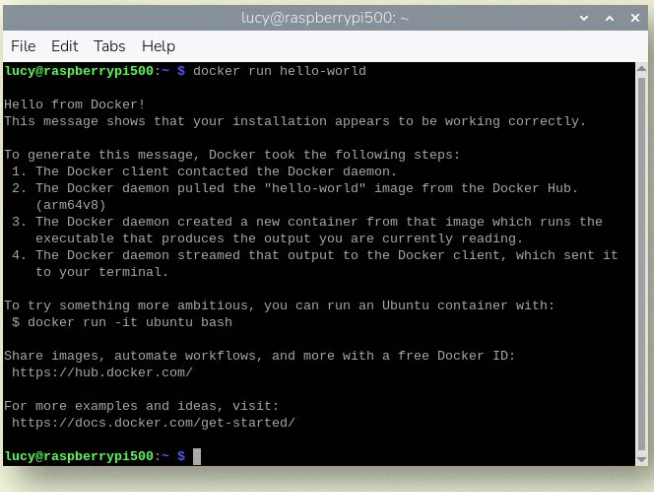
When paired with a Raspberry Pi Camera Module, YOLO forms a powerful means of identifying objects that your Raspberry Pi board can react to. You can use it with sensors and actuators connected to Raspberry Pi to perform real-time identification and reaction. You can also use it to analyse images and video files.

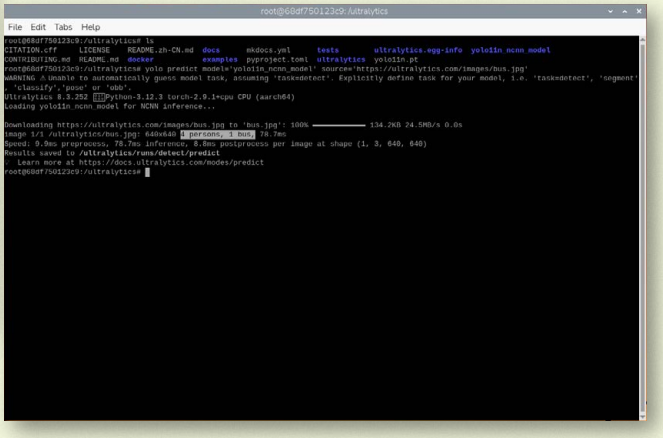
YOLO (You Only Look Once) forms a powerful means of identifying objects that your Raspberry Pi board can react to

YOLO26 has just been released, and we are using it here. It is custom-built for speed, accuracy, and versatility. You can use YOLO out of the box or train your own dataset on it.

In this tutorial, we're going to look at installing the Ultralytics framework on a Raspberry Pi 5 with images and video files, both online and in our computer system. We'll also look at setting up Docker so you can install the environment and the programs needed.

You don't need a Raspberry Pi Camera Module for this, but a reasonably powerful Raspberry Pi will help. We are using a Raspberry Pi 5 for the tutorial. In following tutorials we will look at integration with a Raspberry Pi Camera Module.





▲ Check that Docker is up-and-running
▶ Using YOLO to download an image and perform inference on it

Install Docker

Set up your Raspberry Pi 5 computer with Raspberry Pi OS (see **rpimag.co/docs** for help with these steps). We start by installing Docker Engine in Raspberry Pi OS.

Add Docker to apt

To install Docker Engine, you should be running the latest version of Raspberry Pi OS based on Debian Trixie (it will also work on Bookworm and Bullseye). These instructions follow the Docker documentation guide for Debian (**rpimag.co/docker**). Docker provides separate Raspberry Pi installation instructions, but these are geared towards the old 32-bit version of Raspberry Pi OS, so stick with the Debian install.

First, make sure you have uninstalled any old Docker packages. Open a terminal window and enter:

```
$ sudo apt remove $(dpkg --get-selections
docker.io docker-compose docker-doc podman-docker
containerd runc | cut -f1)
```

Unless you have Docker already installed, `apt` will report that these packages are not found.

Next we'll add Docker's official GPG (GNU Privacy Guard) key to the **keyrings** folder. First, we update `apt`, then install `ca-certificates` and `curl`:

```
$ sudo apt update
$ sudo apt install ca-certificates curl
```

YOU'LL NEED

- Raspberry Pi 5 (rpimag.co/raspberrypi5)
- Raspberry Pi OS (rpimag.co/software)
- Docker (rpimag.co/docker)
- Ultralytics YOLO26 (rpimag.co/yolo)

TIP! TAKE A LOOK

Check in on the **docker.asc** file with:

```
$ ls /etc/apt/keyrings
```

You can also view the public key with:

```
$ cat /etc/apt/keyrings/
docker.asc
```

What is Docker?

Docker enables us to download images with containerised applications. Containers are like a virtual machine, but they contain all the applications needed to run a particular program and share the host kernel. Docker makes it easy to download and run a sandboxed environment with all the software components you need for a particular project.

Read the 'Install Docker Engine on Debian' guide for advice on installing and using the latest version of Docker with Raspberry Pi OS.

rpimag.co/docker

These should be already installed. We make sure our **keyrings** directory has the correct permissions: 0755. This enables the file owner (our admin account) to read, write, and execute. Just read and execute permission is set for groups and others. We do this with a funky install command that is normally used for copying files but in this instance is being used to adjust permissions.

```
$ sudo install -m 0755 -d /etc/apt/keyrings
```

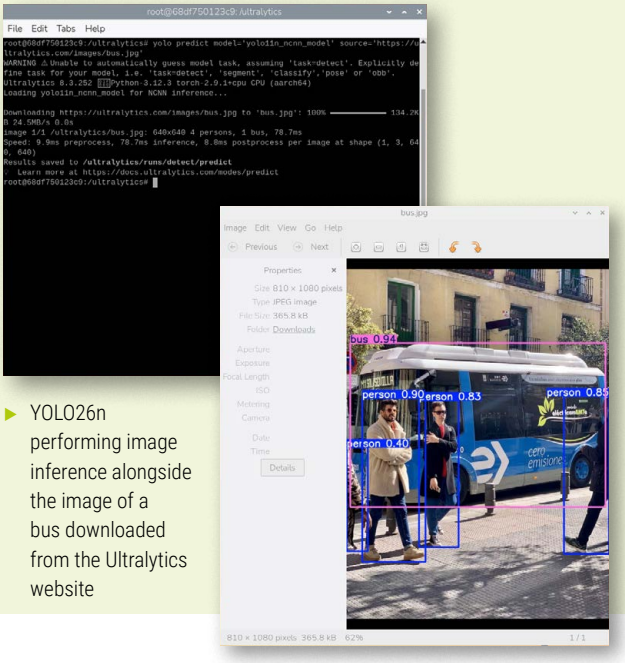
Now we use `curl` to download Docker's GPG key and place it into our **keyrings** directory with the file name **docker.asc**.

```
$ sudo curl -fsSL https://download.docker.com/
linux/debian/gpg -o /etc/apt/keyrings/docker.asc
```

We need to ensure that all users can read the **docker.asc** file. We do this with the standard `chmod` command with `a+r` options:

```
$ sudo chmod a+r /etc/apt/keyrings/docker.asc
```

Next comes a funky multi-line piece of code that creates a file called **docker.sources** in our **/etc/apt/** directory and contains the details of the Docker repository. Enter the first line and you will see a `>` in terminal. Enter each line carefully and press **RETURN** after each one. Each line is added to the **docker.sources** text file until you enter `EOF` (at which point you return to the command line).



- YOLO26n performing image inference alongside the image of a bus downloaded from the Ultralytics website

```
$ sudo tee /etc/apt/sources.list.d/docker.
sources <<EOF
Types: deb
URIs: https://download.docker.com/linux/debian
Suites: $(. /etc/os-release && echo "$VERSION_
CODENAME")
Components: stable
Signed-By: /etc/apt/keyrings/docker.asc
EOF
```

Check that the **docker.sources** file has been created correctly:

```
$ cat /etc/apt/sources.list.d/docker.sources
```

The output is expected to list the following, where `Suites` is the `VERSION_CODENAME` of your operating system (`trixie`):

```
Types: deb
URIs: https://download.docker.com/linux/debian
Suites: trixie
Components: stable
Signed-By: /etc/apt/keyrings/docker.asc
```

If there's a problem, use vim or nano to edit your file:

```
$ sudo nano /etc/apt/sources.list.d/docker.
sources
```

Check the update

Now update the system and check access to Docker downloads:

```
$ sudo apt update
```

The output should include a line like this:

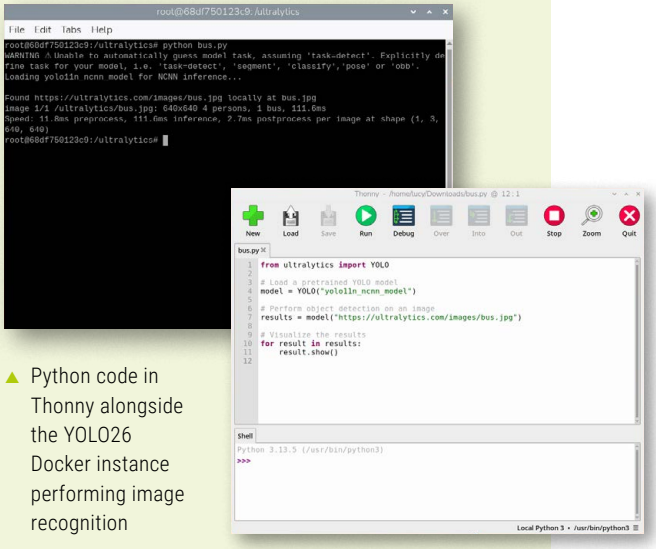
```
Get:5 https://download.docker.com/linux/debian
trixie InRelease [32.5 kB]
```

Install Docker

Now that the Docker repository is in apt, it's time to install the various elements. Enter this line in the terminal:

```
$ sudo apt install docker-ce docker-ce-cli
containerd.io docker-buildx-plugin docker-
compose-plugin
```

Docker should run automatically after installation. To verify that Docker is running, use:



▲ Python code in Thonny alongside the YOLO26 Docker instance performing image recognition

```
$ sudo systemctl status docker
```

Press **q** to exit `systemctl` and return to the command line. Some systems may have this behaviour disabled and will require a manual start:

```
$ sudo systemctl start docker
```

Finally, verify that the installation is successful by running the hello-world image:

```
$ sudo docker run hello-world
```

If this is the first run, it will pull the library/hello-world container from the Docker Hub. You will see a message containing:

```
Hello from Docker!
```

This message shows that your installation appears to be working correctly.

Add user to Docker

Adding a user to the docker group will prevent us from having to use `sudo` with each `docker` command:

First, create a docker group:

```
$ sudo groupadd docker
```

Now, add your user to the docker group:

```
$ sudo usermod -aG docker $USER
```

Note! You are root

The Docker container runs as root – you can tell, as the user and host of the command line will be something like:

`root@cc705521e3e2:/ultralytics#`

Remember that you don't need to preface commands that normally require elevated privileges with `sudo`.

Warning!

Security risk

Adding your user to Docker so you don't have to use `sudo` enables password-free root access for your user from the command line. It's not recommended in shared environments.

Sign out and back in again so that your group membership is re-evaluated, or run the following command to activate changes to the group:

```
$ newgrp docker
```

Finally, test docker again, now without `sudo`:

```
$ docker run hello-world
```

You should see the same `Hello from Docker!` message.

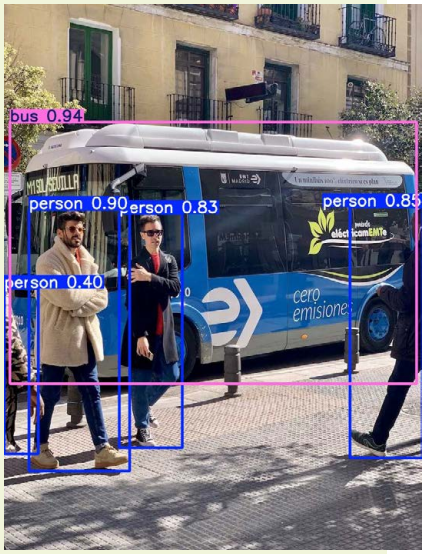
Start with Docker

Now that we have Docker installed, we can pull the pre-built Ultralytics YOLO26 image for Raspberry Pi.

Execute the following commands to pull the Docker container and run on Raspberry Pi. This is based on the arm64v8/debian Docker image which contains Debian 12 (Bookworm) in a Python 3 environment.

```
$ t=ultralytics/ultralytics:latest-arm64
$ docker pull $t
$ docker run -it --ipc=host $t
```

You will now be running a Docker instance; the prompt will be replaced with:



▼ Ultralytics' standard demonstration image features a bus and four people

Resources

Install Docker
- **rpimag.co/dockerinstall**
- **rpimag.co/dockerinstallrpios**

Ultralytics
- **rpimag.co/ultradocs**

NCNN
- **rpimag.co/ncnndocs**

Python usage
- **rpimag.co/ultrapython**

```
root@44214a2e5000:/ultralytics#
```

Enter `ls` to take a look at the files inside the Docker instance. Note the presence of **yolo26n.pt** in the directory. We will be using this PyTorch file in the next step.

Use YOLO26n from the command line

To perform image detection on Raspberry Pi, we export the YOLO26n model from PyTorch format to NCNN (**rpimag.co/ncnn**).

Out of all the model export formats supported by Ultralytics, NCNN delivers great inference performance when working with Raspberry Pi devices because NCNN is highly optimised for mobile and embedded platforms (such as ARM architecture) which makes it ideal for our Raspberry Pi.

The YOLO26n model in PyTorch format is converted to NCNN to run inference with the exported model. Let's convert the file from the command line:

```
$ yolo export model=yolo26n.pt format=ncnn
```

This takes the **yolo26n.pt** file in our directory and creates a **yolo26n_ncnn_model** directory. Take a look inside with:

```
$ ls yolo26n_ncnn_model
```

You will see several files that make up the NCNN model:

```
metadata.yaml  model.ncnn.bin  model.ncnn.param
model_ncnn.py
```

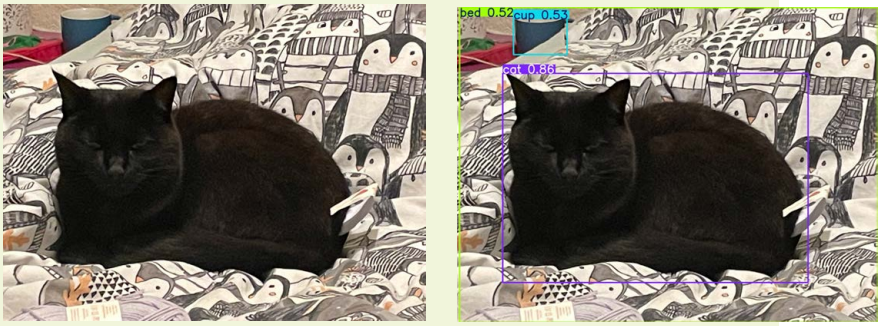
We now use the `yolo predict` command with two options: `model`, which is pointed to the **yolo26n_ncnn_model** folder and `source`, which in this instance is a **bus.jpg** image on the Ultralytics website:

```
$ yolo predict model='yolo26n_ncnn_model'
source='https://ultralytics.com/images/bus.jpg'
```

This code pulls an image of a bus from the Ultralytics website and performs object inference on it. Look for these lines in the output:

```
image 1/1 /ultralytics/bus.jpg: 640x640 4
persons, 1 bus, 194.4ms
Results saved to /ultralytics/runs/detect/
predict
```

This informs us that there is one image, it's called **bus.jpg**, is 640×640 resolution, and inference saw 4 people and 1 bus. The model took 194.4ms to run. It also lets us know where the prediction was saved. Enter `ls` and look in the **/ultralytics** folder again. You'll see the **bus.jpg** file that was downloaded. Enter:



An image of a cat that we uploaded to the Ultralytics instance and performed inference on

```
$ ls runs/detect/predict
```

...and you will see another **bus.jpg** file. This one has bounding boxes around it for viewing.

There's no easy way to view this file inside the Docker instance as we don't have any graphics programs or a windowing system. So this is a good chance for us to learn how to add and remove files from our Docker instance to the desktop.

Copy files to and from Docker

Don't close the Docker instance or the image and model we just used will be lost. Instead, open a new, second, terminal instance to reach inside Docker from our user account. First, we need to know what the name of our Docker instance is:

```
$ docker ps
```

This should give us information on our running docker instances (this is a shortcut for `docker containers list`).

It will return something like this:

```
CONTAINER ID   IMAGE
c1cdfe411f9f   ultralytics/ultralytics:latest-arm64
COMMAND        CREATED          STATUS
"/bin/bash"    4 minutes ago    Up 4 minutes
PORTS          NAMES
               tender_nobel
```

The name of our container is created randomly from two words and will be under `NAMES`. In our case it is `tender_nobel`, but yours will be different. We use this to reach inside the container and grab the **bus.jpg** file with Docker's built-in copy (`cp`) command with source and destination paths:

You can start to integrate the vision detection models into your own code

The number next to the bounding box shows how confident the model is of its identification of the detected object. In this instance, it is 88% sure this is a dog

```
$ docker cp tender_nobel:/ultralytics/bus.jpg
~/Downloads
```

This copies the **bus.jpg** file from our default **ultralytics** folder (where the container starts) to our **Downloads** folder. Take a look at it with:

```
$ xdg-open ~/Downloads/bus.jpg
```

This is the test file downloaded from Ultralytics. What is more interesting is the processed file after we have performed inference on it. These live inside our **runs/detect** folder:

```
$ docker cp tender_nobel:/ultralytics/runs/
detect/predict/bus.jpg ~/Downloads
```

NOTE

If you have run more than one prediction, it may be saved in another folder such as **predict2** or **predict3**. Take a look using `ls runs/detect` inside the Docker container to find your **bus.jpg** file.

Open this file in the Files interface in an image view to see the bus and four people with bounding boxes around them.

Use your own files

You can replace the source reference in our `yolo predict` command with the URL for a different image (although some services will block it). More importantly, you can use `docker cp` to send files to the Docker instance.

We've got an image of our cat sitting on a bed. We can send it to our Ultralytics instance and perform inference on it. In the terminal (outside of the Docker instance), enter:

```
$ docker cp cat.jpg tender_nobel:/ultralytics/
cat.jpg
```

Replace `tender_nobel` with the name of your Docker instance.

Head back to your terminal window running the Docker instance, and perform inference on the image you just sent over:

```
$ yolo predict model=yolo26n_ncnn_model
source='./cat.jpg'
```

Video files

We can also perform inference on video files, both online and uploaded directly to our Docker instance with the yolo26n-seg.pt model. We will use the `yolo predict` command to perform object detection on a YouTube video.

As with our image model, we are first going to convert it to the NCNN format for improved performance on our Raspberry Pi:

```
$ yolo export model=yolo26n-seg.pt format=ncnn
```

NOTE

You can run the yolo26n-seg.pt (PyTorch) model on Raspberry Pi. It just takes longer.

```
$ yolo predict model=yolo26n-seg_ncnn_model
source='https://youtu.be/LNwODJXcvt4' imgsz=320
```

As it runs, you'll see frame detections on the command line:

```
0: 640x640 1 person, 1 potted plant, 1 tv, 134.2ms
0: 640x640 1 person, 1 potted plant, 1 tv, 139.1ms
0: 640x640 1 person, 1 chair, 1 potted plant, 1
tv, 150.7ms
0: 640x640 1 person, 1 chair, 1 potted plant, 1
tv, 1 laptop, 130.4ms
```

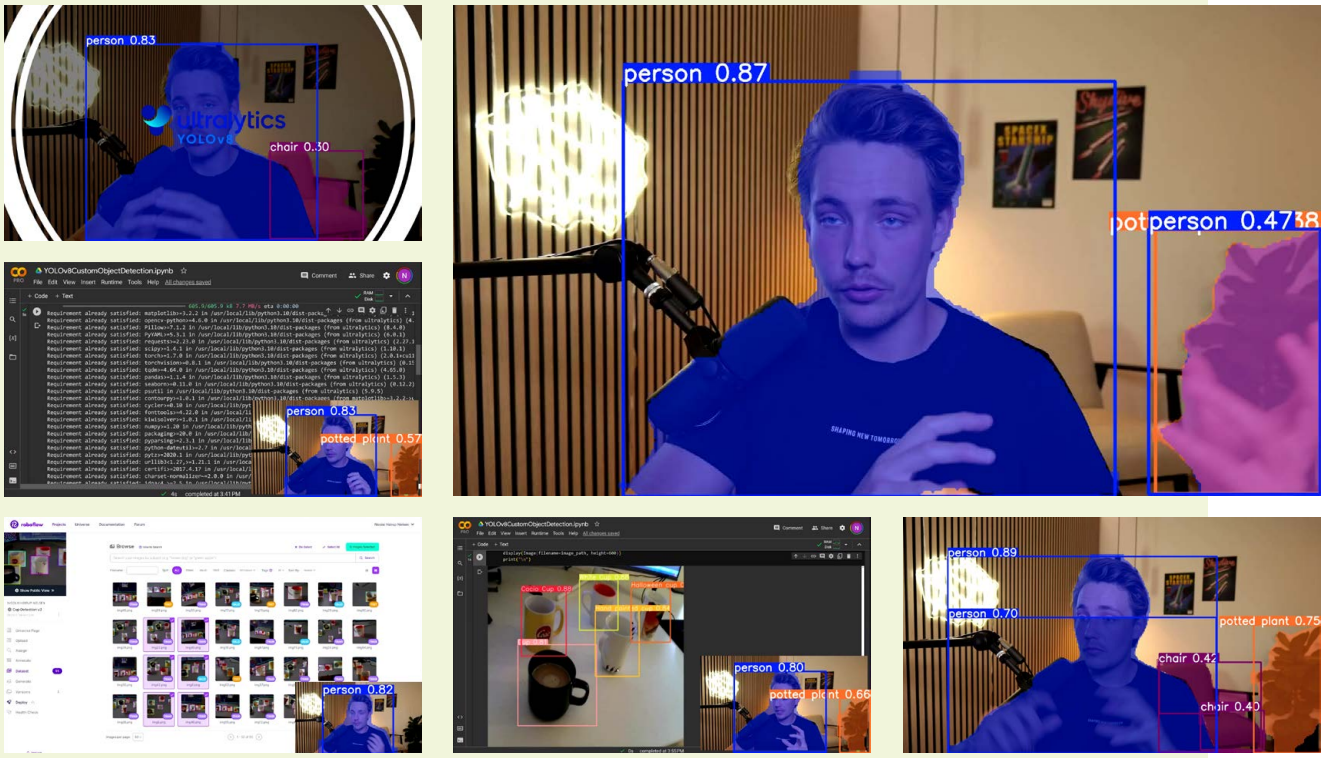
When the inference has finished you can download video with the exported frames and bounding boxes. Head to another terminal window (outside of the Docker instance) and enter:

```
$ sudo docker cp <container_name>:/ultralytics/
runs/segment/predict/https___youtu.avi .
```

...replacing `<container_name>` with your own container (enter `docker ps` to get the name).

Python usage

Performing object detection from the command line is fun, but using inference models in Python is where you can start to explore the power of AI in coding. At this point, you can start to integrate the vision detection models into your own code, and connect it to sensors and actuators.



▲ A selection of images from a YouTube video with image detection performed on it

Enter the code from the **bus.py** listing to replicate downloading the **bus.jpg** file from Ultralytics, performing inference, and outputting the results to the terminal window.

NOTE!

You may want to install a text editor, such as nano or vim, to work on your Python code. You can do this with apt inside the Docker instance.

```
$ apt update
$ apt install vim -y # or nano
```

Or edit the file in your home workspace and use `docker cp` to copy it over. Save it as **bus.py** to the **ultralytics** folder (or a subfolder) in your Docker instance.

To run the code, enter the command:

```
$ python bus.py
```

Now that you have Ultralytics YOLO26n up and running, you can work through the Python documentation and integrate it into your projects. Bookmark **docs.ultralytics.com** and also take a look at the 'Resources' boxout to continue your journey.

bus.py

> Language: **Python**

DOWNLOAD THE FULL CODE:
rpimag.co/github

```
001. from ultralytics import YOLO
002.
003. # Load a pretrained YOLO model
004. model = YOLO("yolo26n_ncnn_model")
005.
006. # Perform object detection on an image
007. results = model("https://ultralytics.com/
     images/bus.jpg")
008.
009. # Visualize the results
010. for result in results:
011.     result.show()
```

ENIAC (Electronic Numerical Integrator And Computer)

Built to do military calculations, it was the first general-purpose digital computer

Maker

Tim Danton

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

dantonmedia.com

If war stymied the development of the ABC, it was the making of the ENIAC. Its story can be simplified to one sentence: the US Army needed a more efficient way to calculate firing tables for its weapons, conventional methods couldn't keep up, and a bright young professor called John Mauchly had the answer in the form of an electronic computer.

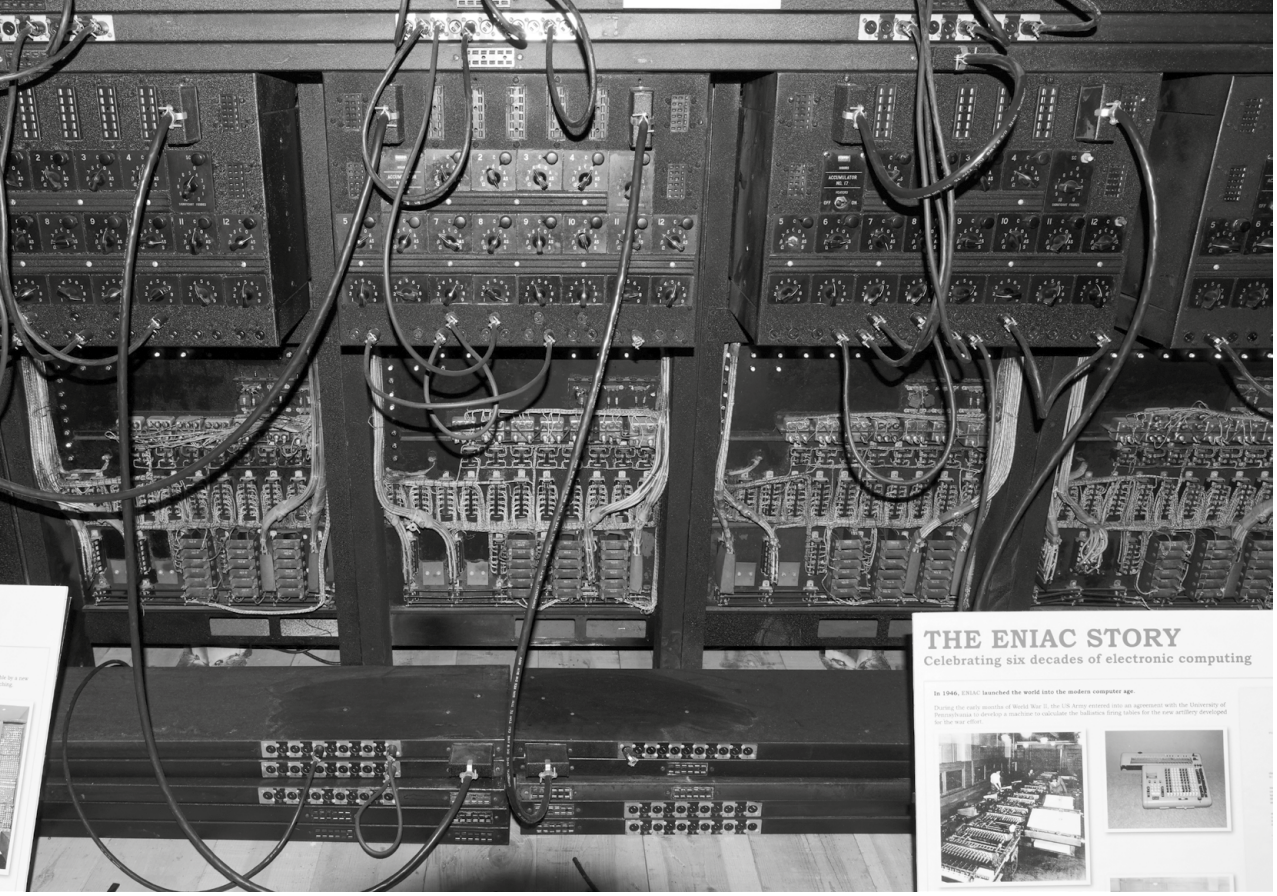
The real story is far more complicated. One that has also been camouflaged by lawsuits, huge corporations, warring egos, and the breadth of time. Here, we will attempt to navigate our way to the truth – but, in the same way that 1940s firing tables could only offer accuracy within certain bounds, we must do the same.

One thing we can state with confidence is that the ENIAC would have remained a concept were it not for John Mauchly meeting Herman Goldstine, a mathematician and officer at the US Army's Ballistic Research Laboratory (BRL). Goldstine saw his primary duty as "finding new apparatus to expedite the production of firing and bombing tables,"[1] which were in hot demand from the US Army and Air Force.

> *If war stymied the development of the ABC, it was the making of the ENIAC*

We will let Harry Polachek, who headed the BRL group that created the Army's firing tables, set out the scale of the

1 Herman H Goldstine, *The Computer: from Pascal to von Neumann* (Princeton University Press, 1993, ISBN 978-0691023670), p138



◀ ENIAC computer on display at Fort Sill Museum, Lawton, Oklahoma, USA **Image:** Judson McCranie, CC BY-SA 3.0

challenge.[2] Using a desk calculator, he explained, "it required an average of two eight-hour days to compute the path of a single trajectory. To compute a firing table, it was necessary to find the solution to hundreds of trajectories in addition to carrying out extensive auxiliary computations of equal difficulty." Depending on the weapon in question, that could take up to three months.

When Polachek joined the BRL in March 1941, the lab had a dozen staff. That itself was double the number from a year earlier. The US had not yet joined the conflict – that would come four days after Japanese planes attacked Pearl Harbor on 7 December – but it was already obvious to the US military that war was only a matter of time. They needed new firing and bombing tables to accompany their new weapons.

A year earlier, the director of the BRL – Colonel Herman H Zornig – had visited the University of Pennsylvania, in particular the Moore School of Electrical Engineering, to request use of its differential analyser to help create firing tables. While differential analysers weren't as accurate as desktop calculators, they could work a single trajectory much more quickly: "about 15 to 30 minutes" according to Polachek, although he added the caveat that "it required a lengthy period (a day or more) to change from one type of trajectory to another."

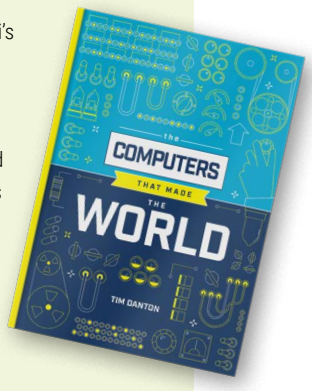
Making a difference

Between them, the BRL and Moore School owned two out of three differential analysers in existence in the US at the time, but it wasn't enough. More manpower was required; or to be accurate, more women power. Men were already being drafted into the Army, and so when the BRL asked the Moore School to provide "a staff of human computers"[3] it was happy to cooperate. "Moore School immediately agreed, and the staff, primarily college graduates who could handle some mathematics, quickly grew to more than 100," said John Grist Brainerd, a professor at the Moore School who was in charge of liaison with the BRL.

The Computers that Made the World

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

rpimag.co/tctmtw



2 Harry Polachek, 'Before the ENIAC [weapons firing table calculations],' in IEEE Annals of the History of Computing, Vol 19, No 2, Apr–Jun 1997, doi: 10.1109/85.586069, pp25-30

3 John G Brainerd, 'Genesis of the ENIAC,' in Technology and Culture, Vol 17, No 3, July 1976, pp482-488

However, the computers' work was dull. "It's like doing your income tax every day of your life, from morning till night," Goldstine explained in an interview with historian Nancy Stern.[4] While the Army wanted the women to be based at the Aberdeen Proving Ground in Maryland, the women themselves were less keen. "Aberdeen was about 32,000 acres of swamp, and the town itself was a little wide place in the road... and it was pretty much a hellhole," said Jean Bartik, one of the ENIAC's initial programmers.[5] Why would someone want to stay there, in barracks, when they could live in an apartment in central Philadelphia?

While differential analysers weren't as accurate as desktop calculators, they could work a single trajectory much more quickly

The answer was they wouldn't, instead working out of the Moore School, and this turned out to be fortuitous. On one of his regular trips to check on the computers, Goldstine spoke to John Mauchly's ex-student Joseph Chapline – who would go on to write an operating manual for the BINAC that is said to have set the template for computer manuals thereafter. Chapline suggested that Goldstine seek out his former lecturer, who had long been espousing the creation of electronic computers to crunch numbers, and wanted to create a machine capable of forecasting the weather.

Mauchly recalled his and Goldstine's first meeting and discussion about an electronic computer in a separate interview with Stern.[6] "After we'd talked a little bit, he [Goldstine] said, 'You ought to write this up.' I said, 'I have.' 'Well, where is it?' 'Well, I don't know. I'll look.'"

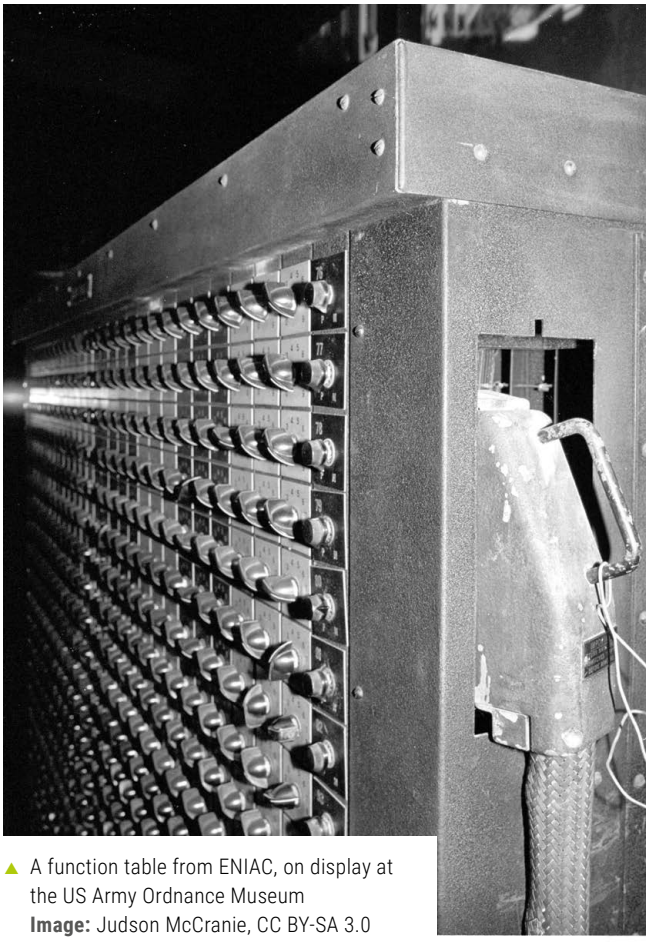
Mauchly had sent the memo to various members of the faculty, including Brainerd, but their reaction had been lukewarm at best. By the time Mauchly met Goldstine for the first time, not only had Mauchly stopped pushing the idea (in writing at least) but all copies of the memo had been lost.

This may seem odd in the light of the ENIAC's importance. Surely Mauchly would be feverishly attempting to turn his electronic computing concept into reality? After all, he was passionate enough about it to drive a thousand miles to visit Dr Atanasoff, as we heard in the story of the ABC, on discovering the Iowa State professor had created a prototype digital computer.

However, according to Goldstine, this fits Mauchly's character as an ideas man rather than a finisher. "Wilks [Samuel Wilks, an American mathematician] said Mauchly would occasionally drive down... and spend the day talking to Wilks about his ideas on applying methods of statistics and probability theory to meteorology, and then would just disappear, and nothing would come out of it in the form of a

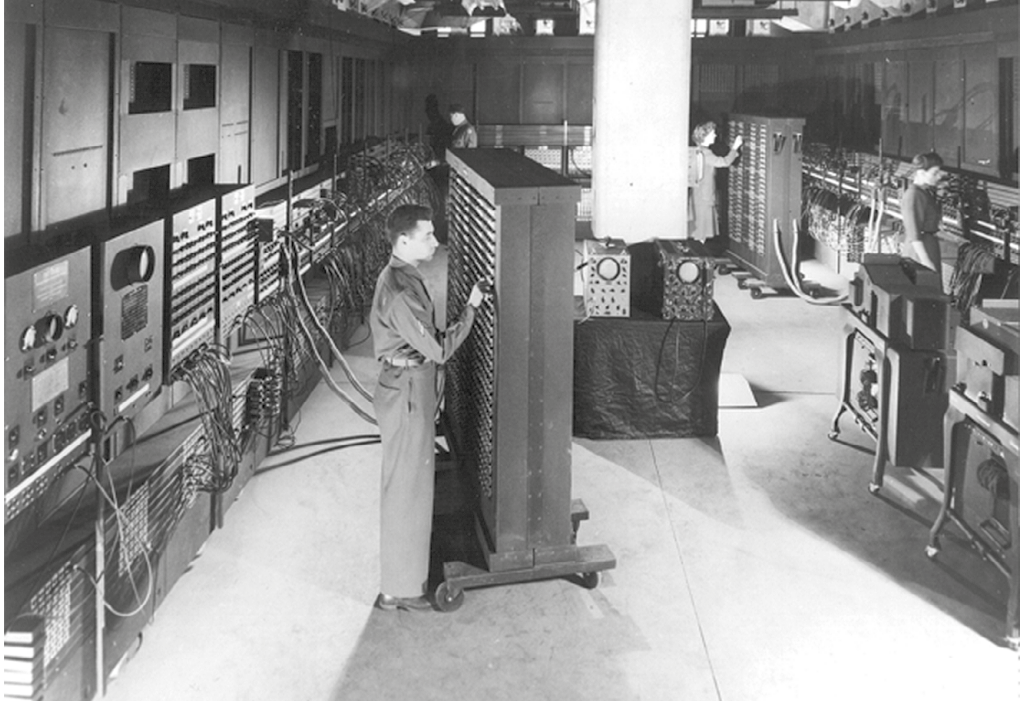
4 Interview with Herman Goldstine by Nancy Stern on 14 March 1977, Niels Bohr Library & Archives, American Institute of Physics, College Park, MD USA, **rpimag.co/goldstineinterview**

5 Interview with Jean Bartik by Gardner Hendrie on 1 July 2008, Computer History Museum, reference number X4596.2008, **rpimag.co/bartikinterview**



▲ A function table from ENIAC, on display at the US Army Ordnance Museum
Image: Judson McCranie, CC BY-SA 3.0

6 Interview with John Mauchly by Nancy Stern on 6 May 1977, Niels Bohr Library & Archives, American Institute of Physics, College Park, MD USA, **rpimag.co/mauchlyinterview**



▲ Cpl. Irwin Goldstein sets the switches on one of ENIAC's function tables at the Moore School of Electrical Engineering
Image: US Army photo, 1946, Public Domain

paper. This is just not… the normal academic method."[7]

Goldstine added: "I'm not saying this in any disrespect of Mauchly's abilities. I'm merely commenting that he was not a person who was good at starting something and seeing it through to fruition… I think that's why Mauchly could have written many proposals to no effect, because he always did them to perhaps entertain himself."

It certainly explains how six months later his proposal for an electronic computing device had evaporated into thin air. Fortunately, Mauchly's secretary still had the shorthand notes for the original memo, and she quickly reconstructed the document. Over the course of the next few weeks, with the help of an outstanding young engineer called J. Presper Eckert, that memo turned into a formal proposal for the building of the world's first general-purpose electronic computer.

On 9 April 1943, Eckert's 24th birthday, Goldstine and Brainerd took the proposal to the Ballistic Research Laboratory at Aberdeen. Despite the proposed $150,000 budget, Colonel Paul Gillon, who was responsible for the computing branch of the BRL, was quick to give the project his backing.

"He [Gillon] was convinced that the computing load at the Proving Ground was not going to be eased by any change of factors of two or three, or things like that,"[8] said Goldstine. "He saw that what really was needed was something like an order of magnitude, and he was prepared to go forward with almost anything, if it was only reasonable."

It would be wrong to give Colonel Gillon all the credit. Oswald Veblen, then the chief scientist at the BRL and a distinguished Maths professor at Princeton University, was another big name to back the project. In his book, *The Computer: from Pascal to von Neumann*,[9] Goldstine describes a BRL meeting in the spring of 1943 involving him, Veblen, and Colonel Leslie Simon, where "Veblen, after listening for a short while to my presentation and teetering on the back legs of his chair brought the chair down with a crash, arose, and said, 'Simon, give Goldstine the money.'"

War, what is it good for?

Still, others in the Army were harder to win around. It would require a huge number of vacuum tubes, still in short supply, and some senior figures believed two alternative projects would offer results more quickly. First, Vannevar Bush's Rockefeller Differential Analyzer, an electronic version of the differential analyser that required minutes rather than a full day to set up for a different weapon. Second, the Harvard Mark I, an electromechanical computer built upon the principles set out by Charles Babbage's analytical engine a century earlier.

Arguably, the doubters turned out to be correct. An early version of the Rockefeller Differential Analyzer was already in service by early 1943, and it was said to be one of America's secret weapons for the war, such was its effectiveness. The Mark I proved less impactful during the war, but we covered this remarkable IBM-backed computer in more detail in the previous instalment (see **rpimag.co/161**).

Despite the reluctance in some quarters, Gillon successfully fought for the electronic computer project, securing a budget of $61,700 for six months. This initial sum would cover "research and development of an electronic numerical integrator and computer and delivery of a report thereon," later wrote Martin H Weik. It was this document, and Gillon, that gave the ENIAC its name.

The agreement for 'Project PX' was signed on 5 June 1943, eight weeks after the official pitch meeting with Colonel Gillon, but this was a formality. A week earlier, Goldstine, together with Professor Brainerd, had formed a team. At its head stood Eckert, the chief engineer, with Mauchly the principal consultant (he would continue in his teaching role).

They were initially joined by a handful of others, including Arthur Burks, Thomas Kite Sharpless, and Jack Davies. Under Eckert's direct and demanding supervision, each was given a section of ENIAC to design.

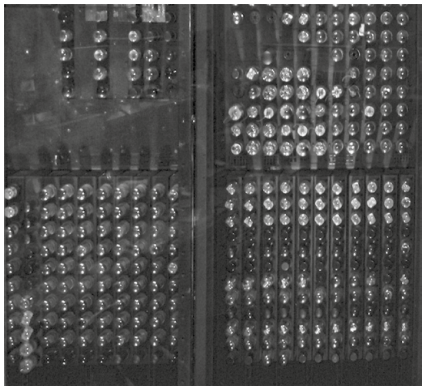
7 Interview with Herman Goldstine by Nancy Stern on 14 March 1977, Niels Bohr Library & Archives, American Institute of Physics, College Park, MD USA, **rpimag.co/goldstineinterview**

8 As above

9 Herman H Goldstine, *The Computer: from Pascal to von Neumann*, p149

At this point, the ENIAC was a concept rather than a blueprint. The team didn't even know what it would look like. But they had already settled on the three key areas: one for mathematical operations, one for storing data, one for programming.

Now, the small team needed to turn theory into practice. That meant devising and testing designs (Burks said the team initially worked on "basic counting circuits and switching circuits – building different counters and testing them"[10]), and finding out existing resources they could draw upon and what they needed to invent themselves. For example, Goldstine describes visits to the RCA Research Laboratories in Princeton, New Jersey, where they saw a function table ("a way of storing a table of numbers in the form of an electrical network of resistances"[11]) that would eventually become a key part of the ENIAC.



▲ Detail of the rear of a section of ENIAC, showing vacuum tubes
Image: Paul W Shaffer, CC BY-SA 3.0

Mauchly and Eckert had long been aware of 'flip-flop' circuits, where a pair of vacuum tubes worked as an electronic on-off switch to represent zero or one. This idea dated back to 1918, when two British physicists – William Henry Eccles and Frank Wilfred Jordan – filed a patent for what would become the Eccles-Jordan trigger circuits.

The bigger issue was reliability. Vacuum tubes were already commonplace in radios, but they were prone to failure because they were constantly heated and cooled, much like domestic light bulbs. Even at this early stage, Eckert and Mauchly knew their computer would require thousands of vacuum tubes, and the ENIAC would be rendered useless if tubes had to be constantly replaced.

Indeed, one of Burks's first jobs was to see if the ENIAC could use thyratron tubes – filled with gas – rather than vacuum tubes, to see if they would be reliable at high speeds. While he eventually succeeded at making the circuits work at 100,000 pulses per second, the ambitious target set by Eckert, they simply weren't reliable enough[12] and Burks's work was abandoned. Another reason the team settled on vacuum tubes was they discovered they "would last a lot longer if we kept them below their proper voltage – not too high or too low," according to Eckert in 1989.[13]

Eckert credits the fact that they started slowly as one of the key reasons for the ENIAC's success, with the team laying the foundations in steps rather than rushing to build anything approaching a full system. They only started building the first two accumulators several months into the project, having secured a second round of funding. Eventually, ENIAC would include 20 such accumulators working in tandem, bringing to life one of Mauchly's core visions: the power of 20 desk calculators working in parallel.

Desk calculators, explained Goldstine, "contained [mechanical] counter wheels which could be turned one stage at a time by the reception of an electric signal. Electronic ring counters are analogous to these wheels." They were called ring counters because the final output fed into the first input of the next digit, akin to carrying over a number in mental arithmetic.

When a new number was fed into an accumulator – via a train of pulses – it was then added to (or subtracted from) the existing number. Each accumulator included ten ten-digit counters – called decade counters – so could hold a number up to 10^{10}, aka 10 billion. To be precise, 9,999,999,999. Along with addition and subtraction, the ENIAC included dedicated units for high-speed multiplication, division, and square rooting.

The ENIAC could compute an addition or subtraction in 200 microseconds, so 1/5000th of a second. Division and square-rooting were more complicated, taking around 3/100ths of a second, while multiplication took up to three milliseconds. By contrast, a desk calculator could complete a multiplication in ten seconds at best. Even the Mark I took around three seconds for a ten-digit complication.[14] In comparison to other methods of the time, the ENIAC was a mathematical beast.

Accumulator value

The team completed the first two accumulators in mid-1944, at which point they felt confident that not only would they complete the project but they were on the verge of something special. "It soon emerged," wrote Goldstine, "that the machine

10 Tom Infield, 'Faster than a speeding bullet', in The Philadelphia Inquirer, 4 February 1996, p26

11 Herman H Goldstine, *The Computer: from Pascal to von Neumann*, p163

12 Interview with Alice R Burks and Arthur W Burks by Nancy Stern on 20 June 1980, Charles Babbage Institute. Retrieved from the University of Minnesota Digital Conservancy, **rpimag.co/burksinterview**

13 'From ENIAC to Everyone: Talking with J Presper Eckert', Alexander Randall, 23 February 2006, **rpimag.co/fromeniac**

14 Herman H Goldstine, *The Computer: from Pascal to von Neumann*, p137

would be much more useful than just a device for solving the differential equations of exterior ballistics. It gradually became clearer that the great advantage of the digital approach was that the ENIAC was going to be a truly general-purpose device."[15]

Even with just a pair of accumulators up and running, they could perform arithmetical problems and solve basic differential equations (the key to calculating firing tables). Meanwhile Jeffrey Chuan Chu, another engineering graduate drafted into the growing ENIAC team from the Moore School, set to work on the divider and square-rooter.

However, the team still needed help for what we would now call input/output, or I/O. And that meant working with IBM to devise a way to integrate its industry-standard punch cards. Colonel Gillon visited IBM's chairman Thomas Watson Sr in February 1944, who helped to arrange a meeting between his engineers and those at the Moore School that eventually led to a solution – the constant transmitter.

"The constant transmitter with its associated card reader reads from punched cards, numbers that are changed in the course of a computation and makes these numbers available to the computer as needed," explained Adele Goldstine in her comprehensive 'Report on the ENIAC'[16] for the US Army's Ordnance Department, printed in 1946.

One apparent backwards step compared to the binary ABC was for ENIAC to use base ten (decimal), but Eckert was quick to explain why in a 1977 interview.[17] "There were many sub-elements of the original ENIAC that are binary in nature and then could be converted to decimal before they came out of the machine. It was a conscious decision forced on us by the fact that the IBM card punch and printers and things that we hooked to were decimal. It had nothing to do with anything but that."

If you can detect a certain spikiness in Eckert's reply, your instincts are correct. Moments earlier in that interview he said: "Some people seem to think that… we didn't learn about the binary system. I assure you that I was familiar with the binary system when I was 14 years old."

While Mauchly came up with many of the ideas, and was a key decision-maker throughout the project, Eckert oversaw day-to-day operations for good reason. He was a stickler for getting things right, overseeing the project with exceptional levels of detail.

"I took every engineer's work and checked every calculation of every resistor in the machine to make sure that it was done correctly," he told historian Nancy Stern.[18] "Normally, I wouldn't want to have to do that. But this was the first time for a machine with in the order of 100 times as many tubes as anybody has ever built electronically. And if it was going to work, one had to be 100 times more careful."

No detail was too small. In an interview with Alexander Randall,[19] Eckert describes how they protected themselves from a basement's worst enemy: mice. "We knew mice would eat the insulation off the wires, so we got samples of all the wires that were available and put them in a cage with a bunch of mice to see which insulation they did not like. We only used wire that passed the mouse test."

Carl Chambers, a research director at the University of Pennsylvania, paints a vivid picture of Eckert as supervisor.[20] "There wasn't a single one of the staff who was doing a breadboard setup that he didn't tell him where to solder the joint," he said. "And he'd come in the next morning and tell them to tear

▲ Two pieces of ENIAC on display in the Moore School of Engineering and Applied Science
Image: Paul W Shaffer, CC BY-SA 3.0

15 As before, p163

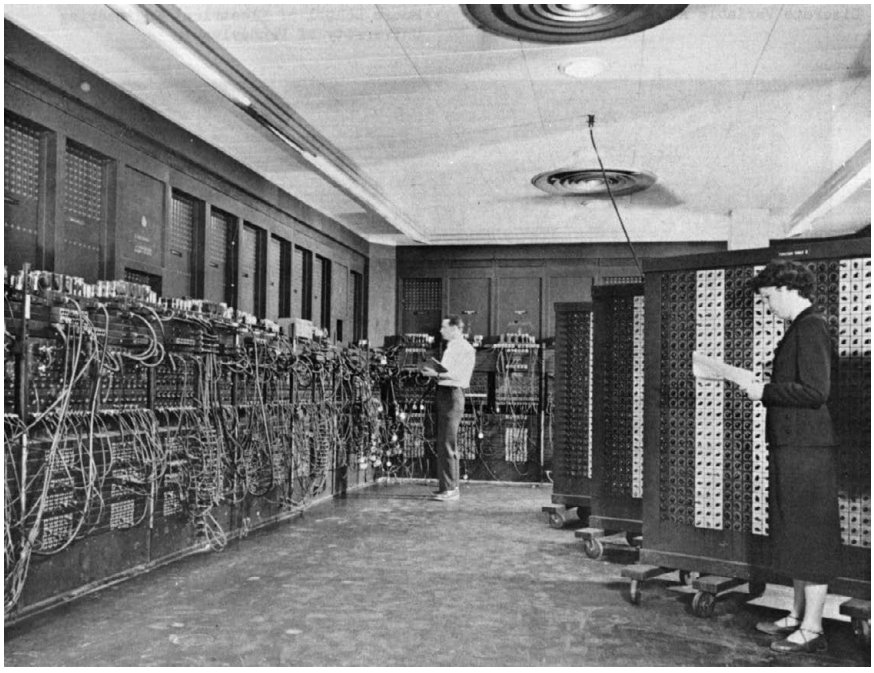
16 Adele K Goldstine, 'A report on the ENIAC', 1946, pp1-12. Retrieved from Smithsonian Libraries, **rpimag.co/eniacreport**

17 Interview with John Presper Eckert by Nancy Stern on 28 October 1977, Charles Babbage Institute. Retrieved from the University of Minnesota Digital Conservancy, **rpimag.co/eckertinterview**

18 Interview with John Mauchly by Nancy Stern on 6 May 1977, Niels Bohr Library & Archives, American Institute of Physics, Maryland, **rpimag.co/mauchlyinterview**

19 Alexander Randall, 'From ENIAC to Everyone: Talking with J Presper Eckert', 23 February 2006, **rpimag.co/fromeniac**

20 Interview with Carl Chambers by Nancy Stern on 30 November 1977, Charles Babbage Institute, University of Minnesota, Minneapolis. Retrieved from the University of Minnesota Digital Conservancy, **rpimag.co/chambershistory**



▶ Glenn Beck and Betty Snyder program the ENIAC in building 328 at the Ballistic Research Laboratory
Image: US Army photo, c. 1947–1955, Public Domain

everything up and change it because it was going to be revised as a result of his overnight idea for improvement."

This might normally cause friction within a team, but Chambers claims that "Eckert's knowledge of circuits and so on was superior. They were glad to get the advice, glad to get the ideas, glad to learn from him."

Despite the workload, group spirit remained strong throughout the project. "Everybody felt this [building the ENIAC] was terribly important to the war and that we get it done," said Arthur Burks,[21] one of the first members on the team. "So nobody hesitated to work all day and then all evening or come in on the weekends if necessary. Typically we worked six days a week and didn't come in on Sundays, I recall, until things later became very busy and then we would come in on Sunday."

Unfortunately, the team's hopes that they would finish work at some point in 1944 proved impossible. And like so many information technology projects, as time wore on so costs increased. The initial estimate was $150,000, but the final cost came in at $487,000.

"Part of this growth was due to expansions of requirements," explained Mauchly at the Los Alamos International Research Conference in 1976.[22] "[As] others saw what we were doing and how we were going about it, they said, 'We don't want just one of these function tables to put in drag functions, maybe for that really flexible, general-purpose device we ought to have three.' So they wanted more accumulators, too, more ten-digit storage devices."

The growth Mauchly refers to is literal too. The plan had always been for the ENIAC to be located in the Aberdeen Proving Ground, and Goldstine first asked for a room "about 20 feet by 40 feet to house the machine." This proved too small, with the computer eventually consuming the 30-by-50 foot (139m²) basement of the Moore School. The BRL ended up building a dedicated annex to house the world's first computer.

All this work happened under the military cloak of secrecy, with even the female computers left to wonder what was being built. The only clues, explained Kay Mauchly Antonelli,[23] was a trail of workmen carrying metal panes into a room. Until one evening, when she was working a night shift on the analyser. "Eckert and Mauchly came down to the analyser room and said, 'Would you like to see what we've been working on?' They said they had achieved a milestone."

The milestone was hooking up the two accumulators so they could communicate, with the men demonstrating this with a simple calculation of five times one thousand. "They were very elated," said Antonelli. She, however, was rather less impressed. "It didn't mean anything to me. I had no concept of how this would fit into the running of a trajectory."

By June 1945, it was time to start training the programmers. That is, the people who would interpret the equations and plug in the appropriate cables. The team started small: Kay McNulty (as she was then called; she would marry John Mauchly in 1948), Betty Holberton (née Snyder), Marilyn Meltzer (née Wescoff), Ruth Teitelbaum (née Lichterman), Frances Spence (née Bilas) and Jean Bartik (née Jennings). All six were sent to Aberdeen for a two-month training session.

It was an inauspicious start. "They just gave us these block diagrams of the ENIAC and told us to study them, learn how it works."[24] The women weren't allowed to see the ENIAC itself as their security clearance was too low. Bartik and Betty Holberton

21 Interview with Alice R Burks and Arthur W Burks by Nancy Stern on 20 June 1980, Charles Babbage Institute. Retrieved from the University of Minnesota Digital Conservancy, **rpimag.co/burksinterview**

22 Video recording of 'The ENIAC by John W Mauchly', The International Research Conference in the History of Computing at Los Alamos, 1976, **youtu.be/OUsc5JnyBYU**

23 Tom Infield, 'Faster than a speeding bullet', in The Philadelphia Inquirer, 4 February 1996, p26

24 Interview with Jean Bartik by Gardner Hendrie on 1 July 2008, Computer History Museum, p20. Retrieved from **rpimag.co/bartikinterview**

picked one of the empty classrooms and "just sat there with this block diagram and we didn't even know how to read it."

Fortunately, a man soon walked in and introduced himself as John Mauchly. "So we almost fell off our chairs," said Bartik, as both women were well aware of who he was. "And we said, 'Boy, are we glad to see you?' You know, 'tell us how this blasted accumulator works'. Well, anyway, he was a marvellous teacher, absolutely a wonderful teacher."

He needed to be, because programming the ENIAC was an incredibly complex job. Especially when it became clear that the women would need to not only understand the maths – something they were more than qualified for as college graduates in the topic – but also translate that into settings and then the physical work of plugging everything in. This was an arduous, complex task that could take hours.

And time was ticking. Despite everyone's hard work, the ENIAC project stretched beyond World War II, with Victory over Japan Day marking its official end for the USA[25] on 2 September 1945. So rather than calculating ballistics trajectories, ENIAC's first proper test started a month later with a problem set by scientists from Los Alamos, which was now working on the hydrogen bomb. "Our problem was with one dimension in space and one dimension in time, of course, to study some of the thermonuclear possibilities," said Nicholas Metropolis,[26] an experimental physicist who played a pivotal role in the Manhattan Project. "These were the first realistic, semi-realistic tests."

> *It should have been a moment of triumph for the women programmers, too, but for Bartik it was bitter-sweet*

Famously, John von Neumann was also part of the Manhattan Project and had, by chance, become familiar with the ENIAC after Goldstine had spotted him at Aberdeen train station. While von Neumann played no notable part in the ENIAC's development, he had met both Mauchly and Eckert, was familiar with the computer's workings and would play a big part in the creation of the EDVAC. He and Edward Teller, the so-called 'father of the hydrogen bomb', suggested that they run a model on the ENIAC, although the actual setup fell to Metropolis and computer scientist Stanley Frankel.

The pair had learned enough about ENIAC to program the problem themselves, but the physical plugging in of cables and switch setting was done by the team of programmers. And while the calculations never took long due to ENIAC's power, the entire process took weeks, with Metropolis and Frankel starting work in late autumn 1945 and not leaving until January the following year. In April, the results were reviewed by a "blue-ribbon panel of current and former Los Alamos scientists,"[27] producing a final report that said a hydrogen bomb "would probably work."

It's testament to everyone behind the ENIAC project that a computer commissioned to create firing tables could tackle such a radically different problem. Right from the start, though, the team were thinking long-term, with many of their implementations echoing down to modern computing. "[Mauchly] pointed out that these problems we were doing were highly repetitious and that obviously you couldn't store the program by cables or switches," said Eckert. "However, we planned to do [solve the problems] over and over again. But what you should do is store the program once and then call on it and use it over."

Programmers will recognise these repeated tasks as subroutines, with Eckert giving Mauchly the credit for inventing them – if not the term. "The ENIAC was nicely equipped with the ability to handle subroutines," said Mauchly at the Los Alamos conference, quoting a paper written by Eckert (who refused to turn up as he was to be introduced by John Grist Brainerd, a man he had come to detest). "I think we should be particularly proud that we did have this flexible programming and we did not hamper the people who wanted to put programs on."

Jean Bartik believes some of the credit for subroutines should go to Kay McNulty. The two were "trying to figure out how the ENIAC could do a trajectory" when they realised there wasn't enough hardware. "And finally I remember one day Kay said, 'Oh, I know, I know, I know. We can use a master programmer to repeat code,' and we did. We began to think about, you know, how we could have subroutines, and nested subroutines, and all that stuff."

Making a debut

In February 1946, the US Army was ready to show the ENIAC – until then a confidential project – to the world. "I was in charge of the demonstration," said Arthur Burks in a 1974 talk.[28] "Seems a

25 In the UK, Victory over Japan (VJ) Day is commemorated on 15 August.

26 Interview with Nicholas Metropolis by William Aspray on 29 May 1987, Charles Babbage Institute. Retrieved from the University of Minnesota Digital Conservancy, **rpimag.co/metrohistory**

27 Steve Leibson, 'Stanley P Frankel, Unrecognized Genius', **rpimag.co/frankelgenius**

28 Arthur Burks, 'Who Invented the General-Purpose Electronic Computer?', 2 April 1974, talk given at The University of Michigan, **rpimag.co/burkstalk**

bit silly, but I told the press, 'I am now going to add 5000 numbers together' and pushed the button. The ENIAC added 5000 numbers together in one second. The problem was finished before most of the reporters looked up!"

The problem chosen for a demonstration two weeks later, to a roomful of military VIPs, was rather more challenging: proof that the ENIAC could indeed calculate a trajectory of a missile. When Goldstine asked Betty Holberton and Jean Bartik at the start of February if the calculation was ready, so that it could be used in the demonstration, Bartik says they replied "You bet."[29] Except it wasn't. "So we went back and began working like mad to put it on and... get it up and running."

Come the 14th of February, one day before the official unveiling to that high-ranking military audience, the demonstration worked – except the virtual missile didn't stop when it hit the ground. It just kept on going deeper and deeper. Fortunately, overnight Holberton – and what Bartik describes as her "nighttime logic" – worked out that a single switch was set wrongly and needed to be flipped. This corrected, the demo worked perfectly.

The ENIAC was a general-purpose computer. In other words, it could be programmed

Unlike the earlier press demonstration, there was an element of theatre this time too, as Mauchly and Eckert had painted numbers onto the bulbs. "[The] people that were sitting there could see the numbers build up as the shell reached altitude and then came down and hit the ground," explained Bartik. And it was a triumph.

It should have been a moment of triumph for the women programmers, too, but for Bartik it was bitter-sweet. While everyone else went out for dinner – Eckert, Mauchly, Burks, dignitaries from the University of Pennsylvania and BRL, plus all the attending colonels and generals – neither Bartik nor Holberton were invited. "We were sort of horrified," said Bartik. "We knew how important it was and felt we had done so much but we didn't [get invited] – and of course in history afterward nobody ever mentions us."

Just to make the pill more sour, Goldstine claimed that the "main calculation and the interrelationship between the various problems were prepared solely by my wife and me"[30] in his book *The Computer from Pascal to von Neumann*, first published in 1972. Even when interviewed 36 years later, this still riled Bartik. "Well, he lied. I mean, he never even mentioned in his book the ENIAC women; all he ever mentioned was our names and who we married, and he called me Elizabeth. Well, my name was never Elizabeth."

To the watching public, however, the launch of the ENIAC was an unadulterated success. And one that would set the tone of media coverage for decades to come. In her 1995 article for the IEEE Technology and Society Magazine, Dr C Dianne Martin studied 43 newspaper articles. "In bold headlines seen around the world, metaphorical images such as electronic brain, magic brain, wonder brain, wizard, and man-made robot brain were used to describe the new calculating machine to an awestruck public," she wrote.[31] This, she argued, set the misleading anthropomorphic tone that still lingers for computers.

One such article appeared in The Des Moines Register.[32] Although it led with the curious headline, "ENIAC All Set to Make Mathematics Lost Art", it went on to describe the computer as "the supersonic quiz kid. It's the poor man's Einstein... ENIAC is the army's new robot brain." Others managed to squeeze all the clichés into their headlines alone, including The Fresno Bee with "Army's Robot Einstein Solves Years' Problems In Hours".[33]

Aside from Einstein references, the tone of most articles was awe. Awe at the ability of this electronic machine, awe at the fact the United States created it secretly in such a short time, and awe at the potential good it could do. Nor did the ENIAC's creators aim to dampen expectations. In one article, Mauchly is quoted as saying that high-speed computing could lead to all sorts of future possibilities, including "better transportation, better clothing, better food processing, better television, radio and other communications, better housing, and better weather forecasting."[34]

The latter was one of his great loves, and the passion that drew him into the idea of electronic computers in the first place.

29 Interview with Jean Bartik by Gardner Hendrie on 1 July 2008, Computer History Museum, p28. Retrieved from **rpimag.co/bartikinterview**

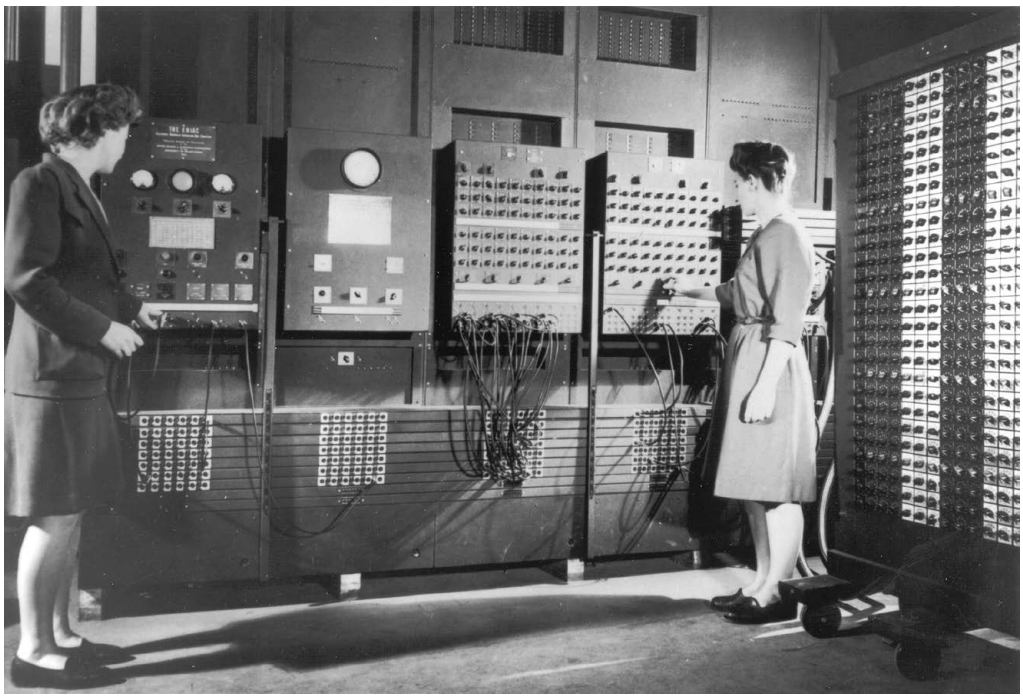
30 Herman H Goldstine, *The Computer: from Pascal to von Neumann*, p230

31 Dr C Dianne Martin, Department of Computer Science, The George Washington University, 'ENIAC: The Press Conference That Shook the World', in IEEE Technology and Society Magazine, December, 1995 **rpimag.co/eniacconf**

32 Jack Wilson, 'ENIAC All Set to Make Mathematics Lost Art', in The Des Moines Register, 15 February 1946, p1

33 Associated Press, 'Army's Robot Einstein Solves Years' Problems In Hours', in The Fresno Bee, 15 February 1946, p5

34 Thomas J O'Donnell, 'Mechanical Mathematician "Brain Child" of Hopkins Man', in The Baltimore Sun, 15 February 1946, p28



▲ Programmers Jean Bartik (left) and Frances Spence operating ENIAC's main control panel
Image: US Army photo, c. 1945, Public Domain

So Mauchly must have been gratified to see that, alongside cosmic-ray studies, wind-tunnel design and many other scientific applications, the ENIAC was indeed used to predict the weather during its ten years of active duty.

By the time it was switched off – at 11.45pm on 2 October 1955 – the machine had also received several useful upgrades. For a start, it had become possible to store programs for "standard trajectory problems",[35] while each function table "became available for the storage of 600 two-decimal digit instructions." Both these changes resulted in major time savings, helping to keep the ENIAC relevant in the face of new electronic computers.

Such improvements happened after the Moore School had released the ENIAC to Aberdeen, and after Eckert and Mauchly had left; they had little choice, as a condition for staying on at the university was that they would lose all rights to the ENIAC's patents. Patents that would eventually lead to a lawsuit that would mean Eckert and Mauchly were, in the eyes of the law, no longer the inventors of the world's first electronic computer: that privilege, the judge decided, fell to Dr Atanasoff for the ABC.

This may be surprising. It should be obvious that the ENIAC represents a huge leap over the ABC, to the extent that Dr Atanasoff didn't recognise any trace of his desk-sized computer on his first encounter with the machine in the late 1940s. Indeed, he claimed in a 1972 interview[36] that when he saw the ENIAC for the first time, "I looked at the ENIAC and said, I'm not interested in having any connection with it… Because it wasn't a very effective machine and I didn't like its end results and I didn't like the insufficiency and I didn't like this and that about it."

It was only in the late 1960s, he states in the same interview, that he read the ENIAC patent and "realised that Mauchly had taken ideas which I had devised and used them in the construction of the ENIAC patents." We covered this in more detail in part one, the story of the ABC (see **rpimag.co/157**).

However, the fact that a legal battle gave so much credit to Atanasoff should not diminish the work done by Mauchly, Eckert, and everyone else involved with the ENIAC. Atanasoff and Clifford Berry never took their ideas beyond a semi-working model that could solve a single type of problem; to create the ENIAC, a computer that continued to put in solid service for a decade, was a truly incredible achievement.

And let's follow in the ENIAC's footsteps and talk numbers. Mauchly and Eckert's machine used almost 18,000 vacuum tubes compared to 300 for the ABC. It used 20 accumulators rather than a pair of magnetic drums. And, although it is difficult to draw a direct comparison, the ABC could perform one complete operation every 15 seconds while the ENIAC could finish 5000 arithmetic instructions per second.

The key difference between the two machines, however, is that the ENIAC was a general-purpose computer. In other words, it could be programmed, unlike a special-purpose computer such as the ABC that could only solve one type of problem. And it had a much bigger legacy.

We're not simply referring to the computer that came afterwards, the EDVAC. By building the ENIAC, the Moore School essentially won the war over analogue devices such as the differential analyser. It proved that digital computing was the way forward, that it could solve problems that were previously unsolvable, and in one swoop moved us into a new, digital era.

35 Martin H Weik, 'The ENIAC Story', in ORDNANCE, Vol 45, No 244, January–February 1961, **rpimag.co/eniacstory**, p575

36 Interview with John V Atanasoff and Alice Atanasoff by Bonnie Kaplan on 17 July 1972, Smithsonian National Museum of American History, **rpimag.co/atanasoffinterview**

STARGAZING

with Raspberry Pi

Build a Raspberry Pi night-sky setup and reach for the stars from your backyard.

By Rosie Hattersley

A Raspberry Pi camera rig can greatly enhance your experience of stargazing

The darker, colder months lend themselves to spending time on projects indoors, but also provide excellent conditions for enjoying looking up at the night sky. Astronomy and photography are both more affordable and accessible than ever, with powerful camera lenses and star-tracking features ensuring this increasingly popular pastime can result in some stunning shots. Deep pockets are needed if you want to set up a professional-grade camera rig with automated sky-tracking capabilities and take photos of deep space.

Arguably, travelling to dedicated dark skies across the globe then becomes a significant expense too. You can also spend quite a lot of hours waiting for the right conditions, and then sitting around while your camera rig takes gazillions of photos, or a few really long-exposure ones.

However, one of the best things about Raspberry Pi is that it has always been a democratising piece of kit: the cost of a Raspberry Pi plus a bolt-on Raspberry Pi HQ Camera and protective case form the basis of an astrophotographer's kit. To this you'll probably want to add a tripod, telescope, and battery pack so you can shoot the stars from a suitable location away from light and other forms of pollution. You may also need an RTC (real-time clock) and GPS (Global Positioning System) module to identify objects in the sky with accurate time and location.

Armed with such a setup, you will potentially be able to capture photos featuring meteor showers, shooting stars, constellations, phases of the moon and, depending on your location, aurora borealis and aurora australis (northern and southern lights). Most of these can also be spotted with the naked eye, but a Raspberry Pi camera rig can greatly enhance your experience of stargazing and the range of what you can see. You might even be able to take a closer look at Comet 3I/ATLAS (**rpimag.co/cometatlas**) and decide for yourself whether it's an alien craft (**rpimag.co/bbccometatlas**).

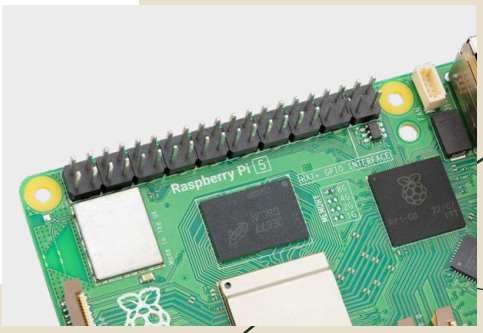
KIT TO CONSIDER

Get the right equipment for your stellar adventure

Raspberry Pi 5

£91/$95 | rpimag.co/raspberrypi5

Now available with up to 16GB RAM, the most powerful Raspberry Pi to date is a compelling option for storing and processing images you've captured of the night sky. It boasts a 2.7GHz quad-core Arm Cortex A76 processor, USB 3.0, dedicated Camera and DSI (Display Serial Interface) ports, while the inclusion of a PCIe 3.0 interface means you can make use of M.2 SSD storage for super-fast read and write speeds as well as nippy boot times. The ideal partner for Raspberry Pi HQ Camera, we recommend getting an 8GB version for astrophotography use (16GB is probably overkill). Pop yours inside either an official or a third-party case to protect it from the elements while you focus on observing majestic skies.



▼ Raspberry Pi 5 8GB is our pick of board for this project

▼ Raspberry Pi High Quality Camera

Raspberry Pi HQ Camera

£48/$55 | rpimag.co/hqcamera / rpimag.co/hqcameram12

The incredible-value Raspberry Pi HQ Camera is a 12MP lens with a C/CS-mount (it's made for Raspberry Pi by Sony Imaging). A version with an M12 connector is also available which accepts most M12 lenses. Unlike the stock Raspberry Pi Camera Module, you will need to attach a lens to HQ Camera. This makes it ideal for attaching directly to a telescope with either a C/CS or M12 mount. Attach HQ Camera to your Raspberry Pi 4 or Raspberry Pi 5 via the camera ribbon cable and then mount it to your chosen telescope or lens, which will magnify whatever is in the frame and augment the amount of light. You then capture images or video from Raspberry Pi OS and automate the capture process.

Lenses

Naturally enough, you need to choose a lens depending on what you'd like to photograph. For all-sky photography, you will need either the 6mm wide-angle lens for Raspberry Pi HQ Camera, or a fisheye lens that is compatible with its C/CS mount.

If you are looking to take photos of meteors and other fast-moving objects that you might be able to spot with a naked eye, you'll want something that captures the whole sky, so a fisheye or wide-angle lens. Capturing photos of the moon or other planets and galaxies involves increased magnification, so telephoto is the way to go.

The Pi Hut has a range of lenses compatible with HQ Camera to peruse on its website: **rpimag.co/lenses**.

▶ Raspberry Pi HQ Camera telephoto lens

◀ Arducam M12 180-degree Fisheye Lens

Arducam M12 180-degree Fisheye Lens

£24 / $32 | rpimag.co/fisheyelens

Arducam's Fisheye Lens is a popular and less expensive choice for capturing the whole sky at once. Because it has an M12 mount rather than the C/CS mount found on the standard version of Raspberry Pi HQ Camera, you will either need an M12 adapter ring or the M12 HQ Camera model.

Telescopes

Almost any Newtonian telescope can be used with Raspberry Pi HQ Camera, which attaches to it with the C-mount as the eyepiece (via a C-mount to nosepiece adapter). Not all motorised telescopes are Raspberry Pi 5 compatible. Notably, the motorised Celestron SCT focuser has a known issue that the manufacturer states is unlikely to be addressed, so a different brand such as ZWO (which also sells kit based around Raspberry Pi 4) could be a better option. GoTo telescopes are ones that can automatically track objects across the equatorial horizon, but many people prefer to control what they're seeing manually. Less high-tech Dobsonian telescopes provide a larger reflecting mirror to draw in more light than standard Newtonian scopes.

Our pick for starting out is the Celestron Royal Observatory StarSense Explorer DX 100 Refractor Telescope (£299) from the Royal Observatory in Greenwich. It has a large refractor which helps it capture faint detail. It is easy to adapt with a C-mount adapter for the nosepiece. Buy direct from the Royal Observatory Greenwich (**rpimag.co/rmgtelescope**). With up to 66× magnification, you should be able to pick out Jupiter, Saturn, Venus, and the Orion Nebula.

Picking a starter telescope is very much a personal choice, though. With a C-mount to nosepiece adapter, you can use almost any telescope. First Light Optics has a great starter guide to picking a telescope (**rpimag.co/firstlightoptics**). Most starter models are made by Celestron, but you can also pick up more expensive ones from Sky-Watcher, Ursa Major, and other brands.

▼ This Celestron Royal Observatory StarSense Explorer DX 100 Refractor Telescope from the Royal Observatory is a good starter option

C-mount adapter

You'll want to connect your Raspberry Pi HQ Camera to a telescope and many entry-level telescopes come with a 1.25-inch eyepiece. What you need is a C-mount adaptor that connects your HQ Camera to the 'nosepiece' of your telescope (this is the cylindrical adaptor that slides into the focuser).

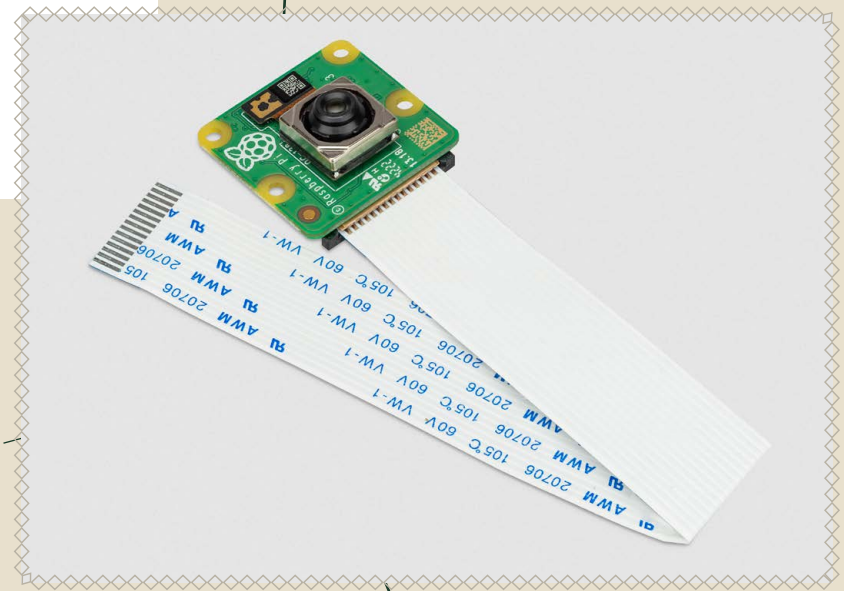
You can pick up a C-mount to 1.25-inch nosepiece adapter for around £15/$15 from most telescope stores. Here are a couple of them, from First Light Optics (**rpimag.co/astro-nosepiece**) and Modern Astronomy (**rpimag.co/astro-mount**).

▼ A C-mount adapter for a 1.25-inch nosepiece

Raspberry Pi Camera Module 3

£24 / $33 | rpimag.co/camera3

This 12MP camera comes with autofocus and a standard or wide-angle lens – with or without an infrared filter. Handy for faster setup and if you want to take photos of phenomena such as the northern lights or the night sky rather than attach to a telescope or lens.



▼ Raspberry Pi Camera Module 3

Dedicated controllers

If are thinking of using a digital SLR camera or automating your entire astrophotography setup, you can probably do with a guiding scope. If you are looking at this level of investment, consider advanced controller kits such as the AStarBox (**astarbox.co.uk**) and Astro Link 4 Pi (**astrojolo.com**). Although buying a pre-built kit takes a lot of the fun out of the setup in our opinion.

▼ An AStarBox controller kit with Raspberry Pi inside

Power bank

rpimag.co/powerbank

Unless you are going to be observing from the comfort of home, you will need to power your Raspberry Pi (and any extras such as a dew heater) using a portable battery pack. Take a spare power bank if possible (as well as plenty of warm, waterproof clothes) so you can maximise your time watching and photographing and do so in relative comfort.

▼ A 10,000mAh 20W PD power bank

Weatherproof casing

Taking your rig out and about, probably for many hours at a time, means you'll need a robust case that can withstand downpours, cold, wind, and storms. Sixfab specialises in industrial kit and equipment to place Raspberry Pi in harsh work environments. Take a look at the Sixfab IP65 Outdoor Project Enclosure (**rpimag.co/outdoorcase**).

◀ Sixfab's IP65 enclosure keeps the water out of your Raspberry Pi

▶ A custom Raspberry Pi HQ Camera tripod

Tripod

Raspberry Pi HQ Camera's C/CS mount attaches to the telescope – via an adapter to keep it in place – so the telescope effectively becomes the camera lens. Be warned, though, that aligning camera lens and telescope is not as precise as a setup in which the camera and telescope have been specifically designed for use together.

Once you have your telescope set up, you'll need something to mount it on and point it around. If you want a cheap option, head over to your favourite photography store and pick up a camera tripod for your telescope. Here's a selection from Jessops in the UK: **rpimag.co/jessopstripods**.

Professionals will want something a little sturdier and AstroShop.eu has a range of tripods and mounts for your attention: **rpimag.co/astromounts**.

If not using a telescope, get a camera tripod for HQ Camera. For instance, the Pi Hut sells small, medium, and large tripods with standard ¼-inch screw mounts that work with it (**rpimag.co/medtripod**).

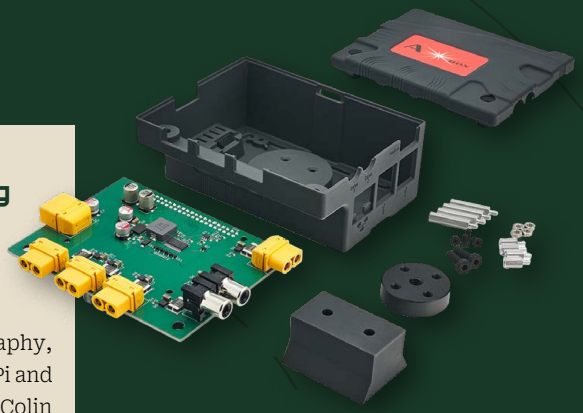
Ready to buy kits

AStarBox Open-Source Astroimaging Controller Kit

£285/$325 | astarbox.co.uk

Ideal for Raspberry Pi 5 astrophotography, the AStarBox powers both Raspberry Pi and photography equipment via a HAT. Dr Colin McGill, one of the two astrophotographers and cosmologists who designed and built the AStarBox in the UK, says Raspberry Pi 5's dedicated RP1 I/O chip makes all the difference since data is saved at 100Mbps. As a result, he is able to run his CMOS cameras at full bandwidth and capture deep sky images. The AStarBox controller kit can optionally be used with an NVMe SSD (**rpimag.co/ssd**) for even faster image transfers of up to 500 MB/s. This also means there's no need for a laptop to process images, making for a smoother experience without interruptions about updates. The lightweight control box includes a guider foot, is compatible with a range of telescope mounts, and can be used with Raspberry Pi OS, StellarMate, and AstroArch, and with well-known astronomy software such as Firecapture, AstroDMX, and TheSky.

The AStarBox kit is available from First Light Optics (**rpimag.co/flastarbox**) and CloudBreak Optics (**rpimag.co/cbastar**).



▲ The AStarBox sold in kit form on First Light Optics; just add Raspberry Pi

▼ AstroLink4 Pi is an all-in-one astrophotography controller

AstroLink 4 Pi

€200 | astrojolo.com

AstroLink 4 Pi is the brainchild of keen astrophotographer Lucas Socha. It can be powered by either Raspberry Pi 4 or 5 and is a 12V DC-powered all-in-one astrophotography controller that can measure and compensate for the sky temperature as well as the prevailing weather conditions. It also has a power management hub for digital SLR cameras, USB hubs, and other essential photography gear and has an integrated real-time clock to make it easier to identify what your telescope picks up. As is the case with many such products, it is designed to be part of a modular setup that can be adapted as the owner's interest and finances allow. As such, it is compatible with several brands of stepper motor in addition to AstroLink for configurations of guiding scope, mount, camera, and telescope. Usefully, it also has a dew heater and a temperature-based focus compensator to combat weather conditions. Lucas uses imaging telescopes such as the Samyang 135mm f/2 telephoto lens, but advises that either Newtonian or refractor telescopes work well with it too. AstroLink 4 Pi is available from the Astrojolo online shop: **rpimag.co/astrolink4pi**.

The PiFinder kit attached to a telescope

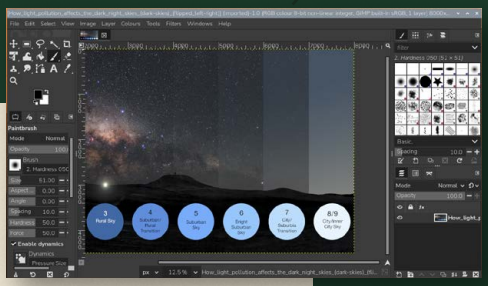
PiFinder

pifinder.io/about

When PiFinder maker Richard Sutherland took his self-built telescope mount and Raspberry Pi 4- and HQ Camera-based photography rig to a 'star party' in rural California in 2023, he was immediately lauded by the assembled astronomy ensemble. Such was the enthusiasm, the astrophotographers spent 32 hours observing the skies over three nights, taking 115,000 photos and locating 65 celestial objects with PiFinder between them. It is now available as a self-build kit at **pifinder.io/build-yours**.

You can read more about PiFinder on the Raspberry Pi blog: **rpimag.co/pifinderblog**.

Editing software



GNU IMP image-editing software looking at light pollution levels on the Bortle scale

Almost all published astronomy photographs consist of multiple images of the same scene and make use of an image-stacking process to optimise sharpness for all the celestial objects within it while reducing unwanted visual noise.

In essence, the stacking process involves opening all the images you want to include in the Layers panel of an image editor and blending them together by adjusting the transparency of all but the bottom layer, then adjusting the levels and curves to get a smooth effect.

The GNU Image Manipulation Program photo editor is an excellent free option for tweaking and combining the images you capture and is available on Raspberry Pi OS. Open Terminal and enter:

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

There's a very useful YouTube primer on image stacking with the GNU Image Manipulation Program here: **rpimag.co/gnuimagestacking**.

Smartphones, software, and apps

iPhone and Android smartphones can be useful companions for your Raspberry Pi photography. For starters, a smartphone can alert you to prevailing weather and celestial conditions and help you decide whether it's worth heading out to look at the stars. A smartphone can help pinpoint constellations, help orientate you as you peer up towards the stars, and let you know when there's a decent chance of spotting something worth photographing.

If you baulk at the thought of carrying around an Apple or Google-powered phone, FairPhone sells a fully de-Googlefied smartphone running /e/OS (**rpimag.co/fairphoneeos**) that is modular and repairable. You can also replace the OS with Ubuntu Touch.

Apps such as Aurora Watch and Aurora provide maps and alerts about solar storms and the chances of seeing the northern lights. Stellarium shows you which constellations can be seen depending on your location and orientation, so you can find them easily.

Better yet, a smartphone can control the software you've installed on your Raspberry Pi-based astrophotography. StellarMate OS (**stellarmate.com**) is an operating system that you install on Raspberry Pi. There's an accompanying Android app that works with multiple astrophotography programs and whatever hardware you're using. It can automatically detect your setup and connect to it. Helpfully, being open source, this works with the Astroberry OS and KStars software on Raspberry Pi too, so you don't even need to change to the StellarMate OS if you decide you like the StellarMate app.

StellarMate OS running on Raspberry Pi

Astro-photography projects

Stand on the shoulders of giants when building a Raspberry Pi-powered all-sky telescope. Others have gone before you

01. A cheap all-sky camera built with Raspberry Pi 3B+ and a Camera Module 2

02. An image captured using an all-sky camera

There are lots of all-sky camera builds that use Raspberry Pi, particularly since HQ Camera launched in 2020. They make use of a wide-angle lens to capture almost an entire hemisphere of sky at once and are typically set up to run overnight, tracking the sky and taking photos automatically. We like this inexpensive version: **rpimag.co/cheapallsky**. Also, check out the Allsky Camera GitHub repo to make one: **rpimag.co/allskygit**. You will need a clear plastic dome to house the camera and fisheye or wide-angle lens. Seal it with caulk to protect it from the elements. Place this on a sturdy rotating mount and add heat and moisture sensors.

Heavy Pan-Tilt System

As we've seen, a tracking mount can be invaluable if you want to follow objects across the skies, but it's not easy to create a setup robust enough to take a hefty digital SLR camera and have it pan smoothly without juddering or losing either focus or the object in the viewfinder. Maker Vito put together his own mechanical designs and software to automate object tracking, with Raspberry Pi 4 providing the critical stepper motor controls and OpenCV libraries for computer vision (**rpimag.co/heavypan**). The rig consists of sturdy aluminium frames, stepper motors with belts and pulleys, plus 3D printed parts to hold Raspberry Pi 4 and the camera on its rotating mount. See the incredible results on YouTube: **rpimag.co/heavypanyt**.

Create your own DIY rig

If you're looking to combine self-assembly with some 3D printing and coding, take a peek at Jaime Machuca's project pages where he shares the case he designed to work with a clear dome in order to capture time-lapses of the night sky: **rpimag.co/allskycase**.

Spot meteors and fireballs

Fireballs occur when a meteor enters the Earth's atmosphere and burns up. Raspberry Pi cameras are great at spotting them (as are Ring doorbells). The UK Fireballs Alliance and the Global Meteor Network consist of both professionals and citizen scientists who track meteors across the sky and are able to triangulate their positions with the result that duly located meteorites have been pinpointed and retrieved. For more info, visit **rpimag.co/fireballs**.

Stellarium

Stellarium (**stellarium.org**) is a desktop guide to the night sky, providing a 3D visual of what you might see using a telescope or binoculars, or even with the naked eye. The key difference is that this open-source planetarium comes with a guide and optional artwork to help you spot specific constellations and auroras, and even has a handy search function. Install the Linux version on Raspberry Pi or simply use it on a PC, Mac, or smartphone to get your celestial bearings. The web version even offers an option to record your observations: **rpimag.co/stellaweb**.

ISS tracker: want to get into space, but not set up all that kit?

The International Space Station orbits Earth 16 times a day and is travelling at 28,000km/h. You can see where it is at any given moment by heading to **rpimag.co/opennotify** and zooming in to the map. The very same page also tells you how many people (and which ones) are currently in space and aboard the ISS.

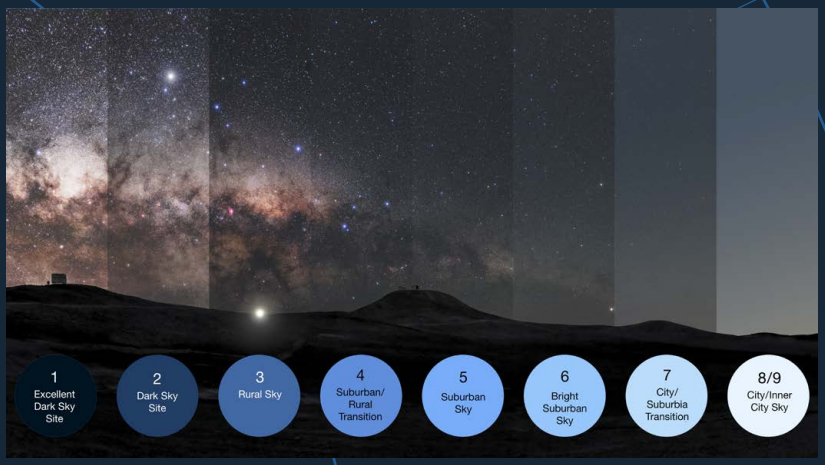
This project using Raspberry Pi and an e-paper display offers a fun way to track the ISS's movements: **rpimag.co/isstracker**.

Astroberry

astroberry.io

As you'll guess from its fruit-based title, Astroberry is specifically designed for Raspberry Pi: full-size models and Raspberry Pi Zero. This Linux-based desktop OS can be used to remotely access and control your stargazing kit (if you have set it up for motorised control, of course) and pan across the skies using a wireless hotspot, change the focus, and trigger snapshots. It works with all the major astronomy programs such as INDI KStars planetarium software and Stellarium. Adafruit's Motor HAT is an ideal focuser to add to the setup – see **rpimag.co/astromotorgit** for the software.

Tips and resources



▼ The Bortle dark-sky scale **Photo credit:** ESO/P. Horálek, M. Wallner, European Southern Observatory (ESO)

Long exposures and a steady tripod or mount – plus the patience and advance planning to be in the right place at a time when weather and night skies are favourable – go a long way towards making it more likely you will get a noteworthy photo.

Composition-wise, having something in the foreground such as a plane, distinctive building, outcrop, or a fortuitously timed ISS flyby really helps. To ensure a sharp overall image and a seemingly infinite depth of field, a succession of shots, automatic focus, and focus stacking followed by painstaking photo editing can also make a huge difference.

Stargazing terminology

- **All-sky camera**. A camera and lens that is able to capture a 180-degree view of the sky.
- **Autoguider**. Locks on to a specific star and follows it across the sky so that deep-sky photos can be taken. Consists of a camera, telescope, and guide scope, plus the mount which continually adjusts its position to ensure a fix.
- **Bortle scale**. Ranges from 1 to 9 and describes how good dark-sky conditions are for viewing and taking photos depending on light pollution and other factors. The lower the number, the better.
- **Close flyby**. A spacecraft or other object potentially gets a boost from flying close enough to be affected by a planet's gravity. A popular astrophotography composition as well as invaluable for data gathering.
- **Dew lamp/heater**. Powered by USB, these often come in the form of a headband-type strip that gently warms the telescope optics or camera lens, preventing dew or condensation forming and potentially ruining the photos you've patiently waited hours to shoot.
- **Drift**. Changes to the position of a tracked star moving across the sky. Successive exposures reveal this drift and the camera position is corrected accordingly.
- **DSO (deep-sky object)**. You will need a fixed-mount camera and telescope to capture photos from distant galaxies and nebulae.
- **Focus stacking**. Combining multiple photos with different focal points in order to create a single photograph with a greater depth of field. Software that automates this.
- **Guiding scope**. A tracking telescope used in more expensive astrophotography rigs that locks on to a star and corrects the movements of the camera and telescope mount that are taking photos.
- **Lens heater**. Prevents moisture building up on the camera and ruining your photos (and damaging the lens).
- **RTC (real-time clock)**. Accurately knowing the time is critical to locating and identifying specific objects in the sky. Raspberry Pi 5 has support built in, but requires a Raspberry Pi RTC Battery to be connected to the port on your board. See **rpimag.co/rtcbattery** for more information.
- **Transit**. An object such as an aeroplane or the ISS passing in front of the moon, sun, or other large object in near space.

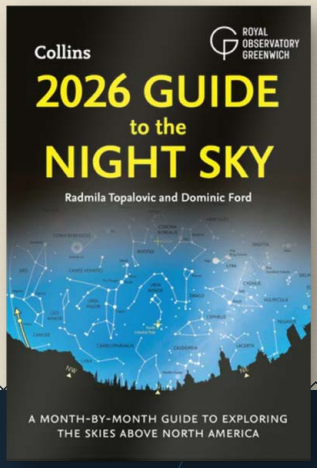
Credit: AstroLink images by Astrojolo (Łukasz (Lucas) Socha)

Collins 2026 Guide to the Night Sky

£7.99 | rpimag.co/nightsky2026
$6.99 | rpimag.co/nightsky2026na

This handy book offers a month-by-month visual guide to stars, meteors and constellations you can see in the skies above Britain and Ireland (or North America) in 2026, along with maps of dark sky sites and useful astronomy terms.

- For more tips, head to **cloudynights.com** or **stargazerslounge.com** forums, or a dedicated Raspberry Pi astrophotography Reddit page.
- AstroBackyard provides useful guides, including how to make the most of live stacking: **rpimag.co/livestacking**.



Credit: AStarBox images provided by Dr Colin McGill

Pull photos from NASA's orbiting EPIC camera

EXPLORE OTHER COLLECTIONS

There are plenty of astronomy photographs available online that have been published under a Creative Commons licence and are available for reuse and manipulation. It's a handy option for experimenting with working with multiple images to show off the celestial subject matter to best effect, perhaps if you can't access places to take night sky photos of your own. You can have plenty of fun browsing shots that others have created and manipulated, or even come up with intriguing imagery yourself by processing and combining such shots to present something new. The golden rule, of course, is to credit and attribute photos you've used.

Pull photos from NASA's own daily feed of images taken by the EPIC camera on its orbiting Deep Space Climate Observatory using this ingenious setup involving Raspberry Pi Zero and a HyperPixel Round Touch Display: **rpimag.co/epicsat**.

If you get a chance, we highly recommend a visit to both an observatory and a trip to the Astronomy Photography exhibition at The National Maritime Museum (part of the Greenwich Museums which includes the Observatory) which runs until 3 August 2026 (**rpimag.co/rmgastro**). Entries to the 2026 photography competition are open now, and there is even a free-to-enter newcomer category, should the astrophotography bug seize you.

Fusion HAT+

An upgraded robotics board with added voice control. By **Phil King**

SunFounder **rpimag.co/fusionhat** £26 / $35

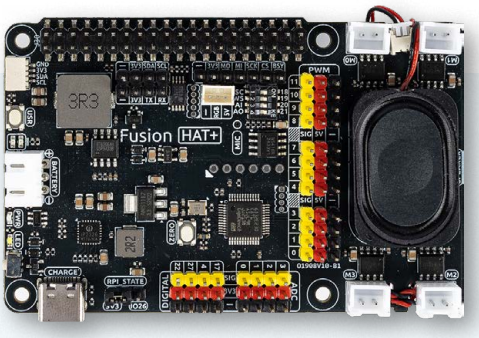
SPECS

DIMENSIONS:
88 × 55 × 11mm (28mm with header)

INPUTS/ OUTPUTS:
12 × PWM channels for servos, 4 × DC motor ports, 4 × digital inputs, 4 × analogue inputs, 7-pin SPI, 4-pin I2C, 4-pin UART, 2-pin WS2812 RGB LED

POWER
6V-8.4V 3-pin battery port, 2 × 18650 battery pack (2000mAh), 5V via USB-C charging port, power button, safe shutdown

There are great possibilities for voice control, with examples detailed in the online documentation



▲ The HAT features an onboard mic and speaker for voice control

▶ It's supplied with a battery pack and fixings to secure it to Raspberry Pi

SunFounder's brand new robotics board is an upgrade on the Robot HAT supplied in many of its robotics kits (such as the PiCar-X and PiDog), cramming in an impressive amount of functionality and some neat features.

For starters, you get no fewer than twelve (three-pin) PWM channels for controlling servos and the like. It works with most micro and standard hobbyist servos. A really handy addition is the 'ZERO' button to test connected servos by setting them to either the mid or zero position, depending on how many times you press it.

There are four motor ports with two-pin JST-XH 2.54mm connectors – four cables are supplied so you can wire up standard DC motors. It should be noted that the motor ports share eight of the PWM channels (4-11). Still, it offers plenty of options for robotics setups. There's also a WS2812 port for a single NeoPixel LED strip.

Then there are the inputs: four digital and four analogue (each with three pins for signal, power, and ground), the latter making use of an onboard ADC – both this and the PWM channels are handled by an Arm Cortex-M23-based microcontroller.

In addition, there are several breakout headers: a four-pin I2C port compatible with Qwiic/STEMMA Qt, seven-pin SPI, and four-pin UART. Not to mention a full 40-pin GPIO pass-through header, although naturally you'll need to watch out for pin conflicts.

Portable power

Another key feature is that you can power both the board and Raspberry Pi with the supplied twin 18650 battery pack, once you've charged it up via the USB-C port – with a capacity of 2000mAh, it takes about two hours. The circuitry features protection for reverse polarity, over/under-voltage input, and overheating.

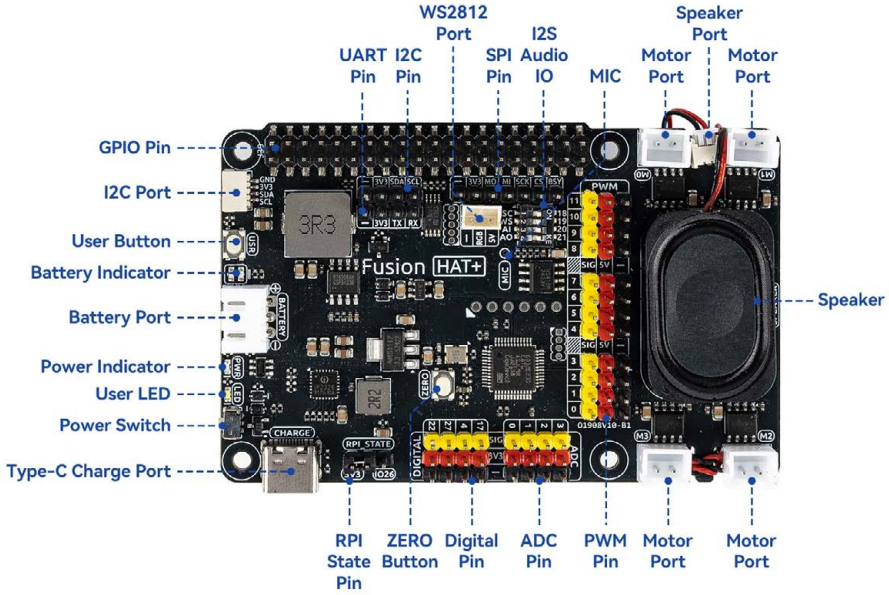
The PWM interface provides up to 5V/5A maximum output, shared with Raspberry Pi. DC motors are powered directly from the batteries (6–8.4V), with each motor channel supporting up to 1.8A maximum current.

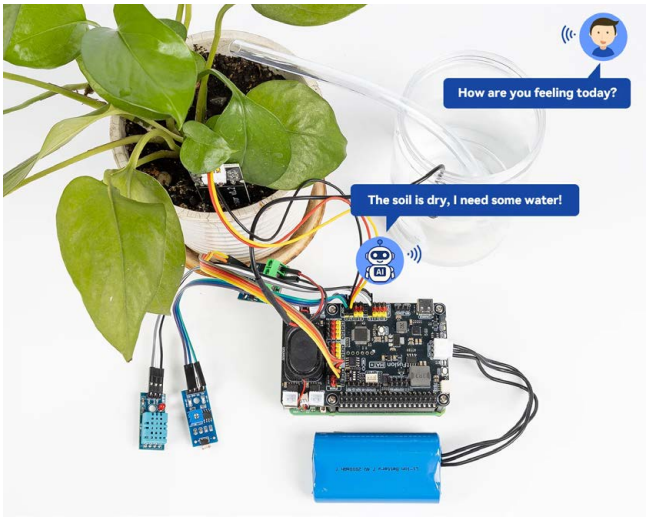
How long the batteries will last on a full charge depends on the load from connected servos and motors, but the power button delivers a safe shutdown, also initiated automatically when a low battery level is detected by the microcontroller.

Voice control

While the product marketing focuses heavily on the Fusion HAT+ being able to work with a range of LLMs (large language models), there's no accelerator chip on board. With a built-in mic and speaker (4Ω 2.5W, using the I2S interface), however, there are great possibilities for voice control, with examples detailed in the online documentation such as text-to-speech (offline and online), speech-to-text (offline), text vision talk with Ollama (run locally), connecting to online LLMs, a local voice chatbot, and an AI voice assistant.

The documentation (at **rpimag.co/fusionhatdoc**) is comprehensive, covering every aspect of the board's functionality and including usage examples for the numerous dedicated Python modules supplied in the fusion-hat package – installed with a single command (plus another for the audio setup). The package also includes submodules covering numerous external sensors and other connected components.





▲ A guide to the Fusion HAT+'s main functionality
◀ An example project with sensors connected (not supplied)

It all adds up to a very versatile board that offers a wide range of possibilities for your robotics projects. Great value for money too, especially considering the battery pack is included.

The Fusion HAT+ is also set to be included in SunFounder's upcoming AI Fusion Lab Kit. Due to launch in February, this is a modular hardware kit with a ready-to-use software environment and step-by-step learning modules. You can check out the documentation and projects at **rpimag.co/aifusionlabdoc**.

Verdict

A feature-packed robotics HAT+ with onboard mic and speaker to add voice control to projects.

9/10

Pitower Gen 1

A mini PC case that emulates a full desktop case, but still leaves room for Raspberry Pi-specific features.

By **Rob Zwetsloot**

Elecrow **rpimag.co/pitower** £37 / $50

▲ No modern PC case is complete without RGB fans inside

SPECS

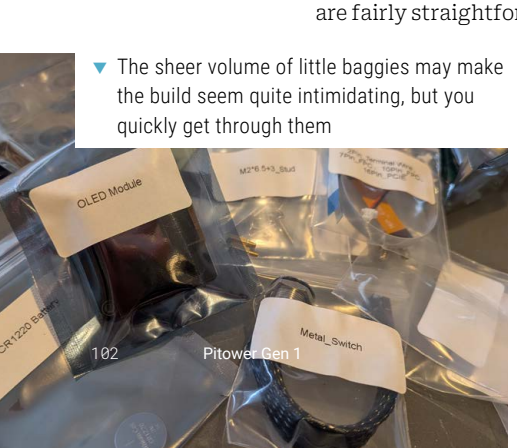
DIMENSIONS:
120 × 120 × 72mm

SCREEN:
1.3" OLED

STORAGE:
SD card and support for NVMe SSD, sold separately

It seems quite popular now for folks to make their Raspberry Pi look like a diminutive desktop PC. This new take from Elecrow manages to do that while also making sure you can still use your Raspberry Pi as, well, a Raspberry Pi.

Construction is fairly simple if lengthy – it took the better part of three *Red Dwarf* episodes to assemble it all physically. For anyone who's built a full-size PC in the last decade or so, the process will be quite familiar, and with the same level of fiddly bits and awkward screws to boot. All the parts are supplied in separate baggies and are properly labelled, which does make the process seem a little daunting at first but mainly saves you time comparing the length of screws against a diagram. The paper instructions that come with the case are fairly straightforward, but there were a couple of times where we had to go back and do bits we'd missed. Thankfully, the build video can help there.



▼ The sheer volume of little baggies may make the build seem quite intimidating, but you quickly get through them

The little parcel box for the GPIO pins allows you to still do digital making with it

Practically sturdy

Once finished, you can boot it up and go. Some extra software is required to get it all working (especially the little OLED screen), but after a few terminal commands you're good to go with your brightly lit and very cool (because of the fans!) case. The aluminium frame is very sturdy and the weight feels good, and the little parcel box for the GPIO pins allows you to still do digital making with it. While it is larger than traditional Raspberry Pi cases, it's not taking up the whole desk. The OLED screen displays some system info, and all the ports are routed to the rear of the box like on a normal PC case – including switching out micro HDMI to regular HDMI for those who prefer it. It's fairly quiet too, which is a big plus.

Verdict

If you like the look of a mini PC case but still want to use some Raspberry Pi features like GPIO pins, this might be worth the fiddly build.

8/10

BOOK OF MAKING 2026

STEP INTO THE WORLD OF MAKING!

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

rpimag.co/BookOfMaking2026

Powered by Raspberry Pi

Raspberry Pi in its various guises is an excellent basis for a whole range of products, whether hobbyist, consumer or industrial-focused. We set up a Powered by Raspberry Pi certification scheme to help customers make confident purchases, safe in the knowledge that any product based around Raspberry Pi hardware and bearing this logo has gone through our detailed accreditation process. Hundreds of products now proudly sport this badge of recognition. You can find detailed information about the compliance regulations and testing procedures for every Raspberry Pi product at **pip.raspberrypi.com**. And for anyone keen to become a Powered by Raspberry Pi manufacturer and proudly sport the bespoke logo signalling your product has our official seal of approval, head to **rpimag.co/poweredbypiapply** to apply.

rpimag.co/poweredbyraspberrypi

APPLY TO POWERED BY RASPBERRY PI

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

Nest Box Live

UK | nestboxlive.com

Raspberry Pi has been following the progress – and the cute nesting/fledging videos and photos – of Nest Box Live since its inception back in 2019. It's grown from a handful of discreet boxes installed at the Owl Trust in NW England to a global network of birdboxes in gardens, at nature reserves and in educational premises around the world. The smart boxes feature networked HD video cameras while the Nest Box app provides AI-assisted species recognition. If you have a box installed, you'll get notifications whenever a feathered visitor arrives – a bit like a Ring camera. Nest Box Live founder Jamie explains that in the past two years alone, Facebook follower counts have increased from 5500 to more than 3 million. As well as highlight videos, the Raspberry Pi-based smart monitoring system has a live viewing app while birdwatchers who don't have their own box can watch webcams that offer real-time videos. No wonder it's such an impressive hit.

Phoenix Organs

Canada | phoenixorgans.com

Phoenix Organs hand-builds organs for church and home use with Raspberry Pi 400 or Raspberry Pi 500 in each and every one. The company – run by two engineers who were also organists – operates in Canada, the UK, and USA, and was specifically developed around Raspberry Pi. They build each digital organ with great care and attention using Raspberry Pi in order to create the most realistic pipeless organ sound using sampling technology. The system they designed can be used to control pipe organs of any size. Each note of every sampled stop can be independently voiced using the on-site voicing tools they've written for both pipe and electronic organists. Installations include the organ at Liverpool Cathedral.

HealthyPi 5

India | rpimag.co/healthypi5

Raspberry Pi hardware setup can be used for all sorts of biosignal acquisition and vital signs monitoring, including ECGs, respiration, body temperature, oxygen levels, and photoplethysmography (a non-invasive means of sensing oxygen and blood volume changes). The flexible setup is based around RP2040 and an ESP-C3 RISC module, with wireless and Bluetooth options alongside machine learning modules.

HealthyPi has a mainboard with 40-pin Raspberry Pi connector, ESP32 wireless, and Qwiic expansion options. Having been developed in India and proved their mettle during the Covid years, these open-source healthcare products are now available for implementation worldwide.

WINGS Black Mechanical Keyboard

Poland | arrowmechanics.com

Split keyboards can offer improved comfort if you find you ache from using a standard layout for a prolonged period. Polish company Arrow Mechanics has designed several keyboards aimed at reducing the discomfort associated with being hunched over a fixed keyboard: carpal tunnel syndrome, RSI, wrist, arm and shoulder pain can all be alleviated with a more ergonomic arrangement that minimises repetitive finger stretches and provides a more natural resting position for your hands. The team call on their own experiences of poor gaming setups and illogical, uncomfortable school typing regimes to come up with Arm-based split keyboards with hot-swappable keys that "free your wrists, relax your shoulders and [allow you to] straighten your shoulders".

Unipi Neuron

Czechia | rpimag.co/neuronc2

Unipi Technology specialises in programmable logic controllers, sensors, and Internet of Things hardware that can be used in smart homes and offices to automate heating and lighting, control building access, and monitor energy consumption. The Czech company's open-source approach means its products can be easily and seamlessly incorporated into existing business configurations or dovetail with a wide range of consumer products across brands and IoT standards. Having built up a successful reputation in Eastern Europe, Unipi now sells internationally, both direct and through B2B partnerships. Many of the company's smart home controllers are Powered by Raspberry Pi-certified, so you can choose the one that best fits your Raspberry Pi configuration with confidence.

Screenly Player

USA | rpimag.co/screenlyplayer

Screenly has become one of the best-known names in digital signage globally, with versions of its software available in free editions (now known as Anthias) right up to installations of more than 100 screens at a single site. Screenly offers high-resolution video and static displays that can be updated and refreshed on the fly, as well as app-based control options. It is ideal for information screens, digital menus, marketing and advertising. The 4GB Screenly Player is based on Raspberry Pi. It offers 1080p video as well as showing photos, slideshows, and website footage, promising a setup process that takes mere minutes. In addition, a slew of apps can be embedded in a Screenly digital display to provide up-to-date news, weather and other timely details: **screenly.io/apps**.

TinyPilot Voyager

USA | rpimag.co/voyager2a

Being able to control a computer from pretty much anywhere can be a boon if you need to remotely access your network, update your digital display, or make use of several PCs at once. TinyPilot is geared up for just this sort of scenario, providing KVM (keyboard, video monitor, and mouse) control and out-of-band network access via a secure server housed in a compact box. Raspberry Pi-certified products such as the 2GB Voyager 2a operate over either 2.4GHz or 5GHz 802.11ac secure wireless networks and runs on the Debian Bullseye operating system. Low latency 24fps video capture at 1920×1080 and a near silent fan make it ideal for server rooms and unobtrusive office use.

ONLY THE **BEST**

Power solutions

By **Phil King**

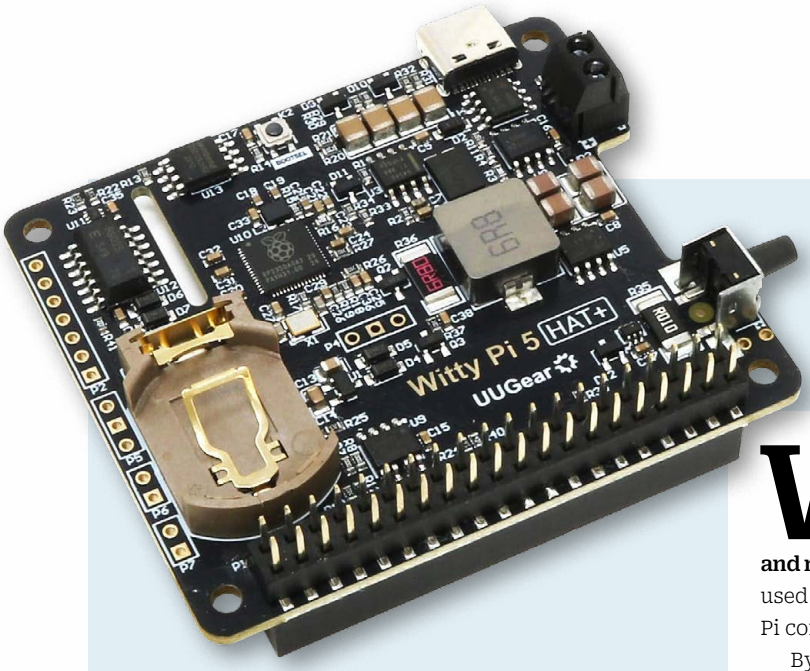
Powering a Raspberry Pi and its connected devices isn't always as simple as plugging in a USB power supply. What if you want to use it to control higher-powered devices? For that, you'll need relay switches. Fortunately, there are plenty of relay HATs and boards available. Or you can use MOSFETs as an alternative.

When it comes to powering Raspberry Pi itself, if it's vital that your project should stay up and running continuously, you'll want an uninterruptible power supply (UPS) which can switch to battery backup. You may also want to schedule system shutdowns and restarts – the Witty Pi 5 has you covered here.

Finally, you may need to convert the voltage of an external power source to supply Raspberry Pi, in which case a simple buck/voltage converter may suffice. Or you could even power a Raspberry Pi 4 or 3B+ over an Ethernet cable using a PoE+ HAT. Get ready to power up...

Witty Pi 5 HAT+

UUGear / The Pi Hut > £41 / $54 | uugear.com / thepihut.com



▲ Packed with features, it's ideal for reliable powering of projects

Verdict

An excellent all-round power management solution for Raspberry Pi.

When it comes to smart power management, the Witty Pi series has been providing it for many years, enabling scheduled shutdowns and restarts of a Raspberry Pi. Based around an RP2350 chip (as used in Pico 2), this latest model can be used with any Raspberry Pi computer with a 40-pin GPIO header.

By connecting it to a PC via its USB-C port, you can easily monitor real-time logs and manage config files and scheduling scripts – which are now stored on the board itself rather than Raspberry Pi, for crash-proof reliability. As a Mode 1 Power HAT+ board, it can deliver up to 5A of current to Raspberry Pi and its peripherals. There's also a DC/DC converter input for up to 30V and it can act as an uninterruptible power supply (UPS) using one or both power inputs.

Along with a high-precision onboard temperature sensor, there's the usual real-time clock that enables the board to retain the correct time even when powered down, although you'll need to supply your own CR2032 coin cell battery.

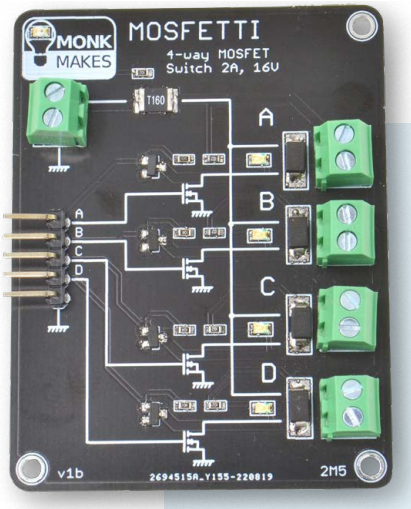
Mosfetti

Monk Makes / The Pi Hut > £10 / $13 | monkmakes.com / thepihut.com

An alternative to using a mechanical relay switch, a MOSFET (Metal-Oxide-Semiconductor Field-Effect Transistor) can be used to amplify or switch electronic signals. This board features four N-channel MOSFETs for fast low-side switching of loads of up to 16V with a maximum current of 2A per channel (2.5A across all four).

The PCB comes supplied with all the surface-mount components attached, although you will need to solder on the screw terminals and five-pin header. The latter can then be connected to four GPIO pins and a GND pin on your Raspberry Pi computer or Pico (or other 3V/5V microcontroller) to control devices connected to the MOSFET terminals – including using PWM to dim lamps, etc. You'll need to add a separate 12V+ DC power supply via two screw terminals.

Detailed instructions and wiring examples help you to get started and there are Python and MicroPython code examples in a GitHub repo.



▲ Switch up to four devices with this MOSFET-based board

Verdict

A handy MOSFET board that can be used with Raspberry Pi or Pico.

Raspberry Pi Relay Board

Waveshare / The Pi Hut > £15 / $21 | waveshare.com / thepihut.com

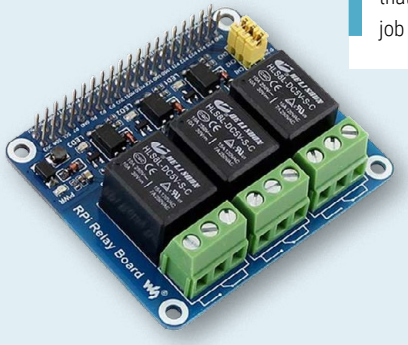
There are many Raspberry Pi relay HATs and breakouts on the market. This Waveshare board (not a HAT, strictly speaking) is a fairly simple affair featuring three high-quality relay switches. While too tall to be stackable, it does have a full GPIO pass-through header which you could use to connect one or more extra relay boards via jumper wires.

As usual, there are inputs for normally open (NO) and normally closed (NC), along with status LEDs. Relays are controlled via a default set of GPIO pins – switched on when they're pulled low, which is unusual; you can also use custom pins by shorting a jumper on the board. The online documentation features a few code examples, including Python. There's even an optional web interface to trigger the relays remotely.

The relay switches can handle a 5A load for up to 30V DC. The spec says they can also switch 250V AC, but you should not even consider doing this unless you have specialist electrical expertise as it's potentially lethal.

Verdict

A simple, low-cost relay board that does the job effectively.



▲ This straightforward board has three relays

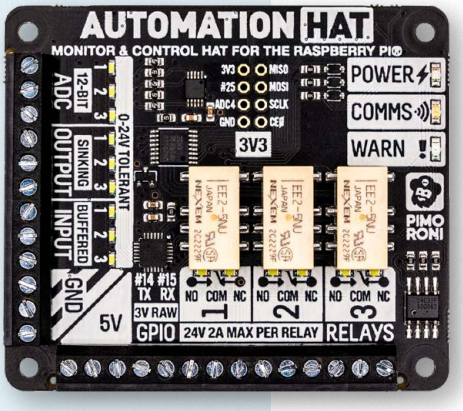
Automation HAT

Pimoroni > £30 / $33 | pimoroni.com

While a Raspberry Pi computer's GPIO pins are great for controlling low-voltage electronic components, they are limited to 3.3V with a maximum output current of 16mA. To switch higher-powered components and devices (such as for home automation) on and off, relay switches are a good option.

This full-size HAT features three mechanical relays that are tolerant to up to 24V (DC) and 2A and have NO (normally open) and NC (normally closed) screw terminals. In addition, there are three sinking outputs (solid-state electronic switches), with a combined output of up to 500mA maximum, and buffered inputs to read the state of devices. These are all 24V tolerant. If you do need to control higher voltages, Pimoroni's Pico W-powered Automation 2040 W can handle up to 40V and offers very similar functionality to this board.

The Automation HAT also features three 12-bit ADC inputs (up to 24V), a breakout header, and lots of status LEDs to show what's going on. A Python library makes it very easy to use in your programs.



▲ As well as three relays, this HAT has lots of 24V-tolerant inputs and outputs

Verdict

A function-packed board for controlling devices up to 24V.

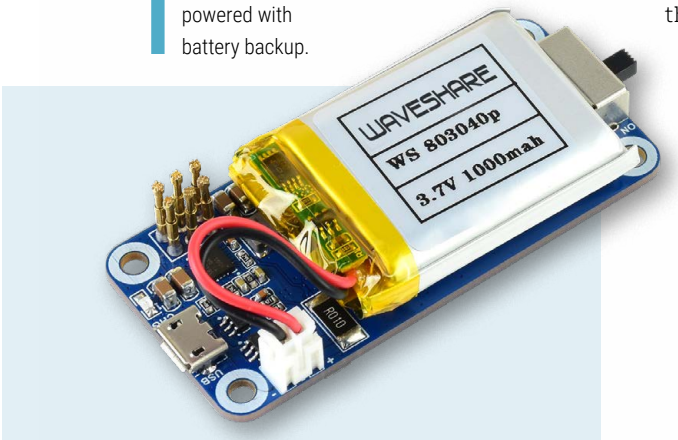
UPS HAT For Raspberry Pi Zero

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

Verdict

Keep your Raspberry Pi Zero project powered with battery backup.

▼ An uninterruptible power supply for Raspberry Pi Zero



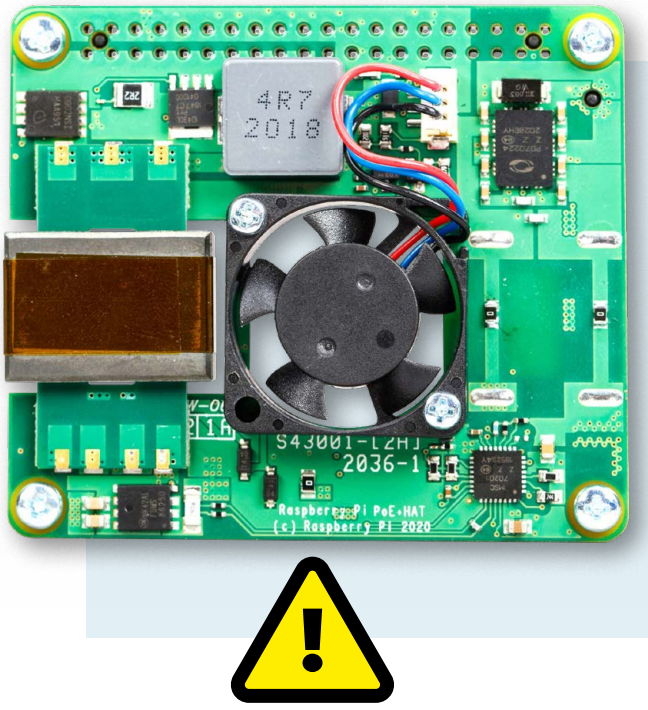
When you have a Raspberry Pi running constantly for an important project, there's nothing more frustrating than a mains power cut. To avoid this, a UPS (uninterruptible power supply) is an essential bit of kit, maintaining power with a battery pack.

This one works with any Raspberry Pi Zero model with a GPIO pin header attached. While it's called a 'HAT', it sits underneath Raspberry Pi Zero and six springy pogo pins connect to the soldered underside of the GPIO header, thereby leaving the GPIO pins free to use (it only uses I2C).

Once in place, you can connect a standard 5V power supply to its micro USB port to start charging the onboard 3.7V 1000mAh LiPo battery. Flicking the switch will power on Raspberry Pi Zero. If the mains power then fails, the battery will instantly take over for up to seven hours of power. You can monitor the battery level by installing a simple Python program.

PoE+ HAT

Raspberry Pi > £19 / $26 | raspberrypi.com



▶ Delivering power and data over the same Ethernet cable

If you take a look at any recent full-size Raspberry Pi computer, you'll notice a mysterious four-pin header behind the Ethernet port. This is for Power over Ethernet (PoE), a way of delivering power (along with the usual data) along an Ethernet cable: up to 5A at 5V.

The PoE+ HAT can be used to power Raspberry Pi without the need for a separate power supply, which may prove useful for remotely deployed projects such as cameras and sensors, along with industrial applications such as digital signage, as you only need a single cable. A fan helps to keep everything cool on the board.

One key thing to note about the PoE+ HAT is that it doesn't work with Raspberry Pi 5 due to the different position of the PoE header on the latter – only Raspberry Pi 4 and 3B+. You will also need a suitable powered Ethernet switch box to provide the power. Alternatively, you can use a $25 official PoE Injector with your regular router.

Verdict

Power your Raspberry Pi 4 or 3B+ over the network.

Warning!

Electrical Safety

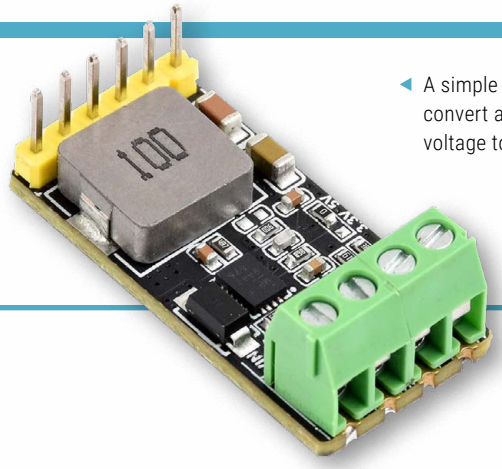
Please be careful when working with electrical projects around the home. Especially if they involve mains electricity.

rpimag.co/electricalsafety

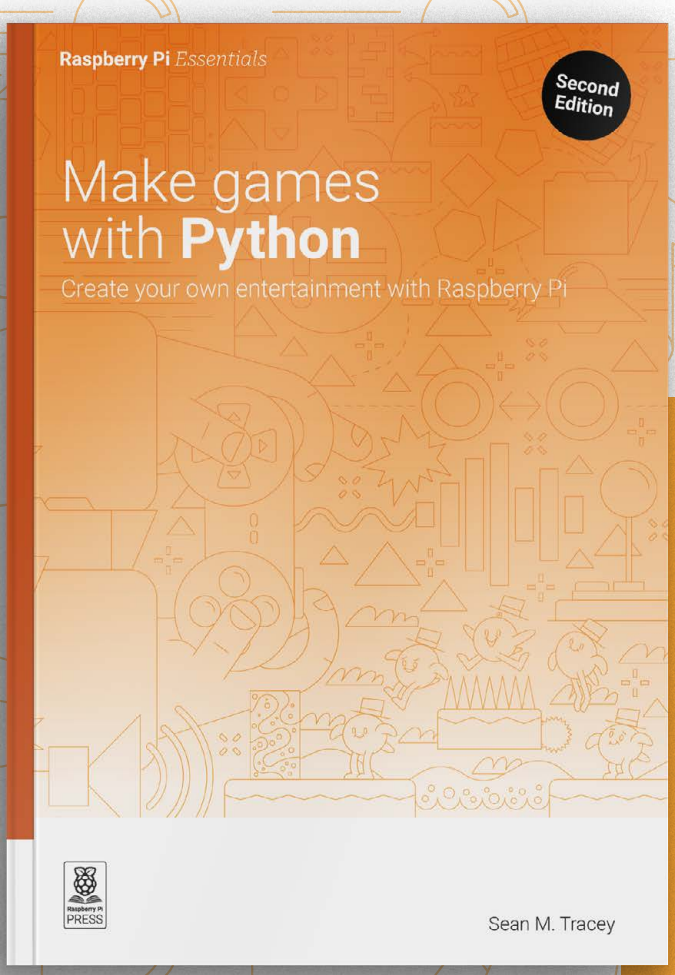
DC-DC BUCK MINI MODULE

Waveshare /The Pi Hut > £3 / $5 | waveshare.com / thepihut.com

Voltage (or buck) converters take an input voltage and change it to another, which can be useful for projects comprising devices with different voltages. This one takes an input of 6V–30V and turns it into 5V – ideal for powering Raspberry Pi from a higher-voltage power source.



◀ A simple device to convert an input voltage to 5V



While millions of us enjoy nothing more than spending hours racking up high scores on our favourite video games, too few are exposed to an even more gratifying way to spend time — making them.

This book teaches Python and Pygame development, helping you to understand the games you play and create almost anything your imagination can come up with.

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

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

10 amazing:

advanced retro gaming projects

Do more than just play old games with these incredible retro gaming builds

Warning!

Copyright

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

rpimag.co/legalroms

We have several little retro gaming devices lying around Raspberry Pi Towers, and in our homes, all powered by a Raspberry Pi – some even by a Pico. That's the easy part though, and it is very easy, so some makers have truly challenged themselves to take the next step and create some truly astounding builds.

01. Recalbox RGB Dual 2

Scanned lines

recalbox.com

Want the true retro gaming experience with a Raspberry Pi? The RGB Dual 2 kit enables you to hook your Raspberry Pi up to CRT TVs using RCA, VGA, and SCART and adds some extra processing to make it look just like the devs imagined.

02. Fancy Octopus Arcades

Beautiful customs

rpimag.co/octopus

It's never been easier to play retro games, or make your own arcade cabinet. These bespoke custom builds are designed for smaller apartments, allowing more people to have one.

03. Robot Pac-Man

Pac Real World

rpimag.co/robotpac

There was a cult classic game on Nintendo's GameCube called Pac-Man VS., where one player as Pac-Man had to avoid player-controlled ghosts. This is very similar, but with physical parts!

04. Star Wars Arcade Cabinet

Perfect recreation

rpimag.co/swarcade

Why track down a (very expensive) classic arcade cabinet when you can create a 3/4 scale replica of it? As well as extensive detailing, the true pièce de résistance of this build is an accurate yoke replica.

05. GamePad Zero

Plug 'n' Play

rpimag.co/gpadzero

A Raspberry Pi Zero is so small, you can fit it into a classic games controller. While we hacked a USB knock-off apart with a Dremel in issue 40, this one uses 3D printed parts on the real deal.

06. DOOM on Pico

It can run

rpimag.co/doompico

A common refrain on the internet is being presented with a piece of technology and asking 'can it run Doom?' Pico, apparently, can. We don't have high hopes for Crysis, though.

07. Giant Fine-Art Game Boy

Games as art

rpimag.co/finegb

These playable art pieces show the beauty of classic technology, at six times the size, while also playing some of the games that made them famous.

08. Soundfighter

Symphony of fists

rpimag.co/soundfighter

Piano duelling takes on a whole new meaning when your ivory keys are tickling the inputs for Street Fighter Alpha 3. We hope you need to perform a sonata for Akuma's Raging Demon.

09. Cupcade

Tiny Space Invaders

rpimag.co/cupcade

While more of a showpiece, this is a fully functional arcade machine that will happily sit in the corner of your desk. Be aware of hand cramps, though.

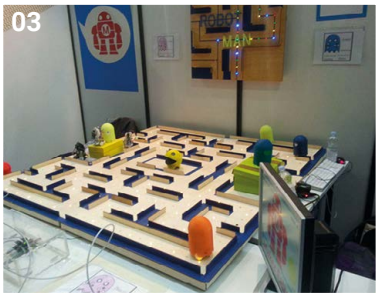
10. Tomy Turnin' Turbo Dashboard Out Run

Upcycled realism

rpimag.co/tomyturbo

Taking a classic children's toy and theming it to a Ferrari, and one of the most famous Ferrari-based video games of all time, is quite an inspired idea.


02

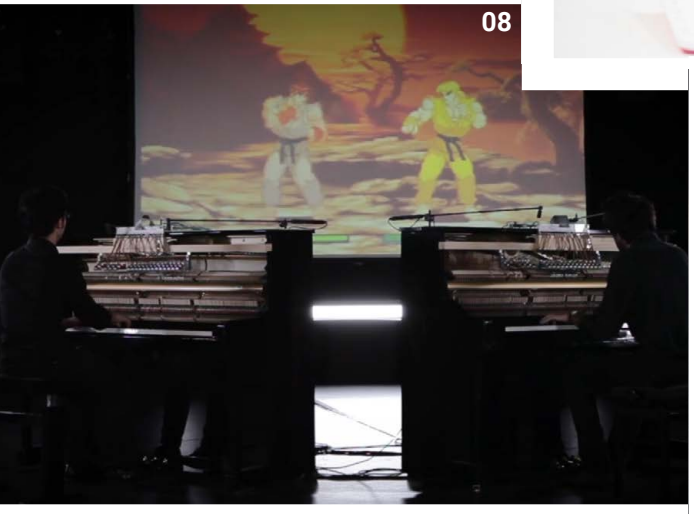

03


04

05

06

07


08


09

10

Richard Kirby

An automation engineer who discovered the joy of quick prototyping on Raspberry Pi, leading to organising one of the longest-running community events

Name Richard Kirby
Occupation System engineer
Community role Event organiser
URL rpimag.co/rpint

Automation is something a lot of Raspberry Pi enthusiasts are into, but this month's subject, Richard Kirby, takes it to a whole other level: "My work time is consumed as a test manager at a multinational company that develops and deploys large railway automation systems throughout the world," he tells us. "The system is largely automated, with signallers and train operators only intervening as needed – this includes automated driving of the trains."

A lot of his work involved the London Underground, and after moving to the city itself from Canada for a 'two-year stint' in 2009, he's stuck around. His projects have been featured in the magazine before – Pi Fighter is a particular highlight of ours (**rpimag.co/85**) – and he also organises the Raspberry Pint meetup in London.

What is your history with making?

In 2014, my daughter asked for a bit of help with a school project. She had decided to use a Raspberry Pi 1 Model B and was struggling with making it drive a DC motor. I was shocked at how quickly we sorted out a simple DC motor system. The DC motor spun a table tennis ball at different speeds using PWM.

▼ A photo from a recent, Christmassy Raspberry Pint

▲ The gamified punching bag, Pi Fighter, powered by Raspberry Pi

The ball was then manually catapulted to measure the Magnus effect. I considered it a major victory, but was brought down to earth when she complained we had no idea of the rotational speed. We ended up building a Raspberry Pi strobe to determine the rotational speed; you increase the flashing frequency until the ball appears stationary. This was all over a weekend.

Needless to say, I got hooked on the pleasure of quickly building something with a Raspberry Pi, followed by spending ages making it more polished. It is addictive to get a quick win, followed by a grind to get something good enough to tell others about. It keeps me sane after working on projects that are four to twelve years in duration.

How did Raspberry Pint start?

Matt Mapleston started Raspberry Pint in a London pub in 2013. I went to my first Pint in 2016 and took over the organising of meetings in 2017. I wanted to make sure it wouldn't fade to black – I had found my people! We have grown since the early days from a handful of people to hybrid in-person and online meetups with between 30 and 80 people.

What are some of your favourite Raspberry Pint memories?

There is a lot of joy and fun at Raspberry Pint as the format is makers telling everyone about their creation. The makers are excited and rightfully proud to explain their creations. We have had a lot of great local presentations of weird and wonderful projects. It is a relaxed venue where the maker can tell their making story; all their trials and tribulations, eventually ending in sweet victory. Even highly accomplished people like the NASA engineers behind the ISS Mimic project (**rpimag.co/issmimic**) were brimming with an obvious love of what they had built. I never imagined a great community from around the world joining us to tell us of their experiences. Even the Australians and Japanese are getting up very early in their morning to tell us about their builds. It is a fantastic way to spend a weeknight with a pint.

A key highlight was Eben Upton joining us in a wide-ranging informal discussion. He was relaxed and engaging over a wide range of topics. He was able to answer all the questions on a huge range of topics at a surprising level of detail. Everything from detailed design decisions to the stock levels of Raspberry Pi 3B+ across the world – it was during the chip shortage, so it was a hot topic.

There is a lot of joy and fun at Raspberry Pint as the format is makers telling everyone about their creation

What are some of your favourite Raspberry Pi/Pico creations?

'Pi Fighter' featured in *The MagPi* issue #85 (**rpimag.co/85**) is still a favourite of mine. It gamified heavy bag workouts. It still works and was a big hit (and took big hits) at the Cambridge Raspberry Jams.

I still regularly use the 'Talkative Tube Dashboard' that was featured in *The MagPi* issue #120 (**rpimag.co/120**). It provides real-time statuses of the Tube lines. It is a nice background for work video calls as we can see railway problems in real time.

My current favourite is my 'Diet Tracker' project, which I have started using seriously again. It tracks what I eat and the amount of calories I consume. I have a fairly healthy lifestyle, but I needed some help in improving my fitness. Naturally, this meant building something using Raspberry Pi. Two years later, I'm significantly healthier, and I'm now using it to get the next level by further fine-tuning of my food intake by closely monitoring protein, fat, and carbohydrate intake.

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. Apparently, this is nearing completion – and looking very good, we might add

02. This reminds us of that one prop that kept appearing in 1990s sci-fi that was just some red tubes. Looked cool, though

03. New year, new robots from Kevin

04. A very cool-looking setup for managing your home automation

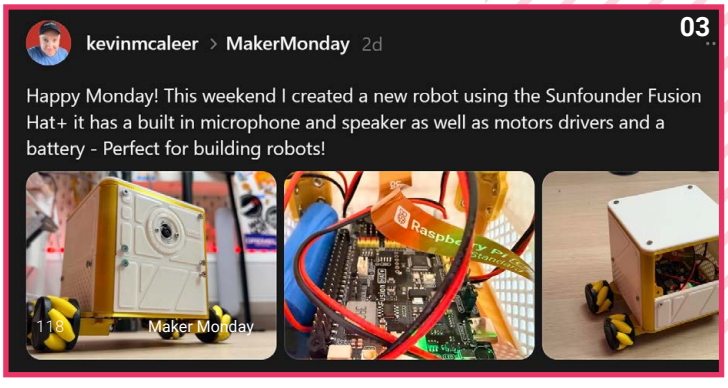
05. As a Disney Adult, we're not immune to the charms of this tree topper

06. At the time of writing, Christmas had been very recent, so please forgive some leftover festive projects

07. Akkie modified some driver code to get this working. Impressive!

08. A powerful yet minimal upgrade to a classic project

09. Not sure why you'd need a lunchbox for your whiskey, but a cyberdeck is always handy!



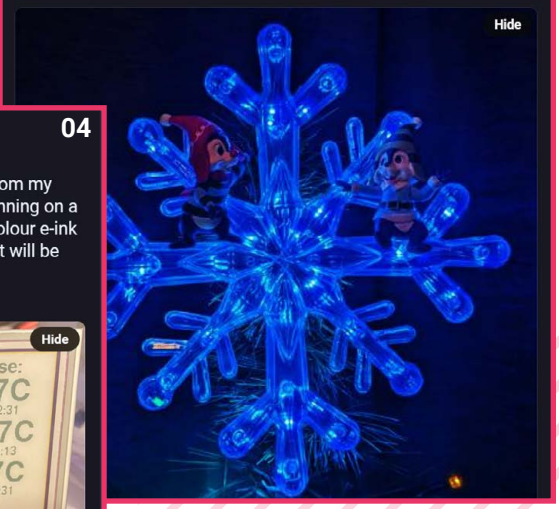
05

PenguinTutor (Stewart Watkiss)
@penguintutor@fosstodon.org

@rpimag
I've upgraded a Disney Christmas Tree topper with a Raspberry Pi Pico.

I used a circuit and PCB I'd originally created for a MagPi article.

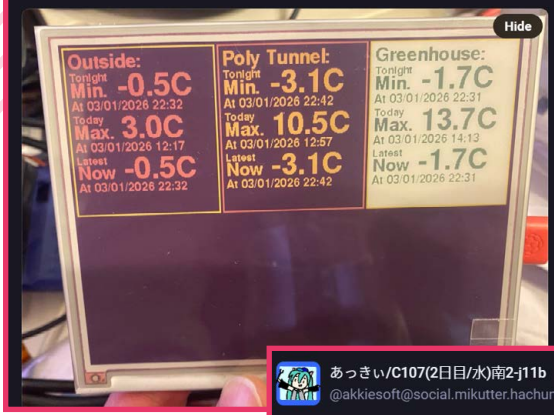
youtu.be/7if3OGkeGTU?si=V-VclZ...



04

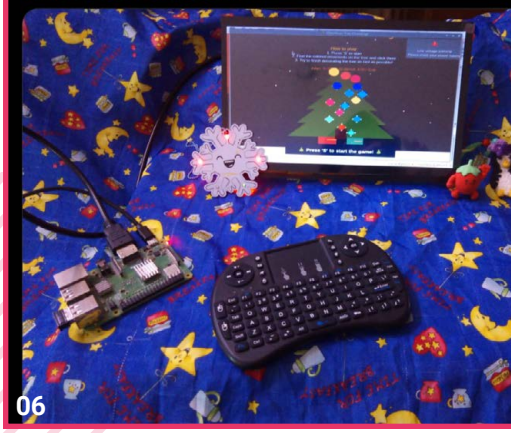
Dr Footleg (he/him)
@drfootleg@fosstodon.org

@rpimag I have been working on a dashboard for data from my newly built Home Assistant server. The server itself is running on a CM4 and this dashboard is using an Inky Impression 7 colour e-ink display attached to a Raspberry Pi 4 for development, but will be run on a Pi Zero 2W once it is installed in the kitchen.
#MakerMonday #HA



Aula Jazmati
@Aula_J2018

Good Morning and happy New Year
hackster.io/aula-jazmati/i...



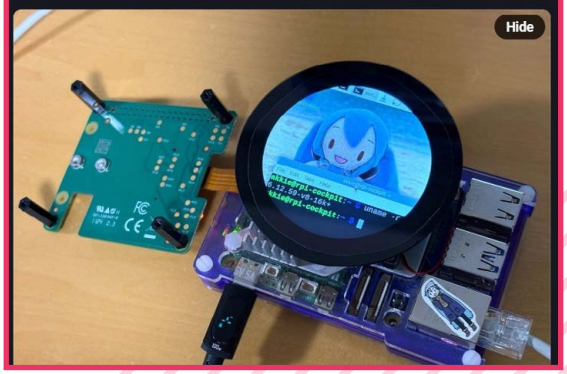
06

07

あっきぃ/C107(2日目/水)南2-j11b
@akkiesoft@social.mikutter.hachune.net

@rpimag Hello!
Last week I attempted to port HyperPixel2r to run on Linux 6.12.
(There is an issue that is not functioning on the current RPi OS: github.com/raspberrypi/linux/i...)

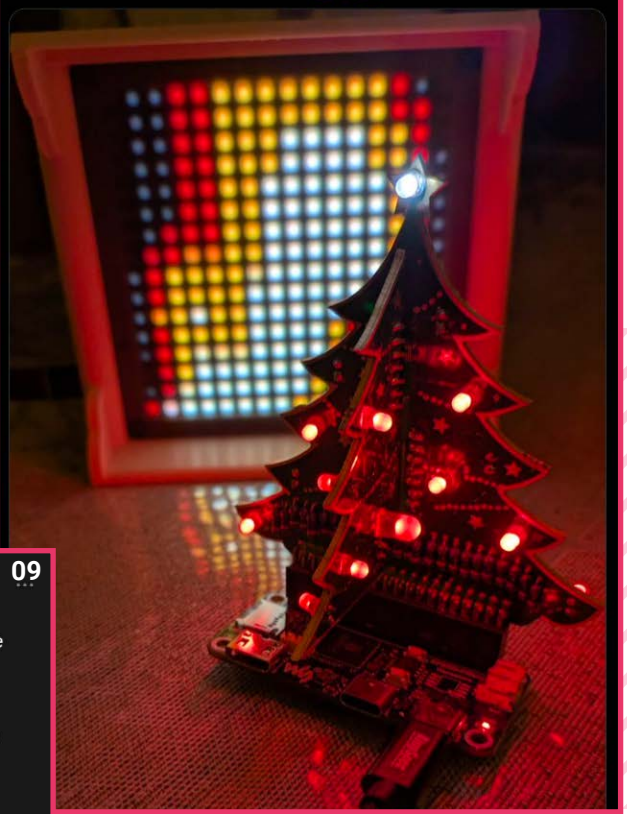
This was my first time modifying the driver's source code, but I managed to get it working :)



08

Pierre-yves Baloche
@FunkyPiwy

Took some time to finally put up the Pi-stMas tree, with an upgrade from a #PiZero to a #Pico setup! With reduced startup time, these little blinkers are making Rudolph jealous

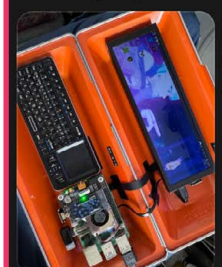


09

maniacfive 08/12/2025

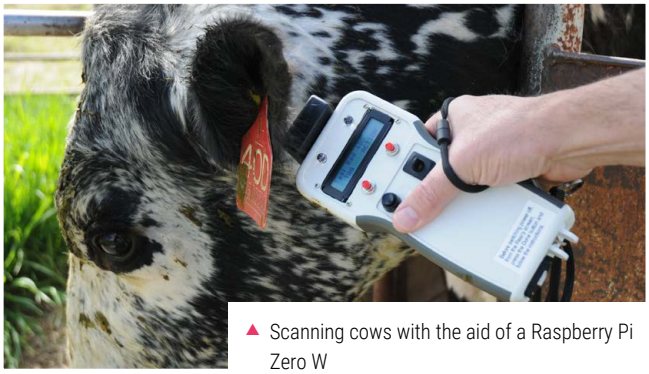
Still tweaking my RPi5 'cyberdeck' project I'm building out of lunchbox that came free with some bourbon.

I'm using it as a torrent downloader. Still working on getting sound, blew the decoder I was intending to use via some clumsy soldering. Oops. Oh well, who needs sound anyway.



Your Letters

For all the cows



▲ Scanning cows with the aid of a Raspberry Pi Zero W

I've just moved house, and had to pay a fee to update the information on my dog's chip – at least, I assume that the data on the chip just links to a website, and it's there that the data is updated. Anyway, it got me thinking about homebrew solutions, and after a while I realised just how difficult it would be. Anyway, that's a long way of saying that I'm impressed by the Flokk herd management system – updating information for one 10kg dog was beyond me, so building a Raspberry Pi Zero-based system to scan and manage data for a whole herd of cattle would be something else entirely.

Neil, via email

It's a shocking admission to make, but we sometimes forget about Raspberry Pi Zero. At one end of the Raspberry Pi stable there's the all-singing, all-dancing power of the Raspberry Pi 5 family, with the clicky keyboard of the 500+ and the AI and storage HATs; at the other end there's the low-power, tiny, cheap Pico group, which is ideal if you want to monitor a sensor, or control logic in a small device. Raspberry Pi Zero W (or 2 W) is somewhere in between these two use cases, with a smaller form factor and lower battery consumption than a Raspberry Pi 5, and the ability to run more software (a complete operating system, in fact) than Pico. If you're out and about, and you need a device that can upload data to a web server, then a Raspberry Pi Zero W is the perfect computer on which to build your device.

3D printer filament scale

I laughed when I saw the 3D printer filament scale. How pointless, I thought, to automate a task that takes only a couple of seconds. Then I thought a bit more, and realised that it's brilliant. What kind of brain does it take to look at something so mundane as taking a spool of filament off the printer, weighing it, and tapping some numbers into a calculator, and thinking: "I could build a machine to do that"? I've lost count of the number of times I've tried to eke out one more print from an almost-finished roll of printer filament only to run out just before the print job is finished, wasting time and material. If you look at the time that the filament scale saves in the best-case scenario, when all you do is weigh the spool, it's a ridiculously over-engineered solution. But as soon as you factor in the time, material, and sheer frustration that it saves long into the future, you see it for what it is: a boon to mankind!

David, via email

Chris Forde, who built the filament scale that we featured in RPOM issue 161, is a maker after our own hearts. Yes, you could just weight the filament before you start, to get an idea of whether you're going to run out halfway through, but in the act of building an automatic system such as Chris has done, you would learn so much more. Not only that, but we're suckers for the chunky switches he used – there's something about digital technology controlled by big, physical bits of metal that we adore.

◀ Never run out of filament mid-print again!

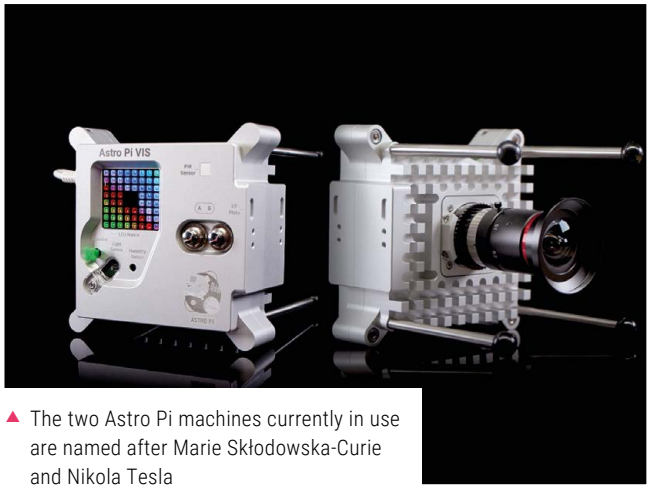
Astro Pi is 10!

A few years back, my son entered a competition to get some code that he'd written running on a Raspberry Pi in space, aboard the International Space Station (ISS). At least, it seems like a few years ago – I learn today that it was actually ten years ago, and I am far older than I thought. It's crazy that something literally space-age is a decade old.

Marie, via email

Yup. In December 2015, astronaut Tim Peake set sail from the Baikonur cosmodrome in Kazakhstan on a Soyuz rocket on his way to the ISS. Waiting for him aboard were two Raspberry Pi computers, ready to be set up as part of the educational project known as Astro Pi.

The world has changed since then; we're no longer hitching rides on Russian rocket ships, for example. But Astro Pi is still up there, comprised of a Raspberry Pi computer (plus a modified Sense HAT, High Quality Camera, Coral AI accelerator, and a filter for the camera to enable it to monitor vegetation coverage.



▲ The two Astro Pi machines currently in use are named after Marie Skłodowska-Curie and Nikola Tesla

Contact us!

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

Community Events Calendar

Find out what community-organised Raspberry Pi-themed events are happening near you...

01. FTS Raspberry Pi Jam

Thursday 5 February
Fruitful Trees Premier School, Ota, Nigeria
rpimag.co/ftsrpj162

Where young enthusiasts come to learn, explore, and interact with the Raspberry Pi 500 + keyboard features. Explore coding with Python, HTML, Scratch, etc.

02. Totnes Raspberry Pi Jam

Monday 16 February
Go Deer, Totnes, UK
rpimag.co/trpj162

Bring craft projects to life with lights and motion. The event will begin with Snap Circuits — clever components that join together with press-studs to create working circuits with lights, motors, batteries, and sensors. Once everyone has the basics, they'll use Raspberry Pi to check the weather and code mini traffic lights.

03. Coding with Blocks and Square Roots

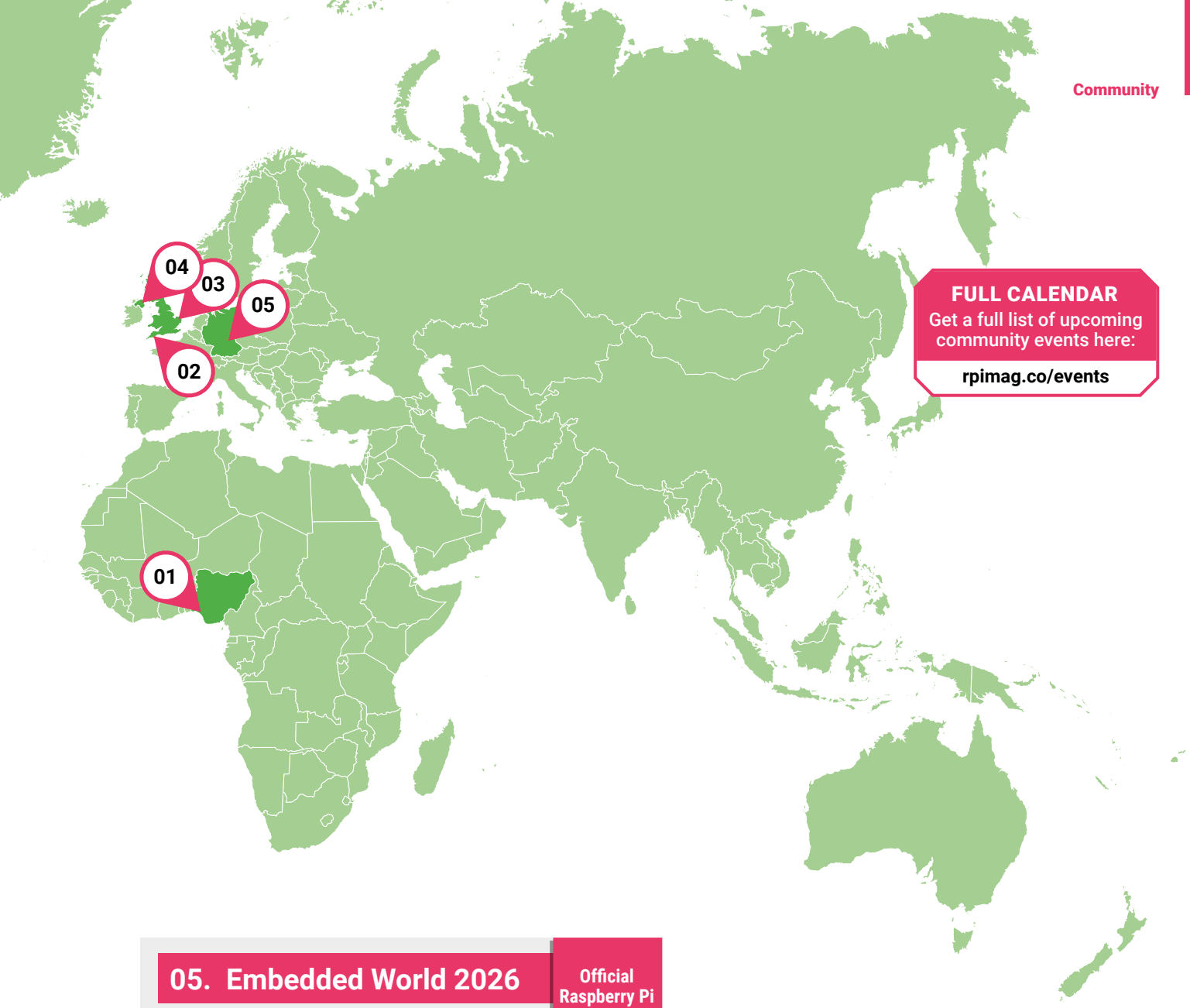
Saturday 21 February
The Forum, Norwich, UK
rpimag.co/cblocks162

Fun for the whole family. Come and explore what happens when technology, creativity, and curiosity collide. This is a relaxed, hands-on session where you can try something new.

04. Northern Ireland Raspberry Jam

Saturday 21 February
Queen's University, Belfast, UK
rpimag.co/nirj162

In collaboration with the School of Physics and Mathematics at Queen's University and the Northern Ireland Raspberry Jam team, these free monthly events involve tinkering, coding, electronics, and generally just having a stack of fun making stuff!



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

05. Embedded World 2026

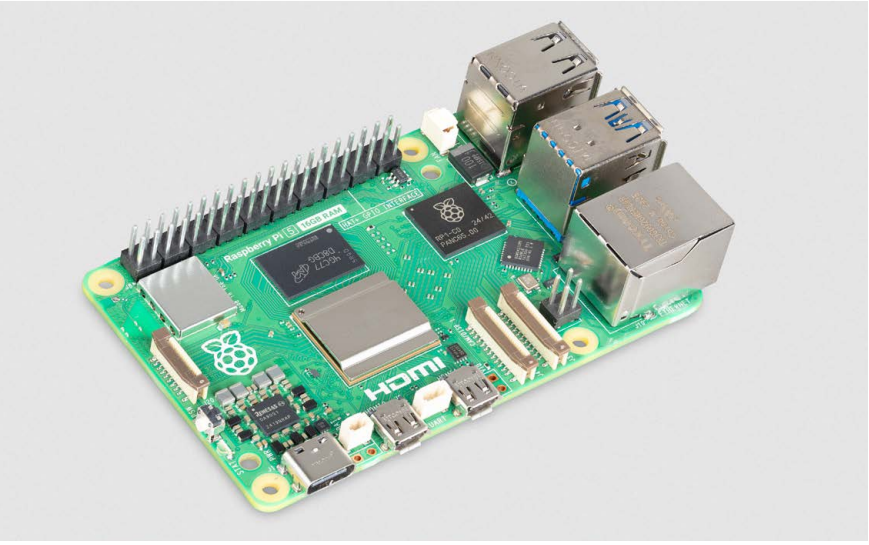
Official Raspberry Pi Event

- **Tuesday 10 March 2026 to Thursday 12 March 2026**
- **Messezentrum 1, Nürnberg, Germany**
- **rpimag.co/ew26**

The Raspberry Pi team is looking forward to returning to Embedded World in 2026. There, you'll be able to meet them and experience demos from across the full spectrum of Raspberry Pi products, including Raspberry Pi Pico 2, the AI product range, RP2350-based solutions, and Raspberry Pi's latest industrial device: Compute Module 5.

Win a Raspberry Pi 16GB

Protect yourself from the ongoing RAM shortage with a Raspberry Pi 5 stuffed full of 16GB RAM. It's perfect for AI inference projects, using as a powerful desktop computer, and running demanding code. Show off to your friends and flex your memory muscle with the most RAM spacious model around.



Head here to enter:

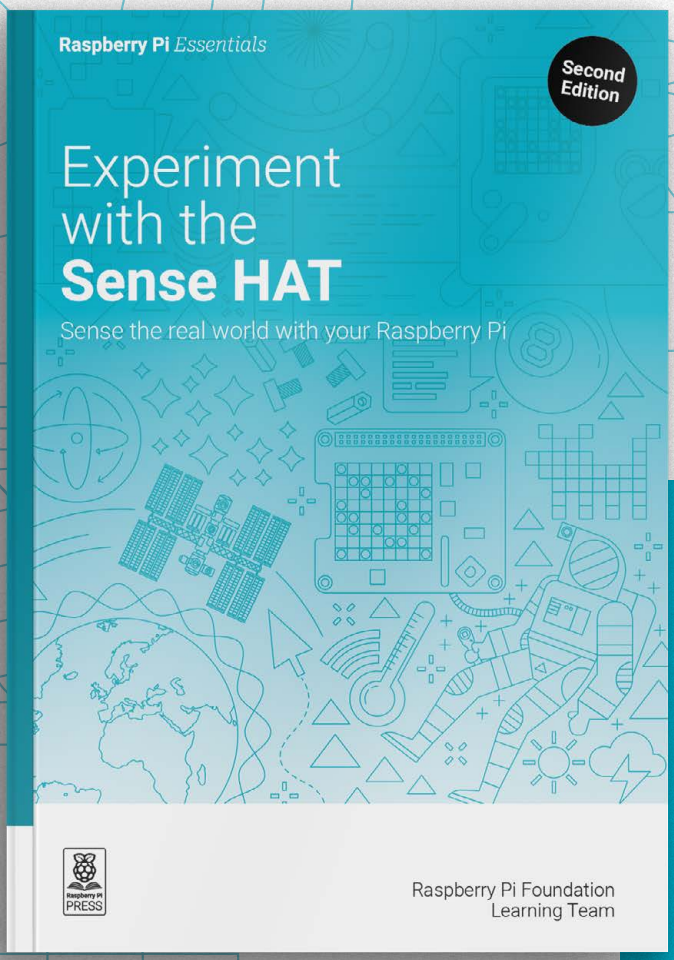
rpimag.co/win

Learn more:

rpimag.co/raspberrypi5

Terms & Conditions

Competition opens on **28 January 2026** and closes on **26 February 2026** Prize is offered to participants worldwide aged 13 or over, except employees of Raspberry Pi Ltd, the prize supplier, their families, or friends. Winners will be notified by email no more than 30 days after the competition closes. By entering the competition, the winner consents to any publicity generated from the competition, in print and online. Participants agree to receive occasional newsletters from Raspberry Pi Official magazine. We don't like spam: participants' details will remain strictly confidential and won't be shared with third parties. Prizes are non-negotiable and no cash alternative will be offered. Winners will be contacted by email to arrange delivery. Any winners who have not responded 60 days after the initial email is sent will have their prize revoked. This promotion is in no way sponsored, endorsed or administered by, or associated with, Instagram, Facebook, Twitter (X) or any other companies used to promote the service.



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

- ***Getting started with Sense HAT***
- ***Learn by building:***
  - *A digital twist on the Magic 8 Ball*
  - *Your own interactive pixel pet*
  - *A sparkly light show*
  - *An environmental data logger*
  - *Flappy Astronaut, a low-res, high-fun video game*

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

REACH A GLOBAL COMMUNITY

Advertise in **Raspberry Pi Official Magazine**

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

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

Take the next step –
advertise with us today!

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

Official Magazine
#163

Learn Python

Modern coding masterclass

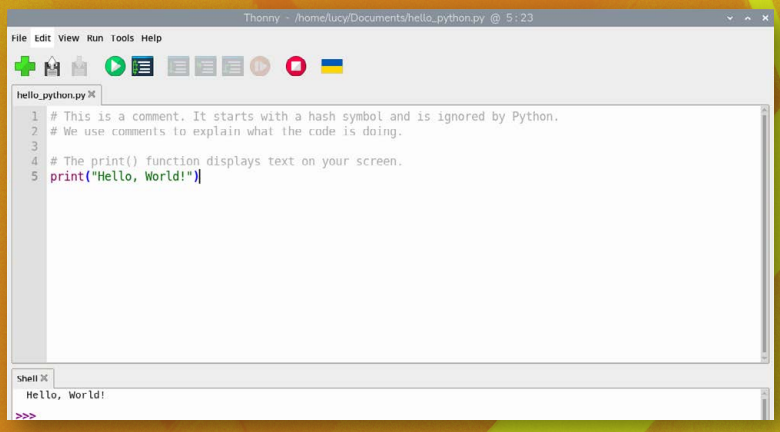
On sale 26 February

PLUS!

Emulating retro computers

Wear Raspberry Pi sneakers

Discover chiptune music



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

Tim Danton, Frank Delporte, Rosemary Hattersley, Jo Hinchliffe, Phil King, Andrew Lewis, Rob Miles, Richard Smedley

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
raspberrypi@subscriptionhelpline.co.uk

ISSN: 2977-4403 (Print)
ISSN: 2977-4411 (Digital)

Let's get stuck into 2026

Lucy Hattersley is limbering up for an eventful year ahead

Thanks to the miracle of magazine time travel, it's presently five days past the new year as I write this article. I'm still full of a little festive cheer, and ready for the year ahead.

I was partying in December and am enjoying my newfound Dry Jan sobriety. What did we achieve in 2025 to warrant all the parties? Quite a lot for the magazine: we've now got a solid year of being *Raspberry Pi Official Magazine* behind us, and although *The MagPi* will never be forgotten, I feel we have definitely earned our new official stripes.

Join the party

I can't thank everybody who subscribed to *Raspberry Pi Official Magazine* last year enough. Becoming a subscriber really helps (**rpimag.co/subscribe**). I subscribe to a few print magazines still, and it's an absolute joy to get a magazine in the post every month. It's so much healthier than doomscrolling the internet.

Speaking of print: I'm knee-deep into writing an AI book this year, and have been playing around with various AI systems. It's a fascinating mix of new and old.

I can't talk about upcoming products for 2026. It's seen as poor form to spoil the surprise and the company lawyer puts on her angry face. Still, there's a lot happening in 2026 and it comes on the back of a busy year.

> *It's such a delight that I've moved away from using non-Linux operating systems*

It was Raspberry Pi 500+ that really stole my heart in 2025. This wonderful clacky keyboard variant of my favourite computer has absolutely everything I need from a desktop: a mechanical keyboard, 16GB RAM, built-in NVMe SSD, and a GPIO header so I can attach sensors and actuators to do stuff in the real world.

It's such a delight that I've moved away from using non-Linux operating systems almost completely. And guess what: Linux is better in every way. Although we already knew that.

Let's get things started

We've kicked off 2026 with an in-depth feature on robotics and astrophotography.

Raspberry Pi now powers robots, video display screens, servers, thin clients, industrial controllers, hotspots, and electronics projects. The list of things it does (**raspberrypi.com/for-industry**) is dazzling. Just don't get lost in the options!

Our favourite computer is one year older, and growing year-on-year. One year it'll be able to get its own round in at a bar. And when it does, I'll be waiting with my Guinness Zero in hand and wanting to know what it's got planned for the rest of its life.

Lucy Hattersley – Editor

Lucy is going to attempt the Chester Marathon in 2026 and has run a total of zero kilometres so far this year. So that'll be something to laugh at if you're in town.

rpimag.co

HiPi.io

HIGHPI PRO

The new case from the HiPi.io team

- Rapid tool-free assembly and disassembly
- Large internal volume for HATs
- Compatible with Pi 2/3/4
- Multiple lid options
- Passive & Active Cooling options
- Secure microSD card cover
- VESA mount support
- Molding-configurable output ports customizable for volume orders
- Printed logo for your branding

Available at these great Pi stores:

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

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

PiKVM

Remote control **redefined**

Manage your servers or PCs remotely!

Powered by Raspberry Pi

PiKVM V4 Mini

Small, cost-effective, and powerful!

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

PiKVM V4 Plus

The most feature-rich edition

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

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

Available at the main Raspberry Pi resellers

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

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

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

HiPi.io

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

List of official resellers by country: