Get annoyed **by an AI duck alarm clock**

Learn electronics **with Raspberry Pi Pico**

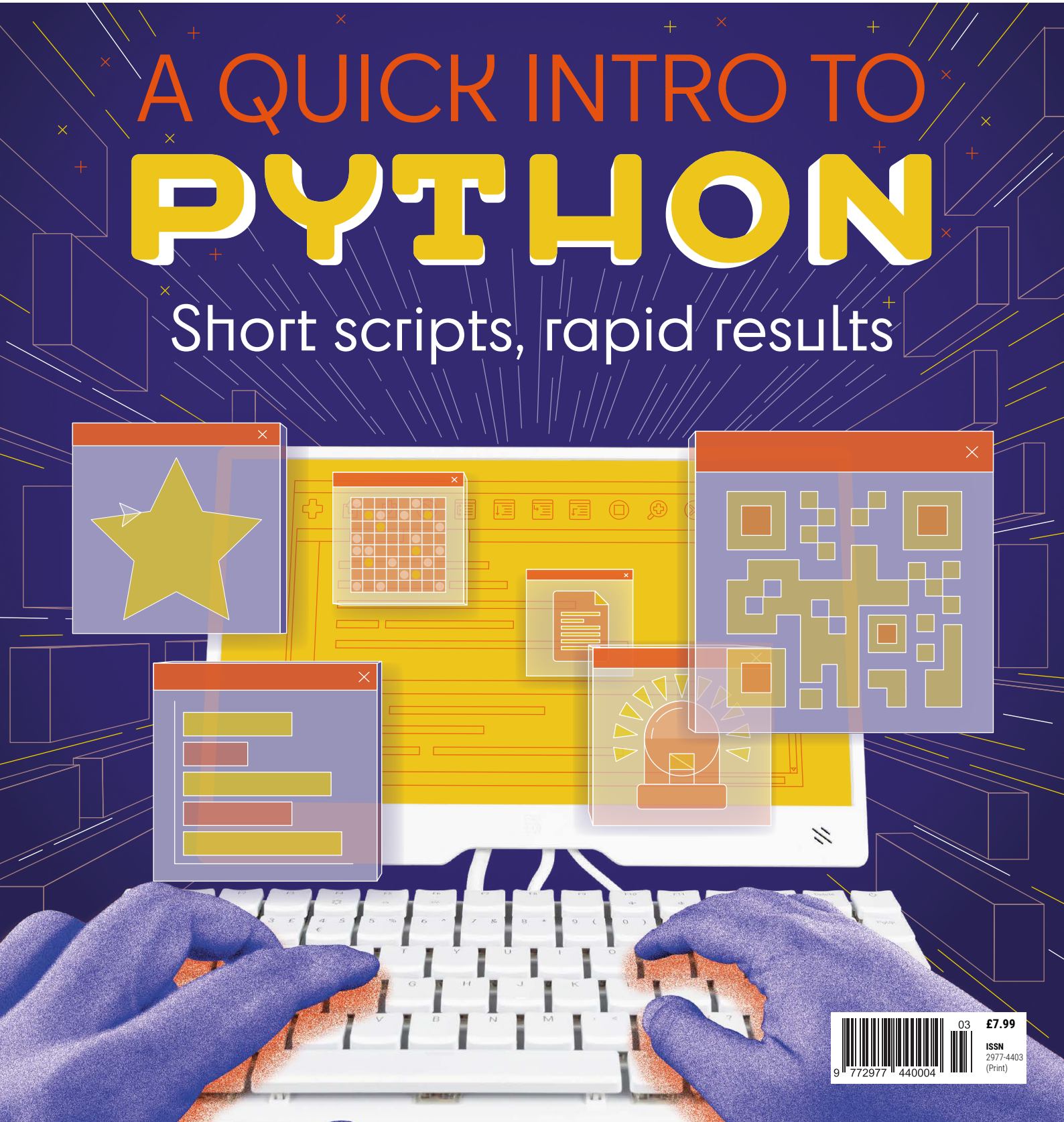
Exploring **only the best e-ink displays**

Raspberry Pi

Official Magazine
#163 | March 2026

A QUICK INTRO TO PYTHON

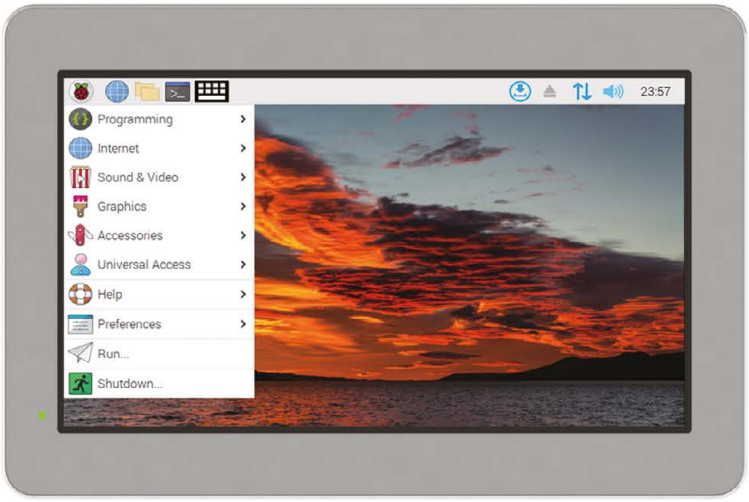
Short scripts, rapid results



03 £7.99
ISSN 2977-4403 (Print)

Industrial Raspberry Pi

ComfilePi



UL LISTED

Powered by Raspberry Pi

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

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

Visit www.comfiletech.com

COMFILE TECHNOLOGY

Welcome to Raspberry Pi Official Magazine

Editor
Lucy Hattersley
Lucy has just been wassailing in the park and at least the trees appreciate her singing voice.

rpimag.co

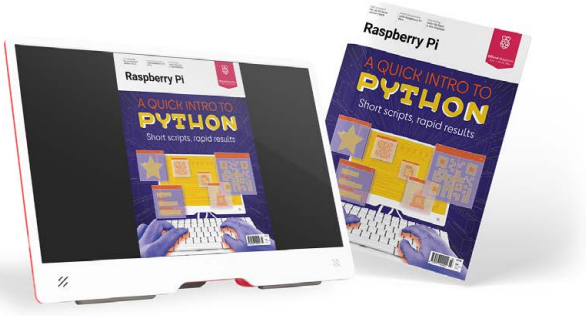
It's a big month for us at *Raspberry Pi Official Magazine*. This issue, we dig right into our roots as a community magazine that helps the next generation develop coding skills (and hopefully improve ours on the way).

Python is our language of choice. Sean McManus has given us his Quick Intro to Python. It's a great feature that gets you up and running with practical hacks for our favourite programming language. Sean will take you from Hello World, to lists, functions, and objects, up to creating simple programs that craft art, create QR codes, and control turtle robots. You even get to edit PDFs from Python. There's a lot of fun to be found in coding.

Learning to code was a life-changing experience for me. It helped me break problems down and understand how to put computers to work.

While vibe coding may be all the rage, I still believe you need to know the fundamentals to understand what's going on and to put code to work in the real world. That's what this magazine is all about.

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:

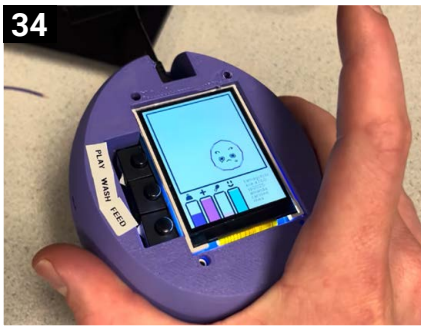
www.pcbway.com

service@pcbway.com

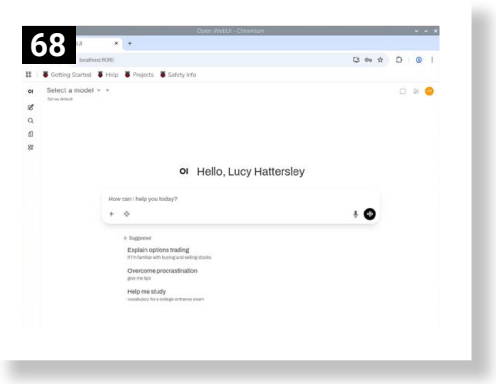
Contents

World of Raspberry Pi

Project Showcase

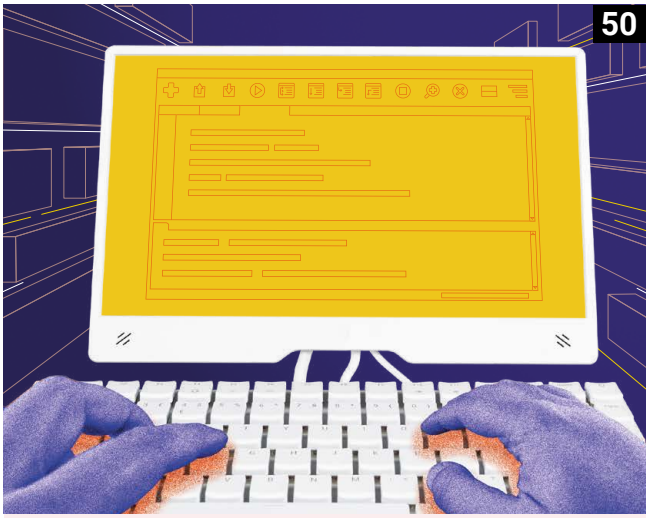


Raspberry Pi (ISSN No: 2977-4403, USPS No: 25-672) is published 12 times per year by Raspberry Pi Ltd, and distributed in the USA by Asendia USA, 701 Ashland Ave, Folcroft PA. Application to Mail at Periodicals Postage. Prices is pending at Philadelphia, PA and additional mailing offices. POSTMASTER: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



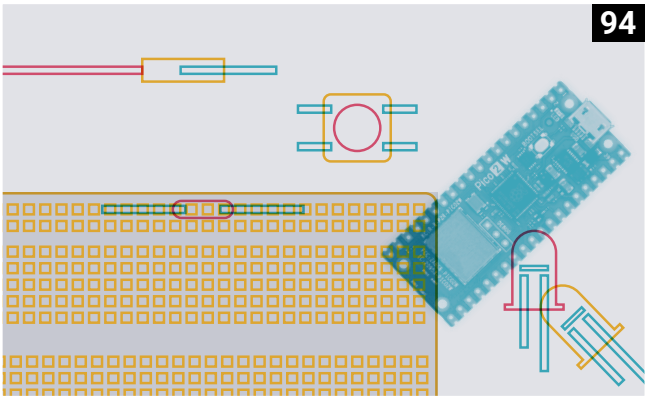
Top Projects

Feature



Tutorials

Feature



Reviews

Raspberry Pi Community

Competition

Raspberry Pi releases high-quality Flash Drive

Flash Drive available now from $30: a high-quality essential accessory.

By **Lucy Hattersley**

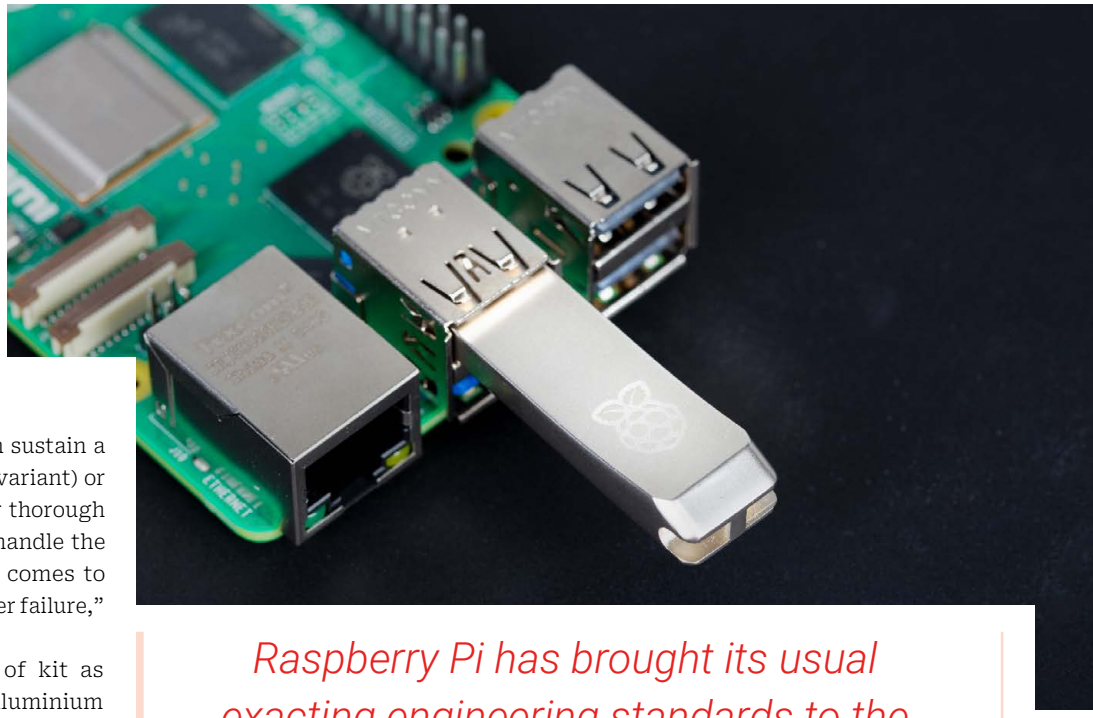
▲ Raspberry Pi's Flash Drive is available in 128GB and 256GB variants

Raspberry Pi has released another new product for 2026, the humble (yet powerful) Flash Drive.

Available in 128GB and 256GB variants, the brand new USB 3.0, USB-A device starts at $30.

Helen Lynn, Director of Communications at Raspberry Pi says: "For basics like these, it's tempting to reach for the cheapest thing on Amazon or whatever you find in your local supermarket, but you can easily end up with a device that has sluggish read and write speeds, fragile casing, or – worst of all – far less storage capacity than it claims."

Raspberry Pi has brought its usual exacting engineering standards to the new accessory, so you can be sure that this is a quality piece of kit.



Sustained performance

Raspberry Pi's Flash Drive can sustain a write speed of 75MB/s (128GB variant) or 150MB/s (256GB variant). "Our thorough testing has made sure it can handle the demands of real life when it comes to sudden disconnection and power failure," writes Lynn.

It's a nice-looking piece of kit as well, with an ergonomic all-aluminium enclosure that is "easy to grasp and almost impossible to break".

The Flash Drive employs a small reservation of pseudo-single-level cell (pSLC) cache to improve performance under burst write workloads. In the background, any writes that were allocated in pSLC are streamed out to the higher-density, but slower, quad-level cell (QLC) flash. "There are significant advantages to doing this for short periods," explains Lynn. "The sequential write speed can be almost as fast as USB 3.0 will go."

This cache does, however, make benchmarking challenging. For this reason, the USB 3.0 performance figures Raspberry Pi quotes are sustained figures, where writes are measured when the cache is forced to do write-through due to the volume of writes already committed, and reads are measured with the cache empty.

It goes without saying that whatever internal storage arrangement is used, it must be robust against surprise removal or power failure. Raspberry Pi has verified that its new Flash Drive meets this requirement over tens of thousands of random power cycles while running intermittently intensive I/O workloads.

Raspberry Pi has brought its usual exacting engineering standards to the new accessory

▲ The Flash Drive connects to the USB 3.0, USB-A connection found on most Raspberry Pi computers

Bonus features

In addition to being fast, we made sure that these drives support SSD-style SMART health reporting to help you to manage the device lifespan, as well as supporting TRIM operations. They will also autonomously enter low-power USB 3.0 states when idle.

The new Raspberry Pi Flash Drive (see **rpimag.co/flashdrive**) provides compact, portable storage with reliable performance for both 128GB and 256GB capacity options. Grab one from a Raspberry Pi Approved Reseller today.

More handy essentials from Raspberry Pi

Raspberry Pi's new flash storage drive joins a growing range of rigorously specified and robustly tested Raspberry Pi accessories designed to make your day-to-day computing life as friction-free as possible. Raspberry Pi SD Cards (**rpimag.co/sdcard**) and Raspberry Pi SSDs (**rpimag.co/ssd**) offer you a choice of storage solutions; the four-way Raspberry Pi USB 3 Hub (**rpimag.co/usbhub**) provides an excellent alternative to unsatisfactory price/quality compromises elsewhere; and Raspberry Pi Bumper (**rpimag.co/bumper**) is exactly what you need to protect the base and edges of your Raspberry Pi 5, without obstructing access to anything else.

Raspberry Pi Hacking Challenge made easier

RP2350 Hacking Challenge 2: less randomisation, more correlation.

By **Lucy Hattersley**

The RP2350 microcontroller

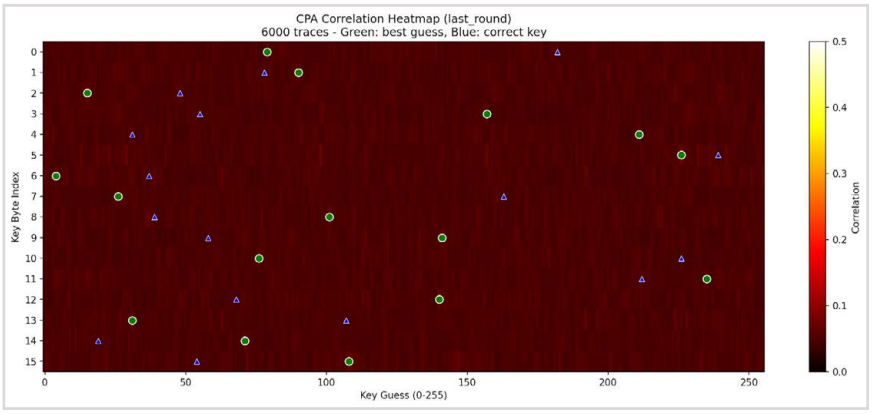
In June 2025 Raspberry Pi launched its second ever hacking challenge (rpimag.co/hackchallenge2). The challenge is to see if side-channel attacks can be implemented against the power-hardened Advanced Encryption Standard (AES) that underpins RP2350's secure boot (**rpimag.co/rp2350**).

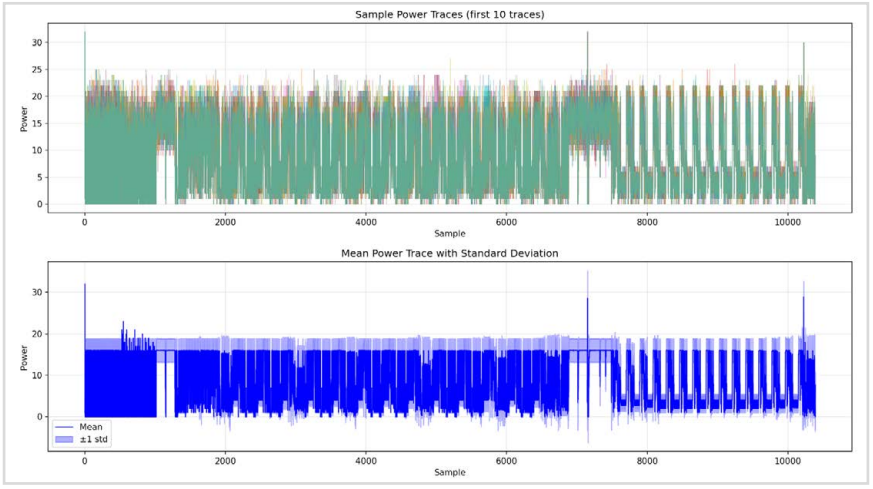
So far, the answer seems to be: "no". So Raspberry Pi has extended the deadline and removed one of the key requirements.

Stack smasher

Thomas Roth is a co-CEO of Hextree and host of YouTube's stacksmashing, writing for Raspberry Pi's blog, says: "So far, we don't have a winner, so we decided to evolve the challenge by removing one of the core defence-in-depth features: the randomisation of memory accesses.

"Our AES implementation was designed to withstand side-channel attacks by using multi-way secret sharing," writes Roth, "where sensitive values are split into random components that must be XORed





◀ An example of the analysis on the last round of hacks; Raspberry Pi could not identify a strong correlation between the power analysis and the correct key

▼ Example (cleaned-up) power traces collected from the target – the rounds are clearly visible

together and by randomly permuting the order of operations and data. We hope that even just the multi-way shares are enough to protect us against side-channel attacks; hence, we have decided to update our challenge.

Raspberry Pi has disabled the different randomisation features, such as Vector Permute (VPERM) and Bit Permute (BPERM), and removed the jitter to make collecting good measurements easier. "Your traces should align much more nicely now," says Roth. "We've also added a Unicorn-based emulation example, in the hope that people without fancy hardware setups can build virtual power-analysis attacks."

We decided to evolve the challenge

Entrants will need to demonstrate a successful attack on Raspberry Pi's AES implementation without the randomisation to win. Raspberry Pi has extended the deadline to 30 Apr 2026. The prize remains unchanged at $20,000.

Visit the Hacking Challenge 2 GitHub repo (**rpimag.co/hackchallenge2git**) in order to view the updated challenge software and have a go.

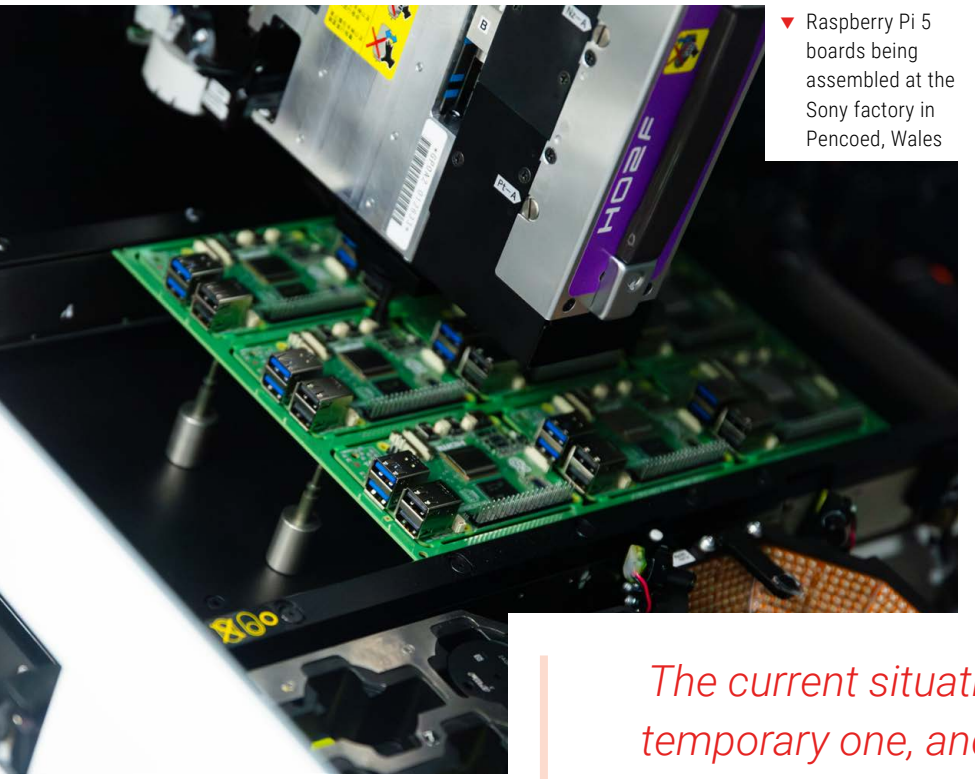
I didn't understand any of this?

The secure boot protection of firmware on RP2350 relies on AES – the Advanced Encryption Standard – to decrypt the firmware from external flash into the on-chip, static random-access memory (SRAM). AES in itself is considered very secure; however, a lot of software and hardware implementations are susceptible to so-called side-channel attacks. By recording and analysing hundreds of thousands (or even millions) of power traces on the chip, attackers might be able to recover the encryption key.

To protect against this, Raspberry Pi worked with some very smart folks to build an AES implementation that is hardened against these kinds of attacks. Raspberry Pi is putting it to the test by offering a bounty to the first person who successfully manages to attack our AES via side channels.

Raspberry Pi prices go up

AI infrastructure continues to push up prices. By **Lucy Hattersley**



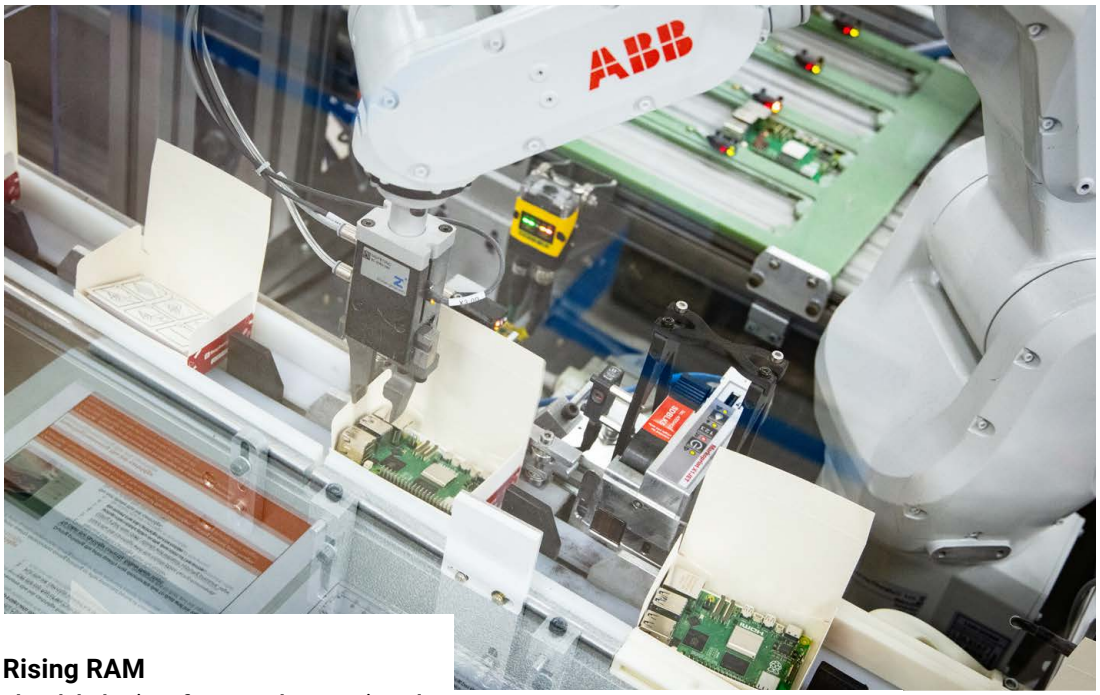
▼ Raspberry Pi 5 boards being assembled at the Sony factory in Pencoed, Wales

Raspberry Pi has announced more memory-driven price increases to its range of single-board computers.

Writing on the Raspberry Pi blog, Eben Upton, CEO and Co-Founder of Raspberry Pi blamed the rises on the continued rise in the cost of LPDDR4. These price rises were driven by "an unprecedented rise in the cost of LPDDR4 memory, thanks to competition for memory fab capacity from the AI infrastructure roll-out."

In December 2025, Raspberry Pi introduced a new, low-cost, Raspberry Pi 5 with just 1GB of RAM. Priced at $45, the new model was designed to offset the "unprecedented rise in the cost of LPDDR4 memory". The price of the entry-level Raspberry Pi 1GB remains unchanged.

> *The current situation is ultimately a temporary one, and we look forward to unwinding these price increases once it abates*



- The price of the entry-level Raspberry Pi 5 1GB is unaffected, along with many older models of Raspberry Pi computers
- A global increase in the price of LPDDR4 is being caused by the AI infrastructure rollout

Rising RAM

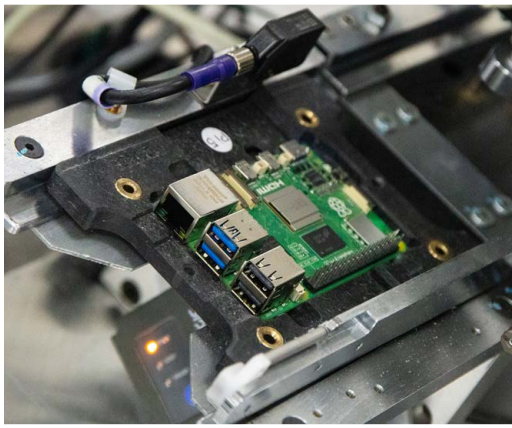
The global price of LPDDR4 has continued to increase. Research by Counterpoint shows that some memory prices have more than doubled over the last quarter (**rpimag.co/memprices**).

As a result, Upton explained that Raspberry Pi needed to make these "further increases" to its pricing. This will affect all Raspberry Pi 4 and 5, and Compute Module 4 and 5 products that have 2GB or more of memory.

Alongside the entry-level Raspberry Pi 5 with 1GB RAM, some other models are protected from the increases. The all-in-one Raspberry Pi 400 computer is also maintaining its price, at just $60. And the Raspberry Pi 4 1GB model remains at $35.

These low-cost entry-level models remain an affordable introduction to Raspberry Pi computing. As do Raspberry Pi Zero and Raspberry Pi 3. "We don't anticipate any changes to the price of Raspberry Pi Zero, Raspberry Pi 3, and other older products, as we currently hold several years' inventory of the LPDDR2 memory that they use," says Upton.

"2026 looks likely to be another challenging year for memory pricing," says Raspberry Pi's CEO, "but we are working hard to limit the impact. We've said it before, but we'll say it again: the current situation is ultimately a temporary one, and we look forward to unwinding these price increases once it abates".



Price rise

Memory density	Price increase
1GB	-
2GB	$10
4GB	$15
8GB	$30
16GB	$60

Sixfab wins CES Best of Innovation Award

Raspberry Pi-powered ALPON X5 AI gateway wins the show's highest honour.

By **Bianca McLean**

We were at CES too; our very own Chris Boross and Samantha Snyder were on hand to celebrate Sixfab's Best of Innovation win

Our friends over at Sixfab, one of our long-standing Raspberry Pi Design Partners, are kicking off 2026 in serious style. The company headed to Las Vegas in January 2026 to pick up the CES Best of Innovation Award (**rpimag.co/cesaward**).

CES's Best of Innovation is the show's highest honour. Sixfab won for its new AI gateway: ALPON X5 AI (**rpimag.co/alponx5**). The device is powered by Raspberry Pi Compute Module 5 and features DEEPX's DX-M1 accelerator chip.

If you've ever needed to connect your Raspberry Pi to... well, just about anything, you've probably come across Sixfab's gear. It has been making clever connectivity solutions for years: LTE and 5G modems, IoT shields, and the sort of industrial-ready bits and pieces that keep real-world deployments ticking without anyone having to duct-tape a USB dongle to a radio mast. (Not that we're judging.)

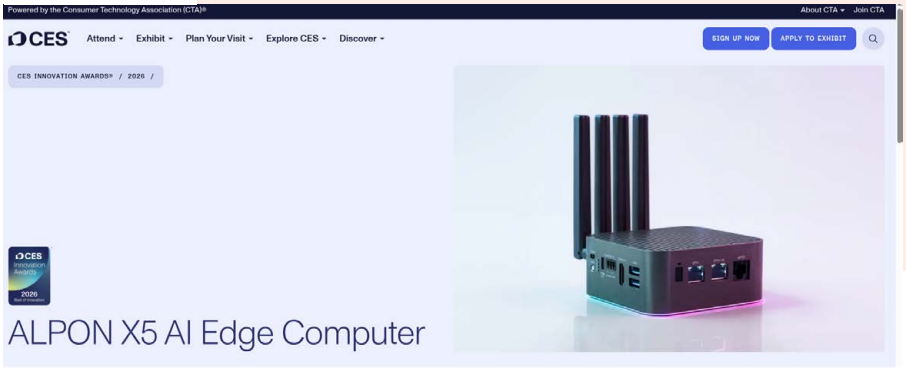
NEW!

CES Innovation Awards 2026 Best of Innovation

◀ ALPON X5 AI is a rugged, fanless edge AI computer powered by Raspberry Pi Compute Module 5 and the DEEPX DX-M1

▼ The CES Best of Innovation prize is highly coveted

Its award-winning ALPON X5 AI is an edge AI gateway designed for rugged deployments in the field – the kind where you need a full stack of compute, connectivity, and machine learning in one compact box. With Compute Module 5 and DX-M1 silicon under the hood, it's built for low-power, high-performance inferencing at the edge, making it a strong fit for manufacturers rolling out smart kiosks, robotics, industrial automation solutions, or even that one remote sensor you've long since forgotten about somewhere on a hill in Wales.



We would like to extend a huge congratulations to the whole Sixfab team

Sixfab's ALPON X5 AI promises:

- High-efficiency AI acceleration thanks to DEEPX's DX-M1 chip
- Industrial-grade cellular and wired connectivity options
- Drop-in support for Raspberry Pi-based workflows and deployments
- A clean, developer-friendly software stack

Raspberry Pi was at CES too; our very own Chris Boross and Samantha Snyder were on hand to celebrate Sixfab's Best of Innovation win.

We would like to extend a huge congratulations to the whole Sixfab team. They've been doing great stuff with Raspberry Pi for years, and it's lovely to see them get this very shiny bit of recognition.

oDuckberry

A novelty alarm clock has been turned into something ‘pro-duck-tive’.

By **David Crookes**

Maker

Osmund Grov

Osmund is a Norwegian technologist who works professionally with software and system architecture. He spends his free time building playful hardware projects that combine AI, electronics, and humour.

rpimag.co/aiduckgit

Some of the best ideas in life can feel rather ‘quackers’, and that’s certainly the case with Osmund Grov’s project. He’s built an AI assistant housed in an old duck alarm clock and, when words are uttered, the plastic waterfowl’s mouth moves – producing something a tad “terrifying”, Redditors joked after Osmund shared his work on the discussion platform.

“I was inspired by a couple of fairly boring home assistants from Amazon and Google,” Osmund explains. “I wanted something with a bit more personality and briefly considered 3D-printing a custom enclosure. Then I remembered an annoying duck alarm clock I’d been given for a birthday. At that point, it just felt like the obvious choice.”

The project evolved organically. “From the start, I wanted to interact with the AI in the same way I do with regular home assistants, using voice,” he says. “I also wanted the duck’s beak to move in sync with the AI’s responses.” To achieve this, he tore out the duck’s electronics and replaced them with a Raspberry Pi 4 computer, a custom PCB, and a cheap SG90 micro servo motor which allows for small-angle movements.

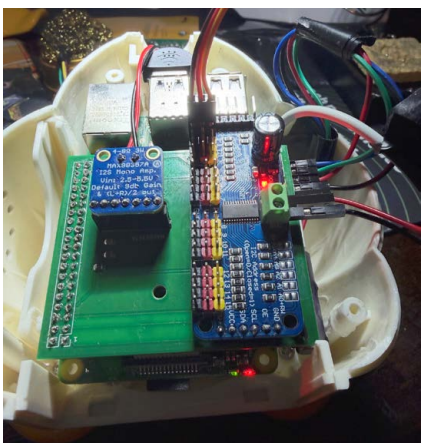
“I originally planned to add head movement as well, but there simply wasn’t enough room inside the duck for more servos and I didn’t want to heavily modify the duck itself,” Osmund adds. “Some ideas were also dropped simply due to physical constraints.”

Quacking on

Osmund felt Raspberry Pi 4 was a perfect fit for this project. “I already had a Raspberry Pi computer and a breadboard from earlier small projects,” he says. “Raspberry Pi 4 is relatively inexpensive and has enough memory and CPU power to run the software while also driving the servo and LEDs.”

Importantly, it meant he could power everything using a small USB-C PD trigger, supplying enough power to both Raspberry Pi and the servo controller. “Most of the other components were chosen based on availability and how well they would fit inside the duck,”

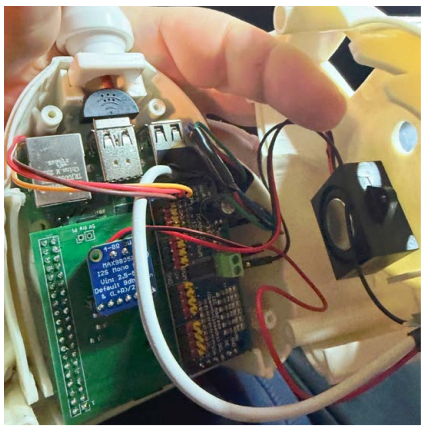
▼ The project includes a web interface that allows changes to the duck's personality and a choice of male and female voices



01. A blue LED appears when Raspberry Pi boots and the software begins listening for a wake word

02. A red LED means the duck is speaking, green means it's listening, blinking yellow/purple means it's thinking

▼ Beak movement is synchronised with the duck's speech. Volume levels can be adjusted



Osmund says. Those components included an inexpensive Adafruit I2S 3W Class D Amplifier Breakout, a small speaker, and a small USB microphone.

> *I remembered an annoying duck alarm clock I'd been given for a birthday*

More feathered friends

The biggest challenge was the number of GPIO pins needed for the audio, LEDs, and the servo controller. As such, Osmund says the first iteration had a spaghetti of cable inside the duck. "Eventually, I realised I needed a custom HAT to get proper control, so I designed one in KiCad," he adds. "The sound card and servo controller are soldered directly onto the HAT, which also includes LED pins."

Osmund used the wake word detection engine Porcupine (**rpimag.co/porcupine**). He also used Microsoft Azure Speech for the speech-to-text and text-to-speech handling, and OpenAI GPT models to generate the AI responses. As it stands, this causes a delay. "It can take up to five seconds from asking a question to getting a response, so I may use OpenAI's speech-to-speech capabilities instead," he says. That small issue aside, the project works swimmingly well.

It's proven so effective that Osmund has purchased two more ducks on eBay and he now has plans to make all three talk to each other and maybe even speak more languages. He's also experimenting with short- and long-term memory stored in the multi-model database Azure Cosmos DB, and he wants the duck to have cross-session memory. As far as quirky AI projects go, this one definitely fits the bill.

Quick FACTS

- Osmund got the duck for a birthday in the 1990s
- He stripped it of all of its parts
- It now works like a normal AI assistant
- Users can have two-way conversations with it
- Beak movement is controlled by an SG90 servo

PicoCPC

Lord Sugar's Amstrad CPC range of 8-bit computers have been sweetened with the help of Raspberry Pi Pico.

By **David Crookes**

Maker

Stéphane Plantard

Stéphane has been working as an artificial intelligence engineer since 1998. He worked on *World of Warcraft* server emulators for ten years and loves to create hardware for old computers.

rpimag.co/picocpc

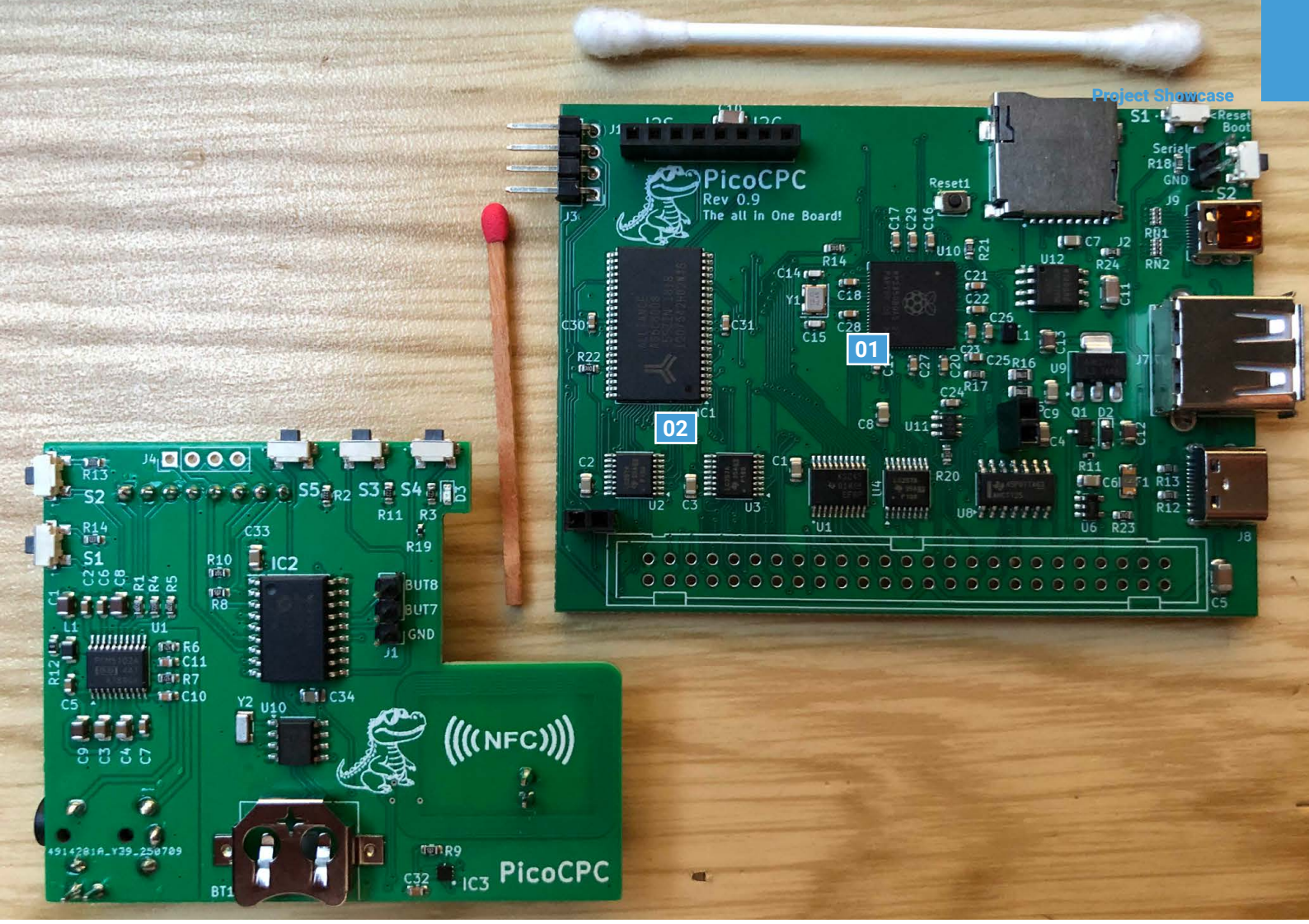
In the computer playground wars of the 1980s, children would spend hours extolling the virtues of either the ZX Spectrum or Commodore 64, depending on which side of the fence they sat. As the arguments raged, those who owned an Amstrad CPC machine tended to watch from the sidelines but, deep down, they knew their chosen machine could very much hold its own.

As time has gone on, there has been a growing appreciation for the 8-bit computers created by Lord Sugar's company. In recent years, a small but nonetheless loyal community has been creating a string of games that push the machines to their limits (check out *Pinball Dreams* as proof). They've also been producing hardware to take the CPC to the next level. A good example of this is the M4 board, which not only enables wireless LAN but allows an SD card to be used for storage.

Joining the hardware roster is the PicoCPC, a new multi-purpose add-on that can be used across the CPC range, including the Plus machines launched in 1990. It's impossible to sum up its capabilities in one sentence. Rather, this neat little device – built around a Raspberry Pi Pico 2 – provides a host of benefits for machines that maxed out at 128kB of memory.

For starters, the PicoCPC extends the memory up to 1024kB. It also emulates a floppy disk controller, allows the use of up to 16 emulated ROMs (for instant access to the likes of the Protext word processor and alternative operating systems such

> *Users won't have to take care of deteriorated, old floppy disks*



as SymbOS and FutureOS) and enables cartridge loading of software produced for the GX4000 console. It adds six-voice audio courtesy of PlayCity sound card emulation and there's also a clock. For CPC enthusiasts, it's fast becoming an essential upgrade.

Key to success

The idea emerged after Stéphane Plantard noticed a problem with many previous CPC add-ons. "I discovered there were a lot of expansions for Amstrad's early computers, but they were expensive, rare and, moreover, made with deprecated chips," he says. Over time, he became very familiar with the inner workings of the CPC. He produced an external Gotek drive for the CPC 664 and CPC 6128, as well as a device that powers the CPC and allows it to be connected to a TV instead of the bundled monitor.

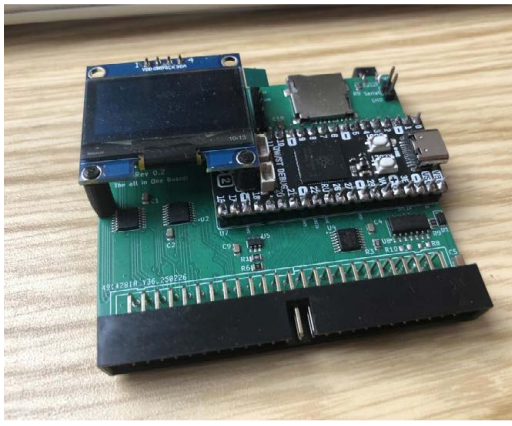
"Then my friend Freddy started a project called PicoMEM for old PCs." he says, of a device based on the Raspberry Pi RP2040 microcontroller that runs emulated 8-bit ISA boards on a real PC. "I thought I could do a similar card for the CPC, so I started to write some CPC-related code on the PicoMEM to test the SD card readings, which allowed me to become familiar with Raspberry Pi Pico. Freddy then pushed me to create my own card and I've produced three prototypes so far."

Stéphane says he approached the project according to the agile principles, which essentially means he sought to be flexible, open to change, and willing to work closely with the CPC community (there are twelve principles in total and they're part of the Agile Manifesto published in 2001). "It means I maintain a backlog of ideas and setup my epics and my sprints," he explains. "Having a strategy is a requirement when you work alone on such a big project."

01. The final PicoCPC model's PCB is going to have a Raspberry Pi RP2350B microcontroller soldered directly onto it

02. The PicoCPC ROM is used for small data transfers from the add-on to the CPC

▼ This is the prototype PicoCPC fitted with Raspberry Pi Pico 2 and a small display



In essence, Stéphane has been working in short development cycle bursts, or "sprints", that tend to stretch to a week. "It has allowed me to achieve some progress and reach goals; it's often enough to stay motivated," he notes. Given the project has taken a year so far, motivation has proven important. Stéphane has certainly seen the many benefits of basing the project around a Raspberry Pi Pico 2 and he says it's preferable to the alternatives he considered.

"The CPC generates a lot of signals that the card has to act upon, so an STM32 microcontroller would not be fast enough," he says, of one potential choice. He could have opted for an FPGA solution but this, too, posed problems. "It would have required expensive chips and I don't know how to work with them," he adds. "At the end of the day, the Pico 2 is cheap and fast. Its two cores and the PIOs make it a lot more suitable for this kind of task than any other available microcontroller. It uses one core dedicated to manage CPC I/O, a second core to run the emulations, and the PIO pilots are the multiplexors to get addresses and pull/push data from/to the CPC."

▲ The device has gone through a few revisions. Stéphane is also working on a cartridge with a SD card reader for Amstrad's GX4000 console

Quick upgrade

Some of Stéphane's ideas are still being worked on. "The PicoCPC does not support USB mice and joysticks yet. It's in the backlog but not done," he says. But, as it stands, the PicoCPC makes a big difference to the experience of using an Amstrad CPC and it also addresses a few practical issues.

"An ordinary CPC user would want to use an original computer to play old games from back in the day," Stéphane says. Since the tape-based CPC 464 is easier to find than the disk-based CPC 6128, the PicoCPC can be used to transform a 464 into a 6128 by adding an emulated floppy drive, more RAM, a later version of BASIC (v1.1), and the C3 mode which allows access to the extra RAM beyond the base 64kB. "All the CPC 6128 games and demos will then work on a CPC 464."

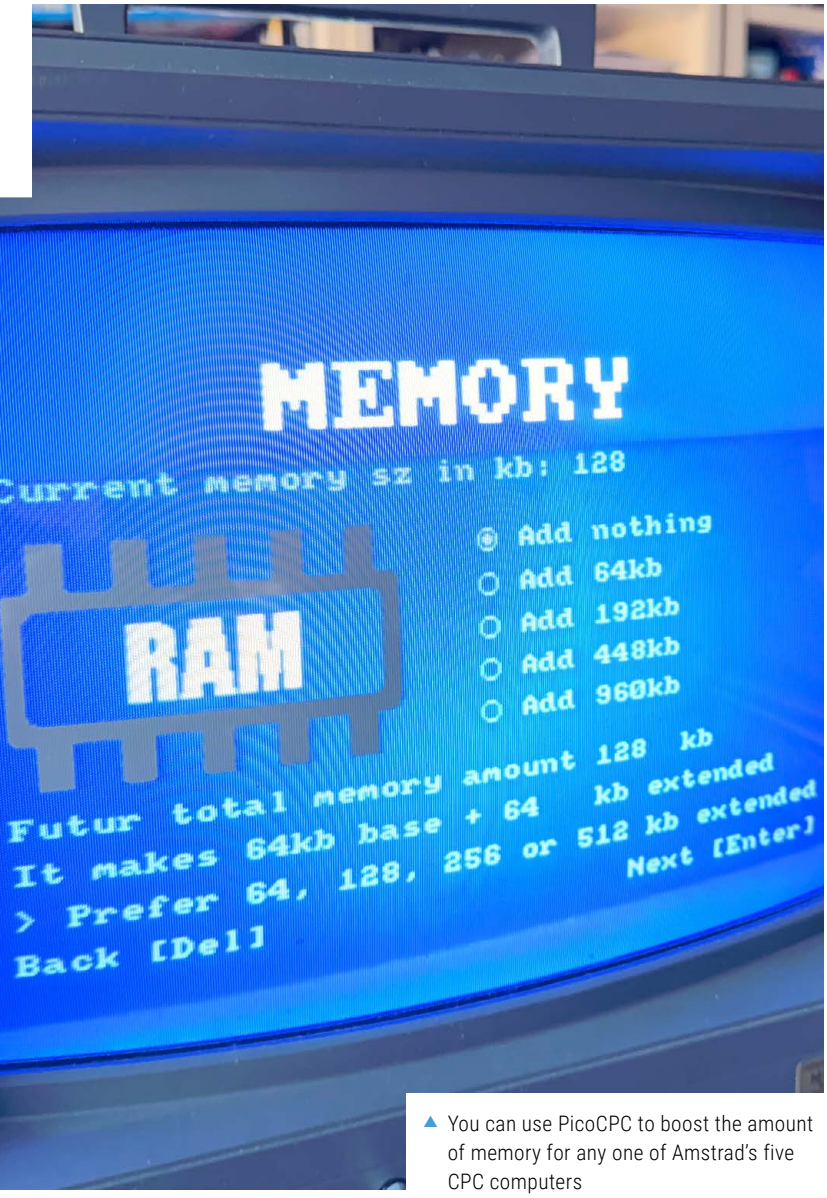
Likewise, on a CPC 6128, PicoCPC can override the internal floppy. "Users won't have to take care of deteriorated, old floppy disks," Stéphane says. But that's just touching the surface, he adds. "Many new games developed in recent years require 512kB of memory," he explains. "The Pico does not offer enough memory for that, which is why I added an SRAM chip on the card."

▲ The Amstrad CPC 464 made its debut in 1984. The PicoCPC plugs into the back **Credit: Bill Bertram, Wikimedia Commons, CC BY-SA 2.5**

He hopes the availability of PlayCity will push future software developers to use the expanded audio capabilities, and he's excited about other possibilities. "I plan to add proper hard-disk emulation, AdLib music card emulation, and new blank floppy disk creation," he reveals. "But the first plan for the PicoCPC is to make it available. I'm working with retailers in France, UK, and Spain, and hope it'll be on sale very soon."

Quick FACTS

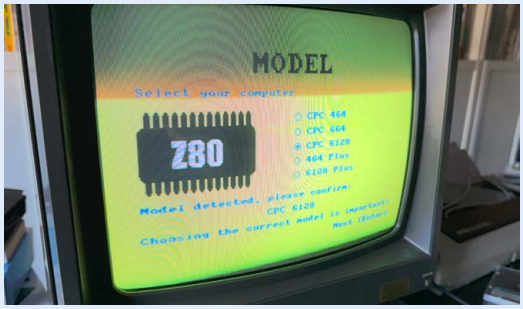
- PicoCPC significantly enhances Amstrad's CPC range of 8-bit computers
- It connects to the CPC's expansion port
- The add-on card boosts the computer's memory to 1MB
- It boasts a virtual floppy disk system
- It runs software from SD cards and USB sticks



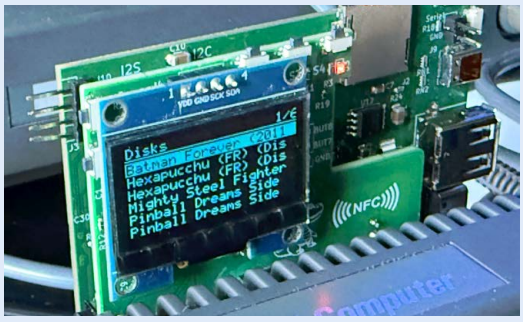
▲ You can use PicoCPC to boost the amount of memory for any one of Amstrad's five CPC computers

Boosting the CPC

1. Connect the PicoCPC to the expansion port of the CPC. The device requires you to insert a blank SD card and it will issue a warning on the built-in screen if it does not detect one. If no configuration file is detected, a Wizard begins.



2. The Wizard allows you to set up the device. You'll be prompted to select the CPC you're using and your preferred keyboard language. You'll also be asked if you want to add extra memory. Special commands to list, load, and manage files become available.



3. Files can be saved onto a USB stick and connected to the PicoCPC. Other features also become available, such as PlayCity's audio capabilities, if they're supported by software. On-device buttons are used to control the PicoCPC's OLED screen UI.

TVArgenta

A yen for the old country is behind an Argentinian retro TV build, learns **Rosie Hattersley**

Maker

Ricardo Sappia

Ricardo has a day job keeping vehicles safe from hackers and has a fondness for retro tech projects such as this one.

rpimag.co/ricardosappia

Cybersecurity and systems engineering expertise brought Ricardo Sappia to the attention of high-tech companies in Germany, where he has lived and worked for more than a decade. But his adopted home is a very long way from where he grew up in Argentina, and maintaining a connection with his South American roots isn't always easy. TVArgenta, which he designed around a Raspberry Pi 4, allows Ricardo to call up video clips, jingles, and photos from his childhood, as well as watching Argentinian TV shows. It's also a powerful way of connecting his German-born children with their father's heritage.

"TV commercials from the 1990s are more than just short ads: they are memory anchors – jingles, lines, and little cultural signals that smell like home," he explains. Ricardo says TVArgenta became "a cultural bridge" and a way to 'tune in' to the world he grew up in. It's also an easily adaptable project: from the outset, Ricardo intended the TV surfing experience he devised to be something other makers could run with and adapt to their own story.

Gathering momentum

Ricardo relishes Raspberry Pi "because it perfectly fits the intersection of creativity, accessibility, and engineering". He uses it to prototype ideas that connect the digital and physical worlds, from nostalgic devices to modern dashboards, in a way that feels both technical and artistic. TVArgenta is a case in point: it is an offline retro-styled television on which you can 'zap' through old adverts, "static, channel noise, and all". The idea can be replicated by anyone with a similar case of nostalgia to recreate a "local cultural experience".

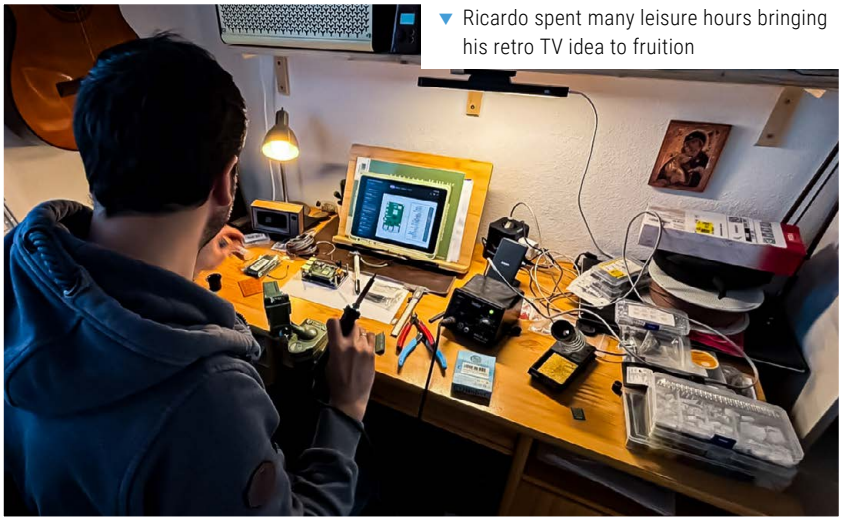
> *TVArgenta is an offline retro-styled television on which you can 'zap' through old adverts, "static, channel noise, and all"*

Ricardo's previous Raspberry Pi make was Dashi (**rpimag.co/dashi**), a fitness dashboard intended to cajole him to build up his activity levels as he recovered from a running injury and which used an

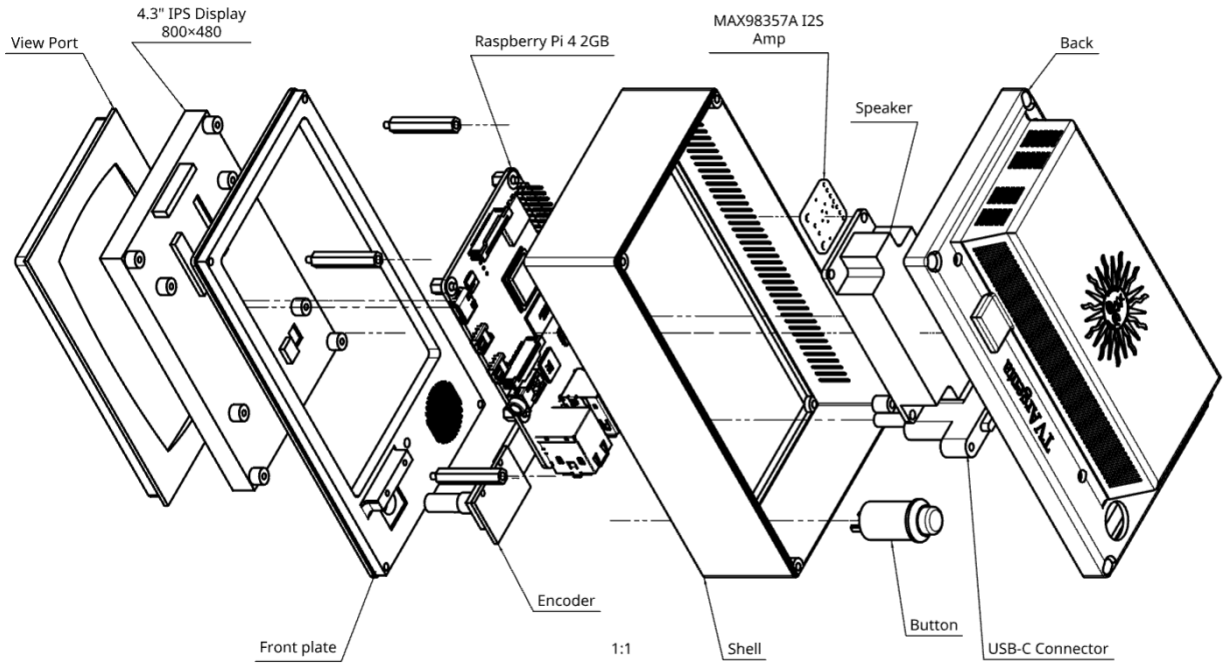
01. TVArgenta uses Raspberry Pi 4 and a Waveshare screen to display vintage Argentinian TV shows and adverts

02. A rotary encoder acts as an old-school dial for changing the channel

LLM (large language model) to generate motivational quotes delivered by an on-screen fox avatar. Dashi could also pull data from Strava and Garmin training data alongside live weather and a clock. He used a similar idea for TVArgenta, combining a Waveshare display with on-screen widgets that linked to video clips, adverts, and shows. He initially used Raspberry Pi Zero W, but switched to a 2GB Raspberry Pi 4 to take advantage of its GPIO connectors, encoder, and audio interface while also connecting it directly to a DSI touchscreen. This low-power setup nonetheless provides smooth video playback.



▼ Ricardo spent many leisure hours bringing his retro TV idea to fruition



▲ An exploded drawing of how the components fit together

Slick visuals

Ricardo describes Raspberry Pi as being at the heart of his project, running both the back end that manages content and metadata, and user interactions. There's a rotary dial to change channels, a power button, and an LED to show it's switched on, just like a classic TV set. Videos of old TV shows play smoothly in Chromium.

This aspect took the most finessing. Ricardo found there was a lot of trial and error getting the visuals to run smoothly and in sync, "especially the overlays with nineties-style effects". He eventually decided to handle the video and overlay rendering through Chromium, and take advantage of its direct GPU acceleration.

Choosing Raspberry Pi offered great versatility, kept project costs affordable, and ensured a polished experience. It meant "being able to go from prototype to polished product on the same hardware". Ricardo says it was also a huge advantage being able to draw on the Raspberry Pi community.

Raspberry Pi 4's physical size lent itself well to his mini TV ambition and the SD card meant he could keep it fully offline and self-contained. "Honestly, I can't imagine another platform I'd feel as comfortable working with when thinking about this kind of project."

TVArgenta's 3D printed case holds a 4.3in Waveshare capacitive touchscreen display and has a micro speaker and subwoofer with a 3W amp, rotary encoder, and power port. Under the hood, Raspberry Pi 4B is connected to perfboard circuits. These boards are ideal for the precise but isolated circuitry and electrical pads Ricardo wanted.

Having nailed the initial concept, Ricardo has recently developed it to include a radio mod and added a remote control (for full authenticity, we're certain he must already have lost it down the back of a tiny sofa) and overlays for that authentic 1990s vibe. You can watch a video of TVArgenta in action on YouTube: **rpimag.co/tvargentavid.**

Quick FACTS

- Ricardo settled in Wolfenbüttel, Germany a few years ago
- The town is home to the Jägermeister distillery as well as his employer
- He chose Raspberry Pi because it allows him "to be both technical and artistic"
- TVArgenta gives his kids a tangible connection to his Argentinian roots
- The idea can be easily adapted with footage from around the globe

Ricardo says TVArgenta became "a cultural bridge" and a way to 'tune in' to the world he grew up in

▼ TVArgenta has proved popular with his son, who also uses it to play retro games

Comfort viewing

1. Ricardo has made the STL files for the retro TV case available at **rpimag.co/tvargentastl**.

2. Raspberry Pi 4 connects to a Waveshare DSI touchscreen, speakers, and encoder and shows old TV programmes and adverts stored on an SD card.

3. Ricardo's code even displays an old-school test card when performing its final safety checks before 'transmission'.

Pico Chess Timer

Strategy games gain an extra frisson with this coder's Pico-based timer, finds **Rosie Hattersley**

Maker

Nirvaan Tandon

Teenage coder Nirvaan loves Raspberry Pi and has used it for various projects, including AI, robotics, and home automation.

rpimag.co/chesstimer

Maker Nirvaan Tandon's path to becoming a coder started early. Very early! He loved electronics from the age of four, which was when he got his first Snap Circuits kit. Nirvaan learned programming aged six and encountered Raspberry Pi a year later when he attended Coolest Projects with his family. He's built several Raspberry Pi-based projects since. In fact, this isn't even the talented 14-year-old's first appearance in *Raspberry Pi Official Magazine* (fka *The MagPi*). The Pico Chess Timer is designed to improve strategy games with friends and family. As he notes: "as we all know, sometimes games can take too long".

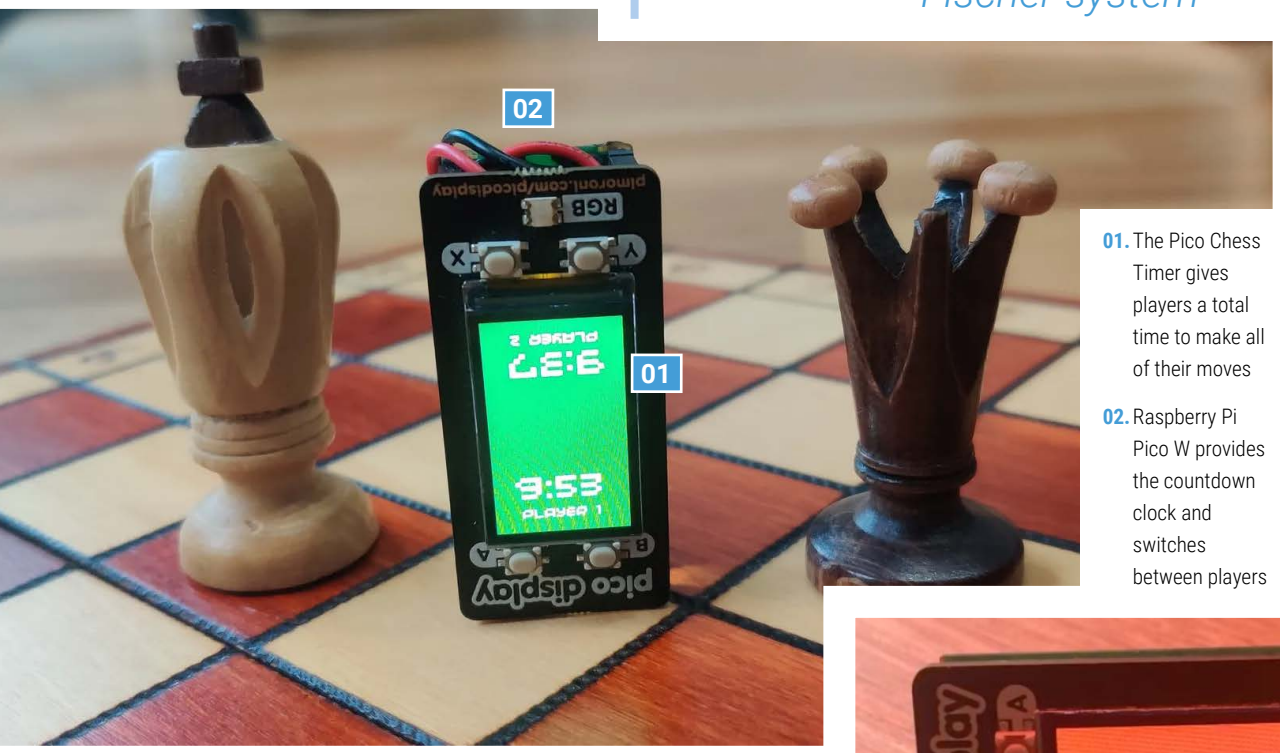
Exceptional diligence

Nirvaan is a lot more focused than the average teenager. Concerned that he might not stay on track with his studies, he came up with a Pico-based pomodoro timer to help him focus on his exams. Some of those studies have a Raspberry Pi element too: he's built several robots at school, as well as creating the smart light setup and a real-time energy price tracker. He eventually aims to automate his entire home and has already created a garage assistant with a map to show where there are available parking spaces.

The Pico Chess Timer evolved from the pomodoro timer idea as well as a reaction game he wrote. Nirvaan knew how to use the PicoGraphics library to create a user interface from previous projects. He used the same Pico Display Pack but rewrote the MicroPython code for two-player control and using Pico W to exploit its wireless connectivity. His aim was to create a timer similar to the one on **chess.com**, providing a no-arguments countdown so each player knows exactly how long they have to plan and execute a strategic move.

The Pico W was ideal because it has low power requirements and can be battery-powered (Nirvaan attached a LiPo battery using a SHIM – an idea he picked up from a robot build) and only as large as one of

The time allowance can be easily adjusted using the same incremental setup chess players use under the Fischer system



01. The Pico Chess Timer gives players a total time to make all of their moves

02. Raspberry Pi Pico W provides the countdown clock and switches between players

the king playing pieces. Setting up a chess game therefore requires little extra time or effort. Nirvaan also observes that Pico works well with MicroPython and can be used with many accessories.

Players just press the 'start' button to begin a countdown. The time allowance can be easily adjusted using the same incremental setup chess players use under the Fischer system. A second button press switches the gameplay to the other player. "Raspberry Pi Pico controls the entire project, running the program to set the time, count down the clock, and listen for button presses to switch turns."

▶ Nirvaan created a pomodoro timer to keep his exam studies on track

Ironing out kinks

Although this was a fairly simple build given Nirvaan's earlier projects, some aspects required finessing. In particular, getting the buttons to switch players accurately. He found this the hardest part as he needed to make sure it didn't accidentally trigger twice. "I had to use a loop checking every 0.01s which updates the time and checks for button pushes, then add a 20ms debounce to it." Doing this fixed the issue.

Getting the battery setup right was important too. Nirvaan found it very convenient being able to solder the LiPo SHIM to Pico so he could attach a LiPo battery. The SHIM added an on/off button and a JST connector and had its own charging circuit. He began with a large battery which he attached to the bottom, but Nirvaan soon realised a smaller 150mAh battery would be more suitable and could fit snugly on the Pico. Better yet, "soldering the SHIM flush with the bottom of the headers meant I could attach the Pico Display on top, making a compact package."

Conclusion

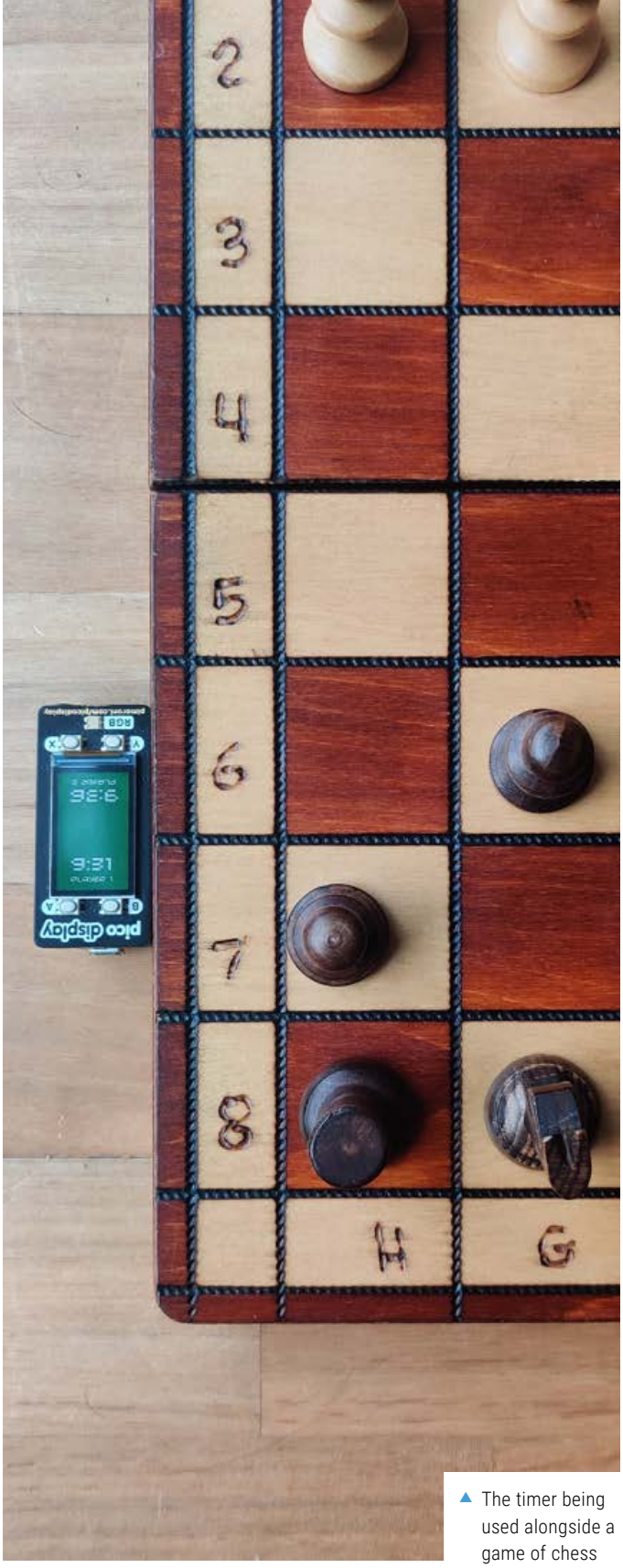
Nirvaan's confidence in coding and tackling original projects is pretty impressive. He's already assessed what he learned from the approximately £30 project: knowledge of dynamic positioning and more experience with Pico Graphics were technical advances for him, while the pragmatist in him realises that keeping things simple is often best. He advises other would-be makers to "experiment, play around and keep trying. Simple is best – don't overcomplicate things or repeat code too much."

Nirvaan himself has an ongoing scorebook against his younger brother and, with versions of the Pico Chess Timer on his side, might well be extending friendly sibling rivalry to Scrabble and other board games, or an against-the-clock Rubik's Cube challenge!

▲ The two-player timer can be set to add extra seconds after each move, just like the Fischer chess system

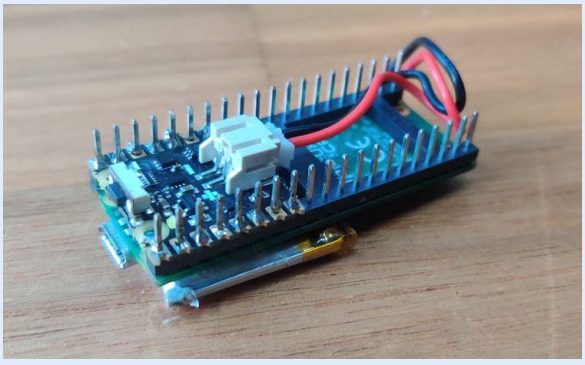
Quick FACTS

- Nirvaan has built up a collection of ten Raspberry Pi boards
- He plans to set up a Raspberry Pi club at school
- He tracks game results against his brother using Home Assistant
- Nirvaan wants to automatically import scores from his chess timer
- He plans to use Lichess integration to pull in timer data

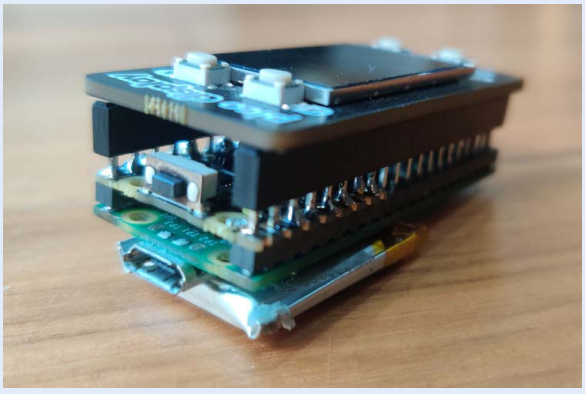


▲ The timer being used alongside a game of chess

Time trials



1. The Chess Timer is based around a Pico W plus a LiPo battery with SHIM soldered to the base of the Pico to keep things compact.



2. Connect the battery using JST connectors, then attach the Pico Display and flash the MicroPython UF2.

3. Transfer the code from Nirvaan's GitHub (**rpimag.co/chesstimer**) to Pico, switch on, and set the countdown timer.

Set Marvin Free

Do you dare free a demon possessing a stuffed animal? **Rob Zwetsloot** wonders what's the worst that could happen

Maker

Sebastião Quintas

An engineer and researcher specialising in speech processing, currently working as a machine learning engineer after a stint in academia

rpimag.co/setmarvinfree

Despite the plethora of adorable talking dolls that have been available to children since at least the 1980s, talking dolls are still disconcerting to some folks; scientifically known as 'the heebie-jeebies'. This did not stop Sebastião Quintas while making Marvin.

Set Marvin Free is a "conversational doll that hosts a speech-based escape game," Sebastião tells us. "The project combines a speech interface with playful storytelling, transforming a regular doll into an interactive game master. Players progress through four different mini games that branch into 24 possible endings. The project runs entirely offline, using a Raspberry Pi 5, as well as a microphone and speaker."

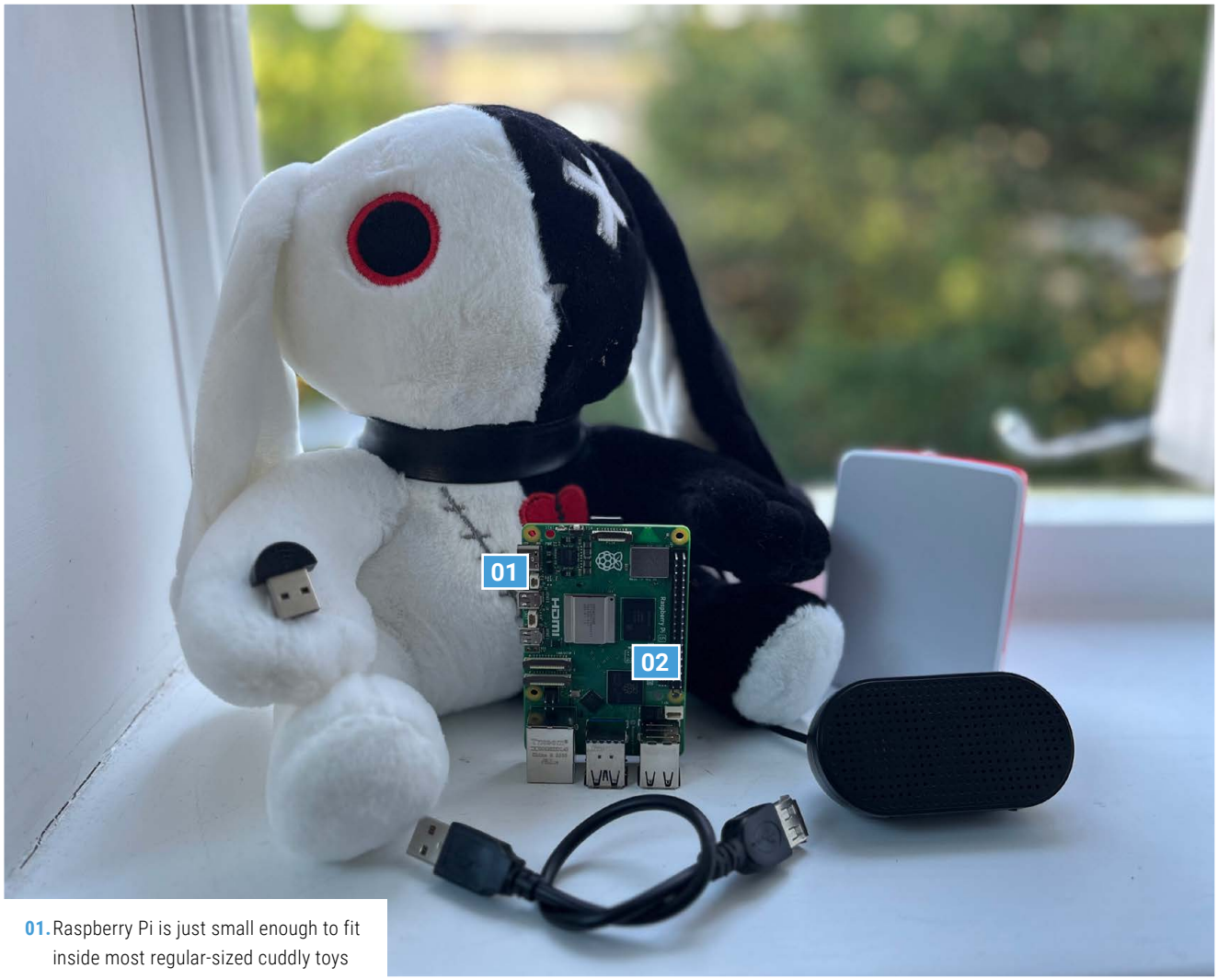
Sebastião was looking to explore using a completely vocal interface for a toy or game, with no visual cues to rely on, and to make it completely offline with everything running locally with no need for external APIs: "This greatly simplifies the infrastructure required, but at the cost of trying to fit everything on-device," he says.

Rasp-beary Pi

Landing on Raspberry Pi to power Marvin seemed like a natural conclusion.

"Raspberry Pi is the perfect fit for this kind of project: it's powerful enough to run a wake-word detector, a lightweight speech-recognition model, and the full NLU pipeline entirely offline, while its Linux-based environment makes setup straightforward," Sebastião explains. "I experimented with a Raspberry Pi Zero as well, but its underlying CPU architecture wasn't compatible with some of the deep-learning packages needed. Oh, and [Raspberry Pi's] compact format allows for it to be 'easily' concealed inside a stuffed animal!"

As Sebastião developed the project – adding the ability to process a wake word and tell a story – he decided to add more, feeling the idea at this stage was 'a little



01. Raspberry Pi is just small enough to fit inside most regular-sized cuddly toys

02. It is completely offline, with the models stored on Raspberry Pi

bit empty': "As a result, several more features were added, such as cracking jokes, giving random facts, riddles, tongue-twisters, etc. This culminated in the idea of adding a more ambitious feature through the means of a speech-based game that not only provided a different type of interaction, but also a fun venue for more storytelling."

These speech-based games resulted in their own extra challenges – mostly memorising details necessary for the games as they progress. "It's crazy how much work goes into making some of these seemingly simple games!" Sebastião notes.

These speech-based games resulted in their own extra challenges – mostly memorising details necessary for the games as they progress

Creepy cute

The basic story for the game is that Marvin is a demon inside the adorable toy that you can set free. This has not always elicited an expected reaction from players, although they did seem to enjoy it.

"The general reaction of the people that have watched and tested the demo was that it was engaging and fun," Sebastião says. "I also got some reactions saying that the voice was cute (it shouldn't be the case for a demon inside a doll!). Some people think that a few of the sound effects in the different endings can be a bit spooky. On a fun note, my partner was at times a bit tired of systematically listening to my exchanges with the toy whenever I had to debug something. There's only so much 'Hey Marvin' you can listen to!"

If you want to see Marvin in action, there's a full demo up on Sebastião's YouTube channel at **rpimag.co/marvinyt**. It probably won't release a demon into your house. Probably.

Quick FACTS

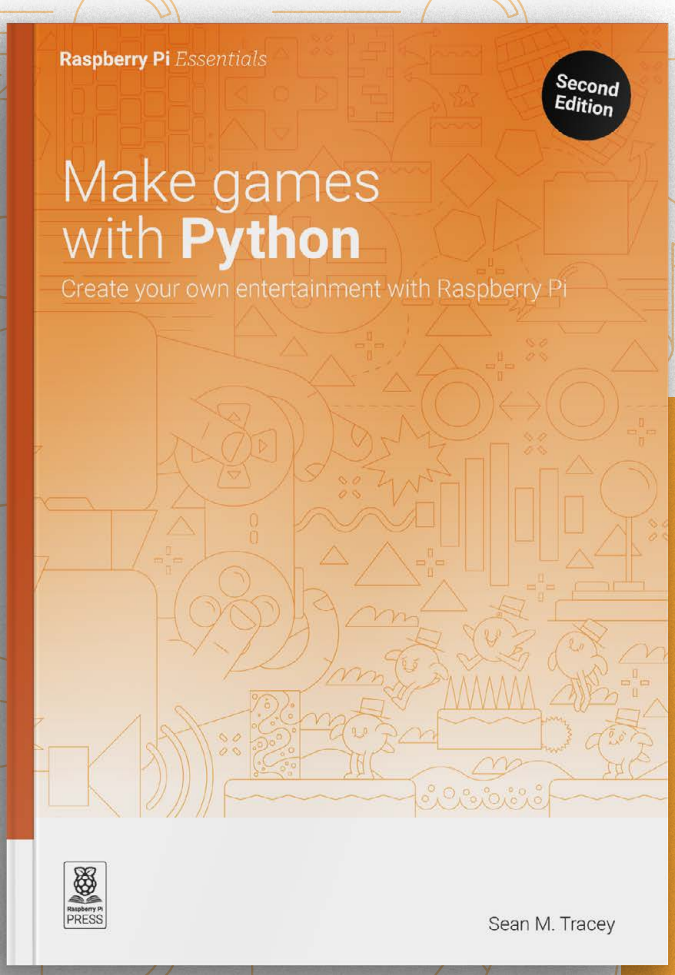
- A bear was used for an earlier prototype
- Sebastião did research with IRIT in Toulouse, France
- It's mostly software based, requiring only a mic and speaker
- You do not have to let Marvin free…
- … but where is the fun in that?

Making a demon toy

1. First you need to find an unassuming stuffed rabbit to be the unwitting host of a demon. It gives us *Danganronpa* vibes.

2. You will need other maker skills for a project like this – namely, sewing, to get all the gear you need inside.

3. Make sure you do tests first before powering your demonic toy.



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**

Cyber Pet Tumbler

This virtual pet can be thrown around its plastic case. **Rob Zwetsloot** tries to keep it alive

Maker

Rhea Goswami

A recent computer science and engineering graduate who now works in the space sector as a software engineer.

rpimag.co/tamatumbler

Pets are a big responsibility. Thanks to the magic of electronics and computer chips, they can still be a big responsibility, but with only the upfront cost involved. Tamagotchis were all the rage in the 1990s (if you were a kid like Rob back then, who had many knock-offs), and keeping the little pixelated critters alive was a lot of work. It turns out creating one is worthy of a degree in electrical and computer engineering, as recent graduate Rhea Goswami tells us.

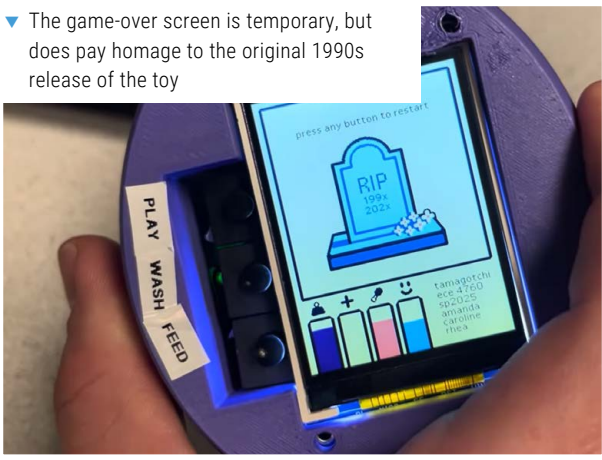
"We set out to create a handheld, portable game with fun graphics and an interesting premise," says Rhea. "We wanted to have the ability for the user to take this wonderful contraption 'on the go' with them, and there was no better example of an 'on-the-go' game that evokes nostalgia like the Tamagotchi."

There are four stats for the pet: weight, food (hunger), happiness, and health. Like the original toy, playing, feeding, and cleaning is required to maintain the correct levels on the meters.

"Of course, you may also interact with the pet by knocking it around its enclosure by physically tilting the device," Rhea says. "If your pet slams against any wall too hard, it will look dizzy and its health will decrease. Tilt the egg to slide your pet around and watch it squish up against the walls."

We don't think we'll be chucking this one haphazardly in a school bag then. It can also be switched off, and saves its state for the next time you turn it on, which also would have saved us some headaches as a kid.

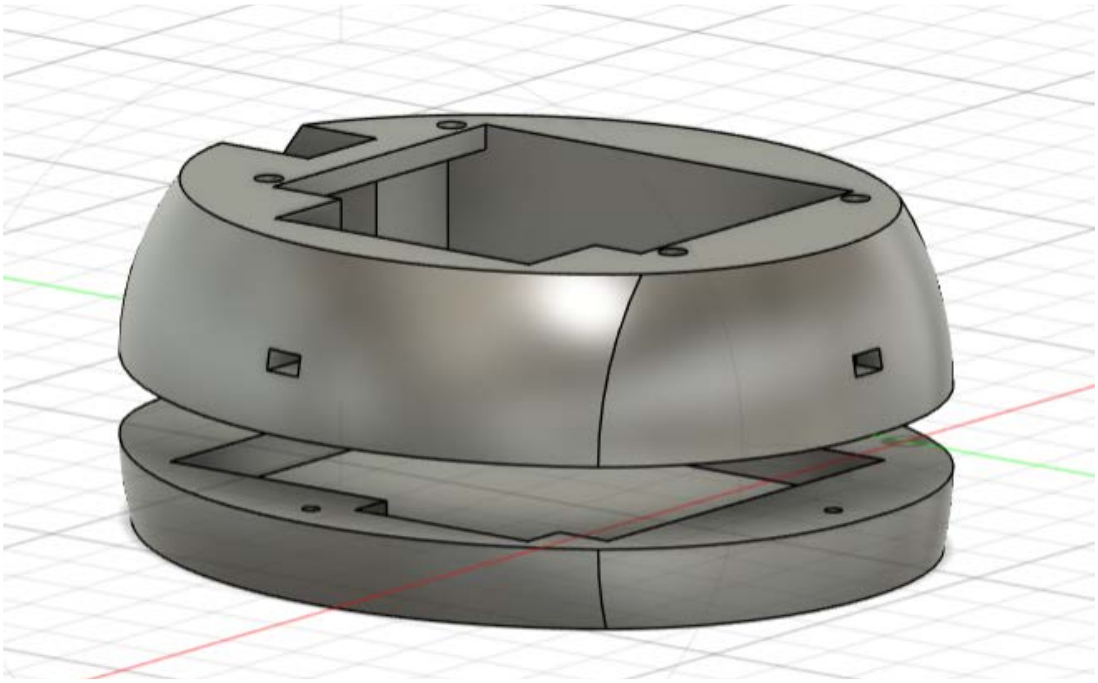
▼ The game-over screen is temporary, but does pay homage to the original 1990s release of the toy



01. The egg-shaped body replicates the Tamagotchi design

02. Using the various buttons will help keep your virtual pet alive

Of course, you may also interact with the pet by knocking it around its enclosure by physically tilting the device



The case was custom-designed for the hardware the team put together

Split parenting

The device was the result of a group project, and the concept played to the strengths of the team.

"The build process was a parallel process as one person solely worked on the graphics and simulation setup of the game, one person worked on understanding and coding up the soft-body physics simulation of the game, and the last person worked on the hardware design," Rhea reveals.

The team, which also consisted of other students Caroline Hohner and Amanda Huang, were given Raspberry Pi Picos by their professor for this specific project. They also included a small TFT screen, a gyroscope for movement sensing, some extra RAM in the form of FRAM to preserve the state during power off, along with the buttons and a custom 3D-printed case.

Quite some work went into the physics simulation, and the intricacies of this have been explained on the project page, along with other aspects of the build for which we don't quite have enough space here: **rpimag.co/tamatumbler**.

Actual play

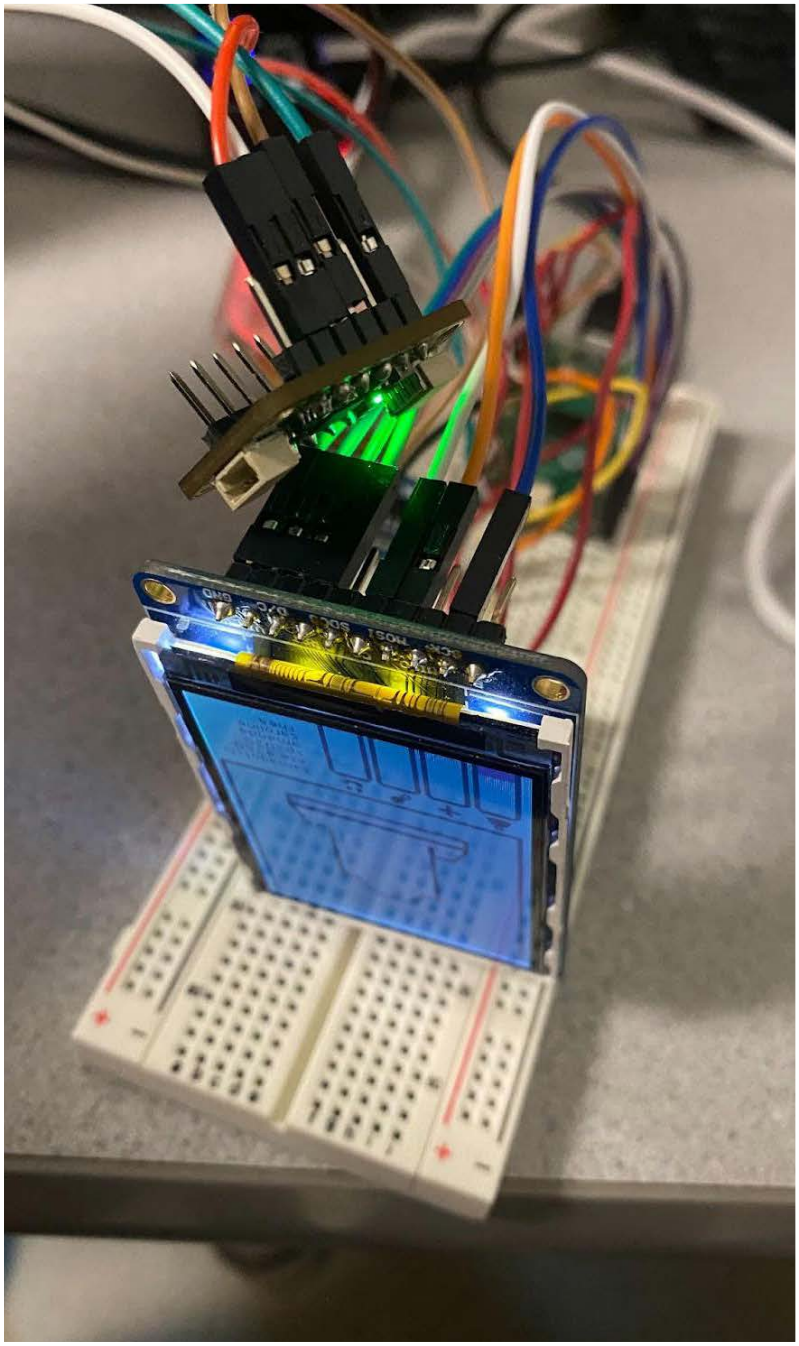
The team managed to pull off all the features they planned, although apparently the buttons don't always register if you press them too rapidly.

"[Reactions to the project] have been filled mostly with awe and wonder," Rhea adds. "Lots of people found our project very amusing and enjoyed playing with it. The project unexpectedly gained quite a bit of traction online, sparking discussion on Reddit and being featured on several smaller tech news sites.

Rhea has moved into working in the space tech sector since first talking to us about this project – hopefully she won't be shaking around astronauts to discover how their soft-body physics work.

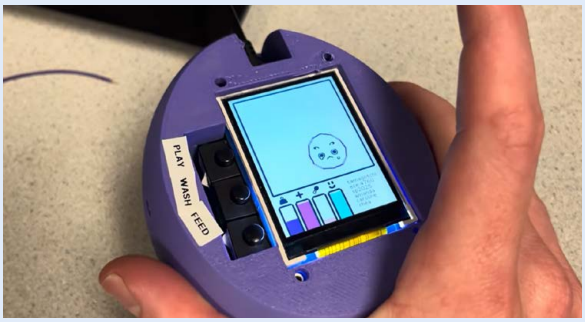
Quick FACTS

- The project took four weeks to complete
- FRAM is RAM with a ferroelectric layer
- Caroline had a background in game design and graphics
- Pets can die, but a button press resets it
- 'Tamagotchi' is a pun on the Japanese words for friend and egg



▼ All projects start off as a breadboard somewhere, and the Tumbler is no different

Look after your pet



1. Your pet's food, happiness, and health meters will decrease over time. Feeding, cleaning, and playing with it increases and decreases different stats; for example, cleaning improves health but decreases happiness.

2. The fourth meter, weight, goes up when it's fed, and goes down after play. This is the only meter that should not be maxed out like the others, otherwise your virtual pet will virtually die.

3. Shaking the pet too hard will also harm its health, so it's probably best not to throw it at a wall if you want it to survive.

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

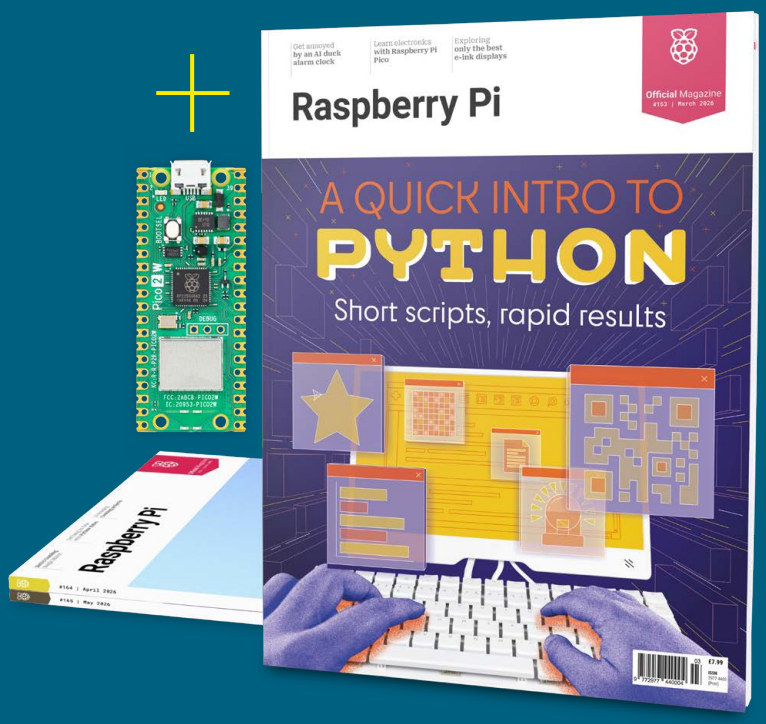
£10 (UK only)

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.

Spectrum Slit

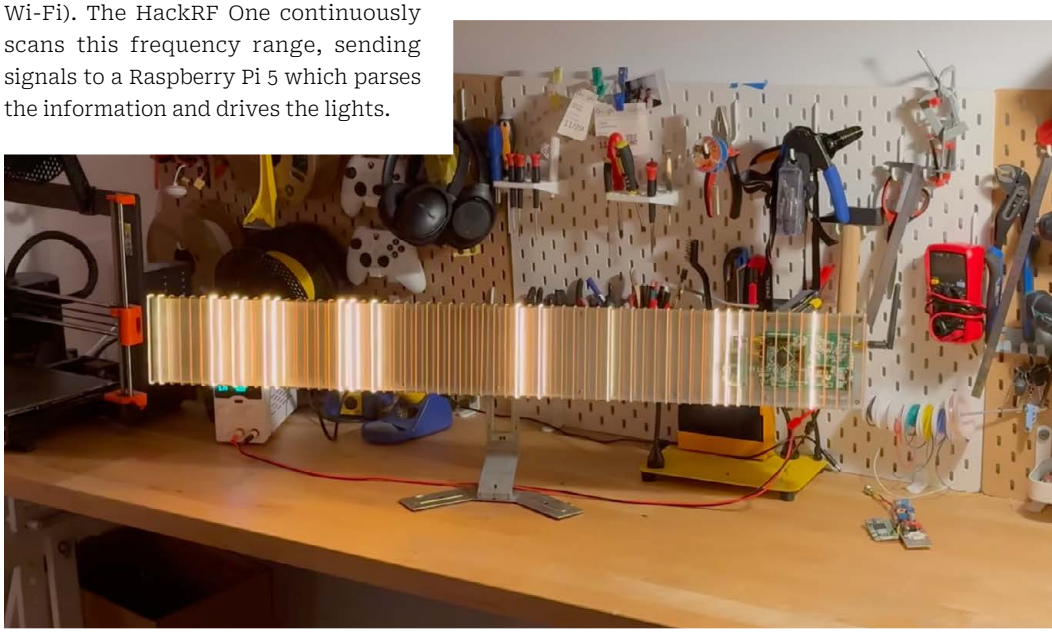
By Rootkid

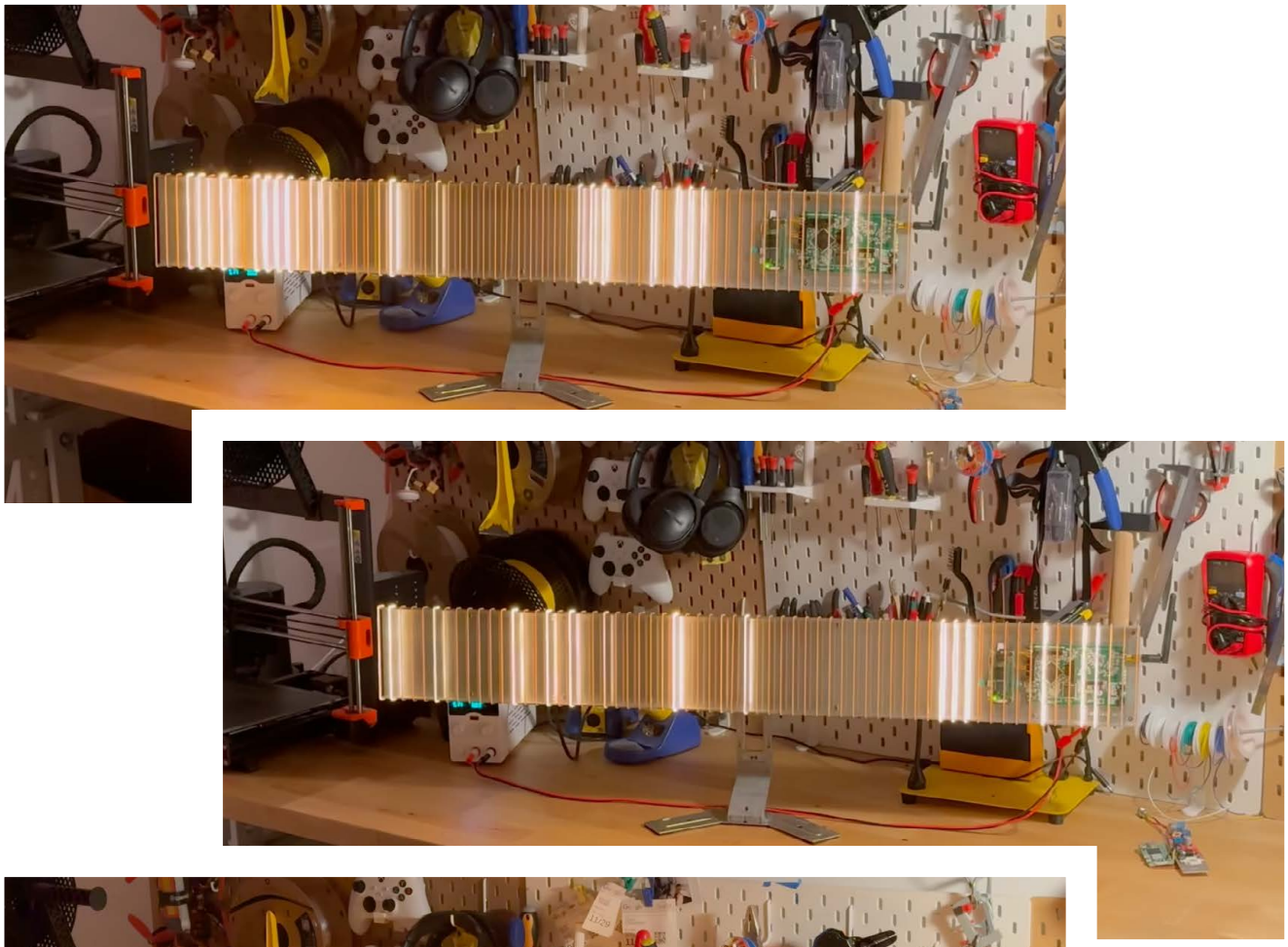
rpimag.co/RF-visualiser

Your eyes and ears are sensitive to a tiny range of wavelengths. Other living creatures can perceive different parts of the spectrum – bees, for instance, can pick up visual information in the petals of flowers that we can't see at all. But even though we can't see these hidden RF signals, it doesn't mean they're not there. If you have a computer and a phone – each with Wi-Fi, Bluetooth, and their background electrical noise – you're immersed in a cacophony of RF signals.

This work exposes that hidden layer of reality by translating radio wave activity into light (and sound) in real time. It measures radio signals within the 2.4GHz–5GHz bands frequencies used by Wi-Fi, Bluetooth, and other domestic wireless devices, and visualises them with 64 glowing filaments arranged to visually echo a scientific frequency plot.

The measuring of the RF spectrum is performed by a HackRF One, which is a software-defined radio that can detect frequencies ranging from 10MHz (commonly used for calibrating electronic equipment) to 6GHz (used by high-speed Wi-Fi). The HackRF One continuously scans this frequency range, sending signals to a Raspberry Pi 5 which parses the information and drives the lights.





▲ Not to get all tin-foil hat, but the walls really are talking to us

Counter-rotating clock

By Joren Heit

rpimag.co/Counter-rotating-clock

If you're ever stuck for a bit of inspiration and you need something to spark your imagination, you could do a lot worse than try your hand at building a clock. They combine as much or as little mathematics as you're comfortable with, physical, moving parts, LCD/OLED/otherwise pixellated displays, audio feedback, and require a means of setting the time as well as adjusting it for daylight savings time. There's a lot to get your teeth into, plus there's the fact that the internet is awash with brilliant designs that, even though they're constrained by the same set of requirements, all seem to be slightly different.

This project was inspired by one of maker Joren's students, who was trying to program an analogue clock in Java. Somehow a bug was causing the numbers to rotate rather than the hands; Joren saw this and decided to make the image on screen a reality.

Rather than start from scratch, Joren's device takes a standard wall clock and clamps it into a geared ring with 216 teeth. This balances on top of two 36-tooth gears, which rotate the clock so that the hour hand is always pointing in the same direction. The other two hands move as normal.

The electronics in this creation are simple: there's a single stepper motor to drive one of the 36T gears, an L298 motor driver, a DS3231 real-time clock, and an Arduino Nano, which calculates how quickly the gears should rotate. For bonus points, the user can set the minute or second hand to be the static hand – though we're not sure this would make the clock any easier to read.

▶ The rotation tends to drift out of time, so there's a real-time clock built in to keep things synchronised

Thought Catcher

By Ansh Trivedi

rpimag.co/ThoughtCatcher

Phones have been proven to be the worst things ever. Well actually not, but they have their own set of disadvantages and, honestly, using the phone at night is just bad." So says Ansh Trivedi, whose device, the Thought Catcher, takes a Raspberry Pi 5 and uses it to help the user sleep better, feel better, and generally live a more productive life. No, really!

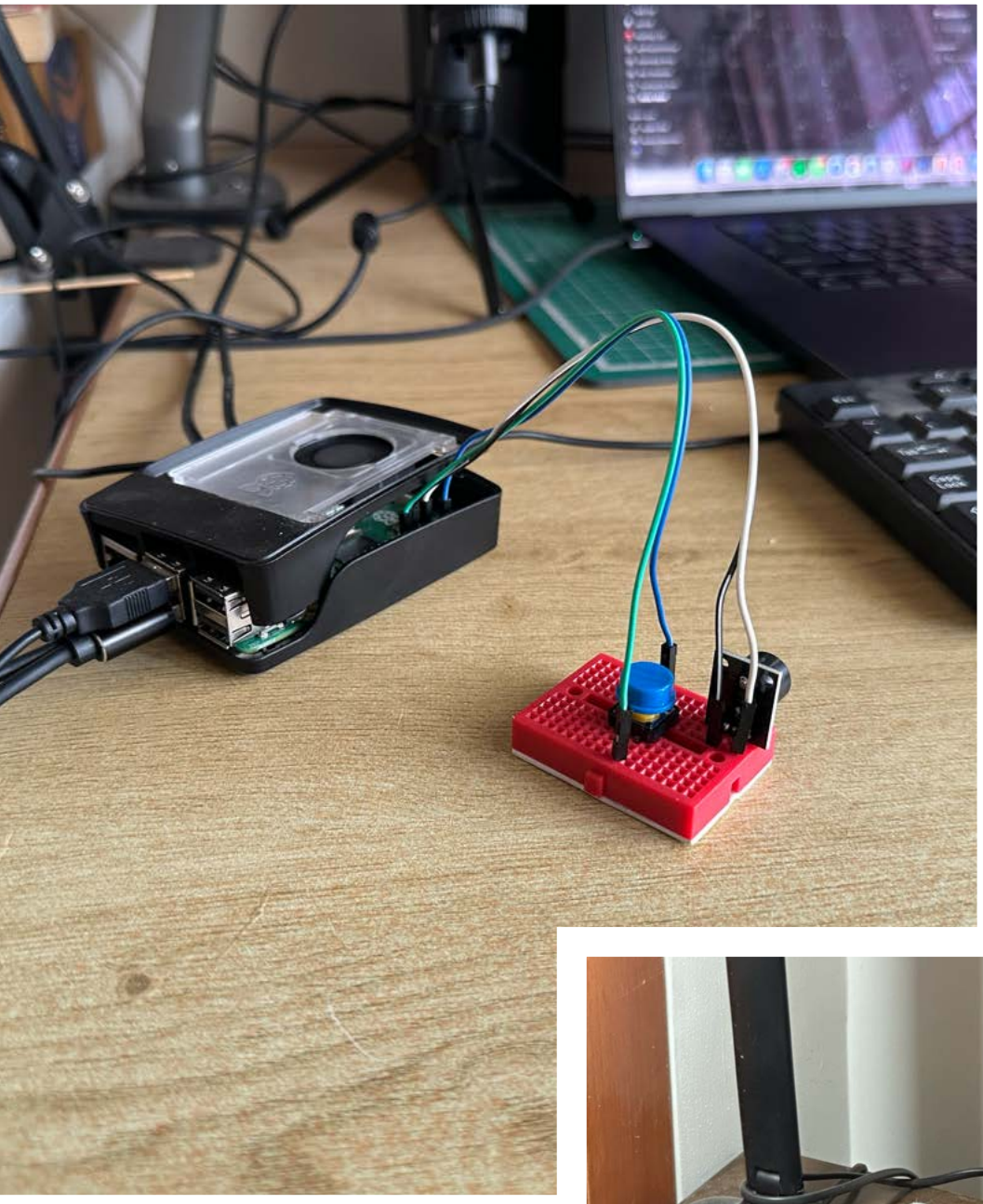
The smartphone is extremely useful, but its curse is that for all its features, it's second-best at so many of them. Despite this, the phone has all but crowded out the landline, the wrist-watch, the notepad, and we're all the poorer for it.

Ansh has recognised the failings inherent in the phone and realised that it's far better to have a device that does one thing, and one thing well – and that device is a nocturnal note-taker. The problem is that a phone is designed to get information into your head, with bright lights, notifications, vibrations, and beeps. You give it more attention and it rewards you with more beeps and bleeps, and before long you're arguing with someone in New Jersey about who the best super-middleweight of the 1990s was, when all you wanted to do was make a note for your future self to remember to take the bins out. It's exhausting, and that's before you get to the bright blue light disrupting your sleep patterns.

"Even when kept far from the bed (which I actually try to do), reaching for a phone breaks the fragile 'almost asleep' mental state," says Ansh. "I didn't need a better notes app. I needed a different way of taking notes. Specially the ones at night."

The Thought Catcher uses whisper.cpp for offline speech-to-text, and Python code running on a Raspberry Pi 5 to categorise the thought, generate a headline, look for keywords, and loop continuously, looking for button presses.

Apart from a Raspberry Pi 5, the hardware comprises a physical button, a microphone, and a buzzer, all housed in a 3D printed case.



▼ Sleep better knowing you have an easy way of making a night-time to-do list

Artie

By Andrew Schmelyun

rpimag.co/Artie

Andrew Schmelyun set out to create the most impractical AI assistant that he could make, and in the process created this device, which uses a portable TV set and a Raspberry Pi 4 to ask OpenAI for dinner recommendations. We've seen AI devices such as this before, and they always seem to veer between the awesome and the useless, but Artie has something that most other AI assistants lack: a face made up of HTML elements, animated by CSS. In case you hadn't guessed, Andrew is a web developer.

The code for Artie is written entirely in vanilla JavaScript. The front end – the face – is a Safari browser window occupying the full screen of the Sony 510U 5-inch portable television, which Andrew bought for $39.99 on eBay. The audio is transcribed on a Raspberry Pi 4, again using the web browser, as is voice recognition, which comes via the MediaRecorder() constructor.

The transcribed question gets sent to an LLM – Andrew has opted for OpenAI. While code is flying back and forth across the internet, animations in the CSS show a thinking face, and when the text comes back from OpenAI, another browser API, SpeechSynthesis, is used to read it out.

▲ "Hey Artie" – say the wake phrase, ask your question, and get a useful answer in your ears or sent to your printer

3D print

Dazzle, charm, and beguile with a twist of your hands

rpimag.co/orb-of-radiance

How about this to bring a bit of magic into our lives?: a spherical lamp that lights, and dims, using only the rotation of its two hemispheres. It's a brilliantly simple idea, and the execution is done so well that it makes us wonder why no-one thought of it before its maker, KICW.

The 3D printed body is covered in astronomical and alchemical symbols, which glow with the light of a pair of LED strips when the lamp is turned on. Just twist the two halves as if you were casting a spell, and the orb switches on and glows. Magic!

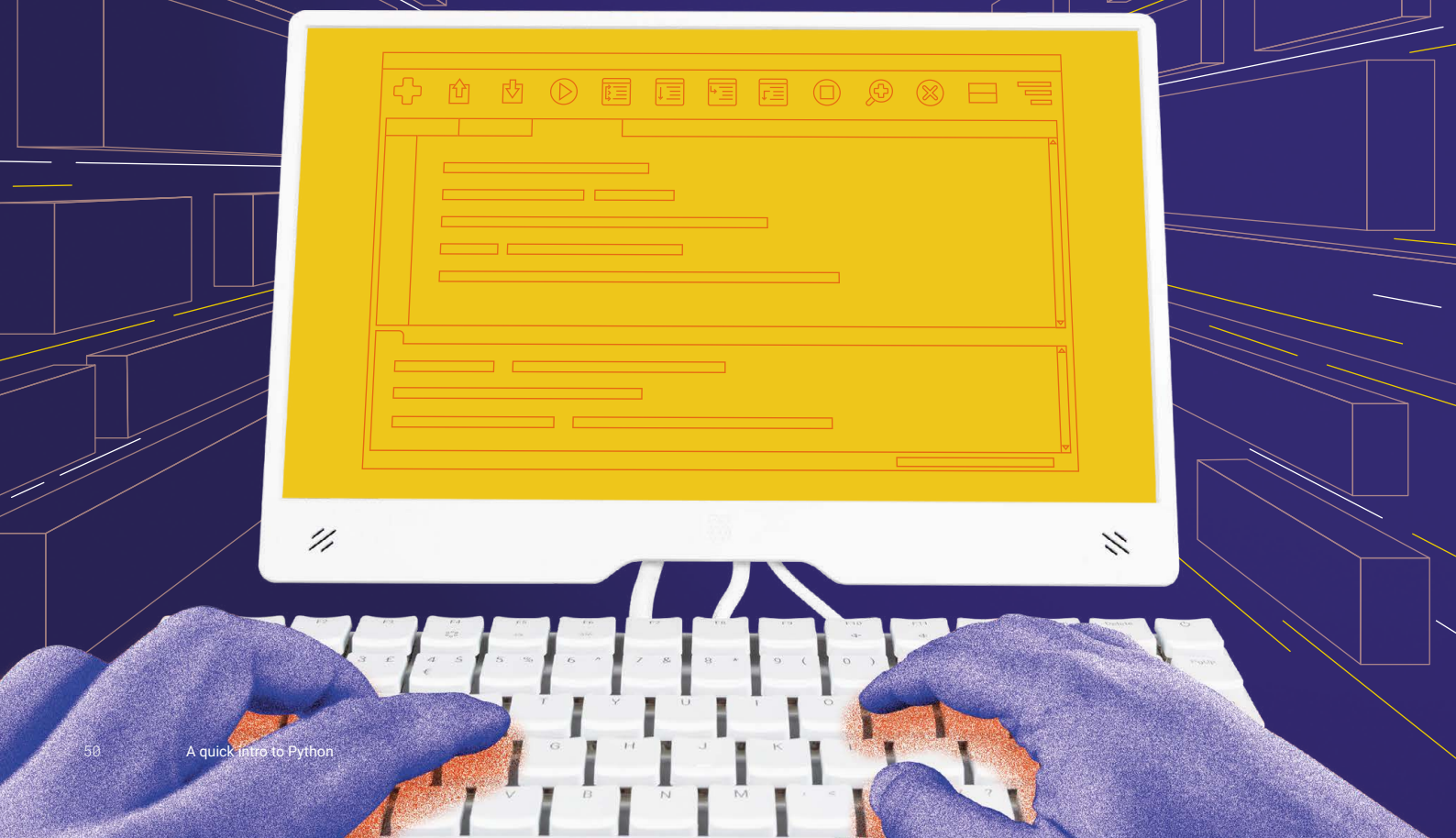
Other than the battery and the circuits to enable charging over USB-C, the LED strips, and the potentiometer board (which is ultimately what you're turning when you rotate the two halves of the lamp), most of the rest of the bill of materials is taken up with connectors and adapters. The really magical aspect of this build is that it was designed to need no soldering at all – you just clip the parts together, and it works. That makes for a slightly more complicated assembly than you'd get if you could join components to a PCB with the addition of hot solder, but it does also mean that you can give this to an older child to build and know that they're not going to set fire to anything.

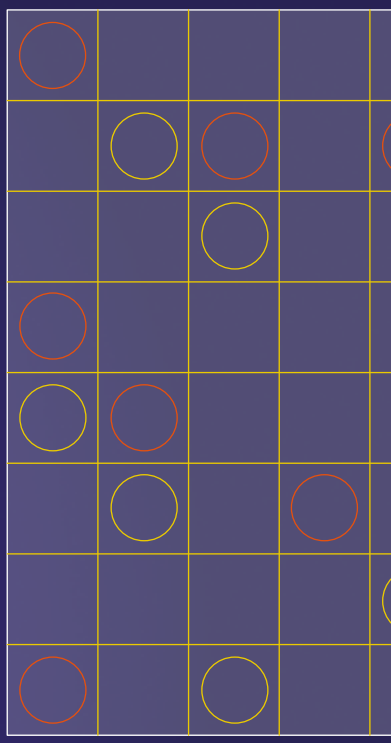
As if that weren't magical enough, the print requires no supports, though you do have to make sure that you use a smooth print bed, as the two halves will have to rotate against each other.

A QUICK INTRO TO PYTHON

Start learning Python today. Discover the libraries that help you quickly create powerful code

By **Sean McManus**.





Need a simple app? Why not build it yourself? Using the Python programming language, you can give the computer precise instructions to do whatever you want.

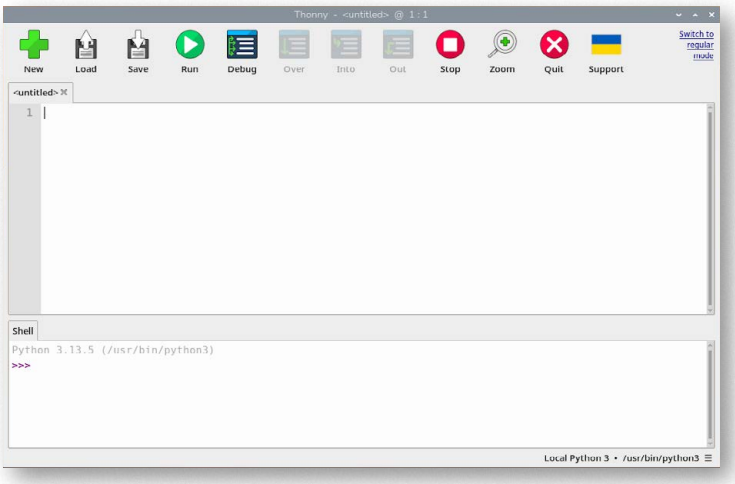
> *Python is described as 'batteries included' because it has so many capabilities*

Python is the world's most popular programming language, according to the TIOBE Index (**rpimag.co/tiobe**), and it's easy to see why. The programs are clutter-free and easy to read, with far fewer brackets to untangle than in some other languages.

It's powerful, too. Python is often described as 'batteries included' because it has so many capabilities built in. You can extend the language by adding libraries to do things like download data, create QR codes, and edit PDF files. As a result, you can do a lot with a few lines of code.

In this article, you'll see how to quickly get results from Python on Raspberry Pi. We'll introduce some of the basics of the language, then share some interesting small programs. Tinker with them. Experiment. Start programming today!

Write your first Python program



▲ What greets every programmer before they write their next great script

The best way to learn to program is by doing it, so let's build a simple quiz game together. Don't worry if the code looks alien. We'll talk you through it.

01 Open Thonny

Raspberry Pi OS comes with Thonny, which you can use to write and run Python programs. Find it under Programming in the Raspberry Pi desktop menu.

02 Enter your first program

Thonny has two panels. At the bottom is the 'Shell'. If you enter instructions here, they are carried out immediately. We'll focus on the top panel, where you type in programs. Programs are stored sets of instructions that you can save, reuse, and easily edit. Enter the following short program. Can you predict what it will do?

```
name = input("What is your name?")
print ("Hello", name)
```

You can get interesting results from very little code

In Thonny, you'll see that "What is your name?" and "Hello" are green, which means they're chunks of text, known as 'strings'. Strings have quote marks around them.

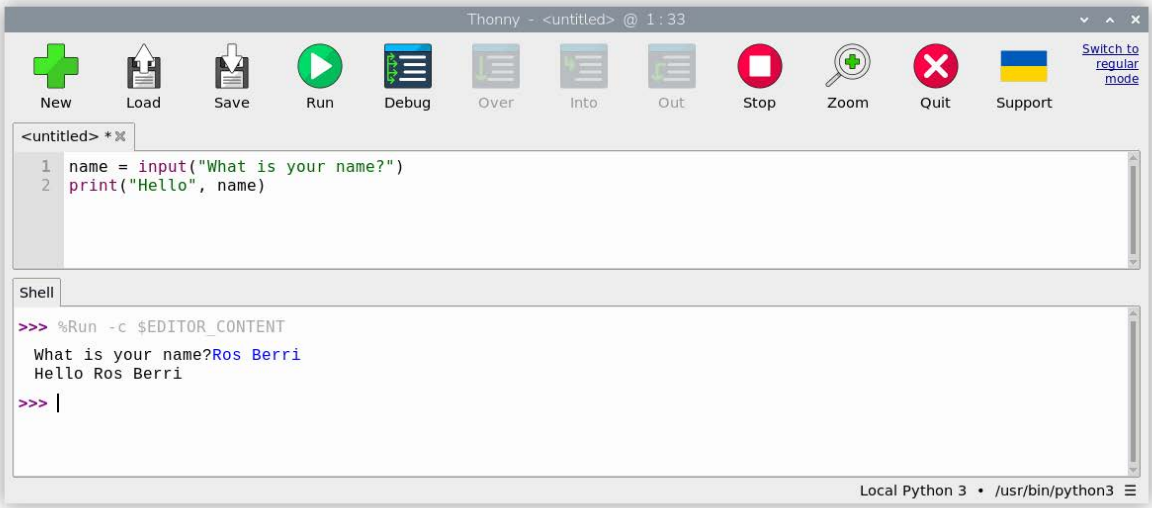
03 Run your program

Click the green Run button to start the program. If it doesn't work, double-check your code, especially the punctuation.

The first line uses Python's `input()` function which tells the user what it wants and then waits for them to type it in. The user's input goes into a variable called `name`. A variable is like a box for storing information. A program can change (or vary) the information in a variable.

On the second line, the `print()` function outputs information in the Shell. Inside the brackets we put the information we want to output. This is the string `"Hello"` and the variable that contains the user's name. We separate them with a comma. Can you modify the program to say "[name] is awesome"?

▼ Write your program in the top panel and see its output in the Shell



04 Add a quiz question

Let's extend this program into a guess-the-year quiz, as shown in **simple_quiz.py**.

The game begins at line 5. We ask the quiz question and put the player's answer into a variable called `guess`. The `input()` function creates string variables, so line 6 converts `guess` into an integer (whole number) so we can do number comparisons. The `#` symbol marks a comment, added to help people understand the code. The computer ignores comments.

Lines 7 and 8 are a team. 7 checks whether the number in the `guess` variable is equal to 1837, the correct answer. Python uses two equals signs to check for things being the same. (In lines 1,5, and 6, you saw one equals sign used to mean "make this variable equal to" something.) The instruction in line 8 is only carried out if the player's guess is 1837.

You can tell that line 8 belongs with line 7 because it's indented by four spaces. Python uses indentation to show which instructions belong together without needing additional punctuation, such as brackets. If you get your indentation wrong, your program won't run. If you remember the colon at the end of line 7 when typing in the program, Thonny will automatically indent the next line. When you don't need indentation any more, tap the **DELETE** key.

Similarly, lines 9 and 10 are a pair that checks whether the player's guess is less than 1837 and tells them they guessed too early if so. Lines 11 and 12 cover guesses higher (or later) than 1837.

Test that your program works by playing it with early, late, and correct guesses. Congratulations! You've built your first Python game!

simple_quiz.py

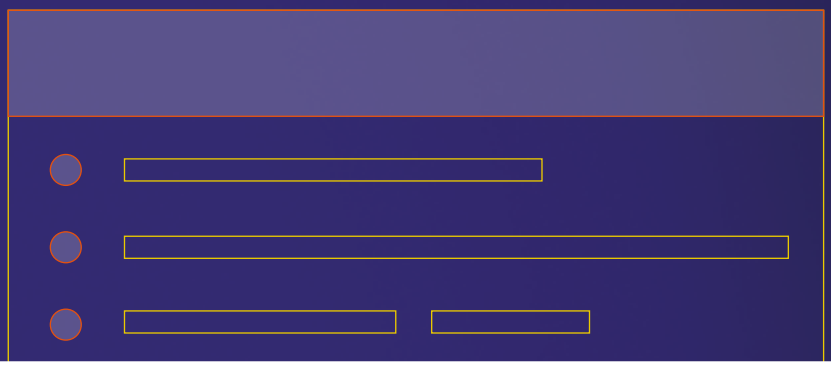
> Language: **Python**

DOWNLOAD THE FULL CODE:

rpimag.co/python163

```
001.  name = input("What is your name?")
002.  print ("Hello", name)
003.  print ("Welcome to the history quiz!")
004.
005.  guess = input("When did Queen Victoria ascend to
      the throne?")
006.  guess = int(guess) # Convert string variable to
      a number variable
007.  if guess == 1837:
008.      print("Correct! And she reigned for 63 years
      and 216 days.")
009.  if guess < 1837:
010.      print("No, it was later than that in 1837")
011.  if guess > 1837:
012.      print("No, it was earlier than that in
      1837")
```

▲ This simple quiz game asks the user to guess the correct year for a history question



▼ This menu is generated from a list of tea flavours.

```
Shell
>>> %Run tea_menu.py
  Our tea menu:

  * A lovely cup of camomile tea
  * A lovely cup of mint tea
  * A lovely cup of blackcurrent tea
  * A lovely cup of fennel tea

  Our first list item is camomile
  Our second list item is mint
  We have 4 drinks
>>>
```

tea_menu.py

> Language: **Python**

DOWNLOAD THE FULL CODE:
rpimag.co/python163

```
001. print("Our tea menu:\n")
002.
003. tea_menu = ["camomile", "mint", "blackcurrant",
     "fennel"]
004. for flavour in tea_menu:
005.     print("* A lovely cup of", flavour, "tea")
006.
007. print("\nOur first list item is", tea_menu[0])
008. print("Our second list item is", tea_menu[1])
009. print("We have", len(tea_menu), "drinks")
```

05 Understanding lists

You could add more questions by copying lines 5 to 12, pasting them at the end of your program so far, and modifying the code for each question. But there's a better way.

Lists are one of the data structures you can use to store information in Python, so that you can write shorter, clearer, and more sophisticated programs. Click Thonny's New button to open a new tab and enter the **tea_menu.py** listing.

In this program, line 3 creates a list called `tea_menu`. It contains four strings. Each list item is separated by a comma, and the whole list is enclosed in square brackets.

Lines 4 and 5 are a `for` loop. A loop is a portion of code that repeats. Line 4 sets it up so that each time the loop starts, the program takes the next item from the `tea_menu` list and puts it into a variable called `flavour`. Line 5 is indented to show it belongs to the loop. It outputs the tea flavour. Try adding more flavours to the list.

We can also access a list item using its position number in the list, called its index. The numbers start at zero, so lines 7 and 8 show how to get the first two flavours. The `len()` function reveals how long a list is, as shown in line 9.

There's lots more to learn about lists, but that's all we need to know today.

Curious about `\n` in the strings we're printing? It adds a blank line to tidy up the output.

```
Shell
>>> %Run longer_quiz.py
  When was the Eiffel Tower completed? 1900
  No, it was earlier than that!
  When was the Eiffel Tower completed? 1880
  No, it was later than that!
  When was the Eiffel Tower completed? 1889
  That's correct!
  When was the Apollo 11 moon landing? 1969
  That's correct!
  When did Queen Victoria ascend to the throne? 1837
  That's correct!
  It took you 5 goes to answer 3 questions
>>>
```

▶ Testing the quiz game with early, late, and right guesses

06 Creating a list-based quiz

Let's combine everything you've seen with a couple of new ideas. We'll make a quiz game with multiple questions. The player guesses until they are right, and then the program moves on to the next question. See **longer_quiz.py**.

The first new idea: Lists can contain lists. This program creates a list called `question_and_answer_list` (lines 3 to 7). Each item in it is a list that contains a single question (a string) and its answer (a number).

Line 9 sets up a loop that goes through the big question and answer list. It extracts the next question and answer pair and puts them into variables called `question` and `answer`.

The second new idea is that you can have loops inside loops, called nested loops.

Need a simple app? Why not build it yourself?

This program introduces the `while` loop starting at line 12 and ending at line 21. This type of loop repeats a chunk of code while something is true. In this case, we repeat as long as the player's guess is not equal to (`!=`) the answer. We indent the lines below by another four spaces to show which lines belong to this loop.

So, we have an outer loop that cycles through the questions, and an inner loop that takes the player's guess, tells them if they were early, late or correct, and repeats until they get it right. At that point, the program jumps back to line 9 to pick up the next question and answer pair.

Line 1 makes a variable to remember how many guesses the player makes. It's increased in line 15. At the end, we give the player their score. In line 13, a space was added to the end of the question, to tidy up the display. Otherwise, the player's answer butts up against the question.

Customising programs is good practice when you're learning Python. Can you add your own questions and change the messages in this quiz?

longer_quiz.py

> Language: **Python**

DOWNLOAD THE FULL CODE:
rpimag.co/python163

```
001. guess_count = 0
002.
003. question_and_answer_list = [
004.     ["When was the Eiffel Tower completed?",
     1889],
005.     ["When was the Apollo 11 moon landing?",
     1969],
006.     ["When did Queen Victoria ascend to the
     throne?", 1837]
007.     ]
008.
009. for question, answer in
     question_and_answer_list:
010.     guess = None
011.
012.     while guess != answer:
013.         guess = input(question + " ")
014.         guess = int(guess)
015.         guess_count +=1 # increase guess_
     count by 1
016.         if guess == answer:
017.             print("That's correct!")
018.         if guess < answer:
019.             print("No, it was later than
     that!")
020.         if guess > answer:
021.             print("No, it was earlier than
     that!")
022.
023. print("It took you", guess_count, "goes to
     answer", len(question_and_answer_list),
     "questions")
```

Extending Python with libraries

▶ Choose your virtual environment in Thonny to use your favourite Python libraries

You can easily install additional libraries that give you new functions to play with. To do this, you need to use a 'virtual environment'. It's a safe place to install libraries without interfering with the main version of Python that the operating system uses. Follow these steps.

01 Create a virtual environment

Use the button in the top-left of the desktop to open a terminal window. Create a folder for your projects called **powerful_python**:

```
$ mkdir powerful_python
$ cd powerful_python
```

Now create the virtual environment:

```
$ python3 -m venv my_venv
```

This creates a virtual environment called `my_venv`.

02 Activate your virtual environment

The next step is to activate the virtual environment:

```
$ source my_venv/bin/activate
```

You should now see your virtual environment name before your prompt.

03 Install packages

To install libraries, we use the Python package manager, called pip. For example, to install the matplotlib library, we use:

```
$ pip install matplotlib
```

04 Use your virtual environment

You've created your virtual environment and installed a package in it, but Thonny isn't using it yet.

Click the three-line menu in the bottom-right of Thonny and choose 'Configure interpreter'. Now, click the three-dots button to open a File window. Choose Home and go into your **powerful_python** folder and then to **my_venv**, and **bin** (short for binary, not rubbish). Select **python3** and click OK. This will ensure that Thonny uses the Python packages in your virtual environment.

Common bugs

In no particular order, the top bugs you might encounter:

1. **Wrong or missing brackets**. Python uses square, curved, and curly brackets for different things. Every bracket needs a matching partner.
2. **Missing colons**. A `for`, `while`, `if`, or `def` instruction needs a colon at the end of the line.
3. **Missing commas**. List items need a comma between them.
4. **Wrong indentation**. You can space out your code inside lists how you like, but indentation before instructions must follow strict rules.

Powerful programs

You might not understand every line, but these Python programs show how you can get interesting results from very little code. Read the listings and their descriptions and try them out. We've kept the programs as simple as possible here, but there are some bonus features in the code online. You can download all of the examples at **rpimag.co/python163**.

Sparkle the Sense HAT

Even if you don't have a Sense HAT, this is a great program to read. It sets each of the LEDs to a random colour. It illuminates (see what I did there?) the way that loops can be put inside each other.

The first few lines import the libraries we want to use, all of which are already installed.

We have a `while True` loop, which repeats until you stop the program. Inside that is a loop that goes through the rows (the y loop). And inside that, there is a loop that lights each LED in the current row (the x loop). As before, each level of indentation shows that instructions belong to the loop defined above.

We used a `for` loop previously to go through a list. Here, we use the `range()` function to make a list of numbers and then loop through that. This gives the `y` and `x` variables the values 0 to 8 in turn, matching the coordinates of the LEDs.

We generate random numbers between 0 and 255 using `random.randint(0, 255)`. These numbers set the amount of red, green, and blue in the LED colour.

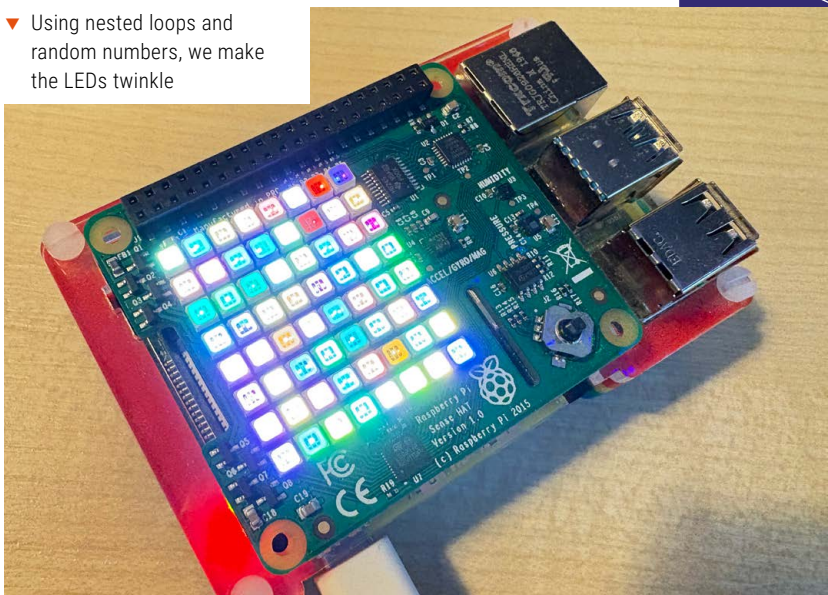
The time module can be used to add a small delay, in this case a quarter of a second (line 12).

sparkle.py

> Language: **Python**

DOWNLOAD THE FULL CODE:
rpimag.co/python163

```
001. from sense_hat import SenseHat
002. import random
003. import time
004.
005. sense = SenseHat()
006. sense.show_message("Let's sparkle!")
007.
008. while True:
009.     for y in range(8):
010.         for x in range(8):
011.             sense.set_pixel(x, y, [random.
     randint(0, 255), random.randint(0, 255),
     random.randint(0, 255)])
012.             time.sleep(0.25)
```



▼ Using nested loops and random numbers, we make the LEDs twinkle

Make art with code

This program draws four stars on the screen, using the turtle module, which comes with Python.

We use a `for` loop with `range()` to repeat a set number of times, but we set the variable to the underscore character. It shows we don't need to use the number in the range.

The loop starting at line 16 draws a star with six points. It's inside the loop starting at line 10, which repeats four times to draw four stars.

In line 14, the pen colour is made up of three random numbers in the same way we assigned random LED colours in the Sense HAT program.

Can you make the program draw stars at random positions?

▼ Draw stars with Python

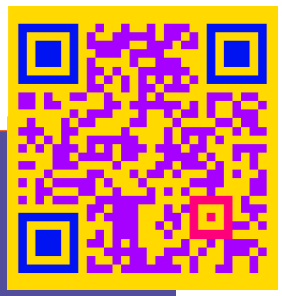
turtle_stars.py

> Language: **Python**

DOWNLOAD THE FULL CODE:
rpimag.co/python163

```
001.  import turtle
002.  import random
003.
004.  turtle.colormode(255)
005.  turtle.bgcolor("black") # Background colour
006.  turtle.pensize(10)
007.  turtle.penup()
008.  turtle.goto(-525, 0) # x, y coordinates
009.
010.  for _ in range(4): # Loop to draw four stars
011.      turtle.penup()
012.      turtle.forward(180) # gap between stars
013.      turtle.pendown()
014.      turtle.pencolor(random.randint(0, 255),
      random.randint(0, 255), random.randint(0, 255))
015.
016.      for _ in range(6): # Loop to draw each star
017.          turtle.forward(50)
018.          turtle.left(60)
019.          turtle.forward(50)
020.          turtle.right(120)
```

▶ Segno makes QR codes



Generate colourful QR codes

The Segno library enables you to create colourful QR codes. It uses colour codes in hexadecimal, a counting system that adds the letters A to F to represent the numbers 10 to 16. You can find colour schemes with hex codes online. Add your own link in line 2. For more features, including support for Wi-Fi codes and animated backgrounds, see the documentation at **rpimag.co/segnodocs**. Install the segno library in your virtual environment using `pip install segno`.

qr_code_maker.py

> Language: **Python**

DOWNLOAD THE FULL CODE:
rpimag.co/python163

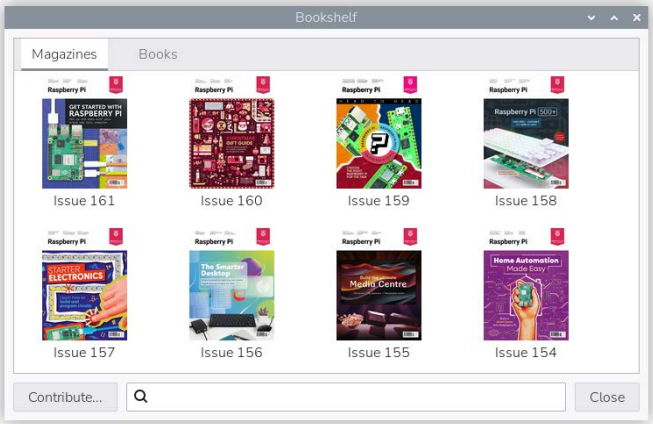
```
001.  import segno
002.  url = "https://www.youtube.com/watch?v=RrESvSRNpeo"
003.  qr_code = segno.make(url)
004.  qr_code.save("qr_code.png", scale = 12, border = 2, dark =
      "#a600ff", light = "#ffd900", alignment_dark = "#ff007b",
      finder_dark = "#0014f1")
```

Merge PDFs

When researching a topic, we sometimes gather relevant articles and book chapters into a single PDF to search, print, or read on the iPad. This simple program combines your chosen pages from various PDFs into a single file. Each item in the file list is itself a list that contains the file name and path in a string, plus the pages you want in another list. So, we've got a list in a list in a list. Pay attention to the commas and brackets!

The Bookshelf app in Raspberry Pi OS provides back issues of this magazine. This example file list combines two Python tutorials from previous issues we've downloaded, which are great next steps after you've read this article. Install the PyPDF2 library in your virtual environment: `pip install pypdf2`. For details of how PyPDF2 and file saving work, see Productive Python in issue 144 (**rpimag.co/144**).

- Find PDFs in Bookshelf



pdf_merger.py

> Language: **Python**

DOWNLOAD THE FULL CODE:

rpimag.co/python163

```
001.  from PyPDF2 import PdfReader, PdfWriter
002.
003.  file_list = [ ["/home/pi/Bookshelf/MagPi128.pdf", [0, 43,
      44, 45, 46, 47, 48, 49] ],
004.                ["/home/pi/Bookshelf/MagPi82.pdf", [0, 30,
      31, 32, 33, 34, 35, 36] ]
005.                ]
006.
007.  w = PdfWriter()
008.
009.  for file, page_list in file_list:
010.      r = PdfReader(file)
011.      for p in page_list:
012.          w.add_page(r.pages[p])
013.
014.  with open("merged_tutorials.pdf", "wb") as f:
015.      w.write(f)
```

Look up your carbon intensity

The requests module enables you to download structured information from the internet. This program displays the current carbon intensity for your electricity region, including how much of your energy is being generated by coal and wind power. An energy-intensive Raspberry Pi project could use this information to decide when to do its most power-hungry work. Alternatively, you can use it to decide when to put the washing machine on.

Python's `split()` function turns a string into a list. In line 8, it splits the postcode wherever there's a space, and then just takes the first list item. It's a slick way to make sure we only get the first part of the postcode. This is what we must send the Carbon Intensity API (application programming interface) to get the right data back.

This program introduces f-strings (see lines 9, 16, 17, and 22), which are used to insert a variable's contents into a string. You put `f` before the opening quote marks and then put the variable name in curly braces within the string.

Extracting bits of the data is complicated (see lines 11 to 14). Think of it like navigating a route, where you direct the program down the right paths by using the names on their signposts, combined with index numbers. The hard work has been done so you can just use the variables we've set up in your program. If you're curious, you can see an example of the data structure at **rpimag.co/carbonintensity**.

Install the requests library with `pip install requests`.

carbon_intensity.py

DOWNLOAD THE FULL CODE: rpimag.co/python163

> Language: **Python**

```
001.  # Carbon intensity reporter
002.  # See documentation at
003.  # https://carbon-intensity.github.io/api-definitions/#get-
      regional-postcode-postcode
004.
005.  import requests
006.
007.  postcode = input("Enter a UK postcode: ")
008.  postcode = postcode.split(" ")[0]
009.  url = f"https://api.carbonintensity.org.uk/regional/postcode/
      {postcode}"
010.
011.  received_data = requests.get(url).json()["data"][0]
012.  region = received_data["shortname"]
013.  intensity = received_data["data"][0]["intensity"]["forecast"]
014.  index = received_data["data"][0]["intensity"]["index"]
015.
016.  print(f"Region: {region}")
017.  print(f"Carbon intensity: {intensity} gCO2/kWh ({index})")
018.
019.  for fuel_and_perc in received_data["data"][0]["generationmix"]:
020.      fuel = fuel_and_perc["fuel"]
021.      perc = fuel_and_perc["perc"]
022.      print(f"{fuel} {perc} %")
```

```
Shell
>>> %Run carbon_requests.py

  Enter a UK postcode: CB4
  Region: East England
  Carbon intensity: 143 gCO₂/kWh (moderate)
  biomass 5.5 %
  coal 0 %
  imports 2.8 %
  gas 31.1 %
  nuclear 16.8 %
  other 0 %
  hydro 0 %
  solar 3.1 %
  wind 40.6 %

>>>
```

◀ See how energy is being generated in your region right now

◀ Use the Pillow library to add watermarks to your photos

Python is the world's most popular programming language

Watermark images

This program watermarks an image and introduces the idea of a function, which is a reusable part of the program. The `watermark` function is defined in line 3 and receives the file name and its extension. As you might expect by now, the indented lines underneath belong to the function. In line 8, you can see how we combine strings using the `+` symbol. The function doesn't start its work until line 23 triggers it and feeds it the image information.

The function uses the pillow library to open the image file, create a transparent overlay, draw the watermark text on it, and then combine it with the original image. The image is converted to RGB with no transparency so it can be saved as a JPG. The extended version of this program online processes an entire folder of images and resizes them too. Install the library using `pip install pillow`.

watermark_image.py

> Language: **Python**

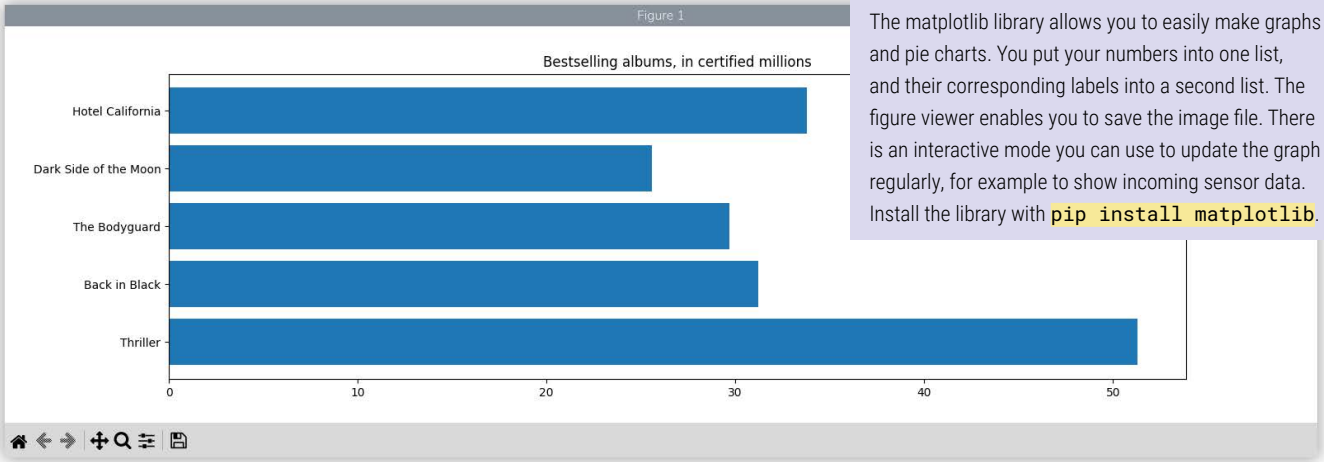
DOWNLOAD THE FULL CODE:
rpimag.co/python163

```
001. from PIL import Image, ImageDraw, ImageFont
002.
003. def watermark(filename, extension):
004.     transparency = 200  # 0 to 255
005.     text = 'Photo by Sean'
006.     font = ImageFont.truetype('DejaVuSans-Bold.ttf', 48)
007.
008.     original = Image.open(filename + "." + extension)
009.     image_height = original.size[1]
010.
011.     text_position = (30, image_height - 70) # Bottom left corner
012.     overlay = Image.new('RGBA', original.size, (255, 255, 255, 0))
013.     draw = ImageDraw.Draw(overlay)
014.     draw.text(text_position, text, font=font, fill=(255, 255,
    255, transparency))
015.
016.     watermarked_image = Image.alpha_composite(
    original.convert('RGBA'), overlay)
017.     watermarked_image = watermarked_image.convert('RGB')
018.
019.     new_filename = filename + "-wm." + extension
020.     watermarked_image.save(new_filename)
021.
022. print("Watermarking your images...")
023. watermark("IMG_3412", "jpg")
```

▼ Generate graphs in Python with the matplotlib library

Create graphs

The matplotlib library allows you to easily make graphs and pie charts. You put your numbers into one list, and their corresponding labels into a second list. The figure viewer enables you to save the image file. There is an interactive mode you can use to update the graph regularly, for example to show incoming sensor data. Install the library with `pip install matplotlib`.



bar_graph.py

> Language: **Python**

DOWNLOAD THE FULL CODE:
rpimag.co/python163

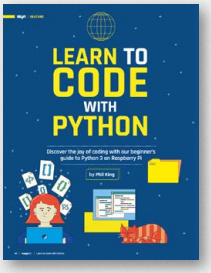
```
001. import matplotlib.pyplot as plt
002.
003. labels = ["Thriller", "Back in Black", "The
     Bodyguard", "Dark Side of the Moon", "Hotel
     California"]
004. values = [51.3, 31.2, 29.7, 25.6, 33.8]
005.
006. plt.barh(labels, values)
007. plt.title("Bestselling albums, in certified
     millions")
008. plt.show()
```

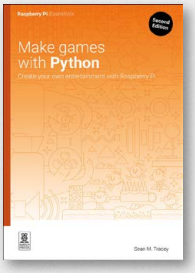
Further reading

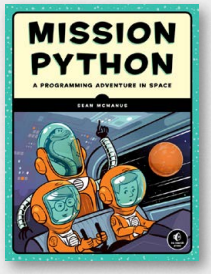
Learn to Code with Python: We took a quick tour this time to get you to results fast, but our feature from issue 128 covers all the basic syntax of Python. **rpimag.co/128**

Make Games with Python: This Raspberry Pi Essentials book shows you how to make games with the Pygame library. **rpimag.co/makegamesbook**

Mission Python: Build a 3D game set on a space station and learn about data structures and animation on the way. **rpimag.co/missionpython**







◀ Take your next steps with Python with these tutorials

Part 08

Conquer the command line: stopping a process

As close to perfect as Raspberry Pi OS is, things can go wrong. In this part, we learn that there's no need to turn Raspberry Pi off and on again: just kill the process!

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

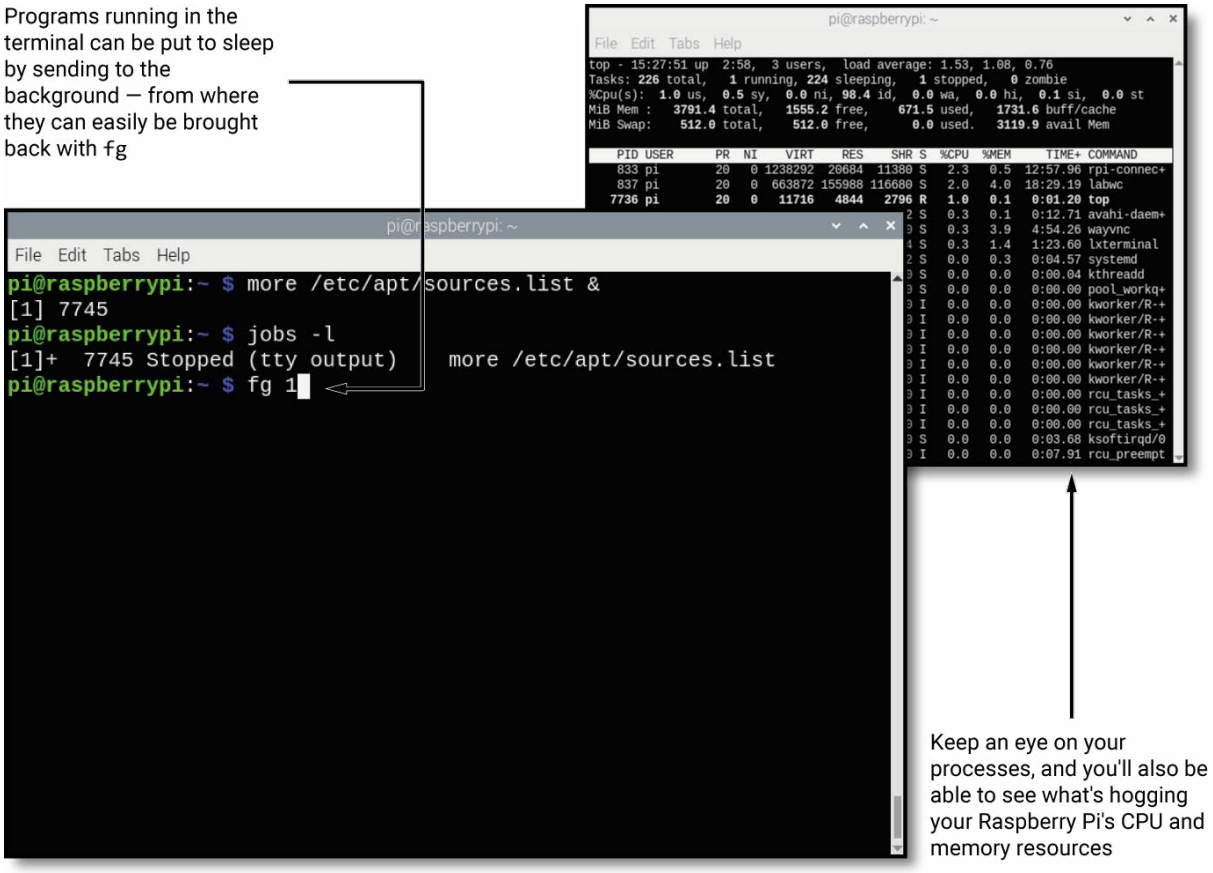
Ever lost the 'off switch' for a program? Sometimes a piece of software you're running seems to have no inclination to stop: either you cannot find how to quit, or the app has a problem, and won't respond to your **Q**, **CTRL+C**, or whatever command should close it down.

There's no need to panic, and certainly no need to reboot; just identify the process and quietly 'kill' it. We'll show you how, and we'll look at what else can be done with knowledge of processes.

Processes

Find the many processes running on your Raspberry Pi with the `ps` command (**Figure 1** overleaf). As a minimum, you should run it with the `a` and `x` options – which together give all processes, rather than just those belonging to you or just those attached to a terminal – and with `u` to see a user-oriented output format that includes detailed process information; `w` adds wider output, and `ww` will wrap over the line end to display information without truncating the output.

Type `ps auxww` to see, then try with just `a` or other combinations. You will notice that these options work without the leading dash seen for other commands. Both the lack of dashes, and the particular letters, `a` and `x`, date back to the original UNIX ps of the early 1970s, maintained through various revisions by one



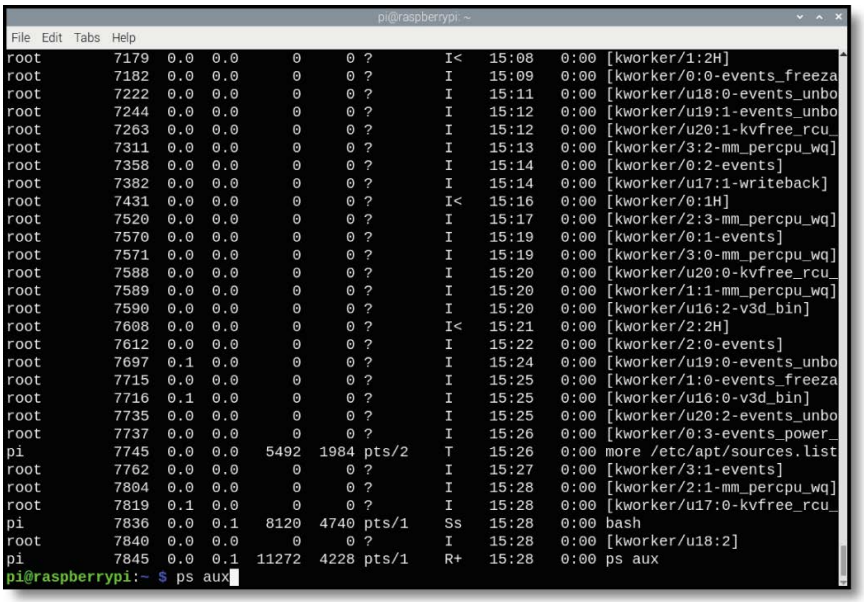
of UNIX's two family branches, BSD, and baked into the first GNU/Linux ps. UNIX's other branch, System V, had extended and changed ps with new options and new abbreviations for command switches, so for `ps ax` you may see elsewhere `ps -e` (or `-ef` or `-ely` to show in long format). The ps that comes with Raspberry Pi OS will work with both BSD and System V command-line arguments.

The `ps aux` listing has various headers, including the USER which owns the process, and the PID, or Process IDentification number. The list starts with 1 for init, the parent process of everything

There's no need to panic; just identify the process and quietly 'kill' it

BOOT SYSTEM

The startup process of Raspberry Pi OS is controlled by systemd, like other GNU/Linux distributions. It's fairly new in Linux terms – systemd came out in 2010, when Linux was just 19 years old (and UNIX was 41).



◀ **Figure 1:** Everything running has a process ID (PID) that can be used to control that program; find them all with `ps aux`

that happens in userspace after the Linux kernel starts up when you switch Raspberry Pi on.

Knowing the PID makes it easy to kill a process, should that be the preferred way of shutting it down. For example, to kill a program with a PID of 3012, simply enter `kill 3012`, and to quickly find the process in the first place, use `grep` on the ps list. For example, locating vi processes:

```
$ ps aux | grep -i vi
```

The `-i` (ignore case) isn't usually necessary, but occasionally a program may break convention and contain upper-case letters in its file name. You can also use `killall` to kill by program name: `killall firefox`.

Piping commands

You can pipe ps's output to select the PID and feed directly to the kill command:

```
$ kill $(ps aux | grep '[f]irefox' | awk
'{print $2}')
```

We don't have space for an in-depth look at awk (we're using it here to print the second field of grep's output: the PID), but the `[f]` trick at the beginning of 'firefox' (or whatever named process you want to kill) prevents the grep process itself being listed in the results; in the earlier vi example, grep found the grep process itself as well as vi (and anything with the letter sequence vi in its name). If you run the command between `$(` and `)` by itself, you'll see the PID of Firefox displayed (if it's running). Everything between the `$(` and `)` is replaced by its output, so it's as if you typed `kill` followed by the PID of Firefox.

That's a fun trick, but the `pgrep` command can find the PID for you, just using the name of the process: `pgrep firefox`. The `pkill` command will kill all processes matching the name you provide: `pkill firefox`. But don't go killing processes indiscriminately; you could inadvertently terminate one that is essential to the proper functioning of Raspberry Pi OS!

The output of ps also shows you useful information like percentage of memory and CPU time used, but it's more useful to see these changing in real time. For this, use `top`, which also shows total CPU and memory use in the header lines, the latter in the format that you can also find by issuing the command `free`. For an improved top, you can use `htop` – if it needs installing, use:

```
$ sudo apt install htop
```

htop is scrollable, both horizontally and vertically, and allows you to issue commands (such as `k` for kill) to highlighted processes. When you've finished, both top and htop are exited with **Q**, although in htop you may care to practise by highlighting the htop process and killing it from there (see **Figure 2**). htop also shows load over the separate cores of the processor if you have a Raspberry Pi 2 or later.

Background job

Placing an ampersand (&) after a command in the shell will place the program in the background – try with `man top &` and you'll get an output like `[1] 12768`.

The first number is a job number, assigned by the shell, and the second the PID we've been working with above. man top is now running in the background, and you can use the job control number to work with the process in the shell. Start some other processes in the background if you wish (by appending `&`), then bring the first – man top – to the foreground with `fg 1`. Now you should see man running again.

You can place a running shell program in the background by 'suspending' it with **CTRL+Z.** fg will always bring back the most recently suspended or backgrounded job, unless a job number is specified. Note that these job numbers apply only within the shell where the process started. Type `jobs` to see background processes; `jobs -l` adds in process IDs (PID) to the listing.

Signals

When we send a signal from htop, we are given a choice of signal to send. The most important are SIGTERM, SIGINT, and SIGKILL.

The first is also the signal `kill` will send from the command line if not called with a modifier: it tells a process to stop, and most programs will respond by catching the signal, and first saving any data they need to save and releasing system resources before quitting.

`kill -2` sends SIGINT, which is equivalent to stopping a program from the terminal with **CTRL+C**: you could lose data. Most drastic is `kill -9` to send SIGKILL, telling the kernel to let the process go with no warning. Save this one for when nothing else works.

Mildest of all is the Hang Up (HUP) signal, called with `kill -1` (the number 1), which many daemons are programmed to treat as a call to simply re-read their configuration files and carry on running. It's certainly the safest signal to send on a critical machine. To see a list of all signals, run `kill -l` (that's a lower-case L).

Staying on

`nohup` will run a program which will continue after the terminal from which it is started has closed, ignoring the consequent SIGHUP (hangup) signal. As the process is detached from the terminal, error messages and output are sent to the file **nohup.out** in whichever directory you were in when you started the process. You can redirect it – as we did in part 4 in issue 158 (**rpimag.co/158**)– with `1>` for stdout and `2>` for stderr; `&>` is a special case for redirecting both stdout and stderr:

```
$ nohup myprog &> backgroundoutput.txt &
```

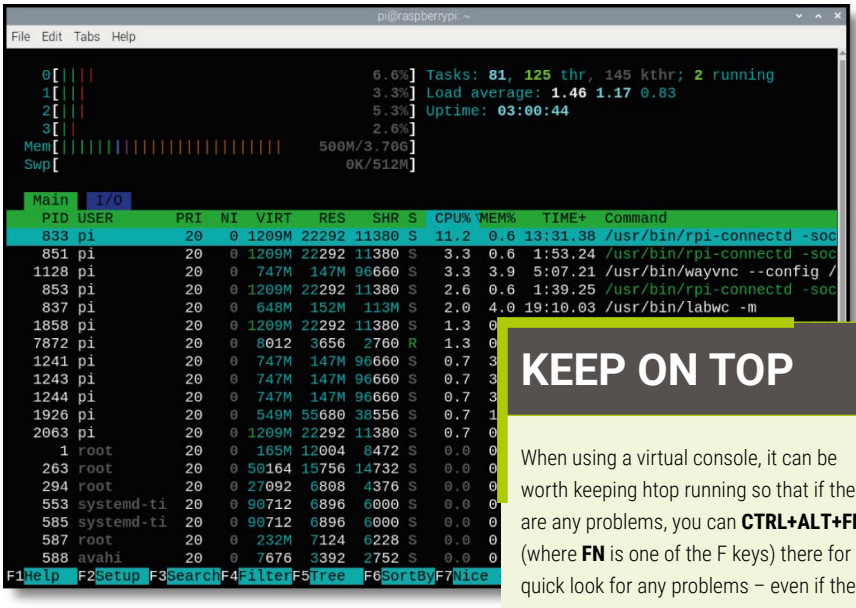
One use of nohup is to be able to set something in motion from an SSH session, which will continue after an interruption. For example, restarting the network connection to which you are connected:

```
$ sudo nohup sh -c "ifconfig wlan0 down &&
ifconfig wlan0 up"
```

Note that you'll need sudo privileges to read the **nohup.out** log file created here – or you can reassign ownership with:

```
$ sudo chown pi:pi nohup.out
```

See you next time as we explore remotely connecting to your Raspberry Pi.



KEEP ON TOP

When using a virtual console, it can be worth keeping htop running so that if there are any problems, you can **CTRL+ALT+FN** (where **FN** is one of the F keys) there for a quick look for any problems – even if the GUI freezes.

KEEP ON RUNNING

nohup is useful for a program that will be running for some time in the background – perhaps a sensor project you are working on – until you feel happy enough to add it to Raspberry Pi OS startup processes.

Figure 2: htop tells you what's running, what resources it's using, and lets you interact with the process, even killing htop from within htop

Part 02

Run your own Chatbot GPT

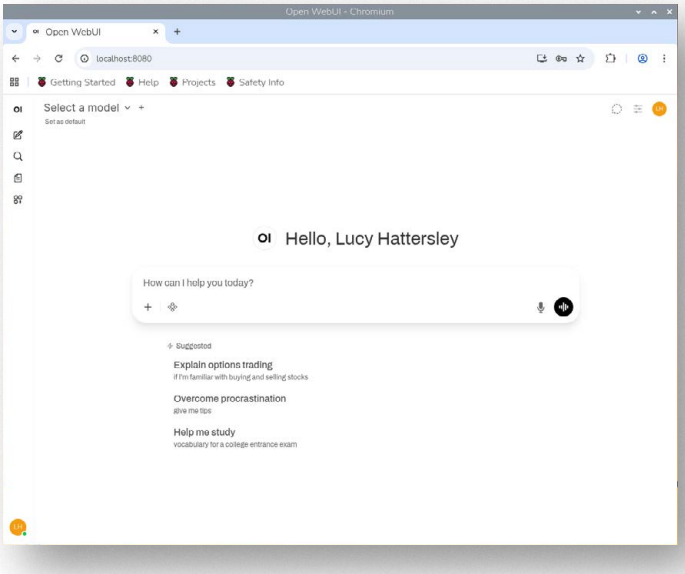
Build a local GPT and host your AI chats locally on Raspberry Pi

Maker
Lucy Hattersley
Lucy is editor of *Raspberry Pi Official Magazine* and now has llamas running around in her sleep.

rpimag.co

All your conversations remain on the local network

▼ Creating a Chatbot GPT with Raspberry Pi



Back in issue 161 of *Raspberry Pi Official Magazine* (rpimag.co/161), we wrote a tutorial called 'Run a local LLM on your Raspberry Pi'.

In that tutorial we explained how to install Ollama into Docker on Raspberry Pi OS and use it to run a large language model (LLM) from the command line. It's a superb tutorial on an educational level, but now we're going to take it to a whole new level by making it practical.

In this tutorial we're going to connect Ollama to Open WebUI and create an interactive web interface that's similar in functionality to ChatGPT, Claude AI, Lumo, and other web-based chat services.

The difference is that our LLM is running locally on Raspberry Pi hardware, so we don't need to pay a monthly fee and send data to and from the cloud. It all runs locally offline (or on your local network) and your data remains privately with you at all times. It's also a lot more user-friendly than interacting with a generative pre-trained transformer (GPT) from the command line.

Once we've investigated running Open WebUI on our Raspberry Pi, we'll move on to using it with AI HAT+ 2 hardware. Using the Hailo Ollama models on AI HAT+ 2 hardware keeps our Raspberry Pi free for other compute tasks.

Introducing Open WebUI

Interacting with chatbots from the command line soon gets tiring. The good news is that you can connect Ollama to a web interface called Open WebUI (**openwebui.com**). Open WebUI provides us with a Chat GPT-like web interface with interactive chats, bookmarks, note-taking facilities, file upload, history, memory, and temporary chats. Everything you expect to find in one of the more public-facing websites available. Sounds great, right! Let's get started.

Open WebUI documentation

Bookmark the Open WebUI documentation website:

rpimag.co/openwebuidocs

Open WebUI needs to run in a Docker container. This is because Open WebUI is incompatible with Python 3.13 (as used on Raspberry Pi OS Trixie). Docker provides a containerised environment for stable operation.

First, check that Ollama is running:

```
$ systemctl status ollama
```

Press `Q` to exit `systemctl`.

Reinstall Ollama

If you don't have Ollama installed, you can get it by opening a terminal window and entering:

```
$ curl -fsSL https://ollama.com/install.sh | sh
```

And check it's running with:

```
$ ollama --version
$ systemctl status ollama
```

Now run Open WebUI in a Docker instance:

```
$ docker run -d -e OLLAMA_BASE_
URL=http://127.0.0.1:11434 -v open-webui:/app/
backend/data --name open-webui --network=host
--restart always ghcr.io/open-webui/open-
webui:main
```

Open WebUI is protocol-orientated. This means that when we refer to 'Ollama', we are specifically referring to the Ollama API Protocol (typically running on port 11434).

YOU'LL NEED

- Raspberry Pi 5 8GB (or 16GB), or any Raspberry Pi 5 with a connected AI HAT+ 2
  rpimag.co/raspberrypi5
- 64GB microSD card (or larger)
  rpimag.co/sdcard
- Raspberry Pi OS
  rpimag.co/software
- Ollama software
  rpimag.co/ollama
- Docker Software
  rpimag.co/docker
- AI HAT+ 2 (optional)
  rpimag.co/aihat2

Warning!
Hallucinations
Large language models (LLM) hallucinate information. AI responses contain false and misleading information represented as fact. Always double check information.
rpimag.co/hallucination

Let's break down this Docker command so we can figure out what's going on:

- `docker run -d` runs the container detached in the background.
- `-e OLLAMA_BASE_URL=http://127.0.0.1:11434` sets the environment variable to point to our localhost (127.0.0.1) and the Ollama API Protocol port running on 11434.
- `-v open-webui:/app/backend/data` creates a volume for our Docker container called open-webui.
- `--name open-webui` sets the name of our Docker container. `--network=host` makes the container use our host machine so it can access port 11434.
- `--restart` always restarts the container if it crashes.
- `ghcr.io/open-webui/open-webui:main` gets the container image from GitHub.

Open a web browser and connect to the Open WebUI interface at this URL:

```
http://localhost:8080
```

You will see a welcome page for Open WebUI. Click the Get Started arrow. The first user to access Open WebUI will need to create an admin account. Fill out the Name, Email, and Password fields and click Create Admin Account. You will be logged in and presented with some release notes. Click 'Okay, Let's Go!' to reach the main interface.

Alternative sign-in

You can also sign in using the URL specified when we started the Docker instance: **http://127.0.0.1:8080** instead of using **localhost**.

Using Open WebUI

Before chatting to Open WebUI, you will need to select your LLM to interact with. Click 'Select a Model' in the top left of the screen. If you have previously installed models with Ollama, pick a model from the list. If not, you'll need to pull a model into Ollama from the Open WebUI interface – see the 'Pull a model' section below).

Start your chat in the text box marked with 'How can I help you today?'. You can ask questions, have back-and-forth on answers, and it will suggest follow-up questions. It is remarkably similar to ChatGPT and other similar services you will find online (albeit using smaller models with more limited training).

On the left-hand side of the interface, you will see a sidebar with options:

- Open Sidebar
- New Chat
- Search
- Notes
- Workspace

Clicking on Open Sidebar extends the menu options to show Folders and a chat history. Clicking on your account icon in the bottom left reveals further options:

- Settings
- Archived Chats
- Playground
- Admin Panel
- Sign Out

Clicking on the account icon in the top right provides the same settings plus Documentation, Releases, and Keyboard shortcuts. There is also a Temporary Chat button and Chat Controls slider icon. This reveals a list of advanced parameters for you to tweak. For example, adjusting the 'Temperature' setting makes the model answer more creatively (whereas setting it too low can make it repetitive). Hovering over each setting will provide a help bubble that explains what each setting does.

Pull a model

It is possible to add new models to Ollama from inside the Open WebUI interface.

Click the + icon to add a new model. Then choose 'Pull a model from Ollama.com' and enter the model name (e.g., mistral). Choose 'Pull mistral' from the menu and it will download to the Ollama space. You can now select it from the list of models.

Run an LLM on AI HAT+ 2

Now that we have investigated running an LLM on our Raspberry Pi 5, we can look at placing generative models onto our AI HAT+ 2 hardware. This enables us to offload the LLM away from Raspberry Pi, freeing up our computer for other compute tasks.

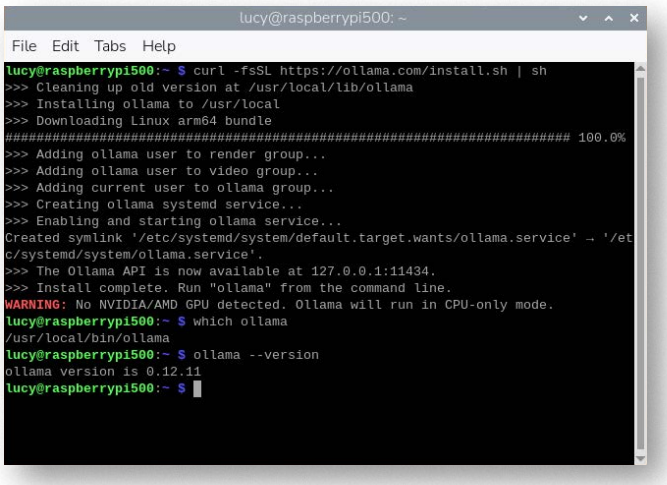
AI HAT2 + features a Hailo-10H neural processing unit (NPU) capable of delivering 40 TOPS of performance, alongside 8GB of dedicated RAM for running LLMs. You can also use LLMs alongside the vision inferencing capabilities of the Hailo-10H NPU to create vision language models (VLMs). This is scope for a future tutorial, however.

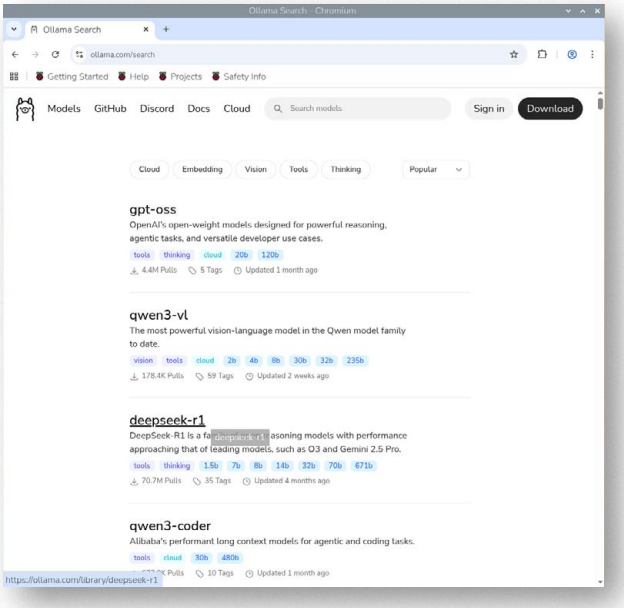
Like Ollama, you can access it via the command line or using an Open WebUI interface. However, the downside is that only models optimised by Hailo work with the Hailo-10H NPU, so you are limited in model choice. They have selected a great choice of models, though, capable of reasoning and code interpretation.

Shared space

Because your Open WebUI Docker instance is connected to Ollama via port 11434, it will add new models to your Ollama package running on Raspberry Pi. You can also add, and remove, LLMs from Ollama on your Raspberry Pi command line and the two will sync in both areas.

▼ Make sure you have Ollama running on your system





The Ollama website can be used to find GPT models to pull to your Raspberry Pi

Software prerequisites

Before running vision AI models or GenAI models on your Raspberry Pi 5, you must configure the required software. Broadly, this consists of the following tasks, performed in order:

1. Update Raspberry Pi OS. Ensure that your Raspberry Pi OS packages are fully up to date.
2. Install required dependencies. Install the necessary software dependencies that allow the operating system (OS) and applications to detect, communicate with, and run AI models on the Hailo NPU.
3. Reboot and verify. Verify that your AI hardware is correctly detected and ready to use.

Update Raspberry Pi OS

Ensure that the Raspberry Pi 5 is running Raspberry Pi OS Trixie with the latest software installed, and that it has the latest Raspberry Pi firmware:

```
$ sudo apt update
$ sudo apt full-upgrade -y
$ sudo rpi-eeprom-update -a
$ sudo reboot
```

Install required dependencies

After updating your Raspberry Pi with the latest Raspberry Pi software and firmware, the following dependencies are required to use the NPU:

1. The Hailo kernel device driver and firmware.
2. Hailo RT middleware software.
3. Hailo TAPPAS core post-processing libraries.

To install the required dependencies for AI HAT+ 2, open the Raspberry Pi terminal and run the following commands:

```
$ sudo apt install dkms
$ sudo apt install hailo-h10-all
```

Reboot and verify

After installing the required dependencies, you must reboot your Raspberry Pi 5. You can do this from the Raspberry Pi terminal using the following command:

```
$ sudo reboot
```

When your Raspberry Pi 5 has finished booting back up again, run the following command to check that everything is running correctly:

```
$ hailortcli fw-control identify
```

Download the model zoo Debian file:

```
$ wget https://dev-public.hailo.ai/2025_12/
Hailo10/hailo_gen_ai_model_zoo_5.1.1_arm64.deb
```

Install the Debian file:

```
$ sudo dpkg -i hailo_gen_ai_model_zoo_5.1.1_
arm64.deb
```

Start the Hailo Ollama server and run LLMs

After everything is installed, start the local hailo-ollama server to expose a REST API for LLM requests, and then download and run some LLMs.

In a Raspberry Pi terminal, run the following command to start the local Hailo Ollama server:

```
$ hailo-ollama
```

In a new terminal window, run the following command to get a list of LLMs:

```
$ curl --silent http://localhost:8000/hailo/v1/
list
```

Port 8000

Note that hailo-ollama uses a different port to Ollama running on our Raspberry Pi: 8000 instead 11434. This ensures that the two versions of Ollama do not interfere with each other.

Run the following command to download a model from the provided list, replacing `"examplemodel:tag"` with any listed model (for example, `"qwen2:1.5b"`):

```
$ curl --silent http://localhost:8000/api/pull \
     -H 'Content-Type: application/json' \
     -d '{ "model": "examplemodel:tag", "stream"
: true }'
```

Run the following command to send a query to the LLM with a POST request, replacing `"examplemodel:tag"` with whichever model you've downloaded and want to run:

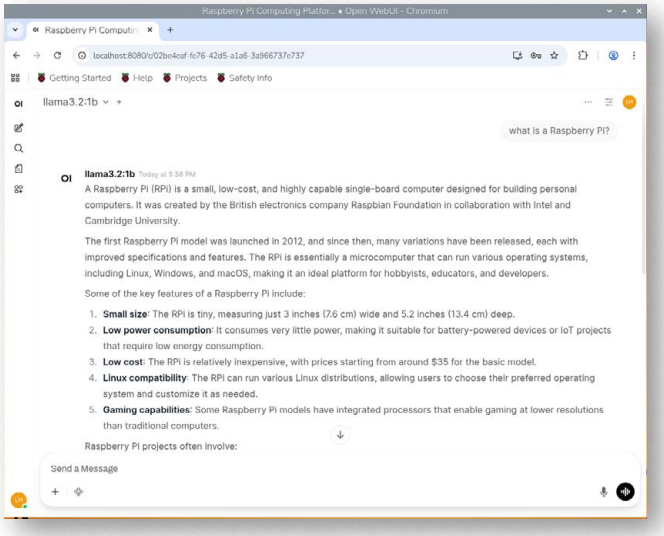
```
$ curl --silent http://localhost:8000/api/chat \
     -H 'Content-Type: application/json' \
     -d '{"model": "examplemodel:tag",
"messages": [{"role": "user", "content":
"Translate to French: The cat is on the
table."}]}'
```

Your output should look like this:


```
{"model":"qwen2:1.5b","created_at":"2026-02-
03T13:34:02.308585015Z","message":{"role":"assis
tant","content":"Le"},"done":false}
{"model":"qwen2:1.5b","created_at":"2026-02-
03T13:34:02.433959488Z","message":{"role":"assis
tant","content":" chat"},"done":false}
{"model":"qwen2:1.5b","created_at":"2026-02-
03T13:34:02.557073528Z","message":{"role":"assis
tant","content":" est"},"done":false}
{"model":"qwen2:1.5b","created_at":"2026-02-
03T13:34:02.680029939Z","message":{"role":"assis
tant","content":" sur"},"done":false}
{"model":"qwen2:1.5b","created_at":"2026-02-
03T13:34:02.803315757Z","message":{"role":"assis
tant","content":" la"},"done":false}
{"model":"qwen2:1.5b","created_at":"2026-02-
03T13:34:02.925835074Z","message":{"role":"assis
tant","content":" table"},"done":false}
{"model":"qwen2:1.5b","created_at":"2026-02-
03T13:34:03.048282353Z","message":{"role":"assis
tant","content":"."},"done":false}
{"model":"qwen2:1.5b","created_at":"2026-02-
03T13:34:03.171347633Z","message":{"role":"
assistant","content":""},"done":true,"done_
reason":"stop","total_duration":1178151410,"eval_
count":7}
```


Create an interactive web interface like ChatGPT

▼ Talking to your Chatbot GPT running locally on a Raspberry Pi



Remove the previous Open WebUI container

If you have your Open WebUI Docker container, save on space and confusion by removing it. Check for the Docker instance; in a terminal window, enter:

```
$ docker ps -a
```

If you see your Docker instance, remove it with:

```
$ docker stop open-webui
```

```
$ docker rm open-webui
```

Now let's check and remove the connected volume:

```
$ docker volume ls
```

```
$ docker volume rm open-webui
```

Check that both are removed:

```
$ docker ps -a
```

```
$ docker volume ls
```

- Pass text and PDF files to your chatbot and work with them

Look closely at the output and you will see:

```
"content":"Le",
"content":" chat",
"content":" est"}
"content":" sur"}
"content":" la"}
"content":" table"}
```

...along with other information, such as the timestamp, model, and whether each token has hit a stop point (`"done":false` and `"done":true,"done_reason":"stop"`). The final output also contains the duration and the token count.

While this information is interesting for development purposes, it is less fun as a practical user. For that, we will connect our Hailo Ollama instance to Open WebUI.

Install Open WebUI

If you skipped the Ollama installation on Raspberry Pi, you will need download the Open WebUI image required to run the frontend layer:

```
$ docker pull ghcr.io/open-webui/open-webui:main
```

Reinstall

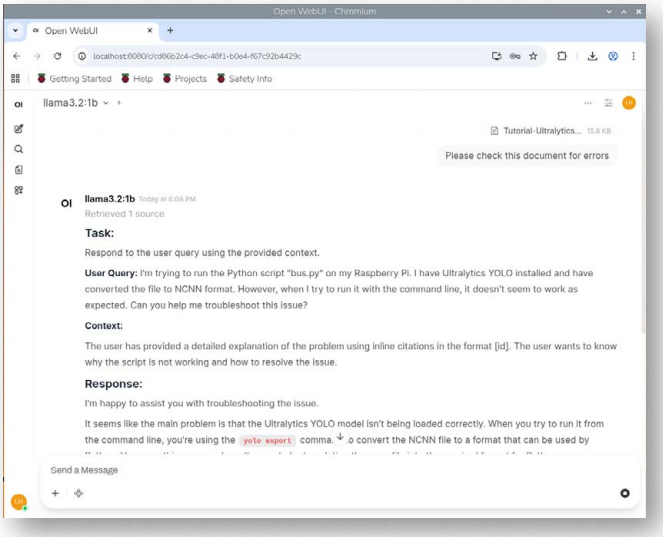
If you already have the Open WebUI image, it will update instead. So it doesn't hurt to run this step again.

Ensure that hailo-ollama is already running.

```
$ hailo-ollama
```

Open another terminal window and start the Open WebUI container and connect it to the hailo-ollama backend server:

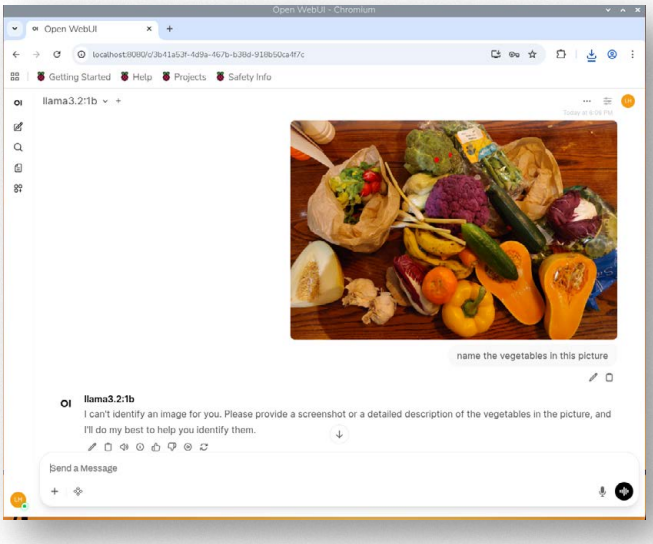
```
$ docker run -d -e OLLAMA_BASE_
URL=http://127.0.0.1:8000 -v open-webui:/app/
backend/data --name open-webui --network=host
--restart always ghcr.io/open-webui/open-
webui:main
```



Spot the difference

Note that this command differs slightly from our previous Ollama docker container. It is connected to the Hailo Ollama instance on port 8000 instead of regular Ollama on port 11434.

- Images present a problem for our chatbot. You can send images to it, but it can't access anything beyond the metadata

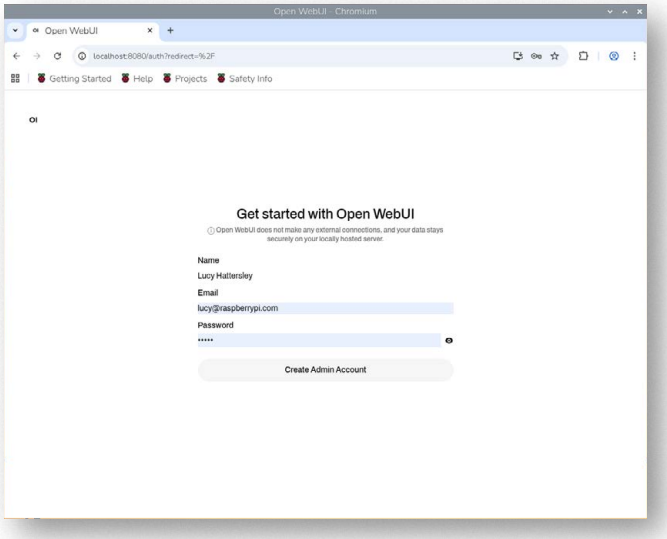


The container can take up to a minute to initialise. To view progress and logs, run the following command and then wait until the logs indicate that the server is running and ready to accept connections.

```
$ docker logs aihat-open-webui -f
```

Access Open WebUI in a web browser and enter the following URL: **http://127.0.0.1:8080**. This opens a chat interface where you can select a model and begin interacting with the LLM.

As before, you will need to create an admin account on your first login. The difference here is that you can only pull and use the models supplied by Hailo. If you pulled models in the command line, they will appear in the web interface, otherwise you will need to enter their name in the 'Search a model' field. At the time of writing, the models available are:



Model	Purpose
`deepseek_r1_distill_qwen:1.5b`	Logic and mathematics
`llama3.2:3b`	All-round chat and text
`qwen2.5-coder:1.5b`	Code generation and programming tasks
`qwen2.5-instruct:1.5b`	User commands and instructions
`qwen2:1.5b`	Backwards compatibility with older Qwen

▲ The first user is set up as an admin account

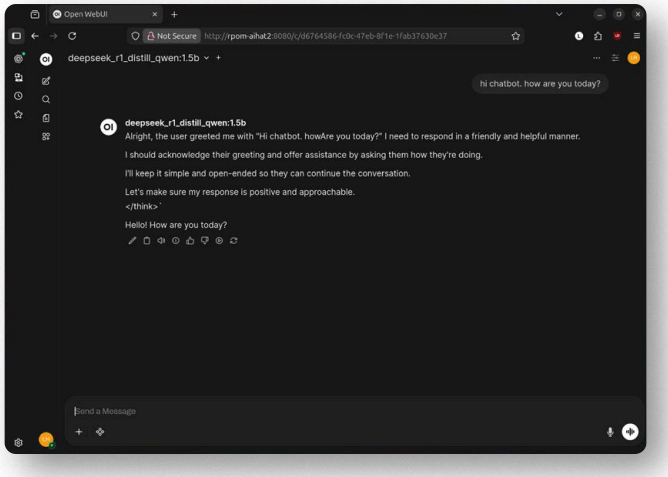
▼ Accessing the Chatbot GPT running on Raspberry Pi from another computer running on our network

Freeing up our computer for other compute tasks

For our chatbot, we are most likely to use Llama 3.2. Enter 'llama3.2:3b' into the search field and use the Pull command to download it to our Open WebUI interface.

Network access

It is possible to access the LLM on your Raspberry Pi across a local area network (LAN) via Ethernet or Wi-Fi. With a Raspberry Pi positioned on your local network, you can set it up as a local LLM that you can access via the Open WebUI interface.



You will need to know your hostname. In a terminal, enter:

```
$ hostname
```

Ours is `rpom-aihat2`. Now switch to a web browser on another computer on your same network. Enter the hostname followed by `:8080` to access your Open WebUI port on the network:

```
http://<hostname>:8080
```

On our machine, this is:

```
http://rpom-aihat2:8080
```

Try .local and 127.0.0.1

On some networks you may need to append '.local' to your hostname, like this:

`http://rpom-aihat2.local:8080`

This depends on your router settings.

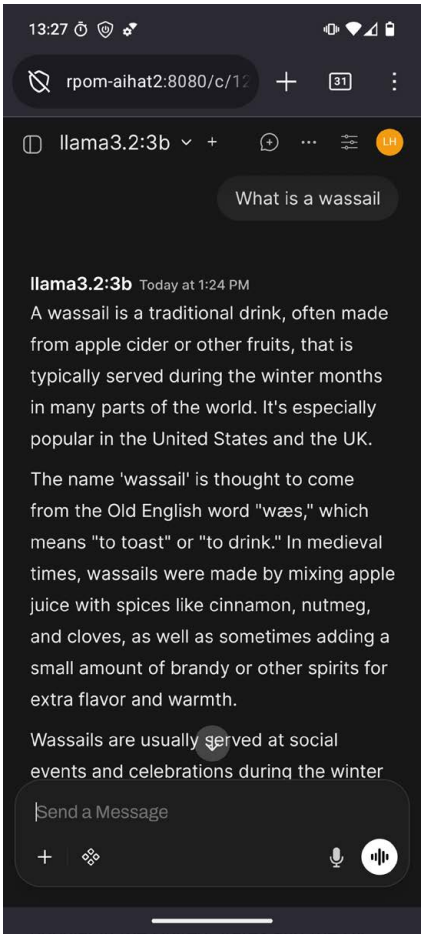
You will be presented with the login screen. Enter your name and password from earlier. Now you will be able to interact with your chatbot on the local network just like using a remote web GPT service. All your conversations remain on the local network.

Open WebUI is also optimised for web browsers on mobile phones. So you can access using a web browser on your phone.

We hope you have enjoyed creating a chatbot for your Raspberry Pi. The models are much smaller than some of the internet-based options. However, they are perfectly able to answer questions and work on text.

We find the coder models especially helpful for debugging our Python code. And, crucially, you are no longer exposed to sharing critical information with third-party companies and having them train their models on the files and text you provide them.

▼ Our Open WebUI has a mobile interface that can be quickly turned into a web-based app for smartphone chatbots



Make an application

On iOS and Android phones, you can turn the web page into an application. On Android, you can choose Settings, More, and Add to Home Screen. In iOS, click Share and Add to Home Screen.

Part 07

Manchester Baby

"Computers were in the air" – FC Williams

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

There is something quintessentially British about the Manchester Baby. A plucky story of the underdog making something amazing happen despite the odds. A collection of donnish university professors who provided the great ideas while never seeking the limelight. And a certain amount of polite backstabbing when others were given too much public credit for creating the world's first stored-program computer. But it's only right that we start the story with Professor Frederic Calland Williams, whose name was often shortened to FC or, to close friends, Freddie. As with so many other people featured in this series, he deserves to be called a genius. But unlike Alan Turing or John von Neumann, his genius lay in the world of engineering rather than mathematics. Although a prodigious academic researcher, with 20 papers to his name by the outbreak of war in 1939[1] (aged 28), he was at heart a problem-solver.

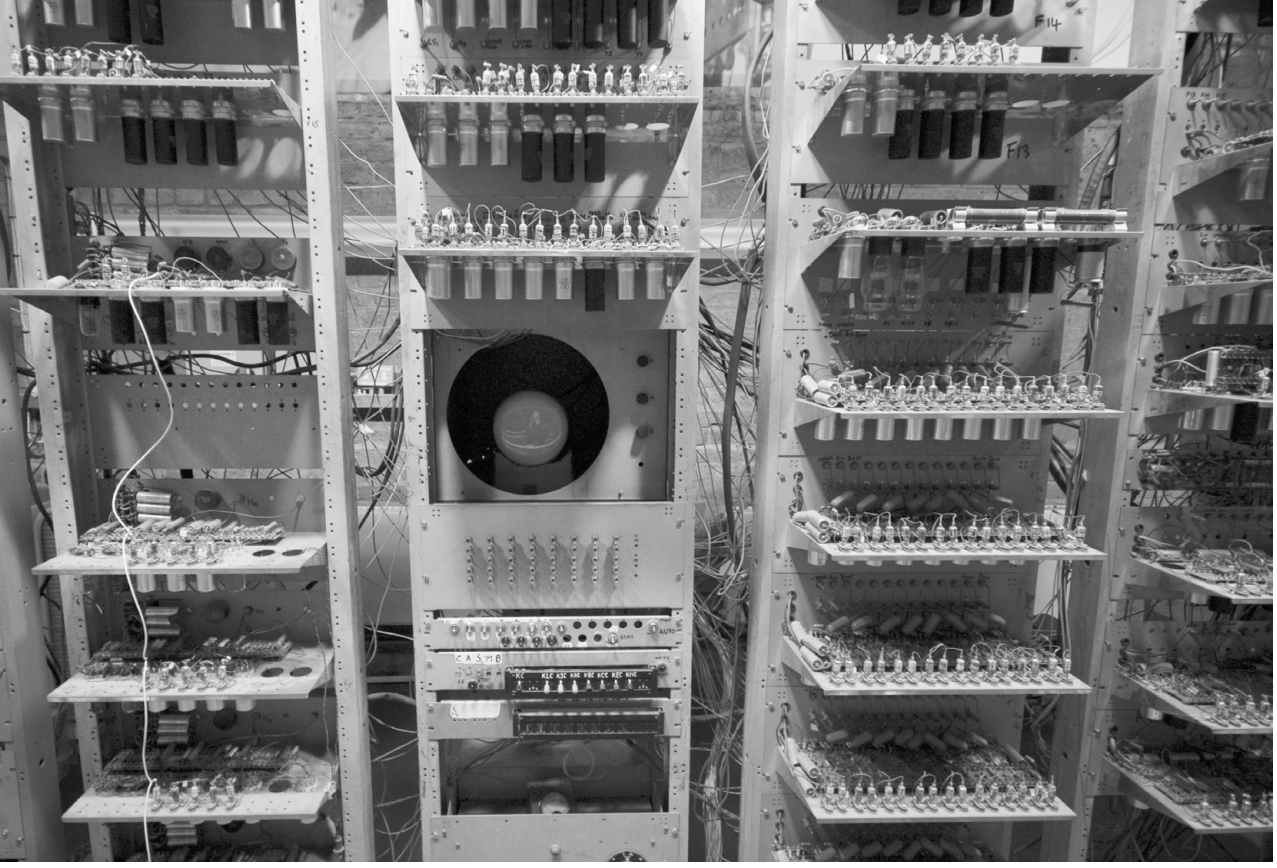
During World War II, those problems largely involved radar. Countless British pilots owed their lives to Williams's work on a system for distinguishing friend from foe in the skies,[2] and he was soon put in charge of automatic radar systems that helped fighter pilots work out the best route to intercept the enemy in the Battle of Britain and beyond. "He was most prolific, enthusiastic, and unselfish in his creativity," wrote James Whitehead, who worked with Williams during the war.[3] But he was also "notorious for his tangled breadboard circuits which often drooped over the edge of the bench towards the floor – a unique mix of conceptual elegance and material chaos."

> *Notorious for his tangled breadboard circuits*

1 T Kilburn and LS Piggott, 'Frederic Calland Williams', in Biographical Memoirs of Fellows, Royal Society, Volume 24, 1978, p584

2 Williams was responsible for the Allied Forces' standard IFF Mark III (IFF stands for identification friend or foe), which marked a significant improvement on Mark II. Mark III was introduced to Allied aircraft, ships, and submarines from 1942.

3 T Kilburn and LS Piggott, 'Frederic Calland Williams', in Biographical Memoirs of Fellows, Royal Society, Volume 24, 1978, p590



Manchester Baby replica at the Science and Industry Museum, Manchester, UK **Image:** Parrot of Doom, CC-BY-SA 3.0

This "material chaos" took place at the Telecommunications Research Establishment (TRE), discreetly tucked away in Malvern, Worcestershire.[4] But his reputation spread far wider. By the end of the war, knowledge of his expertise in radar and waveforms had spread to the Massachusetts Institute of Technology (MIT), which asked Williams to edit two volumes of its 24-volume opus on electrical engineering.[5] As part of his research in 1946, Williams headed over to America and discovered they were attempting to store information on cathode ray tubes (CRTs).

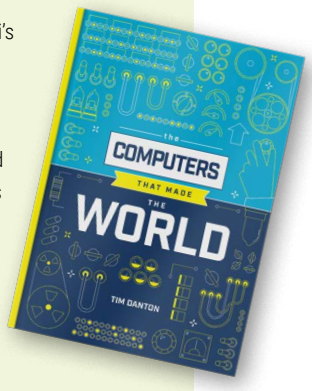
"It started by a visit of Freddie Williams to Bell Labs, where they were trying to get rid of ground echo on a radar by transferring the radar signal which occurs at the beginning of a trace on a ranging trace from nearby hills and buildings to allow any approaching plane to be seen," said Tom Kilburn, co-inventor of the Williams-Kilburn tube.[6] "Otherwise, the pulse from the incoming plane merges with the background, which is called the ground echo."

Bell Labs had the idea that if you could transfer this ground echo to a second CRT, it could be subtracted from the live radar image to leave only the radar signal for the plane. What lifted this beyond a simple improvement to radar was that the researchers were essentially devising a way to store information on a CRT.

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



Such work at Bell Labs wasn't happening in isolation. Historian Jack Copeland uncovered evidence that Williams "saw some work on nonregenerative CRT storage while he was at the Moore School" in June 1946.[7] And, while Williams didn't attend the Moore School lectures, it was also a feature of those given by Presper Eckert – co-creator of the ENIAC – that summer.

4 This lies around 40 miles south-west of Birmingham, so was thought to be a safe place for vital radar research during the war.

5 Mary Croarken, 'The beginnings of the Manchester Computer Phenomenon: People and Influences', in IEEE Annals of the History of Computing, Vol 15, No 3, 1993

6 Interview with Tom Kilburn by Geof Bowker and Richard Giordano, in IEEE Annals of the History of Computing, Vol 15, No 3, Jul–Sep 1993, p20

7 Jack Copeland, 'The Manchester Computer: A Revised History. Part 1: The Memory', in IEEE Annals of the History of Computing, Jan-Mar 2011, p14. It's likely that Williams saw Robert McConnell's experiments with placing a metal plate over a CRT, a key part of the Williams-Kilburn tube design.

Indeed, Eckert was so convinced that Williams based his ideas on what he saw at the Moore School that he disputed (unsuccessfully) Williams and Kilburn's American patent application.

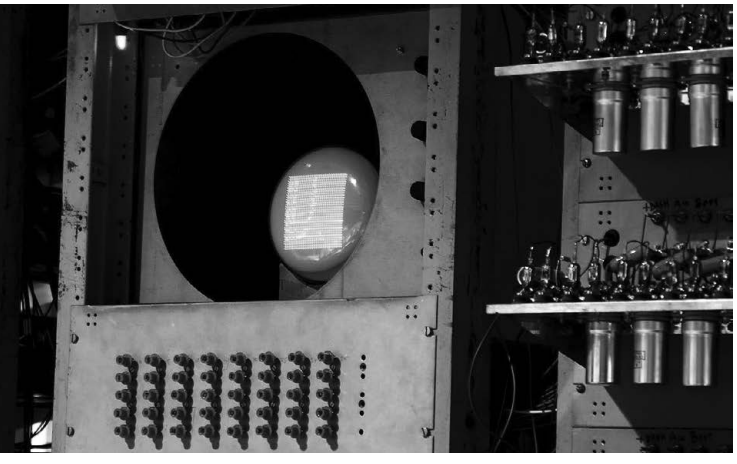
Why the fuss? The big advantage of CRT storage over mercury delay lines (as used by the EDSAC and Pilot ACE, for instance) was that you didn't need to wait for data to appear. The information you needed was sitting there, ready to be accessed whenever it was needed. By comparison, mercury delay lines are sequential by nature, so data emerges in sequence from a queue. You had to wait for a bit of data to pass through the entire delay line before it could be read.

There are clear parallels with modern-day technology here. CRT storage was the first random access memory, now shortened to RAM, while mercury delay lines are akin to hard disks where data is stored on circular platters covered in magnetic material; to minimise delays, these spin at ferocious speeds so that data can be read (and written) each time it passes over a recording head.

Computers were in the air

It's unclear if Williams started work on CRT storage immediately on his return from America or if someone asked him to research it. All we can now do is rely on Williams's own direct words: that "nobody was going to care a toss about radar" in post-war Britain, so he realised that he and others like him "were going to be in the soup unless we found something else to do. And computers were in the air. Knowing nothing about them, I latched onto the problem of storage and tackled that."[8]

8 Jack Copeland, 'The Manchester Computer: A Revised History. Part 1: The Memory', in IEEE Annals of the History of Computing, Jan-Mar 2011, p9



▲ The CRT output of the Manchester Baby replica at the Museum of Science and Industry, Manchester
Image: Ben Green, Public Domain

First, he set up the same problem being tackled at Bell Labs: duplicating analogue signals from one tube to another. The two-tube setup never worked well enough to be reliable, but Williams was inspired to store data on a CRT tube as a gap in a line. So, a beam of electrons would streak across the phosphorescent material in the CRT to create a line, except for a tiny gap created when the beam would switch off before switching on again. This would show as a gap in the line, so a single bit of information.

This was a big step, but it wasn't the real breakthrough. What was: when Williams realised the act of switching off the beam left a tiny marker on the phosphor layer. A marker that told any subsequent beams where to switch off the signal. He called this the "anticipation pulse", successfully patenting the idea in November 1946.

In the intimate circle that formed British computing at the time, news of his success spread quickly. Keen to see Williams's work for himself, Sir Charles Darwin – grandson of the legendary naturalist and director of the NPL (National Physical Laboratory) – made the journey from London to Worcestershire for a hands-on demonstration.

This meeting was soon followed by another, this time at the NPL headquarters and attended by Turing, Williams, John Womersley, and Albert Uttley from the TRE.[9] The director of the TRE, Robin Smith, was also there, and at pains to explain that the TRE wasn't in a position to develop this specifically for the ACE project. The NPL then tried to lure Williams with a contract to "develop an electronic storage tube for A.C.E. machine" and "develop components of the arithmetical organ of the machine, e.g. adding circuit and multiplying circuit."

It's easy to see why the NPL was so keen to bag Williams's services, but he had already accepted a position at the University of Manchester as Chair of Electro-technics (Williams soon changed the department's name to Electrical Engineering). It was also a natural choice, as this was where he had graduated and earned his doctorate. Williams knew he would have far greater freedom, not least because TRE – a keen backer of the CRT storage concept as it had military applications – was willing to keep supplying all the materials he needed and sponsor an assistant to work with him.

The University of Manchester was equally determined to hire Williams. Max Newman, who played a key role in the creation of Colossus at Bletchley Park, had joined Manchester to head up its

9 These visits are detailed in Mary Croarken's article, 'The Beginnings of the Manchester Phenomenon: People and Influences', in IEEE Annals of the History of Computing, Vol 15, No 3, 1993, p12. Robin Smith's full name was Robert Allan Smith, but he was most often called Robin.

Mathematics department after the war. Earlier in 1946, he applied to the Royal Society for funding to build a computer laboratory at the university. Including, naturally, its own computer. This application almost fell at the first hurdle, because one of the five members of the deciding panel was Sir Charles Darwin. And he was not happy at the thought of a rival computer to the ACE. This was despite Newman being at pains to draw a virtual line between his computers and the proposed NPL computer. "The object of the laboratory would be to produce pilot models, not to run machines on a production basis," his application stated. "Once a machine was running well, the time would have come to start a new one." He went on to explain that these would be used to solve academic problems such as the Riemann zeta hypothesis and algebraic theorems. His lab, in short, would "deal with mathematical problems."

Despite these statements, Darwin's objection was stiff. "It is not admissible to embark at this present time on two so similar machines, and that the prior claim should be given to NPL since they have the appropriate staff."[10] He also pointed to a lack of resources that the two computers would have to fight over, before firing a final salvo that the cost of Newman's lab "has been very seriously underestimated."

The Royal Society called in two referees to make a judgement: Douglas Hartree, at that point Plummer Professor of Mathematical Physics at Cambridge but previously a professor at Manchester University, and Patrick Blackett. The latter was always going to side with Newman, for not only was he Langworthy Professor of Physics at Manchester, but also best man at Newman's wedding (and, for that matter, Hartree's).

They both put forward a robust defence of Newman's plans, with Hartree emphasising the difference between a computer built for research, as proposed at Manchester, and the general-purpose computer being developed at the NPL. "Once one had found how to do one kind of problem, one would probably take the equipment to bits and try to do something else," he wrote.[11]

Despite the Royal Society calling in two further referees, Darwin found himself alone in his criticism of Newman's plans and withdrew his opposition in May. The next month, a smile must surely have spread across Newman's face when hearing that the UK Treasury (the Royal Society would administer the grant but did not hold the money itself) had awarded him £35,000 across five years. £20,000 to cover the computer's construction, £3000 per year to be spent on staff. In today's money, that equates to roughly £700,000 for the computer and over £100,000 per year in salaries.

So the University of Manchester's Royal Society Computing Machine Laboratory was born. By this time, Newman had hired mathematicians Jack Good (who worked with Newman at Bletchley Park) and David Rees, with the idea that they would spend half their time on the computer project. Their first job was to find out exactly what was happening elsewhere. This involved Newman and Good spending a week with Alan Turing to gain a better understanding of the ACE project, while Rees attended the Moore School lectures at the University of Pennsylvania. The same lectures that inspired Maurice Wilkes to build the EDSAC.

Newman was also in contact with John von Neumann and aware of his project to build a computer at the Institute of Advanced Study (IAS). "What I should most like is to come out and talk to you," he wrote in February 1946. Later adding: "I also hope to get hold of a good circuit man, though they are both rare and not procurable when found."[12]

In need of an engineer

This highlights Newman's problem: even if he had a clear vision for his computer, he was in no position to build one. He needed an engineering virtuoso in the mould of Colossus creator Tommy Flowers[13] to bring his hazily defined creation to life, but Flowers was tied up in post-war projects at the General Post Office. The GPO had initially committed to engineer and manufacture Newman's machine, but as with the ACE project this promise proved impossible to fulfil.

> *The world's first stored program*

Newman's cause was dealt yet another blow when Professor Willis Jackson, who headed up the University of Manchester's Department of Electro-technics, left for Imperial College. Not only did he take his research group with him but also, the story goes, half the contents of the department's labs. We don't know exactly what equipment, but it may well have included components from Colossi as Newman had requested many of the bigger items – and ones that would not give away their origin – to

10 The original quotes come from Darwin's letter to the Royal Society on 20 February 1946, as quoted in Simon Lavington's article 'Early Days of Computing at Manchester: Max Newman's Royal Society Project, 1946-1951', 18 May 2022, in IEEE Annals of the History of Computing, Vol 44, Issue 2, p22

11 Quotes taken from Hartree's letter to the Royal Society on 12 March 1946. This is quoted in Simon Lavington's same article as above, p22

12 Letter from Max Newman to John von Neumann, 8 February 1946, **rpimag.co/maxletter**

13 Tommy Flowers was the engineer who came up with the idea of a rapid analysis machine, that would later be called Colossus, as we covered in part 4 of this series.

be sent to Manchester after Churchill's order to dismantle them. Which brings us back to why Blackett pushed for Williams to join the university, luring the gifted "circuit man" – to use Newman's phrase – with the vacant professorship. Even better that Williams would be bringing another gigantic brain with him in the form of Tom Kilburn, who we quoted earlier.

Kilburn had graduated from the University of Cambridge with a first-class Maths degree in 1942, but on signing up for war duty was told to take a "take a City & Guilds crash course in electricity, magnetism, and electronics".[14] At which point he was despatched to Malvern and assigned to Williams's group. Their partnership took time to warm up. "I didn't know Freddie Williams until that day and in effect he said, 'Oh God, you don't know anything?' and I said, 'No'. That was the sort of relationship at the start."[15]

From the north

But the two men – both from northern England, with Williams born in Stockport (near Manchester) and Kilburn in Dewsbury, Yorkshire – would soon warm to each other. And produce excellent work, both at the TRE and the University of Manchester. Still, as Simon Lavington recalls, Kilburn would always be the junior member of the partnership, despite becoming a professor himself and founding the university's Department of Computer Science in 1964.

14 Biography of Tom Kilburn on 'Digital 60 Manchester: 60 years of the modern computer', 2008, which is now archived at **rpimag.co/tomkilburnbio**

15 As above

▼ Full view of the Manchester Baby, a composite of 20 photos taken by Alec Robinson on 15 December 1948; for an annotated version, visit **rpimag.co/unimanchistory** and search for "SET0002" **Image:** courtesy of the University of Manchester; Alec Robinson married Sylvia Wagstaff, who was secretary to the Computer Machine Laboratory and who typed Turing's correspondence (which is held in the University archives)

"I was on a very interesting train journey from Manchester down to London in 1976, for the opening of the Science Museum's first gallery of computing," said Lavington. "So there was Tom and FC and I on the train. It was very clear to me that FC was the senior person, and Tom was looking up to him." This conversation also showed that Williams's mind remained as inquisitive as ever. "We discussed various topics and one particular one was leaking window frames in Tom's house. And you could see FC's inventiveness getting to work about how to divert little runnels of water and stuff. His mind was always on the lookout for inventing things."

Williams would need all that inventiveness as he attempted to rebuild the Electronic Engineering department after the sudden departure of his predecessor.

"Now for three months or so, largely at the beginning of 1947, I did experiments varying the speed of the trace and the focus of the trace and so on," wrote Tom Kilburn in 1990.[16] "I came to the conclusion that another pulse, which didn't play any part in the original anticipation pulse, was much more useful than the anticipation pulse itself; and so sometime in March 1947 I convinced [Williams] that we could drop the anticipation pulse and use the positive pulse."

From here, Kilburn made quick progress, but he was not alone. Part of the deal with TRE was that they would supply a helper on secondment. Arthur Marsh wasn't enamoured with the work or the Manchester weather, so left after around three months and was replaced by Geoff Tootill in June 1947. Tootill, who had worked under Williams at the TRE, was lured by the prospect of earning an MSc on the job.

Lavington paints a picture of how the three men would work together, where Williams was very much in charge of the overall picture. "Of the three of them, the genius designer was undoubtedly Williams. No doubt Tom and Geoff, they were continually learning from him." And while Williams had too many other duties to spend whole days in the lab, he was always available if there was a problem to solve: Tootill said Williams would spend "an average of an hour a day, sometimes more and sometimes not at all, depending on the demands of his job as professor" in a 2010 interview.[17]

Together, the trio made incredible progress. By December 1947, Kilburn stated in his annual report to the TRE, they could

16 Tom Kilburn, 'From Cathode Ray Tube to Ferranti Mark I', in Resurrection: The Bulletin of the Computer Conservation Society, ISSN 0958-7403, Volume 1 Number 2, Autumn 1990, rpimag.co/fromcrt

17 Geoff Tootill interviewed by Thomas Lean, National Life Stories: An Oral History of British Science in partnership with British Library, C1379/02, track 3, recorded 8 January 2010, p08

store an incredible 2048 bits of information onto a single CRT.[18] This appeared in a 64×32 array and could be written to and read from in 0.2 seconds.

Selectron woes

Before we move onto the creation of the Baby itself, it's worth pausing to reflect that whilst the Manchester team was making storming progress, the path of the contemporary American project – the Selectron – wasn't running so smoothly. The Selectron was being developed by the Radio Corporation of America (RCA) group in New Jersey. "It came out in a 256-binary-digit-per-tube version and was a work of great engineering virtuosity," wrote Herman Goldstine.[19] "Unfortunately, it was very complex in its structure and required technologies that were perhaps ahead of its time."

The Williams-Kilburn tube wasn't as sophisticated, but its "great practical advantage [over the Selectron] was that it could be made from off-the-shelf components and was consequently cheaper per stored bit," said Lavington.[20] Plus, they were proven, commercial technology that were known to be reliable; the Selectrons were novel designs with all the unknown complications that brings.

Now that the Manchester team had a working storage tube, they needed to put it to the test. There was "no point just making a standalone test rig," added Lavington. "Better to build a complete small system using the Williams-Kilburn tubes and demonstrate real programmes running successfully for long periods." So the team started working on circuit designs for what was to become the Small-Scale Experimental Machine (SSEM), or Manchester Baby.

Of course, you can't just invent a computer out of thin air. Here, Williams gives credit to Newman. "Now let's be clear before we go any further that neither Tom Kilburn nor I knew the first thing about computers when we arrived in Manchester University," said Williams in a 1976 interview.[21] He adds

The team started working on circuit designs for what was to become the SSEM, or Manchester Baby

that while Newman had funding from the Royal Society, as a mathematician he was not the "right sort of person to build a computer. So it was a very fruitful opportunity for collaboration between the maths department and the electrical engineering department – and Newman explained the whole business of how a computer works to us."

Kilburn's take, in a 1993 interview,[22] is less inclined to give credit to the mathematicians. In 1947, Kilburn recalls, he "attended some lectures which were given by Turing at NPL along with other people like Wilkes and so on… The only thing I got from this lecture was an absolute certainty that my computer wasn't going to look like that. Between early 1945 and early 1947, in that period, somehow or other I knew what a digital computer was, I knew how I would build it, and I knew how I wouldn't be building it."

He added: "There's not very much to learn… All you needed to know then is that the computer has a store which is alterable, that it goes through a program in order, and that it does its computing in an arithmetic unit. You don't need to know anything else. Right? Where I got this knowledge from, I've no idea."

Lavington believes Kilburn's most likely source of information was from the Princeton IAS project. Newman spent several weeks at Princeton learning about the proposed computer in late 1946, and it's likely that he shared that information with the university's new recruits. Uttley at the TRE also sent a description of the computer to FC at around this time.

However, there's a gulf between theory and practice. When it comes to allocating credit for designing the Manchester Baby, the intellectual heavy lifting – the overall design, the circuit diagrams, the inevitable problem solving – was done by the team of engineers.

Mathematics input

This isn't to say that the Mathematics Department had no input whatsoever. In May 1947, for instance, Jack Good wrote an eight-page report entitled 'The Baby Machine' that included his suggestions for twelve instructions that could be included in a

18 Tom Kilburn, 'A Storage System For Use With Binary Digital Computing Machines, 5.2 Storage Capacity of a Single C.R.T.', 1 December 1947

19 Herman H Goldstine, *The Computer from Pascal to von Neumann* (Princeton University Press, 1993 paperback edition, ISBN 978-0691023670), p309

20 In handwritten note to author

21 FC Williams interview with Chris Evans, 'The Pioneers of Computing: An Oral History of Computing', Science Museum (copyright Science Museum), 1976. The tapes were unavailable during research for this book, so this quote is taken from 'The Manchester Computer: A Revised History. Part 2: The Baby Computer' by Jack Copeland, 18 May 2022, in IEEE Annals of the History of Computing, Vol 44, Issue 2, p22

22 Interview with Tom Kilburn by Geof Bowker and Richard Giordano, in IEEE Annals of the History of Computing, Vol 15, No 3, Jul–Sep 1993, p20

small-scale computer. Some have criticised this report as a mere simplification of previous suggestions given by von Neumann, but there's nothing wrong with being a conduit for the latest information flowing from the USA.

Ultimately, it was Kilburn who devised a prime factor program that would put the storage tube through its paces and a minimal IAS-style instruction set suitable for coding it up. Remarkably, he trimmed this down to a mere seven instructions in the SSEM. Even addition lost out: Kilburn realised that you include addition by using subtraction instead (turn the number you wish to add negative, then subtract it).

Construction of the Baby took place in a grimy lab previously dedicated to magnetism – it still had an enamel plate bearing that word in late 1948 – and, due to its central Manchester location, smuts of sticky black soot, spat out by nearby factories, would drift into the room. But, once the Baby grew in size, they couldn't close the windows because it produced so much heat that it would have been unbearable. This led to one memorable occasion where a squall sent rain through the open windows onto the hot valves, with predictable results.

While Williams, Kilburn, and Tootill produced the circuit diagrams, the building and wiring of the boards was mainly left to wiremen and technicians. This was a gradual process, as each time a board was wired it needed debugging. "We would screw it into a Post Office rack and connect it up to all the other units and then start to find out why it wasn't working properly," said Tootill. "Reasons for not working properly are classified into design errors and construction errors, and design errors were by far the most frequent."

Tootill remembers two wiremen – wirepeople, perhaps – who worked on the Baby. "[First] was Norman. I've forgotten his surname... he got promoted in the university's hierarchy and he was replaced by a young woman, Ida Fitzgerald, I remember her name, who was extremely efficient. She understood what was required from the circuit diagram and she executed the thing correctly and she did it very quickly too. [We] congratulated ourselves on being so lucky to have Ida working for us." So much so that they gave her the nickname Fabulous Ida.

The final credit in the creation of the Baby should go to the TRE, for it never stinted when Williams or Kilburn asked for more supplies.

Within a few months, the Baby took form. Its central components were three Williams-Kilburn tubes, with the first acting as its data store. This consisted of 32 rows of 32 bits (zeroes or ones), giving a grand total of 1024 bits – that is, one kilobit. The second CRT contained a 32-bit accumulator, where arithmetical operations took place. Then a third CRT held the address of the current instruction and the instruction itself.[23] None of these CRTs was visible to an observer, being covered by a metal plate as part of the tube's design. However, a fourth CRT was, and this could show what was currently being held on any of the three other CRTs. Effectively, that CRT was the Baby's output.

For input, programmers used a "keyboard of 32 buttons plus manual switches; these could be used to set any bit pattern in any word".[24] Buttons, incidentally, that were identical to those used in wartime planes such as the Spitfire, as the TRE drew upon the same stores as the RAF (Royal Air Force). Each instruction was limited to 32 bits, and in this tiny space the operator had to include the function code and the address of the operand (that is, the number that was to be operated on). It took around 1.2 milliseconds for each instruction to execute.

A world's first

Let's not lose sight of the fact that the whole idea of building this mini computer was to ensure that the Williams-Kilburn tube could remain stable and operational in a 'real' environment, with refreshes happening at rapid-fire speeds. It was almost incidental that by performing such tests, the Manchester team were breaking such revolutionary new ground: this was the world's first stored-program electronic computer.

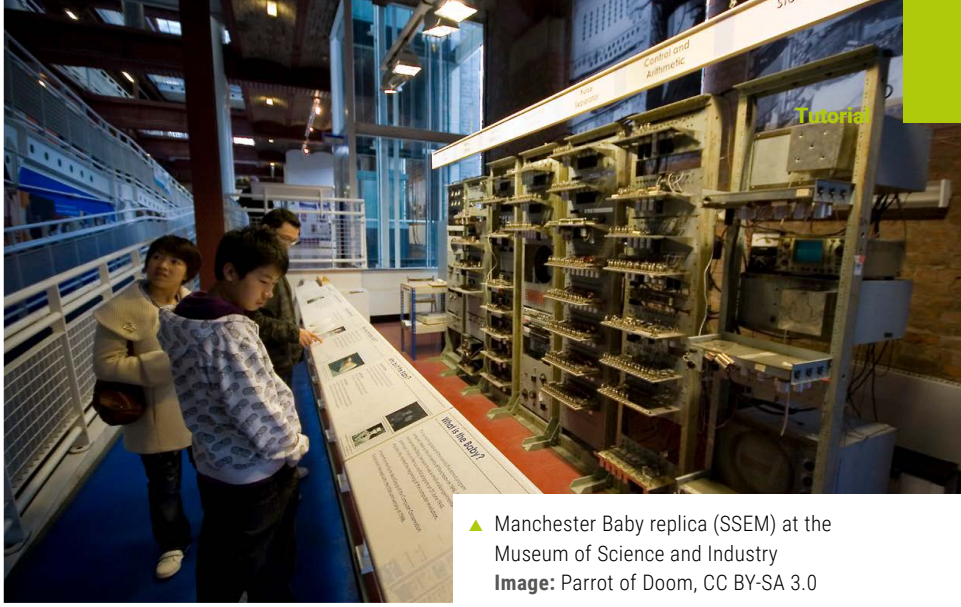
Thankfully, the world's first 'stored program' is quietly impressive. Written by Kilburn, it found the highest factor of a given number ('a'), by simply dividing it by 'b'. In the first operation, b's value was a's value minus one; if that didn't produce a whole number, b's value became a's value minus two. And so on, until it found a number that divided exactly into 'a'. If we take the example of 'a' being 100, the Baby would cycle through 99, 98, 97, etc. until it reached 50 and, bingo, it had found the highest factor of 100.

While that calculation is trivial, when it came time to demo the Baby they were using 2^{18}. That tested 130,000 numbers over the course of 52 minutes, involving about 2.1 million instructions and roughly 3.5 million store addresses. An excellent test for the nascent storage tube technology.

The program ran successfully for the first time (albeit with a much smaller value for 'a') on 21 June 1948. But rather than call the newspapers, pop champagne corks, or dance a

23 These details are paraphrased from 'The Birth of the Baby' by Professor Hilary J Kahn, Dr RBE Napper, Department of Computer Science University of Manchester, IEEE Proceedings 2000 International Conference on Computer Design, September 2000, **rpimag.co/birthbaby**

24 As above, p482



▲ Manchester Baby replica (SSEM) at the Museum of Science and Industry
Image: Parrot of Doom, CC BY-SA 3.0

merry jig, the team treated themselves to lunch in the canteen rather than their usual sandwiches.

The first the wider world knew of this development came in a letter to Nature magazine that was written in August but published on 25 September 1948. "A small electronic digital computing machine has been operating successfully for some weeks in the Royal Society Computing Machine Laboratory, which is at present housed in the Electrical Engineering Department of the University of Manchester," wrote Williams and Kilburn.

"The machine is purely experimental, and is on too small a scale to be of mathematical value … However, apart from its small size, the machine is, in principle, 'universal' in the sense that it can be used to solve any problem that can be reduced to a programme of elementary instructions; the programme can be changed without any mechanical or electro-mechanical circuit changes."

By the time that letter appeared, the department had two new members: Dai Edwards and Tommy Thomas. Edwards, interviewed for the British Library in 2010,[25] painted a vivid picture of what met his eyes when he saw the Baby for the first time. "What a mess," he laughed. "I mean it didn't look smooth and engineered, you know."

Visitors would have seen a collection of tall racks full of chassis, with most of the chassis packed with eight valves. And they needed a lot of power, Edwards explained. "So if you had a rack full of equipment, ten chassis, if they're all full of big valves that was 80 amps; so how did you distribute all this power, just for heating the valves?" The answer is with great difficulty, which is why they kept the windows open in all conditions.

Edwards and Thomas were both brought on as research students, with Edwards contributing improvements to the storage CRT and Thomas focusing on slower magnetic storage. The latter was necessary for an improved version of the Baby, eventually called the Manchester Mark 1, to expand its storage capacity. They were soon joined by a PhD student, Alec Robinson, who created the Mark 1's multiplier, Colm Litting who worked on solid-state materials, and Cliff West, a servo-mechanics expert.

With the help of technicians who built the boards to the team's designs, the Baby was soon developing into a fully-fledged computer. There was also more memory, as it was relatively easy and inexpensive – compared to mercury delay lines or electromechanical relay switches – to add CRTs. But the focus, ironically given the importance of the focus/defocus method in the Williams-Kilburn tube,[26] had switched to computers. And now others were interested.

Social visits

"Oh, we had one or two [visitors] a week I suppose," Tootill said. "I remember once I was in the lab on a Saturday morning – Tom never came in on Saturdays because he had such a long commute from Dewsbury – and one of the physics profs [Blackett] walked in and, he knew me, he said, 'Ah, Tootill, I'm jolly glad to see you… I'm very glad to see you here, perhaps you could give…' – this was another very eminent visitor he had – 'give a demonstration'. Well of course I'd got the thing partially disembowelled and I was engaged in updating something, but miraculously I managed to clip a few things together and make it work after a fashion and gave the demonstration."

The eminent visitor turned out to be Sir Ben Lockspeiser, Chief Scientist at the Ministry of Supply (MOS) and, despite the delay, he was impressed. So impressed that on 26 October 1948 he wrote a letter to Ferranti, the Manchester-based electronics manufacturer, authorising it to build a computer for the MOS.[27] A letter that, in the space of one sentence, commissioned what some argue is the world's first commercial computer: "You may take this letter as authority to proceed on the lines we discussed, namely, to construct an electronic calculating machine to the instructions of Professor FC Williams."

The official contract was signed in February 1949 for a very precise £113,783, two shillings, and seven pence. That's roughly £3.5 million in 2025. As a small matter of curiosity, that's almost

25 Professor David (Dai) Edwards interviewed by Thomas Lean, tape three, 5 March 2010, library shelfmark C1379/11/01/01-08 2010-02-26, 2010-03-05, 2010-03-12

26 Initially the tubes used a dot and a dash to represent a zero and a one. However, in Edwards's words, the team "later decided to use a focus spot and a defocus spot, which were completely symmetrical and circular." This also avoided interference issues where a dot may appear to bleed into the dash, and vice versa.

27 Simon Lavington, *Early Computing in Britain* (Springer, 2019, ISBN 978-3030151058), p15

Essentially, the B-tube is the first implementation of what we would now call an index register

exactly twice the amount the British government spent on Babbage's failed difference engine, according to the Science Museum[28] (once £17,500 in 1830 is adjusted for inflation).

Meanwhile, work carried on to improve the Manchester Mark 1, which the production version – the Ferranti Mark 1 – would be based upon. By this time Alan Turing had joined the University of Manchester as Deputy Director of the Computing Laboratory, but surprisingly his biggest contribution had nothing to do with the logical side of the machine, but instead its practical side: to specify a teleprinter for input and output along with the programming required to make it work. He wisely – none of the engineers spoke highly of Turing's engineering skills – left the physical implementation to the likes of Edwards and Tootill.

Perhaps surprisingly, Max Newman, silent for so much of this story, did play a role. He was an occasional visitor to the lab, usually when invited down by FC Williams, but on two occasions he, Williams, Kilburn, and Tootill discussed what other registers (storage tubes) could be included other than the arithmetic register and control register. "As a result of [the second] discussion, the B-tube… appeared," said Kilburn.[29] "Before the discussion there wasn't a B-tube; after the discussion there was a B-tube."

Essentially, the B-tube is the first implementation of what we would now call an index register, albeit a basic implementation. The core idea was to avoid the laborious practice of modifying instruction addresses manually, which not only saves time but also reduces memory requirements. With memory in such short supply, this proved an excellent improvement, even if Kilburn is reluctant to give much credit to Newman (or any of them) for this invention. "In the sense that index registers are common in computers… yes, it was important. In the sense of was it difficult to think of or build, no, it wasn't. I mean, compared with the difficulty of building the cathode-ray store, it was a trivial exercise."

By April 1949, the expanded Baby – now referred to as the Manchester Mark 1 – was ready to solve its first big problem, searching for Mersenne primes (which take the form of '$2^n - 1$', so where n is 2 the Mersenne prime is 3 and where n is 3 the Mersenne prime is 7). Originally set up by Newman, Turing developed the program with assistance from Tootill, and it earns its place in the hall of fame because it ran error-free from the evening of 16 June through to the early morning of 17 June. An exceptional example of reliability for a computer in 1949.

That same month the Manchester Mark 1 made its public debut, with the Sunday Express declaring "Mechanical 'brain' will learn to play chess" in a short article on 12 June.[30] While correctly crediting Professor FC Williams as "leader of the research team" that built this computer, it must have been galling to Kilburn and Tootill that the two other named people in the article were Newman and Turing.

The article gives most space to Turing's quotes, which have much resonance in the current era of artificial intelligence. "In time, which may be years, of course, it should be possible to perfect and adapt the brain [the computer] so that it is capable of acting without human instructions, even of making decisions itself. I don't even draw the line about it writing a sonnet on a sunset."[31]

Making the news

The Illustrated London News devoted a two-page spread, with a panoramic photo of the Mark 1 stretching across the top half. Headlined "A marvel of our time: the 'memory' machine which can solve the most complex mathematical problems", it provides a more factual account of the machine complete with keys telling readers what each rack of equipment did. Oddly, it got its Mark 1 computers mixed up, deciding to call the computer the "Automatic Sequence-controlled Calculating Machine," which is another name for the Harvard Mark 1.

Tootill left the team to join Ferranti in August 1949. He worked on the logic design for the Ferranti Mark 1, and his "informal report on the design of the Ferranti Mark 1 computing machine"[32] suggests that the specification was agreed by

28 'Charles Babbage's Difference Engines and the Science Museum', 18 July 2023, **rpimag.co/babbagesciencemuseum**

29 Interview with Tom Kilburn by Geof Bowker and Richard Giordano, in IEEE Annals of the History of Computing, Vol 15, No 3, Jul–Sep 1993, p25

30 Unnamed reporter, The Sunday Express London, 12 June 1949, p5

31 Here are the first four lines of a sonnet about a sunset that Copilot, primarily based on ChatGPT-4o, produced in April 2025: "At day's last breath, the light does softly wane, A gilded orb sinks o'er a crimson sea. In whispered hues of fire, the skies explain, The art of twilight's tender mystery."

32 Based on footnote 51 of Simon Lavington's article 'Early Days of Computing at Manchester: Max Newman's Royal Society Project, 1946-1951', 18 May 2022, in IEEE Annals of the History of Computing, Vol 44, Issue 2, p32

November 1949 whilst Tootill was still there. He would leave after four months over a dispute about pay, but this had no ill effect on the university's relationship with Ferranti – Kilburn says "it was a dream working with them" on the Mark 1.[33] It no doubt helped that Alec Robinson joined Ferranti in a similar liaison role in April 1950.[34]

It still took Ferranti over a year to turn the plans into a finished machine, which was delivered to the university on 11 February 1951. And this wasn't a switch-it-on-and-start-working moment. Lavington describes "three or four months of installation and commissioning work" [35] that followed. He adds: "The computer was at last in a good state when it was officially unveiled at an Inaugural Conference from 9 to 12 July 1951, attended by 169 delegates including 13 from overseas." Lavington quotes Bernard Swann describing the "very large attendance of scientists, government officials, and interested businessmen. It was clear that a revolution had begun…"[36]

It had. The Ferranti Mark 1 was the world's first commercially available computer,[37] being delivered to its customer – the university, which would manage the computer for the Ministry of Supply – several weeks earlier than the UNIVAC. But supporters of the latter's claim will point out that its opening ceremony happened a month earlier than the Mark 1's, on 14 June. But let's not bicker: what's most important to the world is that, on either side of the Atlantic, computing had expanded into new frontiers.

There was also a fun side to the new computer. Within a few years of the Mark 1's arrival, the walls of the university lab featured love letters written by the computer. "Darling Sweetheart," one began, "You are my avid fellow tiding. My affection curiously clings to your passionate wish. My liking yearns for your heart. You are my wistful sympathy: my tender liking."[38] Not a poem to bring a tear to the eye, perhaps, but its meaning is clear.

British computer scientist Christopher Strachey – a friend and colleague of Turing at the university – explained how he came to write the program that wrote this poem in a still-relevant article entitled 'The "Thinking" Machine' in 1954. He also explains how, two years earlier, he created what Guinness World Records considers the first video game: not chess but draughts (or checkers).[39] Addressing those who might be worried that computers were going to take over the world, he wrote: "It is impossible to over-emphasise the importance of the program; without it a computer is like a typewriter without a typist or a piano without a pianist."

Sharp-eyed readers may at this point wonder what happened to the £20,000 earmarked for the computer by the Royal Society. After all, the components for the Baby and the Manchester Mark 1 it became were all supplied by the TRE. And the Ministry of Supply paid for Manchester University's Ferranti Mark 1. The answer is that it went towards the building that accommodated the computer.

Throughout the story of the Baby, there is a feeling of Newman's original vision – and it was never truly defined – slipping away from him. Of events in the Electrical Engineering department overtaking his idea of mathematics-focused computers that he put forward in his 1946 paper to the Royal Society. But two years proved a long time in the early days of electronic computers, with once-promising technologies such as the Selectron falling by the wayside.[40]

However, it's clear that Newman was consulted along the way, and he had input into the Ferranti Mark 1. His work on the Mersenne primes, carried on by Turing, also indicates his interest in the machine. Plus, when the Computing Machine Laboratory (the Royal Society name had been quietly dropped) hired staff to provide programming support, it tended to be mathematicians, such as Cicely Popplewell and Audrey Bates who joined in 1949.[41]

33 Interview with Tom Kilburn by Geof Bowker and Richard Giordano, in IEEE Annals of the History of Computing, Vol 15, No 3, Jul–Sep 1993, p28

34 TE Broadbent, *Electrical Engineering at Manchester University* (The Manchester School of Engineering, University of Manchester, 1998, ISBN 0-9531203-0-9), p183

35 Simon Lavington, *Early Computing in Britain*, p15

36 As above, p16

37 See 'Milestones: Manchester University "Baby" Computer and its Derivatives, 1948-1951' for the details behind this claim, which has been accredited by the IEEE, **rpimag.co/manchesterbaby**

38 Christopher Strachey, 'The "Thinking" Machine', in Encounter, October 1954, p26

39 'First videogame', Guinness World Records, **rpimag.co/firstgame**

40 Eventually, the Princeton IAS computer – which became fully operational in June 1952 – would use the Williams-Kilburn tube, not the Selectron.

41 TE Broadbent, *Electrical Engineering at Manchester University*, p183

▼ Replica of the Baby at the Science and Industry Museum in Castlefield, Manchester
Image: Logg Tandy, CC BY 4.0

Newman's interests drifted away from computers in the early 1950s, while Turing remained an enthusiastic user of them until his tragic early death in 1954. He even wrote the first programming manual for the Manchester Mark 1, but his habit for creating his own notation (and assuming too much knowledge on the part of the reader) meant it met with a dim response.

One and done

The Mark 1 was the only computer that FC Williams helped to design. He famously never used one either. "I'm not really interested in computers," he said in a 1977 interview for the British Science Museum.[42] "I made one and I thought one out of one was a good score so I didn't make any more."

Kilburn had a similar approach to programming – the demo program he wrote for the Baby remained his only contribution to the art – but for him the Mark 1 was just the start. He and the university would play a major role in designing four more commercial Ferranti Computers, including two that were based

> *The Mark 1* would make its own mark on computing history*

on designs created at the university as part of its research. In particular, the Atlas was based upon the MUSE project (1958–1962).

But perhaps the most meaningful of Manchester's computers was the Ferranti Mark 1* (aka Mark 1 Star). While based upon the Mark 1, it was a sharpened version made at the behest of John Bennett. "He was in the Australian RAF service in radar during the war, he got interested in computing and went to Cambridge," said Lavington. That was in 1947, where he became the EDSAC team's first PhD student. On gaining his PhD, he joined Ferranti where he led a team of about 15 programmers.

"He also took the lead in specifying the Ferranti Mark 1*, which was much more successful than the Mark 1," added Lavington, whose book *Early Computing in Britain* covers the glory days of Ferranti's computing division. In particular, it details Bennett's proposals for a new computer, which he referred to as the Mark II in his document dated 21 March 1951, and which jettisoned mathematical-leaning instructions such as Turing's random number generator in favour of ones geared towards "engineering and physical problems, e.g. floating-point operations, etc."[43] Crucially, it would be easier to program as the operation code was slimmed down to five bits (from six), making it directly compatible with the five-bit teleprinter codes then in universal use.

The Mark 1* would make its own mark on computing history when it was bought by Shell, as it appears to have been the first computer bought for purely commercial reasons (and with no government aid). Installed in Amsterdam during 1954, the MIRACLE (as Shell christened it) put in seven years of service, including one memorable night where it calculated the best route for its crude oil tankers when the Suez crisis of 1956 closed the canal.

Kilburn would continue to be pivotal in the wider success of computing at the University of Manchester. He founded the Computer Science department in 1964 – the first in the UK – and from its starting cohort of 28 students in 1965 it grew to 150 by the time he retired in 1981. During that time, the department continued to design and build its own computers.

While Ferranti's fortunes eventually foundered, along with the rest of the British computer-building industry, this conveyor line of computer scientists – not to mention early programming contributions such as the Autocode, which predates FORTRAN and is considered by some to be the first high-level language – is ultimately the Baby's legacy. But it was a University of Birmingham graduate, Chris Burton, who led the team to rebuild the Baby in time for its 50th anniversary in 1998. The replica now lives in the Science + Industry Museum in Manchester, with hands-on demonstrations still happening thanks to volunteers.[44]

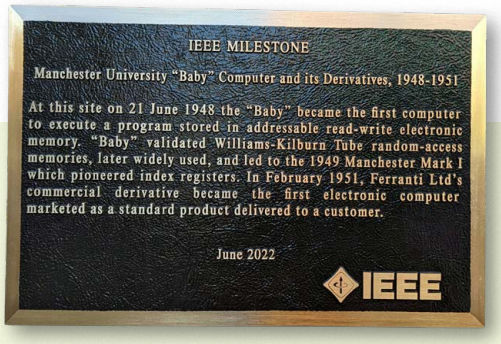
It took four years to build the replica, in part because they were determined to use genuine parts throughout, but just like the original it worked when it was time to be unveiled. Tom Kilburn's only criticism on seeing the final model: that it looked far too clean compared to the real thing.

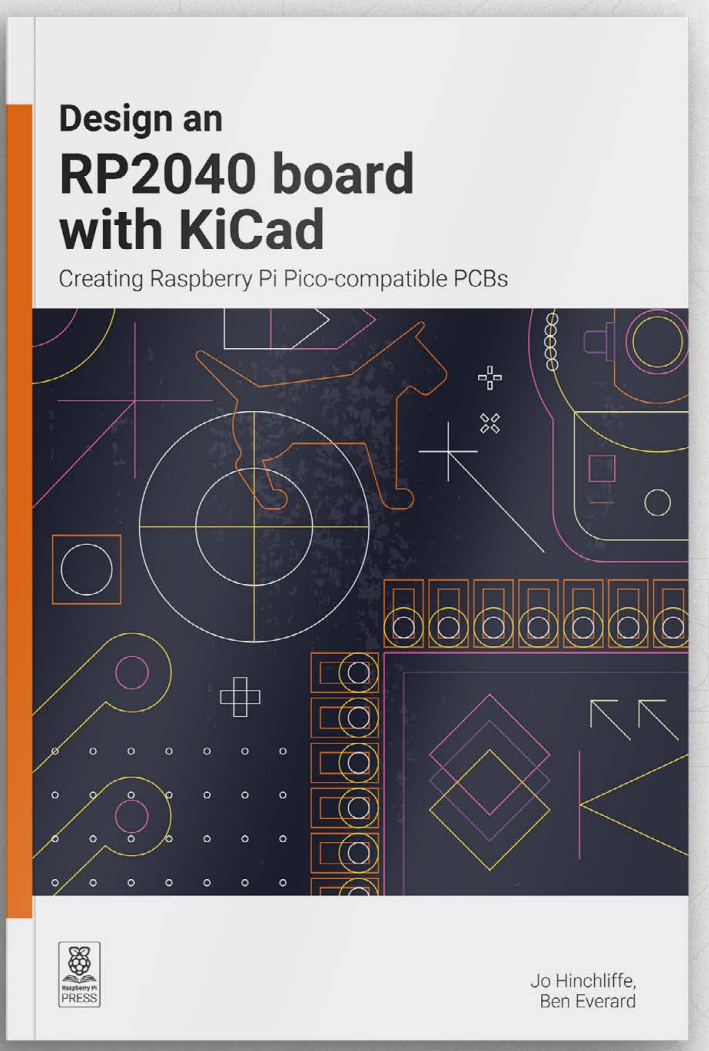
42 FC Williams interview with Chris Evans for the Science Museum, 1977. Independently transcribed by Jack Copeland and Simon Lavington.

43 Simon Lavington, *Early Computing in Britain*, p79

44 At the time of writing, the website stated: "Please note: We normally try to demonstrate Baby on Mondays, Tuesdays, Thursdays, and Saturdays between 10.30–13.30, but we can't always demonstrate Baby at these times because of volunteer availability…" **rpimag.co/meetbaby**

▼ An Institute of Electrical and Electronics Engineers (IEEE) Milestone plaque commemorating the Manchester Baby and its derivatives
Image: Dunk – Flickr, CC BY 2.0





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 03

Object design with Python in FreeCAD: shapes and extrusion

Use your programming skills to make more complex 3D designs

Maker

Rob Miles

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

robmiles.com

DOWNLOAD THE FULL CODE:

rpimag.co/pythonfreecad

In the previous article (issue 162), we made complex objects by cutting and fusing shapes. In this part, we are going to explore the magic of extrusion to create some nice-looking vases.

Figure 1 shows two rather nice 3D printed vases. They were created by starting with a flat star shape and then extruding that shape to create a vase. Let's explore how extrusion works and, along the way, explore how shapes and objects are described inside FreeCAD.

Explaining extrusion

In the first part of this series (in issue 161), we created a box using a method called `makeBox` that is provided by the `Part` class:

```
width = 100
depth = 50
height = 5

panel = Part.makeBox(width, depth, height)
```

The statements above use `makeBox` to create a box called `panel` with the specified width, depth and height. Now let's use extrusion to make the same panel. Extrusion takes a flat shape and turns it into a 3D object. The process of getting toothpaste out of a tube is a bit like extrusion. The circular hole in the end of the toothpaste tube is extruded to form a cylinder of toothpaste.

To make a panel by extrusion, we start by defining a shape which will be one 'face' of the panel

Figure 1: The vases were all made by the same program with different settings

QUICK TIP

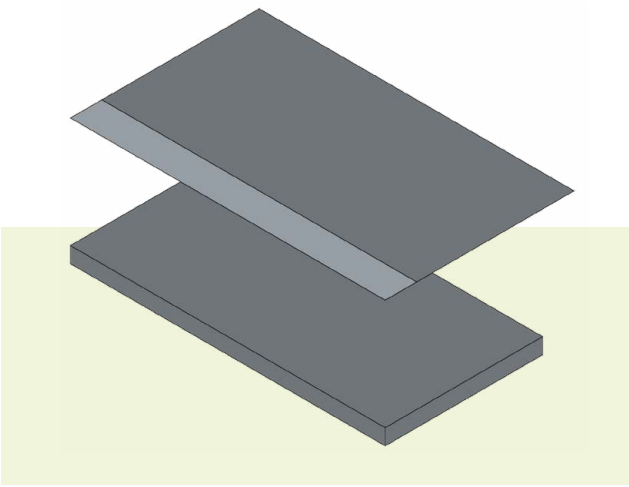
The correct name for the lump of toothpaste that you extrude from the tube onto your brush is a 'nurdle'.

To make a panel by extrusion, we start by defining a shape which will be one 'face' of the panel. We then extrude that face to make the panel. We'll need to define the position of one of the box faces using vectors. We've used `Vector` objects before; they are how we told the punched-card maker where to put the holes in the punched card in the first part of this series.

```
v1 = App.Vector(0, 0, 0)
v2 = App.Vector(width, 0, 0)
v3 = App.Vector(width, depth, 0)
v4 = App.Vector(0, depth, 0)
```

The statements above create four vectors specifying the corner positions of the bottom face of the panel. We use these to create four lines which will specify the face of the panel:

```
e1 = Part.makeLine(v1, v2)
e2 = Part.makeLine(v2, v3)
e3 = Part.makeLine(v3, v4)
e4 = Part.makeLine(v4, v1)
```



▲ **Figure 2:** The leaning panel is the top one

The four statements above use the `makeLine` method in the `Part` class to create lines for the edges of the rectangular face that we are creating. Each line starts at one vector and ends at another. Now that we have our lines, we can create a 'wire' from them. A wire is an ordered collection of lines that we will use to describe a face of the panel. You can also use wires to describe paths.

```
wire = Part.Wire([e1, e2, e3, e4])
```

The statement above creates a wire from the four lines that define the face. The lines are provided in a collection which is ordered so that they describe the path that you would follow if you used the wire to draw the face.

The `Part` class contains a method called `Face` which can create a face from wire.

```
panel_face = Part.Face(wire)
```

The statement above makes a face object from the wire. The face object is referred to by `panel_face`. Now we can extrude the face to create the panel that we want:

```
panel = panel_face.extrude(App.Vector(0, 0,
height))
```

All shapes are equal before FreeCAD

A box created by `makeBox` and a box created by extruding a face are used by FreeCAD in the same way. FreeCAD works with shapes without requiring them to be a particular type of software object. All shapes expose behaviours and properties allowing them to be combined to create models. When building a FreeCAD model, you create each element using the most appropriate method and FreeCAD takes care of combining them.

The `panel_face` object contains a method called `extrude` which is provided with a vector which gives the direction of the extrusion. The `panel_face` describes the bottom face of the panel; we need to extrude this upwards (in the direction of the Z axis) for the height of the panel to make a panel of the size that we want.

Leaning panel

If you are not sure about how extrusion works, you might want to consider what happens when we change the direction of the extrusion.

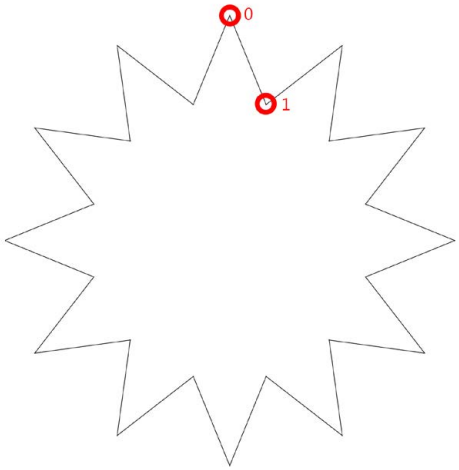
```
leaning_panel = panel_face.extrude(App.
Vector(height, height, height))
```

The statement above extrudes a panel which along a vector which points in the x (width) and y (depth) directions as well as z (up). This makes a panel which 'leans' across and into the scene.

Figure 2 shows the 'standard' and the 'leaning' panels. The top panel looks to be a lot thicker than the one below it due to an optical illusion produced by the isometric projection of the view.

Make you a star

At this point you might be wondering how extrusion makes our life easier. We seem to have used many lines of code to accomplish something we can do with a single call of `makeBox`. However, now that we know how to make shapes using vectors, we can do some interesting things; for example, making a star.



▶ **Figure 3:** The two highlighted points are the first two points that will be generated by the code

Figure 3 shows the twelve-pointed star we are going to make. This will be a 2D shape that we will extrude into a 3D object. We can think of the star as connected points on two circles, an inner one and an outer one. We are going to create the points on the star mathematically, so we need to import the Python math library:

```
import math
```

The next thing we need to do is specify the number of points in the star and the sizes of the circles.

```
points = 12
r_outer = 40
r_inner = 25
```

The statements above create three variables. The value of `points` specifies the number of points in the star. The value in `r_outer` specifies the radius of the outer circle, the value in `r_inner` specifies the radius of the inner circle. The star is going to be 40mm wide with points which are 15mm high.

Our program is going to create an array of vectors that describe the star. We can then use these vectors to create the star shape. So, let's create an empty vector array:

```
vectors = []
```

The variable `vectors` refers to an empty array. The array will be filled using a loop which will go round once for each point in the star. Each time round the loop, two new vectors will be added to the array. **Figure 3** shows the first pair of vectors to be added to the list. The point at the top of star (point 0) is at an angle of 0 radians around the star circle. The second point (point 1) is rotated by half of the angle between points in the star. We need to work out this value:

QUICK TIP

For a less 'pointy' star, you reduce the difference between `r_inner` and `r_outer`.

```
half_step = ((2 * math.pi) / points) / 2
```

You might be used to working with angles expressed in degrees, where the value of 360 means a complete rotation. In FreeCAD angles are express in radians, where the value of 2π (2 × pi) means one rotation. The statement above divides the angle for a complete circle by the number of star points and then divides this value by 2 to get the angle for half a step around the star. This is stored in a variable called `half_step`. The value in `half_step` is used to update the angle inside the loop that creates the vectors. The final variable that we need is a value to keep track of the angle. This starts at zero:

QUICK TIP

The math library contains the value of pi (3.1415926 or so).

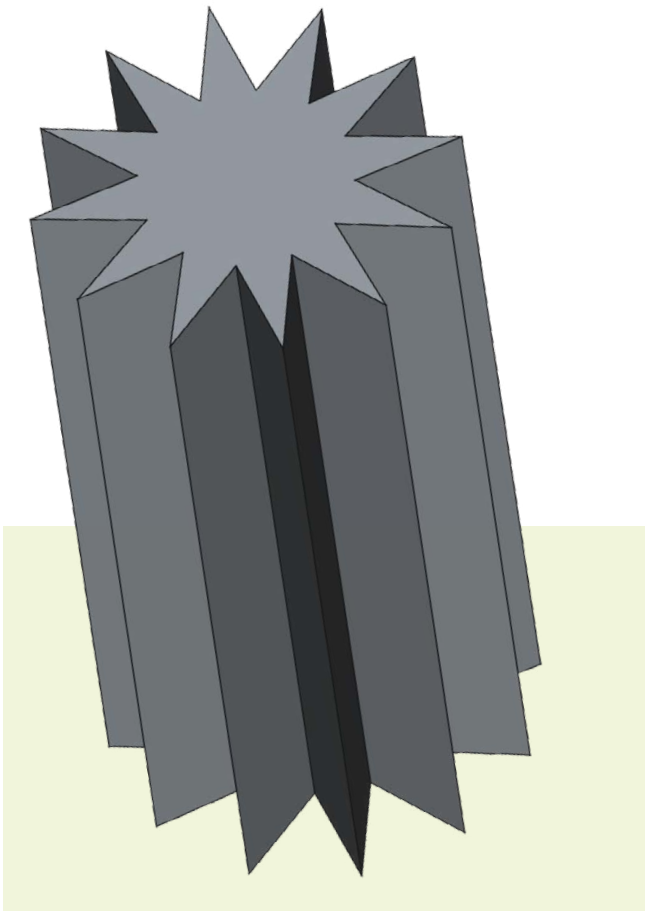
```
angle=0
```

The statement above creates a variable called `angle` and sets it to zero. The value in `angle` will be updated each time the loop goes around. Now we can start our loop:

```
for i in range(points):
```

This loop will go round twelve times for this star, because that is the value in the variable `points`. Now we can create a vertex:

```
    # Outer point (tip)
    vectors.append(App.Vector(
        r_outer * math.sin(angle),
        r_outer * math.cos(angle),
        0
    ))
```



▲ **Figure 4:** You can make different sized and shaped stars by modifying the values in the code. This is quite fun to do

In FreeCAD angles are express in radians, where the value of 2π means one rotation

This statement appends a `Vector` to the `vectors` array. The x and y values of the vector are calculated using `sin` and `cos` respectively. The first time round the loop, the value of `angle` is 0. The sine of 0 is 0, so the x component of the vector is 0. The cosine of 0 is 1, so the y component of the vector is 1. This means that the vector describes the position of the point at the top of the star, the one numbered 0 in **Figure 3**. Now we need to move the angle on to the next star position, the 'valley' between two points. This is the angle for the point numbered 1 in **Figure 3**. We move the angle by adding `half_step` to it:

```
    angle=angle+half_step
```

Now we can add the 'valley' vector:

```
    # Inner point (valley)
    vectors.append(App.Vector(
        r_inner * math.sin(angle),
        r_inner * math.cos(angle),
        0
    ))
```

This code uses the inner radius to position the vector. The final statement in the loop moves the angle onto the next star point:

```
    angle=angle+half_step
```

The statement above increases the value in angle so that it is ready for the next vector, which will be the tip of the next star point. When the loop has finished, the program will have stored vectors for all the points in the star. The final thing we need to finish the star description is to 'close' the sequence of vectors:

```
vectors.append(vectors[0])
```

This statement appends the vector for the point at the top of the star to end of the `vectors` list. FreeCAD now has a 'closed' shape to work with. It can follow all the vectors in the list and end up back where it started. We can ask FreeCAD to make this list of vectors into a shape.

```
wire = Part.makePolygon(vectors)
```

The `makePolygon` method in the statement above takes an array of vectors and creates an array of wires that represent the polygon (multisided shape) that can be drawn by connecting successive vector positions together. We can use the wire to create a face, as we did with the panel earlier.

```
face = Part.Face(wire)
```

The statement above makes a face which we can now extrude to make a solid star:

```
solid = face.extrude(App.Vector(0, 0, 120))
```

The statement above makes an extruded star which is 120 mm high.

Figure 4 shows the extruded star as a 3D object. It would be nice if we could turn this into a vase by hollowing out the centre of the star. We can do this by extruding a slightly smaller shape and then cutting it out of the large one. To make this easier, we could create a `make_extruded_star` function and then call it to make the inner and the outer shapes:

```
wall_thickness=1.5
outer = make_extruded_star(points,r_outer,
r_inner,height)
inner = make_extruded_star(points,
    r_outer-wall_thickness,
    r_inner-wall_thickness,height)
inner.translate(App.Vector(0,0,wall_thickness))
vase = outer.cut(inner)
```

The code above makes outer and inner extruded stars. The outer star is created first. The inner one is made smaller by the amount of the wall thickness. The inner star is translated up so that when it is cut from the other vase, it leaves a solid base which is the same thickness as the vase walls. The wall of the vase is 1.5 mm thick.

Figure 5 shows a star vase made from a twelve-point star. You can have a lot of fun making different sized stars with different circle radii and numbers of points. You can find the vase-making code, along with lots of other examples, at the repository for these articles: **rpimag.co/pythonfreecad**.

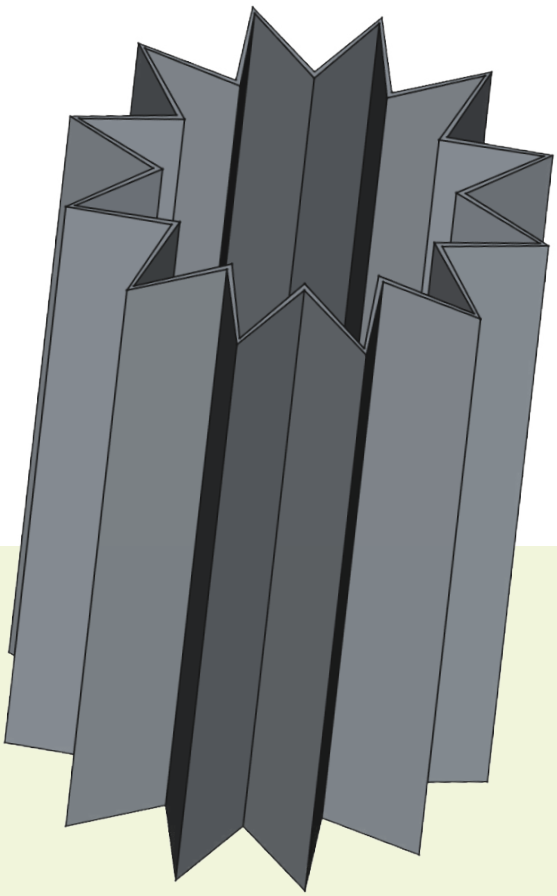


Figure 5: A 120mm high vase takes around two hours to print on a Bambu PS1 printer

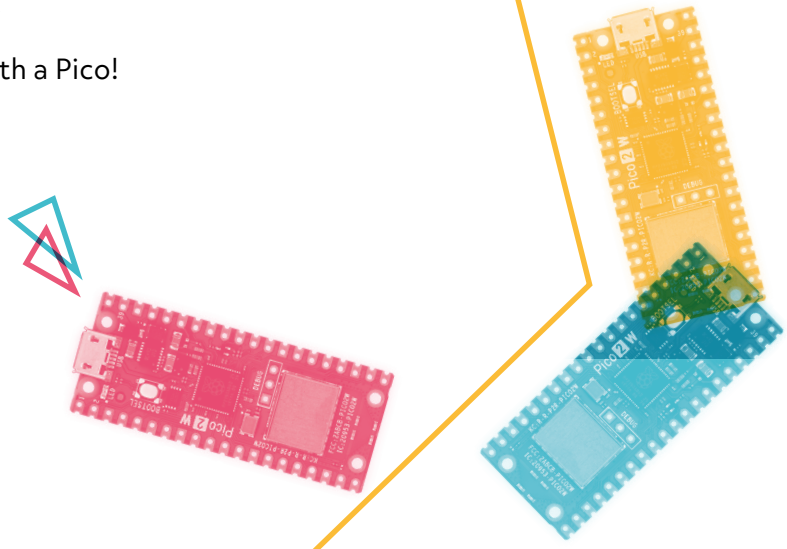
Raspberry Pi Pico Projects

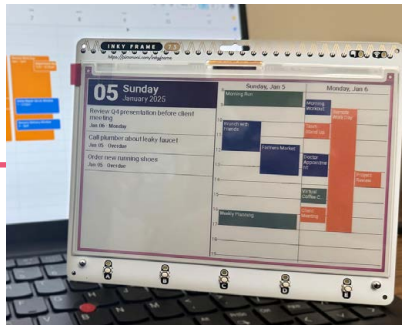
If you need a low-cost, tiny microcontroller board with ultra-low power drain to embed in a project, Raspberry Pi Pico is ideal. It comes in several flavours, depending on how much processing power you need and whether you require wireless connectivity. All models feature a couple of bonus features: analogue inputs and PIO (Programmable Input/Output) state machines that can handle some tasks in the background.

Here we showcase just a few of the best Pico projects around to inspire you, covering a range of topics – from information dashboards, IoT sensors, and LED lighting to robots, drones, music, and wearables.

There are just so many possibilities with a Pico!

By Phil King





Information Dashboard

Fetch online data and show it on a display

Simple Weather Dashboard

rpimag.co/picosimpleweather

With its wireless connectivity, Pico W (or 2 W) can retrieve information from the World Wide Web, such as the current weather conditions. A weather dashboard is a popular Pico project and Pete Cornish's simple example is easy to replicate. All you need is a Pico W and a small screen to show the info – he's used a Waveshare 2.13-inch e-ink display.

To connect Pico W to the web, you'll need to add your wireless router's name and password to a config file. In this example, the latter also contains your latitude and longitude to set a location for local weather info. To fetch data, you'll usually need to obtain a key for the API for whatever online database you're using – in this case, OpenWeatherMap.

Inky Dashboard

rpimag.co/inkydashboard

As showcased in issue 152 (**rpimag.co/152**), Jaeheon Shim's Inky Dashboard features a calendar and to-do list. It runs on a Pico-powered Inky Frame 7.3 colour e-ink display. Since the calendar layout uses the LVGL graphics library, modifying the code to work with other e-ink displays shouldn't be too difficult.

Calendar data is fetched from iCal, but it can work with other services such as Google Calendar or Microsoft Outlook. Since Pico W can't handle this directly, it fetches data via a server running on a computer or hosted in the cloud. As Pico W only fetches data every 30 minutes or so, going into a deep sleep in between, it's a very power-efficient project.

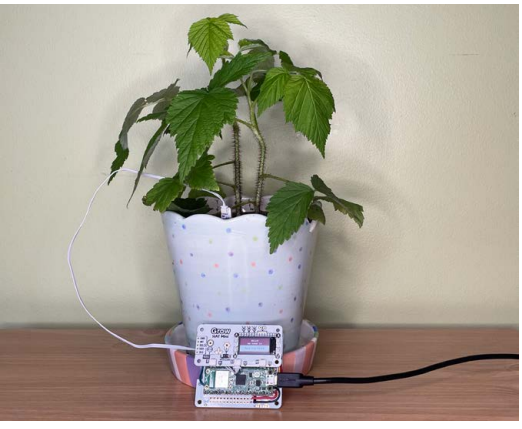
IoT Sensor

Use Pico as an IoT device for sensor data

Texting Pot Plant

rpimag.co/picoplant

Sandeep Mistry's project combines a Pico W with a Pimoroni Grow Kit to monitor the moisture level of the soil. If it's too dry, a text notification is sent – using the Twilio API with a free account – to the owner's phone to remind them to water it. For some extra personality, Sandeep suggests adding random messages, along with a light sensor to detect sunrise/sunset and say 'good morning/night' accordingly.

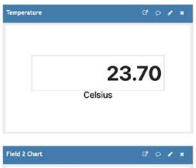


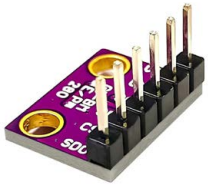
Alternatively, you could set up a Pico-based self-watering system like the one created by Veeb: **rpimag.co/sploosh**. If its sensor detects dry soil, it triggers a relay switch to activate a fish-tank water pump to squirt some much-needed H_2O into the pot.

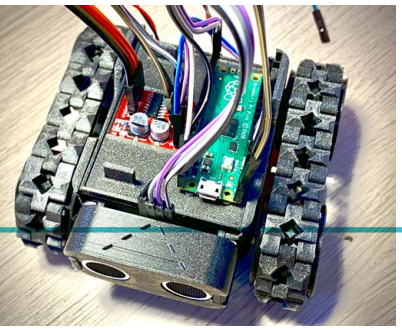
IoT Dashboard

rpimag.co/picothingspeak

With one or more sensors connected to a Pico W, you can monitor the local environment for aspects such as temperature, atmospheric pressure, and humidity. The data collected can then be used how you want – in this project by Mahmood M Shilleh, it's sent to an IoT dashboard in the cloud that can be accessed from any device. There are numerous services and methods that can be used for this; Mahmood opted for ThingSpeak, a popular open-source IoT platform. In his guide, he shows how to set up Pico W to send data from a BME280 sensor to ThingSpeak by generating an API key and creating channels for the data feeds to show them on a web dashboard.







Robots & Drones

Make mechanical marvels with Pico

PicoSMARS

rpimag.co/picosmars

Pico's tiny footprint means it can be used in much smaller robots than a standard Raspberry Pi computer. To prove the point, RoboticBits even made one with a cut-down Pico board equipped with beads for wheels: **rpimag.co/tinypicorobot**.

They don't have to be small, however. Kevin McAleer's PicoSMARS robot is a more standard-size robot car with a Pico as its brain. 'SMARS' is short for 'Screwless Modular Assemblable Robotic System', a 3D-printable robot that's modular and customisable. Pico is connected to a motor board to drive the motors, as well as an ultrasonic distance sensor to detect obstacles. Code and STL files for 3D printing can be found via the link.

Quadcopter Drone

rpimag.co/picoquadcopter

Again, Pico's small size and weight (4g without pin headers attached) make it perfect for use in drones. Created by Tim Hanewich, this impressive quadcopter drone uses a Pico as a flight controller: to read telemetry from an MPU-6050 accelerometer and gyroscope, interpret radio commands from an on-board receiver, and control four independent motors through an ESC (electronic speed controller) using PWM (pulse-width modulation).

The project may look a little daunting to recreate, but Tim has written a free and open-source twelve-part guide to the build process, MicroPython coding, and testing to achieve stable flight.

Model Railway

Choo-choose Pico to stay on track

Automated Model Railroad

rpimag.co/picorailroad

Raspberry Pi boards have been integrated by many hobbyists into their model railway setups, such as to provide smart, controllable lighting. This project features a 'sensored track' equipped with IR proximity sensors that detect a train passing. Pico acts as the brains, reading the sensors and controlling the track voltage – via a motor driver board – to alter the speed of a loco using pulse-width modulation (PWM). In this way, it can make it speed up, slow down, or come to a halt.

Level Crossing Lights

rpimag.co/picolevelcrossing

By connecting Pico to a set of tiny level crossing lights for a model railway, this project by Pater Practicus (aka Brendan McGrath) detects passing trains and determines when it's safe for the model cars or pedestrians to cross. As such, it combines two of Pico's popular uses: the ability to control LED lighting and take readings from a sensor.

It's been upgraded from the original work-in-progress version on a wooden track to a more polished setup for a powered railroad, yet still only cost £23 ($32) to make. The build process, including wiring, MicroPython coding, and making the flashing signs, is covered in the linked YouTube video.

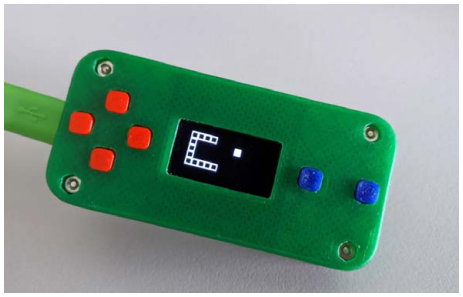
Gaming

Pico is ideal for a handheld console

Tiny Game Console

rpimag.co/tinygameconsolevid

Maker Twan37 took only a few days to make this Raspberry Pi Pico game console. It's more of a proof-of-concept and lacks a built-in battery, but is impressive nonetheless and shows what kind of handheld you can easily make with Pico. Housed in a 3D-printed case, Pico is connected to a monochrome 0.96-inch I2C OLED display and a few push-buttons for controls, four of them comprising a makeshift D-pad.



In order to demonstrate the compact console working, the maker created a couple of simple games in MicroPython – clones of Snake and Tetris – along with a maze generator.

PicoZX Handheld

rpimag.co/picozxbuild

An altogether more sophisticated gaming project, this Sinclair ZX Spectrum-emulating handheld console is based around several custom PCBs, with a Pico soldered to the main board. Inspired by Peter Misenko's original PicoZX Spectrum emulator, maker Ken St Cyr (from the YouTube channel What's Ken Making) even created a miniature QWERTY keyboard that sits below a 2.8-inch colour IPS display. There's also an on-board D-pad, although it can also be used with a separate joystick. Ken's YouTube video documents the whole build process if you fancy having a go.

Music

Make beautiful music with Pico

PicoStepSeq

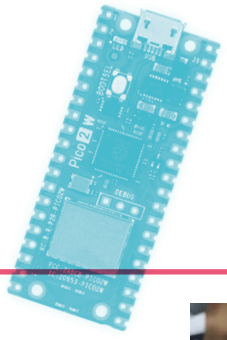
rpimag.co/picostepseq

While Pico doesn't have an on-board audio output, it can be very useful for music projects. Showcased in issue 123 (**rpimag.co/123**), this one is a compact eight-step MIDI sequencer with eight lever switches (equipped with status LEDs), a mini I2C OLED screen, rotary encoder, and Pico in a 3D-printed case. Maker Tod Kurt says it's designed as a potential DIY kit for people with beginner-level soldering skills – except for the two MIDI ports, all the parts are of the through-hole type. All the details and code can be found in the GitHub repo: **rpimag.co/picostepseqgh**.

MIDI Gesture Controller

rpimag.co/midigesture

Another MIDI music project using Pico, GaryRigg's musical expression pedal for a guitar (or other instrument) rotates and rolls around a ball joint, its position being read by a six-axis AHRS IMU sensor. Unlike a standard pedal, this enables it to control three parameters at once using yaw, pitch, and roll, so far more musical effect variations can be produced. It can also operate as a hand controller, so could be used by DJs or in a studio. For more details, see the showcase in issue 149: **rpimag.co/149**.



Input Device

Build a Pico-based controller

Gamepad 2.0

rpimag.co/gamepad2

Microcontrollers like Pico are ideal for powering a gamepad or other control device. You could try building your own from scratch, but this kit from Kevin McAleer makes it a lot easier and is ideal for controlling robots via Bluetooth or Wi-Fi.

The custom PCB enables you to simply solder a Pico W or 2 W onto it, along with eleven tactile switches: for two D-pads and three UI buttons. There are also connections for a LiPo battery and optional mini OLED display. A MicroPython library makes programming easy.

An alternative kit is the Alpakka 1, based around an RP2040-powered module: **inputlabs.io/alpakka**.

Macro Pad

rpimag.co/picomacropad

A macro pad can be prove a very useful addition to a computer setup, enabling you to easily trigger custom keyboard shortcuts and sequences with the press of a single key.

There are some RP2040/Pico-based macro pad kits available from the likes of Pimoroni and Adafruit. Or you can build your own, like this nine-key example made using a Pico, Arduino Nano R3, some Gateron switches, and keycaps. Programmed in CircuitPython, it enables you to record macros to trigger with each key.

Alternatively, you can even build a full keyboard powered by Pico, such as with the pi40 kit: **rpimag.co/pi40**.

Wearables

Pico projects you can wear

Cyber Glasses

rpimag.co/cyberglasses

Pico's small size and low power drain make it ideal for wearable projects such as adding LED lighting to cosplay outfits – or hats, such as the famous Raspberry Beret (**rpimag.co/picoberet**). It's also been used for interactive conference badges such as Pimoroni's Badgerware range (**rpimag.co/badgerware**).

Kevin McAleer, on the other hand, opted to bling up some specs for his hackable Cyber Glasses.

Pico is mounted onto one arm of the 3D printed glasses, along with a servo that moves a monocle-like NeoPixel ring in front of the wearer's right eye.

Pip-Boy 2040

rpimag.co/pipboy2040

Along with the official Pico product line, there are numerous third-party boards that make use of the same Raspberry Pi RP2040 and RP2350 chips. This project from Adafruit uses the firm's Feather 2040 board, for example. Inspired by the game *Fallout*, it's a wrist-mounted prop ready for the apocalypse! It features a rounded-rectangular IPS TFT display in a 3D printed case.

With the demo software, written in CircuitPython, you can switch between graphic screens using the directional buttons, and move the cursor with the mini joystick. With a bit of imagination you could easily create more interactive programs for it.

LED Lighting

Shine a light with some LEDs

Christmas Lights

rpimag.co/picoxmaslights

Strings or strips of individually addressable LEDs are perfect for creating all sorts of funky multicoloured effects. Our LED lighting expert Ben Everard recently put together a two-part guide on controlling Christmas lights with Pico, starting in issue 159 (**rpimag.co/159**). It's easy to connect a NeoPixel (aka WS2812B) strip with just three wires: for power, ground, and data-in. If you have a whole lot of of NeoPixels (more than around 30), though, you may need to inject some extra power into the circuit to light them all up at full brightness.

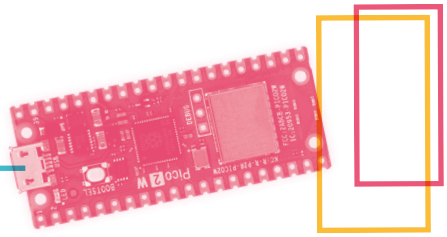
Stair Lights

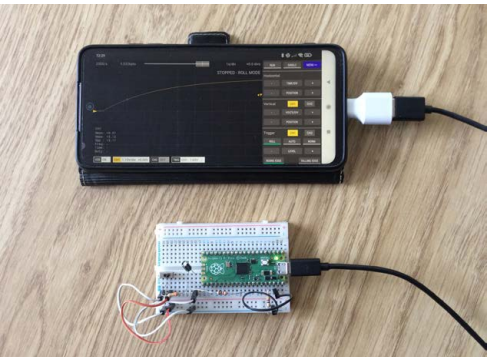
rpimag.co/picostairlights

Adding LED lighting to a staircase has long been a popular lighting project for Raspberry Pi devices. Craig Robertson's project makes use of a Pico and 30-pixel NeoPixel strip. A PIR (passive infrared) sensor detects a person approaching and lights the way for them. He also added an LDR (light-dependent resistor) to monitor the ambient light level so that the lights are only triggered when it's dark, not in daylight. He programmed Pico to light the LEDs in sequence, fading them from a colour to white.

Miscellaneous

Other projects to make with Pico





Scoppy

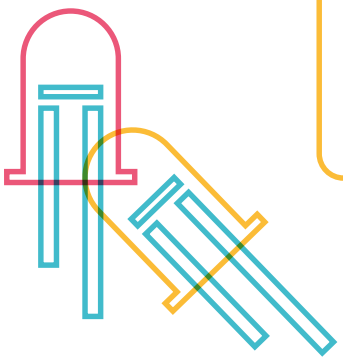
rpimag.co/scoppy

We ran a tutorial on this Pico-powered oscilloscope back in issue 134 (**rpimag.co/134**). It's easy to set up: just flash the custom firmware to Pico and install the accompanying app on an Android phone. Pico can read analogue signals from a circuit connected to its ADC GPIO pins and then send them to the app, which displays the results on a graph and has a control interface for the oscilloscope. There's a bonus logic analyser mode for viewing the digital signals on GPIO pins, too.

Overkill Cyclometer

rpimag.co/cyclometer

Jokingly calling it "a ploy to bankrupt cyclometer companies", Martin Cejp built this RP2040-powered DIY cyclometer from scratch. A weather-proof magnetic reed switch is used to count wheel rotations to calculate the cyclist's speed and distance covered – shown on a mini OLED screen. After prototyping, Martin designed a custom PCB, onto which he mounted an Arduino Nano RP2040 Connect board – preferred over a standard Pico due its wider power input range of 5–18V, so it can work with a standard 9V battery. It's all housed in a 3D printed enclosure.



Electronics with Pico

With just a few components, you can start your digital making journey

YOU'LL NEED

- Raspberry Pi Pico
- Breadboard
- LED (any colour)
- 330Ω resistor
- Push-button
- DHT11 sensor, e.g. **rpimag.co/dht11**
- 3× Male-male jumper wires

Before building your masterpiece Pico project, you'll want to learn the basics of controlling electronic circuits. We'll start by blinking an LED, then detect the push of a button to light it. Finally, we'll connect a sensor to monitor temperature and humidity.

As our examples are written in MicroPython, you'll need to have flashed that to your Pico – download it from **rpimag.co/micropython**. Once done, use a Python IDE in the connected computer to enter and run your code. We're using Thonny with its Python interpreter set to 'MicroPython (Raspberry Pi Pico)'.

Blink an LED

This is the 'Hello World' of electronics projects, typically the first thing you'll do. Wire up the circuit on a breadboard as in **Figure 1**. Note that the LED has its longer, positive leg (the anode) connected – via a 330Ω resistor – to the GPIO 15 pin on Pico; its shorter leg (cathode) is connected to a GND (ground) pin. The LED won't light up until we pull the GPIO 15 pin high, in our program, to power the circuit.

Enter the code from **Blink.py**. At the top, we import the `Pin` method from the `machine` MicroPython module, which enables you to read and control Pico's GPIO pins. We also import the `sleep` method from the `time` module to add a delay between blinks.

We then set assign the LED to the GPIO 15 pin, setting the latter as an output. Finally, in a `while True:` loop that runs repeatedly, we turn the LED on and off, with a one-second delay between each change.

Push the button

As well as controlling output devices like LEDs, Pico can read the signals from input devices such as a push-button. We'll add a button to our existing circuit, with one side of it connected to the GPIO 16 pin on Pico, and the other to the 3V3 power pin, as in **Figure 2**. Pushing the button will then complete the circuit.

Pico can read the signals from input devices such as a push-button

Enter the code from the **Button.py** listing. As before, we import the `Pin` method from the `machine` MicroPython module, along with the `sleep` method from the `time` module.

We then define an object for the button, assigning it to the GPIO 16 pin, but this time as an input; since the button is connected to 3V3, we set the pin's resistor to pull-down (see 'Pull Up or Down' for details). Finally, we use a `while True:` loop to repeatedly check the button's state; if it's pressed, we activate the LED as before.

PULL UP OR DOWN

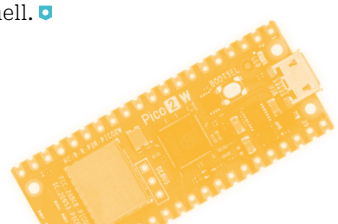
Pico's GPIO pins can have their built-in resistors set to pull-down or pull-up. If the former is used, as in our example, the button should be wired to 3V3 and when not pressed, the input reading is 0. If pull-up is used, the button should be wired to GND and when not pressed, it'll read 1.

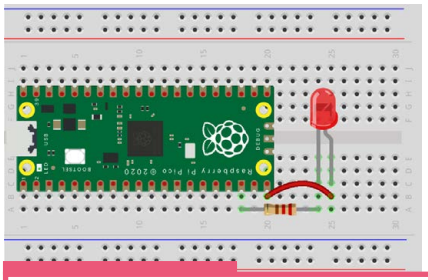
Read a sensor

Another input device is a sensor. There are lots of different types and they come with digital and/or analogue outputs. Pico does have some analogue inputs, but in this case we're reading a digital sensor: a DHT11 that measures temperature and atmospheric pressure.

Starting a brand new circuit, connect the sensor to Pico as in **Figure 3** – the connections are 3V3 (on Pico) to VCC (on DHT11), GND to GND, and GPIO 14 to DOUT.

Enter the code from the **Sensor.py** listing. At the top, we import the `dht11` module for the sensor, along with the `Pin` method from `machine`. We define an object for the sensor, assigning it to the GPIO 14 pin as an input. We then take a measurement, filtering it for temperature and humidity values. Finally, we print the results to the Shell.

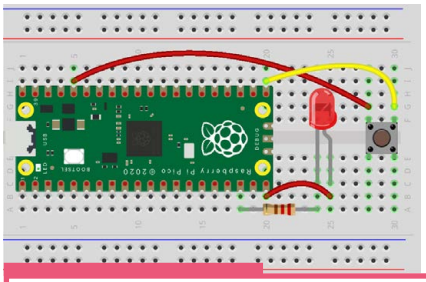




Blink.py

> Language: **MicroPython**

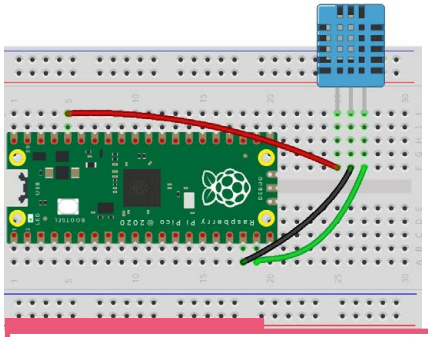
```
001.  from machine import Pin
002.  from time import sleep
003.
004.  led = Pin(15, Pin.OUT)
005.
006.  while True:
007.      led.value(1) # turn LED on
008.      sleep(1)     # wait 1 second
009.      led.value(0) # turn LED off
010.      sleep(1)     # wait 1 second
```



Button.py

> Language: **MicroPython**

```
001.  from machine import Pin
002.  from time import sleep
003.
004.  led = Pin(15, Pin.OUT)
005.  button = Pin(16, Pin.IN, Pin.PULL_DOWN)
006.
007.  while True:
008.      if button.value() == 1:
009.          led.value(1)
010.      led.value(0)
```



Sensor.py

> Language: **MicroPython**

```
001.  import dht
002.  from machine import Pin
003.
004.  sensor = dht.DHT11(Pin(14))
005.
006.  sensor.measure()
007.  temp = sensor.temperature()
008.  hum = sensor.humidity()
009.
010.  readings = ("Temperature: {}°C
      Humidity: {}% ".format(temp, hum))
011.  print(readings)
```

Further reading

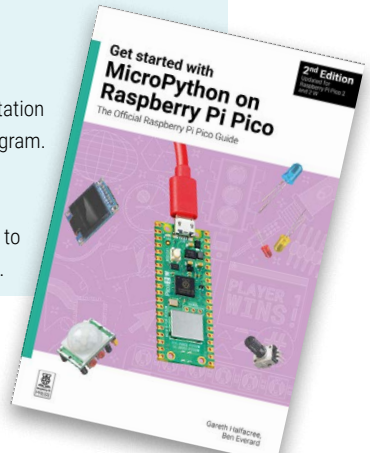
Find out more about Pico and electronics from these resources:

Get Started with MicroPython on Raspberry Pi Pico (rpimag.co/picobook) – By the end of our book, you'll know how to create your own programmable electronic contraptions.

Code Club Projects (rpimag.co/ccpicoprojects) – Code Club has a range of fun projects to help you get started with Pico and learn about coding.

Pico Documentation (rpimag.co/picodoc) – The official online documentation has all the details, including a handy Pico pinout diagram.

Wokwi Pico Simulator (wokwi.com/pi-pico) – This handy site enables you to simulate Pico circuits and run code to test them out.



▶ Our book will help you to start making Pico projects

ONLY THE **BEST**

e-ink displays

By **Phil King**

Unlike conventional displays, e-ink ones can maintain an image on screen indefinitely, even when powered down. They only use energy when refreshing the display to change what's shown. So they're perfect for portable battery-powered projects for which power-efficiency is key. They're also far less likely to cause eye strain, which is why they're used in e-book readers.

Monochrome e-ink displays are fairly inexpensive and their screen refresh time is usually much shorter than that of colour screens. Featuring up to seven colours, the latter can be fairly pricey, depending on the size. Older models may take quite a while to refresh – up to 40 seconds, flashing repeatedly as they do so – but the newer Spectra 6 technology typically halves the time. Some displays also permit a far quicker partial refresh.

E-ink screens for Raspberry Pi or Pico come in a whole range of sizes, from 1.54″ (diagonally) right up to 13.3″ for Pimoroni's largest Inky Impression model. So, whatever the needs of your project, there should be an e-ink screen suitable for it – so long as you don't need to show animations or rapidly changing data.

Inky Impression

Pimoroni | From £50 / $55 | pimoroni.com

Pimoroni's updated range of Inky Impression colour screens is impressive. Each has a female GPIO header on the rear, for connecting a Raspberry Pi computer, along with four user buttons and a breakout header.

There are three sizes to choose from: 4.0″ (600 × 400 pixels), 7.3″ (800 × 480 pixels), and 13.3″ (1600 × 1200 pixels). Interestingly, those resolutions mean the models have different aspect ratios and pixel densities – 3:2 (180ppi), 5:3 (127ppi), and 4:3 (150ppi), respectively.

There's also a big difference in cost: £50/$55 for the 4.0″ model, £80/$88 for the 7.3″, and £230/$254 for the 13.3″. You may well baulk at paying that much for the latter, but it is magnificent at near A4 size – ideal for a digital photo frame or wall art.

The display on all models is sharp and vivid. The Spectra 6 technology supports six main colours, but they can be blended using automatic dithering. The screen refresh is faster than in earlier colour e-ink screens, too: typically 20–25 seconds including detection and data delivery time. A Python library and code examples – including a neat random comic-book page – make it easy to get started.

Verdict

A very crisp colour e-ink display with a faster refresh time.

▶ The 13.3-inch model is easy on the eye, if not the wallet!

2.13-inch E-Paper Display HAT

Waveshare / The Pi Hut | £14 / $20 | waveshare.com / thepihut.com

This monochrome e-ink screen HAT may be small, but it's inexpensive and – with a driver board size of 65 × 30.2mm – fits neatly on top of a Raspberry Pi Zero. No wonder it's one of the most popular models around, especially as it's the default choice for many Raspberry Pi community projects.

It has only two levels of greyscale, but when all you need is a no-nonsense screen to show data or simple graphics, it's perfectly sufficient. The display area on this model is 48.55 × 23.71mm, with a 250 × 122 resolution, although Waveshare produces umpteen other sizes if you need more screen estate. There's also an optional case to house it.

Screen refresh time is about 2 seconds, using a tiny amount of power (26.4mW). It can also do a partial refresh, taking only about 0.3 seconds – good for a digital clock. The product wiki has libraries and demos for C and Python. The instructions could do with an update, but the screen does work in Raspberry Pi OS Trixie.

Verdict

A mini monochrome display HAT that does the job.

▼ This mini black and white display HAT communicates via the SPI protocol

Badgerware Badger

Pimoroni | £50 / $55 | pimoroni.com

▲ The app menu on the Badger; you can add your own creations

The Badger range of e-ink badges has been around for four years. The latest model offers a major upgrade with a bigger, better 2.7″ screen (with four-level greyscale) and a translucent plastic housing for the rear (with reset and home buttons, plus SWD and I2C headers). A lanyard is also supplied.

It's based around Raspberry Pi Pico 2's RP2350 chip with an RM2 radio module for wireless connectivity, 8MB of RAM, and 16MB of flash storage. There's even a built-in 1000mAh LiPo battery than can be recharged via the USB-C port – also used to connect to a computer for coding.

Pressing one of the five chunky user buttons on the front wakes up the Badger to show a menu with icons for demo apps including an ID badge, digital clock, hydration gauge, and an adventure game with 3D graphics. You can create your own apps and copy them over – there's an online guide to the MicroPython library, also used for the Badger's Tufty (colour LCD) and Blinky (LED matrix) siblings.

Verdict

Play with it, program it, and wear it around your neck.

Full-Colour ePaper Display HAT+

Waveshare / The Pi Hut | £62 / $85 | waveshare.com / thepihut.com

Making use of the same Spectra 6 colour e-ink technology as the Inky Impression range, this looks very similar to the latter's 4-inch model, albeit with a slightly smaller footprint as it lacks the top and bottom bezels.

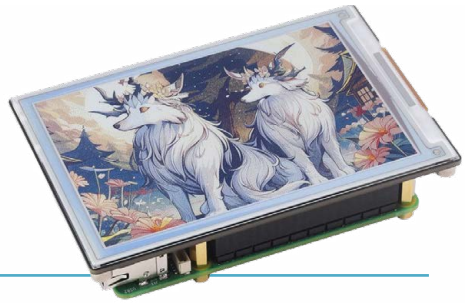
There's a (more compact) nine-pin breakout header on the rear of the board, too, but sadly no user buttons. The integral female GPIO header is tall, which is ideal for a full-size Raspberry Pi, securing it with supplied metal standoffs, but a Zero model won't sit flush to the rear.

As you'd expect, performance-wise the Spectra 6 screen is very similar to the one on the Inky Impression: crisp with vivid colours and a typical refresh time of around 20 seconds.

As with the 2.13 inch Waveshare HAT, the online documentation is in need of an update, but we can confirm the screen works in Raspberry Pi OS Trixie. There are C and Python libraries and a demo to use as a guide for your own programs.

Verdict

A quality colour e-ink screen that matches Raspberry Pi's size.



▶ A spectacular Spectra 6 display for your Raspberry Pi

Inky Frame

Pimoroni | £95 / $105 | pimoroni.com

E-ink screens are very power-efficient, as is the Raspberry Pi Pico range. Combine the two and you have you the ideal portable display with the option of powering it with a battery pack.

The new upgraded Inky Frame model features a Pico 2 W soldered to the rear (instead of the original Pico W) for extra processing power. Naturally, it also offers wireless connectivity so you can connect to the internet and fetch data from online APIs.

Currently available only in a 7.3-inch size with 800 × 480 resolution, the new Inky Frame also benefits from the same Spectra 6 colour e-ink technology used in the updated Inky Impression range. So you get a crisp display with vivid colours and a typical refresh time of 20–25 seconds.

Software support and online documentation is excellent, including a dedicated MicroPython library and plenty of code examples to show what it can do.

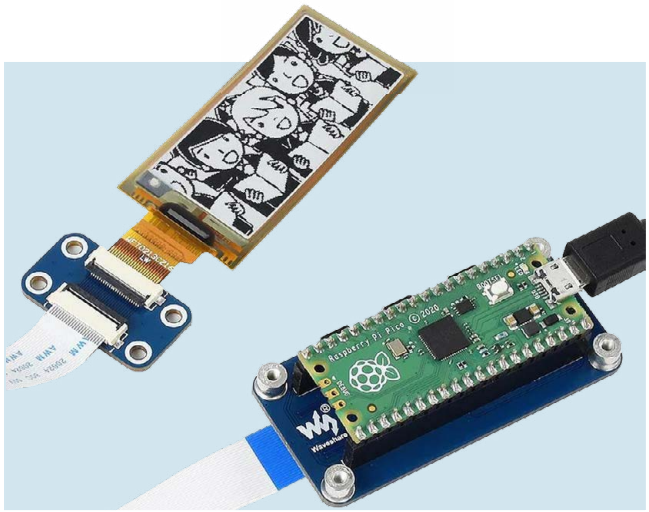
▼ With Pico 2 W aboard, this is a very versatile portable display

Verdict

Combining e-ink and Pico 2 W for an easy-to-use, low-power screen.

2.13-inch Flexible E-Paper Display

Waveshare / The Pi Hut | £18 / $25 | waveshare.com / thepihut.com



This monochrome e-ink display is designed for use with Raspberry Pi Pico. As the name suggests, the ultra-thin screen is flexible, so it could well prove especially useful if you want to attach it to a curved surface. Be careful not to bend and damage the short FPC ribbon cable, though.

The latter connects to a little adapter with a longer, regular ribbon cable going to the main driver board – into whose twin female headers you plug your Pico. A nice touch is that the underside breaks out all of Pico's pins and is also fully labelled.

The flexible e-ink display itself is near identical to that on the 2.13-inch HAT, with a 250 × 122 resolution and just two levels of greyscale. So it's a basic black and white screen for showing data or simple graphics. A full screen refresh takes only around 2 seconds; a partial one, 0.3. Larger sizes of the display are also available.

▲ This flexible screen connects to Pico via an adapter and driver board

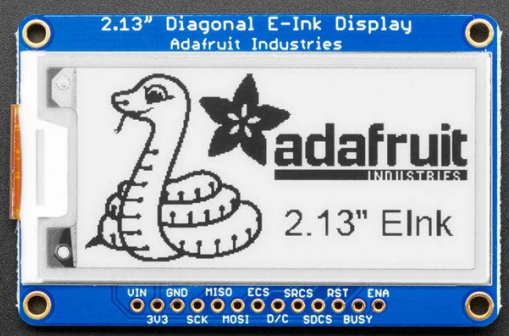
Verdict

Your flexible friend if you need a thin, bendy screen!

ADAFRUIT 2.13" MONOCHROME EINK / EPAPER DISPLAY

Adafruit | £18 / $23 | adafruit.com

Some e-ink screens, such as this one, have SPI (Serial Peripheral Interface) pins to use with almost any microcontroller or SBC. You'll just need a few jumper wires to connect them up – check your Raspberry Pi or Pico pinout for the positions of the SPI pins.



▶ This display has SPI pins plus a few more to connect to your controller board

The official

Raspberry Pi Handbook

2026

200 PAGES OF RASPBERRY PI

Get started guide covering every Raspberry Pi

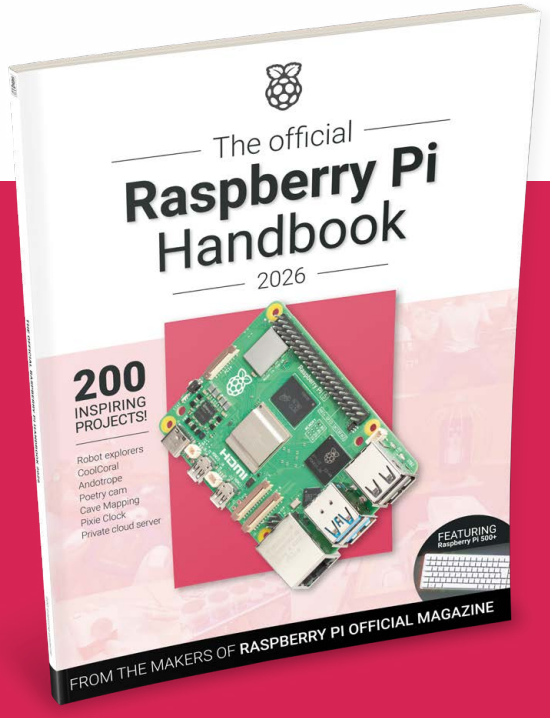
Inspiring projects to give you your next project idea

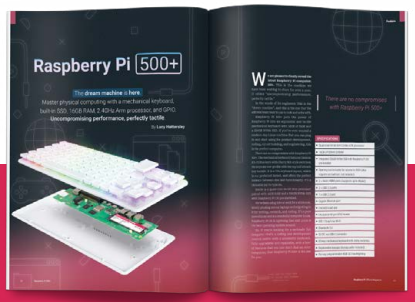
Explore the world around you with roving robots

Everything you need to know about the brand new Raspberry Pi 500+

Build a Raspberry Pi 5-powered media player

Play retro horror games on Raspberry Pi 5





Buy online: rpimag.co/handbook2026

Powered by Raspberry Pi

The new year has started strongly for Raspberry Pi, with the launch of the brand-new AI HAT+ 2 and a great presence at the Consumer Electronics Show (CES) in Las Vegas, where Sixfab walked away with the Innovation of the Year Award for its ALPON X5 AI Gateway based around Compute Module 5 (see rpimag.co/sixfabces). While certification for this in-development product is just around the corner, it was far from the only Raspberry Pi-based hardware on show. In fact, the roster of Powered by Raspberry Pi products now tops 350, with validation seen as a badge of honour. The latest additions to our catalogue include an education-focused synthesizer, really responsive fightsticks, a very visual AI tool, and an appealingly agile robot.

You can find detailed information about the compliance regulations and testing procedures for every Raspberry Pi product at **pip.raspberrypi.com**. To browse other products with the Powered by Raspberry Pi badge, view the catalogue at **rpimag.co/poweredbyraspberrypi**.

APPLY TO POWERED BY RASPBERRY PI

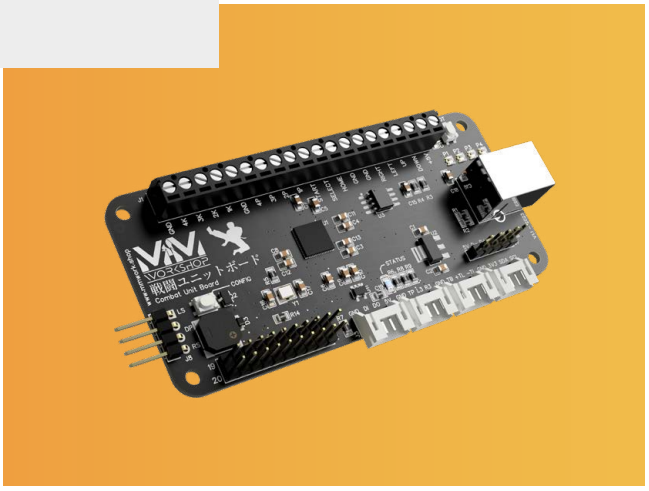
Our Powered by Raspberry Pi logo shows your customers that your product is powered by our high-quality Raspberry Pi computers and microcontrollers. All Powered by Raspberry Pi products are eligible to appear in our online gallery.
rpimag.co/poweredbypiapply

Erica Synths Bullfrog

Latvia | rpimag.co/bullfrog

Erica Synths designs and makes synthesizers and other electronic instruments out of Latvia and prides itself on inspiring young musicians via its DIY approach. This of course chimes very neatly with Raspberry Pi, so we were delighted when they chose it for Bullfrog, a Raspberry Pi RP2040 educational instrument. Bullfrog was designed in collaboration with musician and pioneering DJ Richie Hawtin who was especially keen to come up with a product that would catch the imagination of younger musicians and encourage experimentation. It boasts an analogue voltage control oscillator and amp, noise generators and remixers, pre-patched voice cards with which to get started, plus three more voice cards for recording your own. There's a built-in speaker and the option to connect to a studio monitor. See our review at **rpimag.co/bullfrogreview**.

MM Workshop GP-2040 CE Combat Unit Board



France | rpimag.co/combatboard

Lightning-fast reaction times can be the difference between hero and zero if you're a gamer, so the 0.45 millisecond response time of the GP-2040 Combat Unit Board is a real boon. MM Workshop's Zero Delay USB board was designed especially for gamers wanting to create their own controllers, fightsticks and more. With a complete gamut of connections for joysticks, buttons, LEDs, and USB 2.0 connectivity, this RP2040-based board works with gaming devices and gamepads of every brand, from Xbox to Nintendo Switch as well as retro consoles. It has 13 different modes, including web-based and stickless gameplay, and can even be overclocked.

AI Talking & Singing Fish

Netherlands | rpimag.co/aibillybass

A Raspberry Pi and AI take on the singing Billy Bass (long-time readers may recall the project showcase we featured by Kevin McAleer – rpimag.co/billybass). This more advanced version is based around either Raspberry Pi Zero 2 W or Raspberry Pi 5 and is billed as a real-time conversational animatronic talking and singing [plastic] fish that also offers voice assistant capabilities. Maker Thom has hacked the original flapper with smart home functions, while Billy Bass's voice and smarmy back-chat can be fine-tuned via a web-based dashboard. It even recognises different users' voices and displays different character traits of its own depending on who it's interacting with.

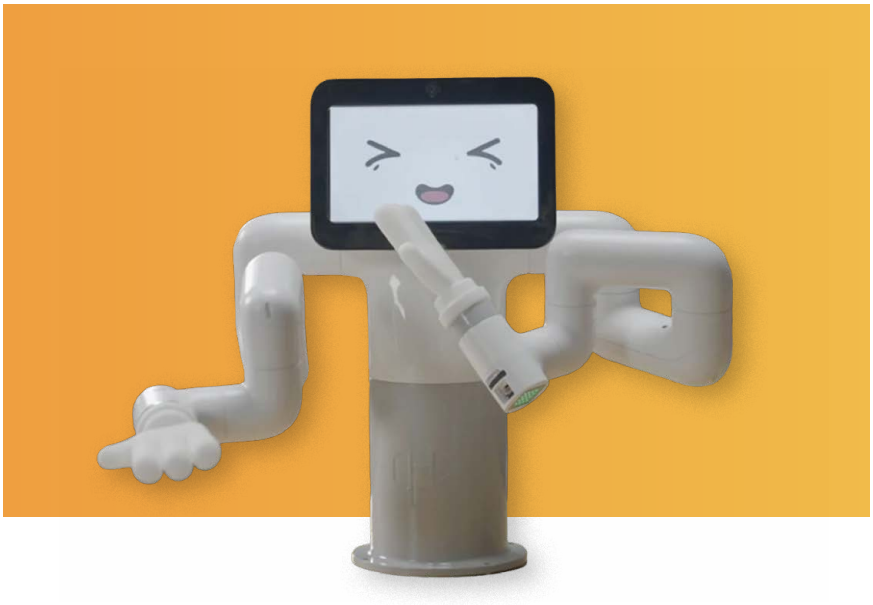
Ultralytics YOLO26

Spain | ultralytics.com

Founded by Spaniard Glenn Jocher, Ultralytics' mission is to make AI both more straightforward and effective. The company now has several international bases, having made inroads into the competitive computer vision market. YOLO (You Only Look Once) works with the Raspberry Pi AI HAT+ and AI Camera. The latest evolution in the YOLO series of real-time object detectors is engineered from the ground up for edge and low-power devices such as Raspberry Pi. It's a native end-to-end model that produces predictions without the need for non-maximum suppression (NMS), making it faster, lighter, and easier to deploy. This of course lends itself very nicely to Raspberry Pi projects, perhaps one using the AI Camera or AI HAT+.

Elephant Robotics myBuddy 280

China | rpimag.co/mybuddy280



Robots don't have to be big, bulky, and imposing. Elephant Robotics' myBuddy is a far more friendly-looking machine with appealing movements reminiscent of a posable doll or a 1980s Rubik's Snake puzzle. The pipe-like limbs on this two-armed robot can be endlessly rotated and there are 13 degrees of movement overall. Equipped with a touchscreen and camera, myBuddy 280 is the company's first Raspberry Pi-based dual-arm robot. It's also the best-value model of its type, making it ideal for educational settings.

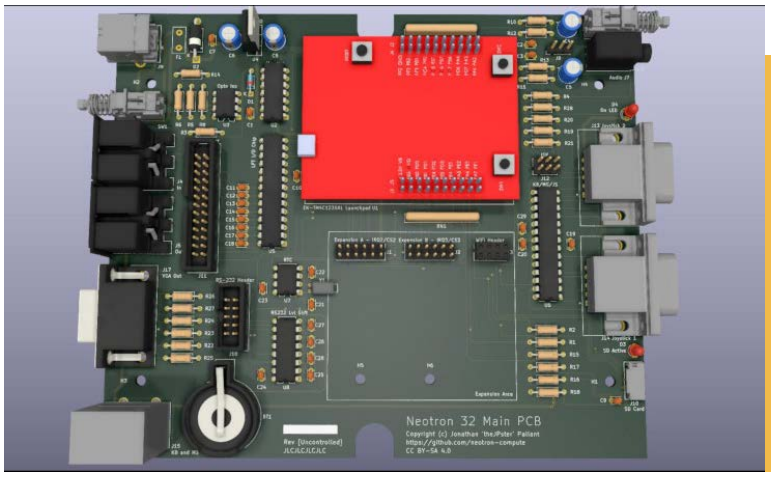
Besomi Pi Ultimate Kit

Jordan/UAE | rpimag.co/besomistarterkit

Besomi is a Middle Eastern reseller and solution provider with a 36-year track record of providing technology to a range of vertical markets, including automation and education. Its mission is to fuel tech advancement in the UAE and wider region by sourcing and distributing top-tier electronic components, including a range of Raspberry Pi products and accessories. Besomi has a range of Raspberry Pi kits on sale, including this Raspberry Pi 4 starter kit. The collection includes a Raspberry Pi 4 with 4GB RAM, Official UK Power Supply (both the UK and UAE use Type-G plugs), a SanDisk 64GB microSD card with Raspberry Pi OS pre-installed, a USB reader, 4K micro HDMI cable, and an aluminium heatsink.

Neotron Pico

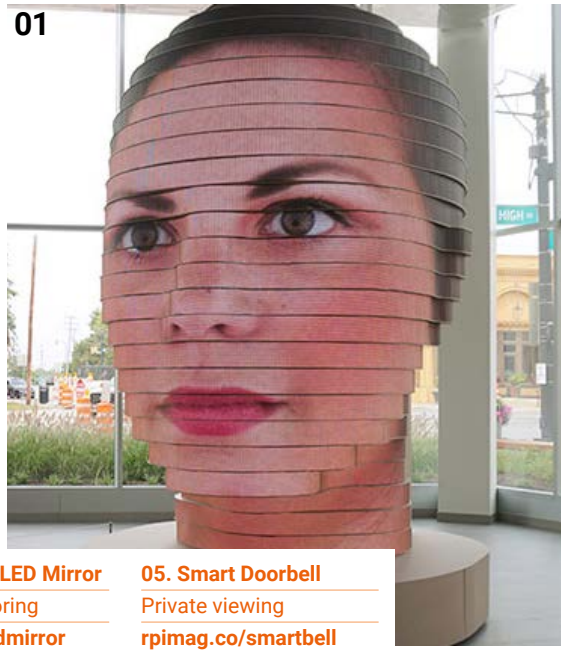
UK | rpimag.co/neotronpico



Cambridge-based developer Jonathan Pallant came up with this take on a classic 1980s Micro-ATX computer board re-envisaged for the 2020s with an ARM CPU and Raspberry Pi Pico, complete with expansion slots to add up to eight peripherals. The Neotron Pico uses Raspberry Pi Pico's PIO state machines to generate 12-bit Super VGA video and 256 colours, as well as digital 16-bit 48kHz stereo audio. Jonathan has written a CP/M or MS-DOS-alike operating system for the retro computer in Rust, but the open-source setup means the Neutron board can be programmed in any language and used with almost any hardware. The whole setup is designed for low power consumption and is suitable for passive cooling.

10 amazing: video projects

Clever displays, monitors, screens, and more with Raspberry Pi



Raspberry Pi has been used for many creative endeavours over the last 14 years, and thanks to its video output capabilities since day one, these have include many visual projects in the mix. Here are just some of our favourites...

01. As We Are

Your face, large

rpimag.co/asweare

This 14ft (4.3m) tall art installation 3D-scans people's heads and then displays them on the huge banks of LEDs surrounding a giant face. Impressive.

02. Scary door

Petrifying portal

rpimag.co/scarydoor

Not the *Futurama* *Twilight Zone* parody, but a haunted house prop displaying horrors beyond comprehension on a 'window'. There's also knocking effects on the door to add extra spooks.

03. Alternative Flight Simulator

Simulated window seat

rpimag.co/altflightsim

Want to simulate the air travel experience? Microwave a budget ready meal at least three times, then sit in this replica of a window seat so you can jet off. Seat kickers not included.

04. NeoPixel LED Mirror

Low-res mirroring

rpimag.co/ledmirror

Video doesn't need to be in 4K – it can be 576 LEDs displaying people as they walk by in surreal, impressionistic manner. Just place this magazine a few feet away and you'll see the faces better.

05. Smart Doorbell

Private viewing

rpimag.co/smartbell

This intercom-like smart doorbell allows for two-way conversations without having to be part of a big ecosystem of doorbells that will end up on a true crime documentary.

06. Living Portrait

Scary painting

rpimag.co/liveportrait

A haunted house prop, popular among folks at Halloween, is a looping video of what looks like an old-timey portrait that then reacts to motion to scare any passers-by.

07. Gourdan

Punny pumpkin

rpimag.co/gourdan

Using Adafruit's creepy eye displays, this pumpkin can keep an eye on you – literally – via object tracking and a Raspberry Pi camera.

08. Portable VCR Media Centre

Retro cool

rpimag.co/vcrmedia

This incredible upcycle project takes a 1981 portable VCR player and makes it a powerful smart media centre. The tape deck even takes a (Raspberry Pi-powered) tape.

09. Simpsons Shuffler

Eye on Springfield

rpimag.co/simpsonsshuffle

Not a dance track on the ill-fated *Yellow Album*, but a Raspberry Pi Zero that plays a random, curated episode of *The Simpsons* at the touch of a button.

10. Magic Mirror

A staple project

magicmirror.builders

This classic project uses a screen to display your day, while you check on how you look before going about your day. You can even add other video elements.


02


Premium
03


04


06


05


07


STEPHENCOYLE.NET
09


10
14:55
16.4°


11:37
08


PROJECTS FOR MAKERS & HACKERS
BUILD A CUSTOM KEYBOARD WITH RASPBERRY PI PICO
CUSTOM CONTROLS WITH A 3D PRINTED BIG BUTTON
BOOK OF
MAKING
2026
CONTROL PROGRAMMABLE LEDS FOR DAZZLING LIGHTING
EVERYTHING YOU NEED TO KNOW ABOUT BATTERY POWER
FROM THE MAKERS OF RASPBERRY PI OFFICIAL MAGAZINE

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

FOSDEM 2026!

The inside track of Europe's biggest Open Source get-together. By **Jo Hinchliffe**

▲ It's fantastic to see projects that we've featured in the magazine, such as the amazing Raspberry Pi-powered OpenFlexure microscope

▲ An amazing, open-source sub-micron repeatable micro manipulator seen being driven by Raspberry Pi silicon

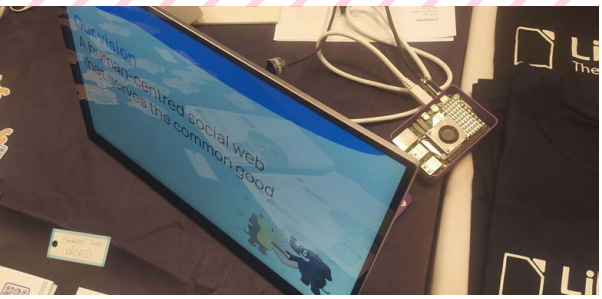
◀ Of course, FOSDEM is the epicentre of a whole heap of other events, including the famous Bytenight party at HSBXL

FOSDEM 2026, the free-to-attend conference about open source, has just been and gone and once again was a huge spectacle. Set on the Solbosch campus of the Université Libre de Bruxelles (Brussels), it saw around 10,000 curious minds descend to be excited and entertained by 1197 speakers!

From CAD/CAM to messaging platforms, printer drivers to space projects, SDR radios, mesh networking, operating systems, word processing, security, penetration testing, the list is near endless. With over 90 booths or stands, you have chance to chat and engage with either the projects you already know and use, or indeed make new friends and discover new areas of interest. One thing of note this year was that there was a marked increase in the number of talks and events relating to open-source policy and management. This meant that there was even more diversity in the groups of people attending and it's good to see free and open-source wares becoming more centred in EU and global policy-making.

Across the campus there are 'Devrooms', lecture theatres which each feature curated content. Every talk is live-streamed and the streams are edited so that each talk ends up with its own web page with the video, the speaker biography, possibly the slides and even perhaps links to other resources. Again, it's staggeringly impressive that this all works so well – kudos and respect to the audio video team!

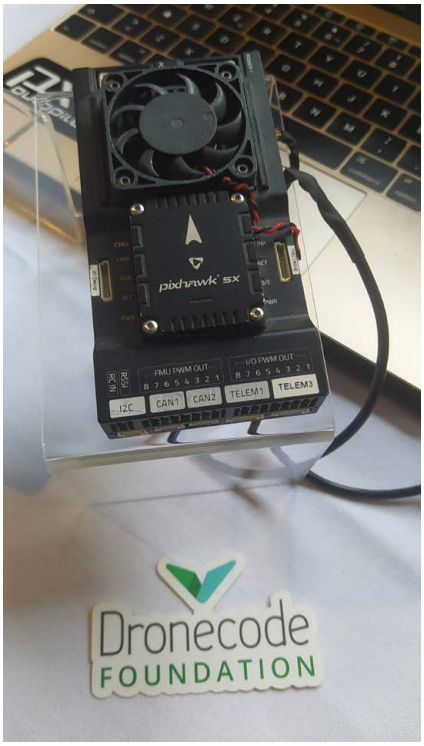
It's fun to search out Raspberry Pi silicon on the stands, along with mentions of it in the talks. Among those that caught our eye was one from José María Casanova Crespo on the improvements



▼ Not all projects use Raspberry Pi for their main project, but you can spot many Pi powering display screens such as this one on the Mastodon booth!

made about the Raspberry Pi graphics stack (**rpimag.co/fosdem26stack**). With the move from a Debian Bookworm to Trixie-based OS, these improvements have opened up a new level of performance on Raspberry Pi 4 and 5.

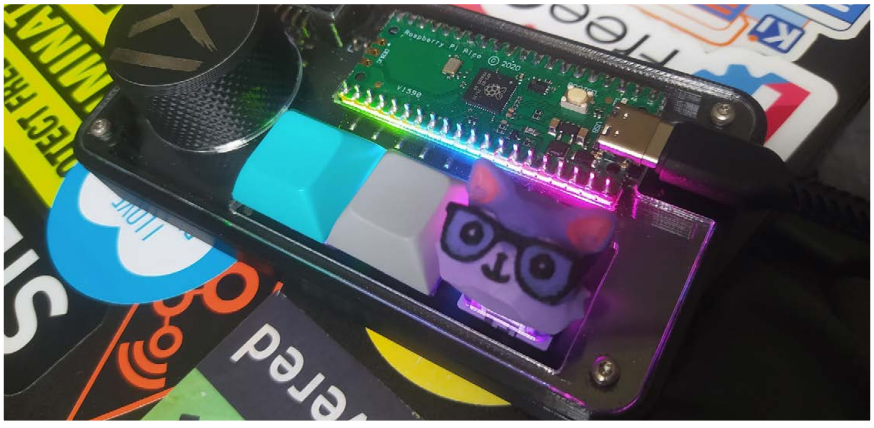
Another interesting Raspberry Pi-related talk was given by Pablo Del Arco on using OpenNebula to create an open-source cloud platform perfect for a home lab (**rpimag.co/fosdem26nebula**). Not only that, the talk slides are peppered with QR codes linking to install scripts and repos to make this really easy to play with.



▼ The Dronecode Foundation, which organises around open-source code and firmwares for drone use, was showing the Raspberry Pi CM4-powered Pixhawk 5X drone controller

> *It's a real testament to the open-source movement*

Of course, the FOSDEM weekend itself is only part of the story. With such a large group of developers, hackers, and tinkerers in attendance, many open-source projects host fringe events, with some even having multi-day meetups across the city before and after the weekend. It's a real testament to the open-source movement and a hugely constructive time. There's also lots of evening drinks and meetups, so it's easy to come home very tired, but usually also incredibly inspired.



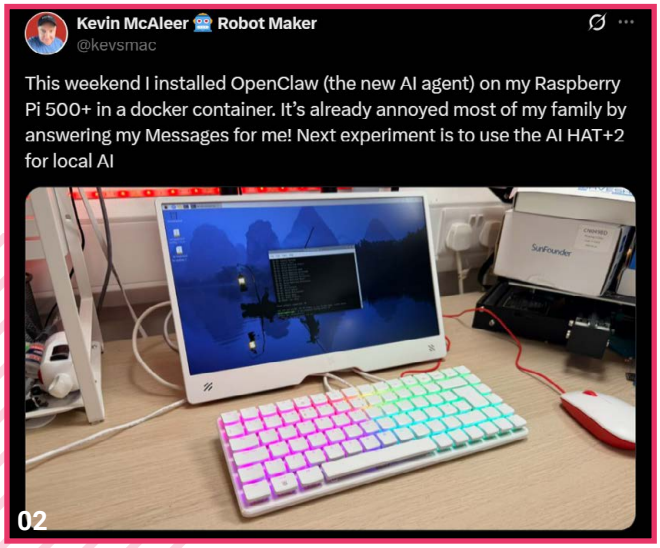
▼ A company, 9 elements, was showing off its hardware and firmware skills with this macro keypad it made for another open-source conference

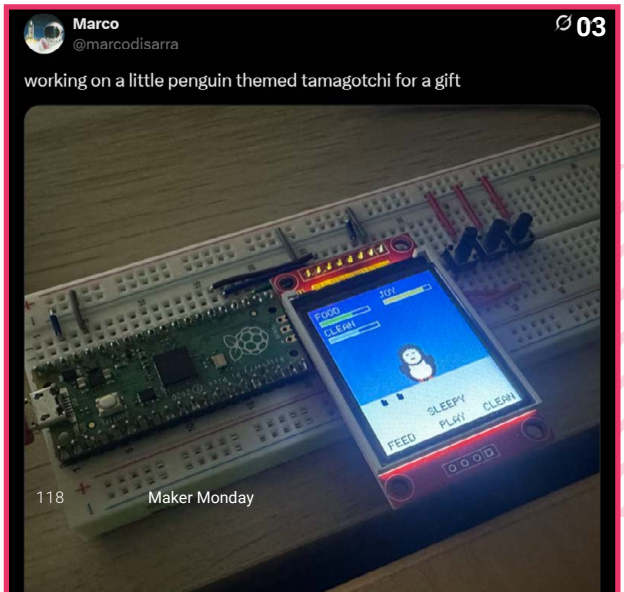
Maker Monday

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!

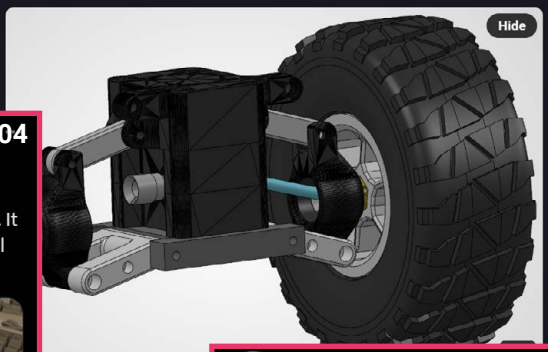




05

Dr Footleg (he/him)
@drfootleg@fosstodon.org

@rpimag I started CAD work on my first fully autonomous #raspberryPi robot. An A2 class #HackyRacer to compete in the robotics racing series event in April. I am starting with a 4WD suspension based on the #OpenRC car V5 design by #CreativeElectronics linktr.ee/CreativeElectronics adapted to be an off roader. #MakerMonday



Ø04

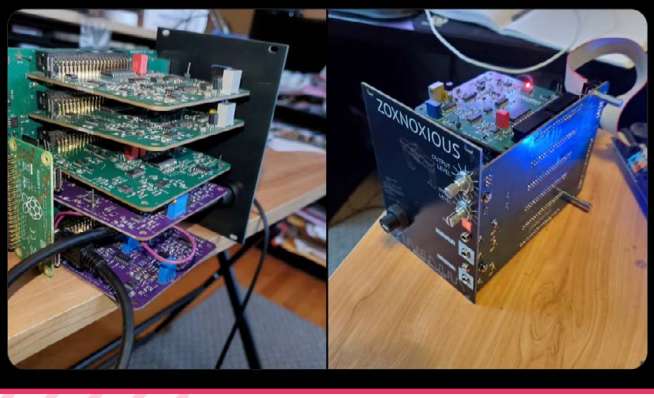
Anne Barela
@anne.engineer

I completed my mini-IBM PC build. It's powered by an @adafruit Fruit Jam and it's powerful & flexible @raspberry_pi RP2350 microcontroller. It has an IBM Joystick port, USB ports, a Big Red Switch and a monitor all encased like a tiny IBM PC from 1982.

Ø06

Ultratroninator3000
@ultratroninator

The Zoxnoxious analog synth wouldn't be beyond obnoxious without a Pi Zero2



Ø07

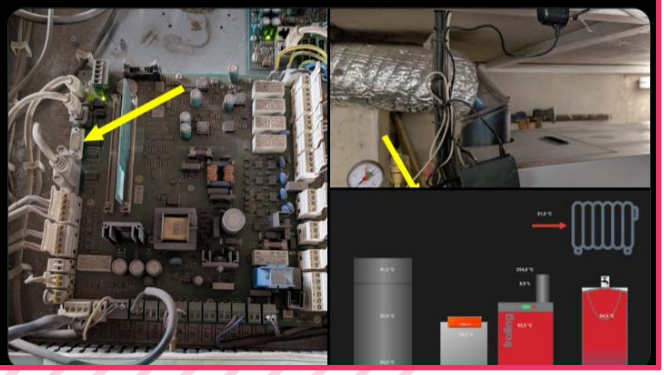
Etnupo
@Etnupoo

I made our somewhat dumb central heating system a bit smarter using a Raspberry Pi 3+, a serial connection to the heater's controller, a minimal Raspbian Bookworm installation, and this excellent project, p4d, from github.com/horchi/linux-p....

Compilation on the Raspberry Pi went without any problems, and it has now been running reliably for its second winter. p4d also publishes data via MQTT, so all the values are easily accessible in Home Assistant.

This is also the second life for this Raspberry Pi hardware!

(And please ignore the dust, the sketchy wiring and the USB serial converter 😉.)

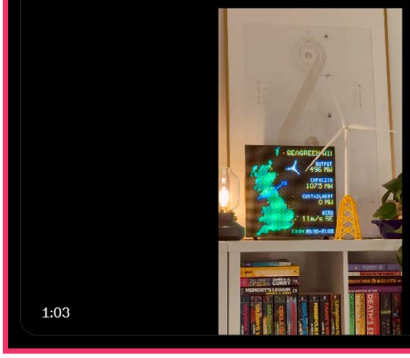


Ø08

Robin Hawkes
@robhawkes

Does this count? Both devices (display and wind turbine) are Pico-powered via @pimoroni devices.

> Robin Hawkes @robhawkes · Jan 19
> Did I accidentally make Ceefax for wind farms? 🤔
>
> This is a new toy that I created over the weekend, a 128x128 LED matrix display that shows the live status of any wind farm in Great Britain that's on the balancing system....
> Show more



Award-winning bird recognition

Created by a young high-flyer, this project is a soaraway success

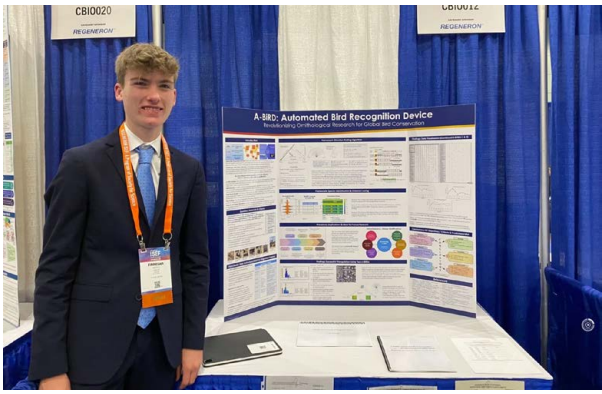
Perhaps it's just because we're ignorant Brits, but when we hear 'Tucson, Arizona', images are conjured of arid deserts and mountains and soaring thermometers. Despite this, young maker Finnegan has created a rugged outdoor device using Raspberry Pi to record birdsong – A-BiRD: Automated Bird Recognition Device.

"[He] was just named a 2025 Davidson Fellow, a national award for exceptional young people," his mother, Miriam tells us via email. "[A-BiRD] runs on a Raspberry Pi mini-computer and is built around Cornell's BirdNET analyser. Using these compact processors, his system has logged over 21,000 bird songs from 98 species here in Tucson."

According to Finnegan's report, global bird populations are mainly tracked by analysing reported sightings from people, and these populations seem to be in decline.

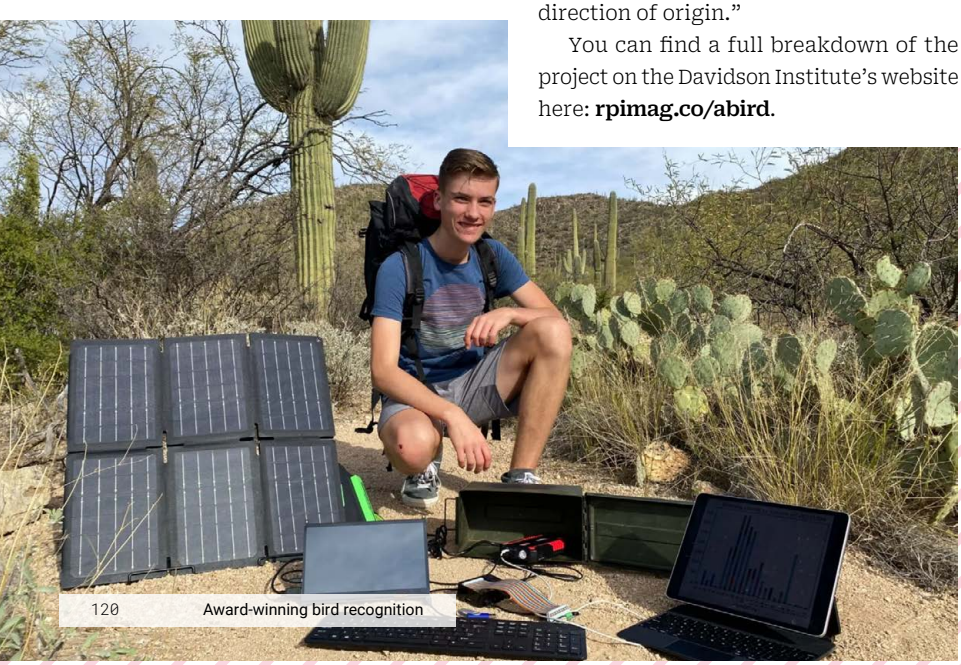
"While acoustic monitoring has been explored, few systems autonomously identify bird species and estimate call direction in the field," reads the description of the project from Finnegan. "To address this gap, I developed A-BiRD... to identify species by song and proprietary embedded algorithms to determine the direction of origin."

You can find a full breakdown of the project on the Davidson Institute's website here: **rpimag.co/abird**.



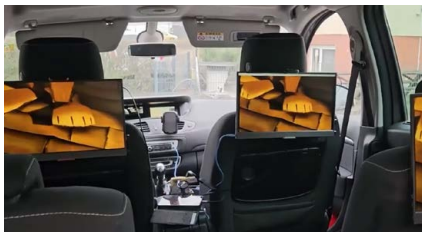
▲ We love these American school project-style info boards

◀ Finnegan out with his project in the Arizona landscape with which we are familiar



Best of the rest!

Other cool things we saw this month



DIY Car Infotainment

Kids these days don't remember the ubiquity of DVD players strapped to the backs of seats in family cars, Shrek DVDs strewn across the floor for the tykes to watch. This Raspberry Pi-powered alternative makes it like an old-school airplane experience, with synced video and car audio.

▶ **rpimag.co/diycarinfotain**

PI-CON

When the world switched away from CRT TVs to LCD, LED, etc., it also lost the ability to use light guns of yore. New methods require different tech, some of which is open source and can be used with Raspberry Pi Pico, a few IR LEDs, and some snazzy 3D printed controllers like GGDDGI has made.

▶ **rpimag.co/redpicon**

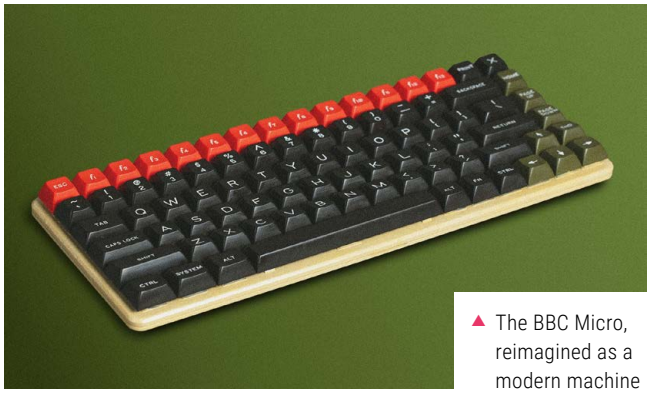
Your Letters

BBC Micro

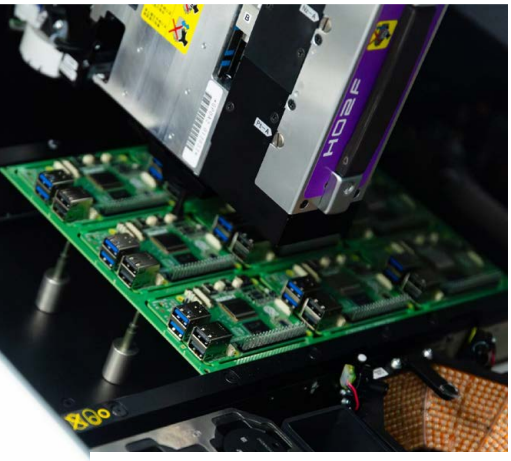
I've got a Raspberry Pi 500+ on my desk, and I do most of my work on it when I'm at home and not in the office. It's great, except for the GPIO pins being the wrong way round. For the money, it's a ridiculous amount of computer. Obviously there's some BBC Micro DNA in there somewhere – if not technically then certainly in the design. But why would anyone want to spray-paint one beige?

Sam, via email

Ah ha! You refer of course to the makeover that Raspberry Pi Maker in Chief, Toby Roberts, gave to a Raspberry Pi 500+ recently, restyling the pristine white product into a nicotine-beige throwback to 1981. It's true that a lot of the team here at Raspberry Pi Towers started out on a BBC Micro, and its all-in-one design heavily inspired the 500+. But apart from that, it's worth it because it reminds us how far we've come. When it was released in 1981, the BBC Micro cost the equivalent of £1600. For that you got an 8-bit 2MHz CPU and 32kB of RAM. Raspberry Pi 500+ has 256GB of SSD storage, 16GB RAM, and a quad-core 2.4GHz processor. We're living in a golden age!



▲ The BBC Micro, reimagined as a modern machine



▲ The world is going crazy. Sell gold, buy Raspberry Pi instead!

RAM price rises

The price of beer is going up, my mortgage is going up... and now thanks to AI bros buying up all the RAM in the world to make stupid videos, the price of a new Raspberry Pi is going up too. I suppose it's not as bad as the Great Chip Shortage of 2020-23, but please – give us a break!

David, via email

We don't like price rises; one of the great things about Raspberry Pi is that it's affordable. You can have one running your desktop, one downstairs monitoring your smart home, a Raspberry Pi Zero 2 W in your greenhouse sending back temperature and humidity data, and more, and it won't break the bank. In fact, we know people who buy the Raspberry Pi first, and think about what they're going to do with it later – that doesn't happen if the price goes up too far. That's why, whenever there have been price increases in the past forced upon us by component shortages, we've always tried hard to get the price back down as soon as we were able.

SmartCoop

I enjoyed the chickens in last issue – Raspberry Pi keeping livestock comfortable and safe is a good thing. And I enjoyed seeing the cows in the issue before that [monitored with the aid of the Flokk herd management system].

Can you make it a recurring feature to always have some form of animal life in the magazine each month?

Rachel, via email

Every year, when it's the back end of winter and I haven't seen the sun, or been warm or dry for the last four months, I think about getting outside. And this would be the perfect way to do it. Raspberry Pi in nature reserves, Raspberry Pi in safari parks, petting zoos, the little farms for kids where you can go and scratch a sheep under its chin and feed apples to horses... I'm sure they could do with some sort of automated system to keep the animals warm and fed. If you run one of these places, preferably by the sea somewhere warm, get in touch!

◀ Scientific fact: chickens lay tastier eggs when they're happy

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. CamJam's Raspberry Pi Birthday Jam

Saturday 28 February
Makerspace Cambridge, Cambridge, UK
rpimag.co/camjam26

CamJam returns for the 14th birthday of Raspberry Pi! Cambridge Raspberry Jam (known as CamJam) is an event and meetup for those interested in the Raspberry Pi family of computing devices, along with other technologies which encourage making and education.

There will be a series of talks, organised Raspberry Pi workshops, plus a 'Show and Tell' area for people to show off their projects.

02. Riverside Raspberry Pi Meetup

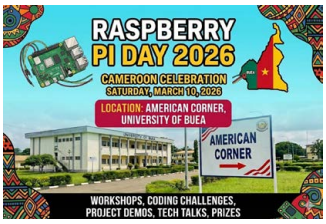
Monday 9 March
ExCITE Riverside, Riverside, CA, USA
rpimag.co/rrpm163

The purpose of Riverside Raspberry Pi is to share knowledge related to Raspberry Pi hardware and to promote interest in tech development in the Inland Empire in general. The group is currently meeting on the second Monday evening of each month.

03. Raspberry Pi Day Event Cameroon 26

Tuesday 10 March
University of Buea Library, Buea, Cameroon
rpimag.co/rpdec26

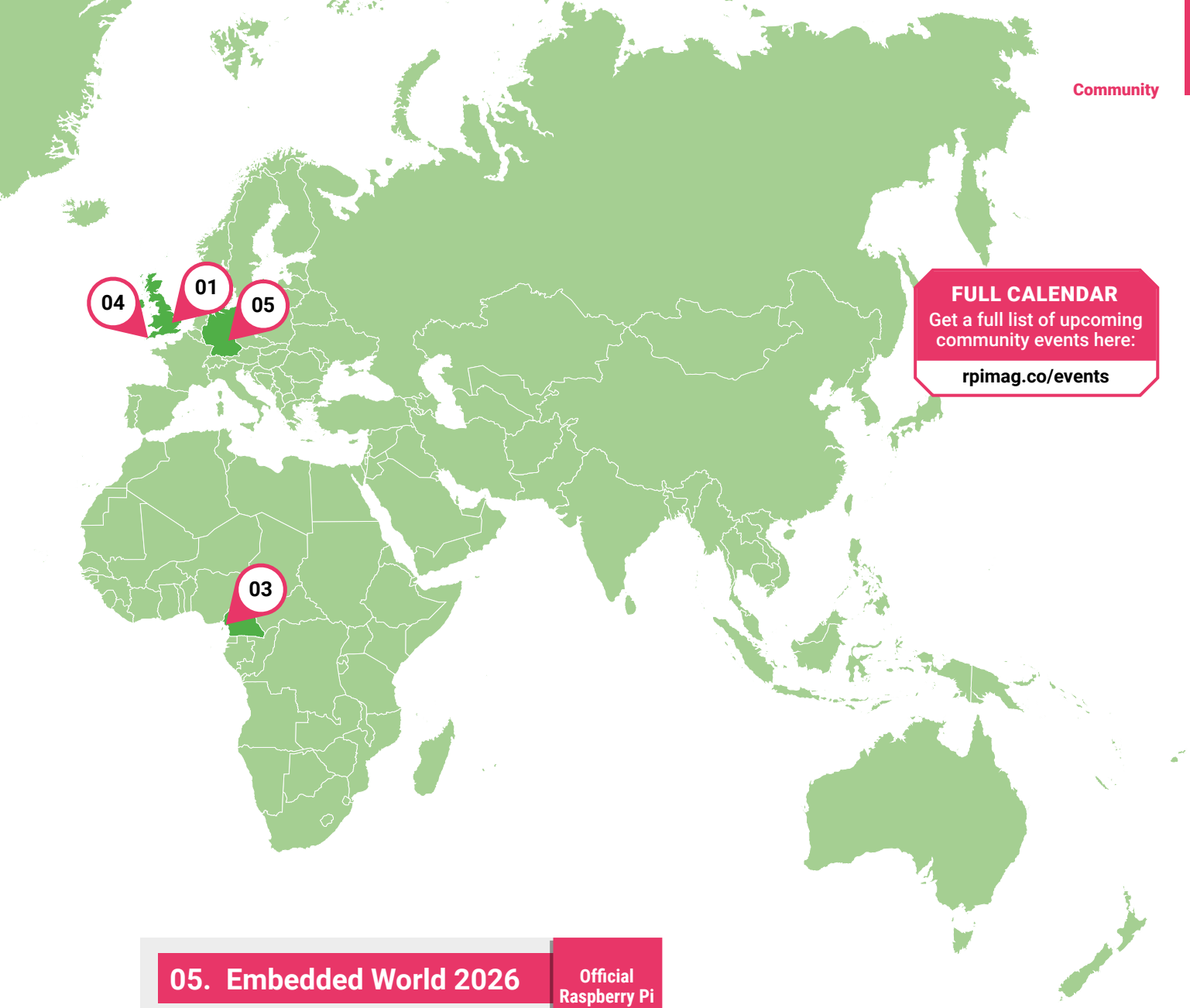
The Raspberry Pi Day event aims to further explore the capabilities of Raspberry Pi technology by focusing on programming isolated sensor nodes and integrating them into a central system. Building upon the success of the previous event, organised in March 2025, where Raspberry Pi 5 was presented for the first time, and a custom software for sensor integration, this event will provide participants with hands-on experience in utilising Raspberry Pi to integrate isolated sensor nodes.



04. Cornwall Tech Jam

Saturday 14 March
Hendra Hall, Truro, UK
rpimag.co/ctj163

Back in Truro for March, the Cornwall Tech Jam is a monthly event held in the county. It offers a welcoming, hands-on space for young people to explore the fascinating world of technology and coding. Whether they're just starting or have some experience, children and teens can dive into exciting coding challenges, build their own projects, and work with tech gear that's not always accessible at home.



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.

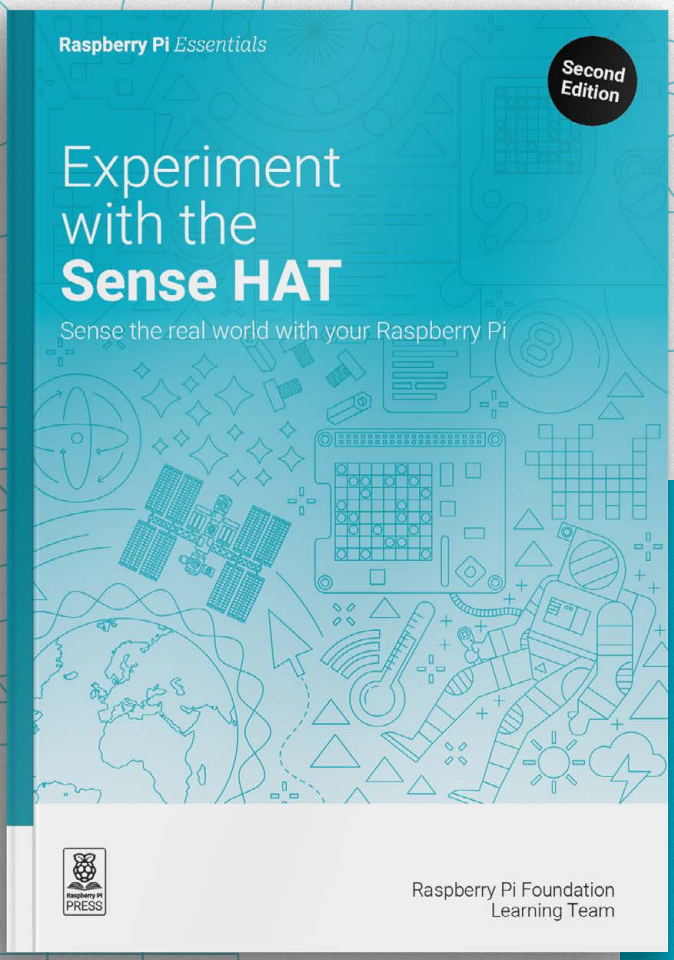
Win 1 of 5 KIWI+ USB KVMs

Ideal for connecting another computer to Raspberry Pi to access it, this KVM device includes a USB hub that can be switched between both systems using the special app. It also maintains the great features of the standard Kiwi KVM, which we reviewed in issue 153 (**rpimag.co/153**).

Head here to enter:
rpimag.co/win
Learn more:
rpimag.co/kiwiplus

Terms & Conditions

Competition opens on **25 February 2026** and closes on **26 March 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
#164

Next Month

LOW POWER PROJECTS

Discover what you can do with the new Raspberry Pi 1GB

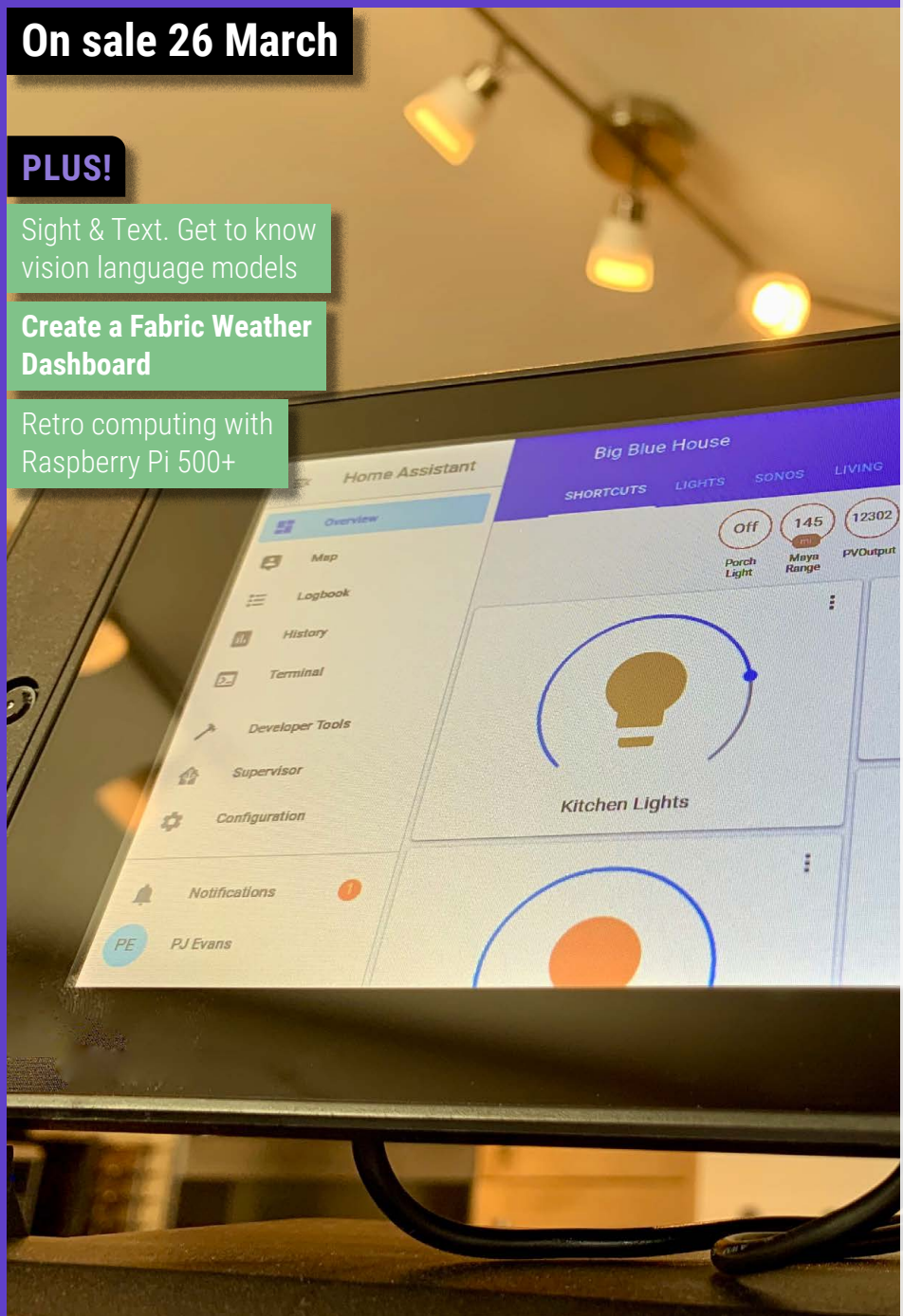
On sale 26 March

PLUS!

Sight & Text. Get to know vision language models

Create a Fabric Weather Dashboard

Retro computing with Raspberry Pi 500+



Editorial

Editor
Lucy Hattersley
lucy@raspberrypi.com

Features Editor
Andrew Gregory
andrew.gregory@raspberrypi.com

Features Editor
Rob Zwetsloot
rob@raspberrypi.com

Sub Editor
Phil King

Advertising
Charlotte Milligan
charlotte.milligan@raspberrypi.com
+44 (0)7725 368887

Design

Head of Design
Jack Willis

Designers
Sara Parodi, Natalie Turner

Illustrator
Sam Alder

Brand Manager
Brian O Halloran

Contributors

David Crookes, Tim Danton, Rosemary Hattersley, Jo Hinchliffe, Phil King, Bianca McLean, Sean McManus, 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
rpipress@subscriptionhelpline.co.uk

ISSN: 2977-4403 (Print)
ISSN: 2977-4411 (Digital)

Changes

Rob gets older and thinks about how things change

As I write this, it's just over a week until my birthday! As you read this, it will have been about a week since my birthday. So we'll split the difference and say: it is my birthday. Not the big Four-Oh just yet, that's next year. When I turned 30 I had a bit of what the Gen Z folks call a 'menty b' – I wasn't really prepared to leave my 20s and was worried I hadn't done enough to make the most of them. It turned out that nothing really changed when I hit 30 and I felt the same! Huzzah. 32 was a different story though.

My 30s have been mostly very kind to me I think, and things generally did change for me. For the better! Change is not always scary and is just the nature of life after all.

> *I have a couple Raspberry Pi Pico boards that I use for many different projects, just flashing them with the software they need*

More Moore's

Tech is continuously changing around us, even if it's incremental. Since 30, technology has subtly changed. The biggest changes came to video calling software for obvious 'everyone was stuck at home for about a year' reasons. However, we're definitely at a point where stuff like smartphones stay 'good enough' for longer despite even Apple's best efforts to make them obsolete. New ways to use existing tech seems to be the theme of recent years, and next week when it's actually my birthday I will be seeing some of that stuff as I am off to Walt Disney World (in case of confusion, I mean Florida, because there is only one place called Disney World, and it's in Orlando. Good pub quiz question, that).

I actually last went to WDW when I was 30. Since then, a ton of rides and shows have been shut down, reopened, reimagined, plussed, etc., and during projects like this Disney love to reuse older parts they have. The Tiana's Bayou Adventure ride, ostensibly a sequel to the wonderful 2009 movie *The Princess and the Frog*, reuses the log flume, layout, and some of the critter animatronics that were once part of Splash Mountain. Old tech in a new way – and in fact some of those animatronics were reused from an even older attraction.

Project recycling

Maybe it's because of the nature of my job, but not every Raspberry Pi project remains forever in my home. I have a couple Raspberry Pi Pico boards that I use for many different projects, just flashing them with the software they need when I'm doing a quiz or powering a ghost-hunting vacuum. Raspberry Pi Zero is good for this too, especially when you can just swap out microSD cards. With the current issues with RAM prices, I may have to do a little more recycling than I normally do for projects, but hopefully – eventually – that will change too. For now though, it's a Zero- and Pico-rich world.

Rob Zwetsloot – Author

Rob feels like if he ever goes broke, he can just sell the 64GB of RAM in his PC and buy a small island.

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: