Recreate the **Apollo Guidance Computer**

Train a **robot dog with Pupper v3**

Use AI to **control GPIO pins with Needle**

Raspberry Pi

Official Magazine
#171 | November 2026

Incredible Weather Projects



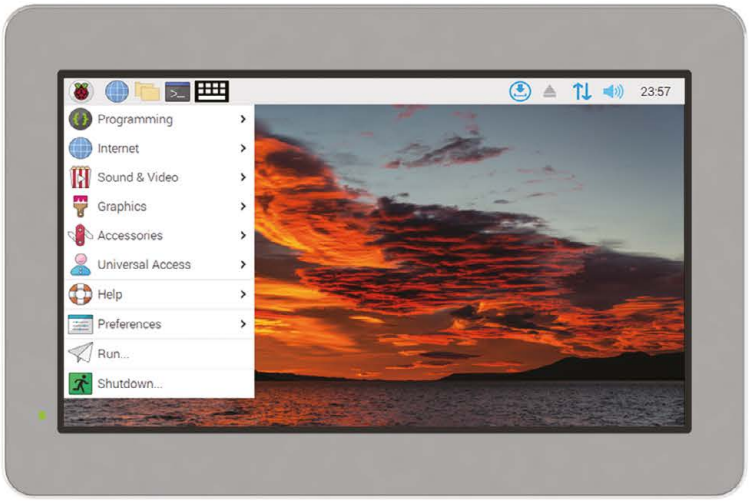
Monitor rain, wind, heat, and earthquakes!

11
£7.99
ISSN
2977-4403
(Print)
9 772977 440004

Industrial Raspberry Pi

ComfilePi





UL LISTED

Powered by Raspberry Pi

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

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

Visit www.comfiletech.com

© copyright COMFILE Technology, Inc. ALL RIGHTS RESERVED

COMFILE TECHNOLOGY

Welcome to Raspberry Pi Official Magazine



Editor
Lucy Hattersley
Lucy spent a small fortune re-roofing this year. She hopes it rains to get her money's worth.

rpimag.co



As I write, it's currently a lovely summer's day. Well, it's technically autumn, but the weather here has been a bit weird of late.

Britain is famous for getting all four seasons in a single day. And chatting about the weather with a cup of tea is our national pastime. According to the Met Office, we will have a lot to talk about later this year. We are set for an especially wild winter. It could be stormy, rainy, windy, and warm – all at the same time.

Therefore, we thought this would be a great time to dust off our weather projects. From measuring the warmth and rain in your garden, through to calling weather service APIs, Raspberry Pi is a brilliant barometer.

So, next time you chat about the weather, you'll be able to do so from a technical perspective. And remember: there's no such thing as bad weather – only bad clothes.

Lucy Hattersley – Editor

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 Charlotte Milligan at: **charlotte.milligan@raspberrypi.com**

STATEMENT OF OWNERSHIP, MANAGEMENT, AND CIRCULATION (required by Act of August 12, 1970: Section 3685, Title 39, United States Code). 1. Raspberry Pi. 2. (ISSN 2977-4403). **3. Filing Date**: September 21, 2026. **4. Issue Frequency**: Monthly. **5. Number of Issues Published Annually**: 12. **6. Annual Subscription Price**: $72.93. **7. Complete Mailing Address of Known Office of Publication**: Asendia USA 777 Henderson Blvd Folcroft, PA 19032. **8. Complete Mailing Address of Headquarters or General Business Office of Publisher**: 194 Science Park Cambridge, CB4 0AB UK. **9. Full Names and Complete Mailing Addresses of Publisher, Editor, and Managing Editor. Publisher**: Raspberry Pi 194 Science Park Cambridge, CB4 0AB UK; **Editor**: Lucy Hattersley Raspberry Pi 194 Science Park Cambridge, CB4 0AB UK; **Managing Editor**: Brian Jepson Raspberry Pi 194 Science Park Cambridge, CB4 0AB UK. **10. Owner**: Raspberry Pi Ltd 194 Science Park Cambridge, CB4 0AB UK. **11. Known Bondholders, Mortgagees, and Other Security Holders Owning or Holding 1 Percent or More of Total Amount of Bonds, Mortgages, or Other Securities**: . **12. Tax Status**: Has Not Changed During Preceding 12 Months. **13. Publication Title**: Raspberry Pi. **14. Issue Date for Circulation Data Below**: August 01, 2026. **15. Extent and Nature of Circulation**: B. Paid Circulation. 1. Mailed Outside-County Paid Subscriptions. Average No. Copies Each Issue During Preceding 12 Months: 1402. No. Copies of Single Issue Published Nearest to Filing Date: 1385. 2. Mailed In-County Paid Subscriptions. Average No. Copies Each Issue During Preceding 12 Months: 0. No. Copies of Single Issue Published Nearest to Filing Date: 0. 3. Paid Distribution Outside the Mails Including Sales Through Dealers and Carriers, Street Vendors, Counter Sales, and Other Paid Distribution Outside USPS. Average No. Copies Each Issue During Preceding 12 Months: 0. No. Copies of Single Issue Published Nearest to Filing Date: 0. 4. Paid Distribution by Other Classes of Mail Through the USPS. Average No. Copies Each Issue During Preceding 12 Months: 0. No. Copies of Single Issue Published Nearest to Filing Date: 0. C. Total Paid Distribution. Average No. Copies Each Issue During Preceding 12 Months: 1402. No. Copies of Single Issue Published Nearest to Filing Date: 1385. D. Free or Nominal Rate Distribution. 1. Free or Nominal Rate Outside-County Copies. Average No. Copies Each Issue During Preceding 12 Months: 0. No. Copies of Single Issue Published Nearest to Filing Date: 0. 2. Free or Nominal Rate In-County Copies. Average No. Copies Each Issue During Preceding 12 Months: 0. No. Copies of Single Issue Published Nearest to Filing Date: 0. 3. Free or Nominal Rate Copies Mailed at Other Classes Through the USPS. Average No. Copies Each Issue During Preceding 12 Months: 0. No. Copies of Single Issue Published Nearest to Filing Date: 0. 4. Free or Nominal Rate Distribution Outside the Mail. Average No. Copies Each Issue During Preceding 12 Months: 0. No. Copies of Single Issue Published Nearest to Filing Date: 0. E. Total Free or Nominal Rate Distribution. Average No. Copies Each Issue During Preceding 12 Months: 0. No. Copies of Single Issue Published Nearest to Filing Date: 0. F. Total Distribution. Average No. Copies Each Issue During Preceding 12 Months: 1402. No. Copies of Single Issue Published Nearest to Filing Date: 1385. G. Copies not Distributed. Average No. Copies Each Issue During Preceding 12 Months: 2. No. Copies of Single Issue Published Nearest to Filing Date: 2. H. Total. Average No. Copies Each Issue During Preceding 12 Months: 1404. No. Copies of Single Issue Published Nearest to Filing Date: 1387. I. Percent Paid. Average No. Copies Each Issue During Preceding 12 Months: 100%. No. Copies of Single Issue Published Nearest to Filing Date: 100%. **Publication of Statement of Ownership**: will be printed in the October 22, 2026 issue of this publication. **Signature and Title of Editor, Publisher, Business Manager, or Owner**: John Dotchel, Director of Data Processing/ Lettershop/Domestic. I certify that all information furnished on this form is true and complete. I understand that anyone who furnishes false or misleading information on this form or who omits material or information requested on the form may be subject to criminal sanctions (including fines and imprisonment) and/or civil sanctions (including civil penalties).

Contents

World of Raspberry Pi

008 Raspberry Pi OS desktop GUI updated

014 Subscribe to Raspberry Pi Official Magazine

Project Showcase

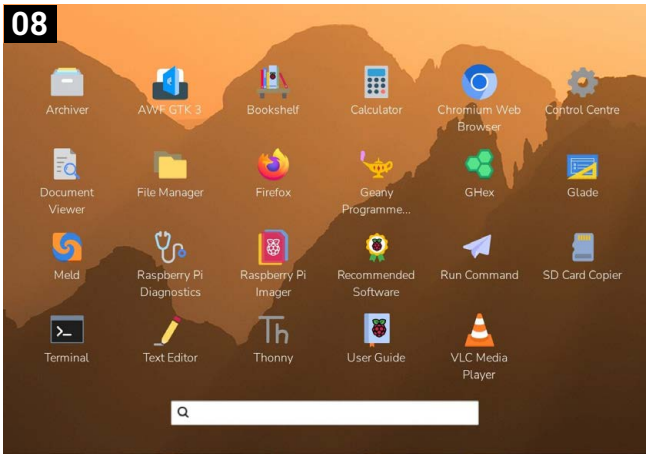
016 Recreate Apollo

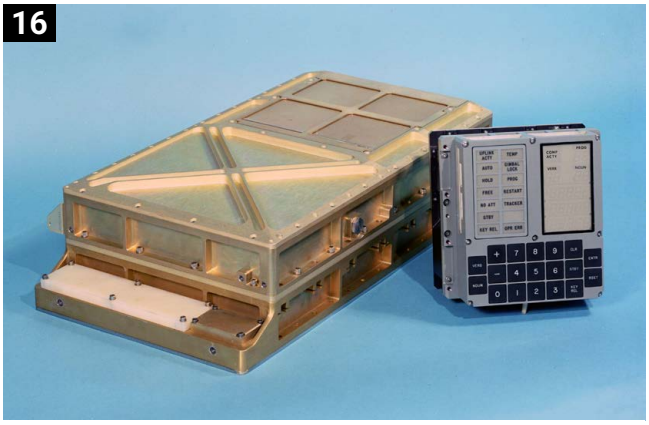
020 Pupper v3

024 Self-sufficient mobile weather station

028 D&D Book

030 Archive 88 project







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





Top Projects

036 NixieGraph

038 WallCalToDo

040 Boombox Cyberdeck

042 Raspberry Pi Cinema Camera

044 3D print showcase

Feature



048 **Incredible Weather Projects:** Builds that feel the pressure

Tutorials

058 Build a weather station

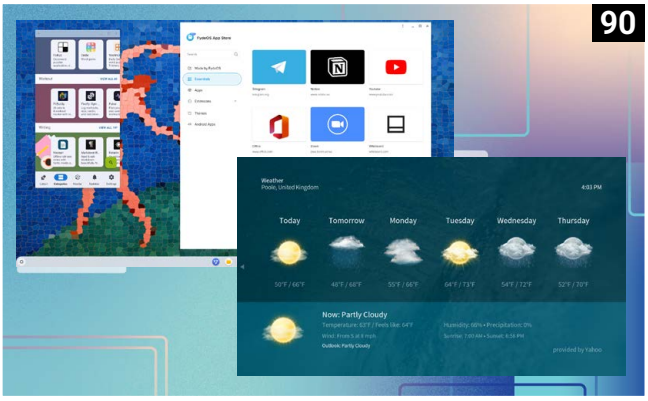
068 Make a Custard Cream-themed photo server

076 Add tech to a doll house – part 3

082 Control GPIO with Needle

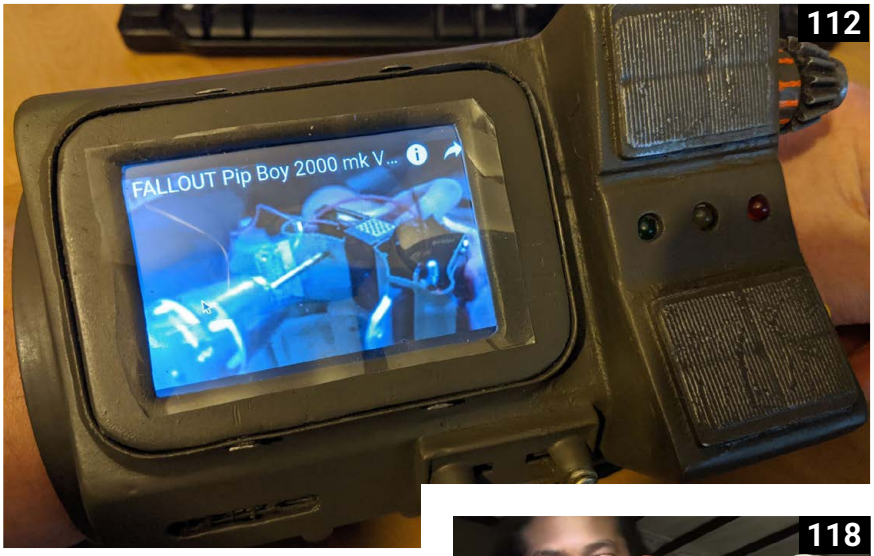
086 Emulate Apple II on Raspberry Pi

Feature



090 **Raspberry Pi operating systems:** Different ways of using Raspberry Pi







Reviews

100 **Only the best:** retro gaming kit

106 Meshstick

108 Powered by Raspberry Pi

112 **Ten amazing:** wearable projects

Raspberry Pi Community

116 Liverpool Makefest report

118 Interview: Favor Brunner

120 This Month in Raspberry Pi

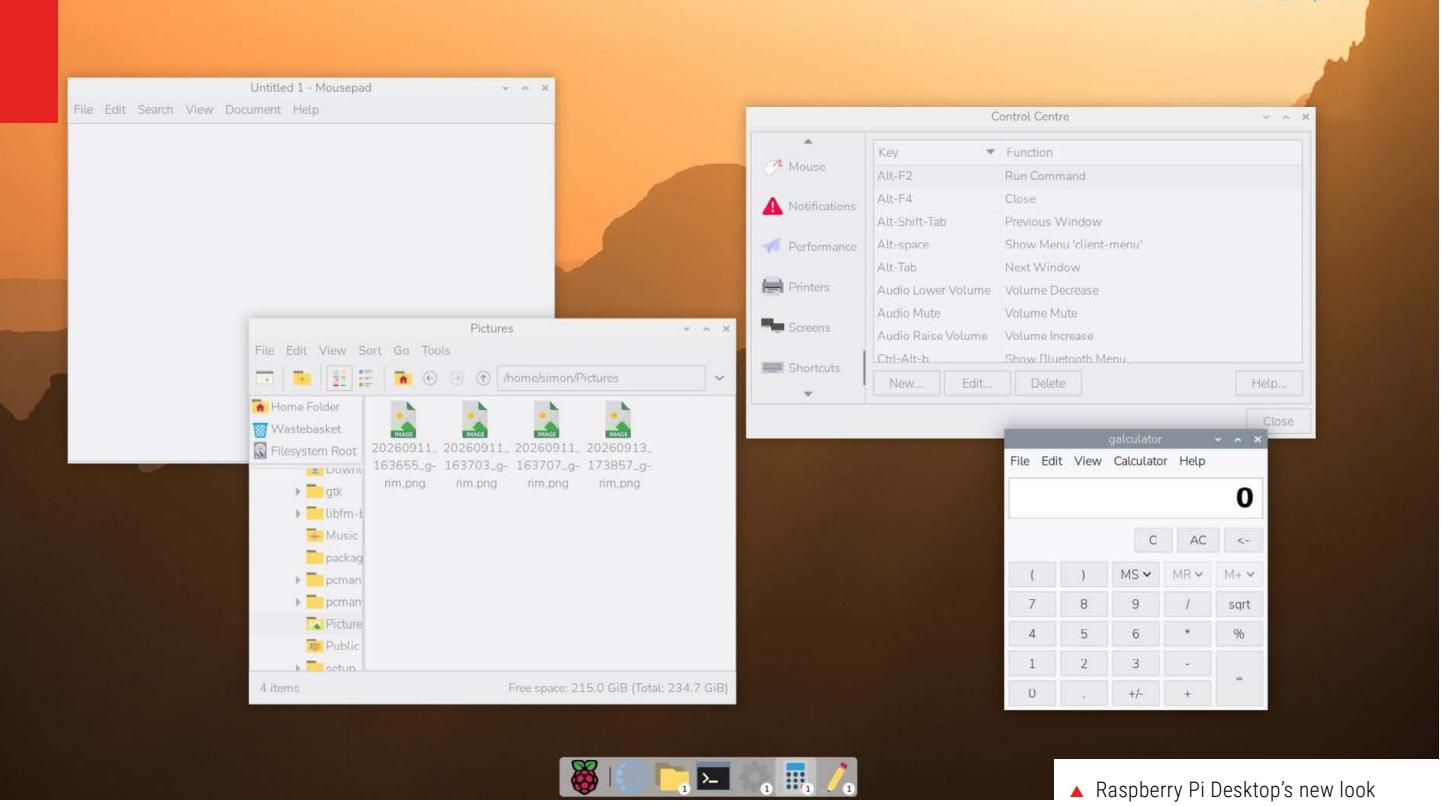
124 Your Letters

126 Community Events Calendar

Competition

128 **Win** 1 of 5 Sense HAT bundles

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



▲ Raspberry Pi Desktop's new look

Raspberry Pi Desktop GUI updated

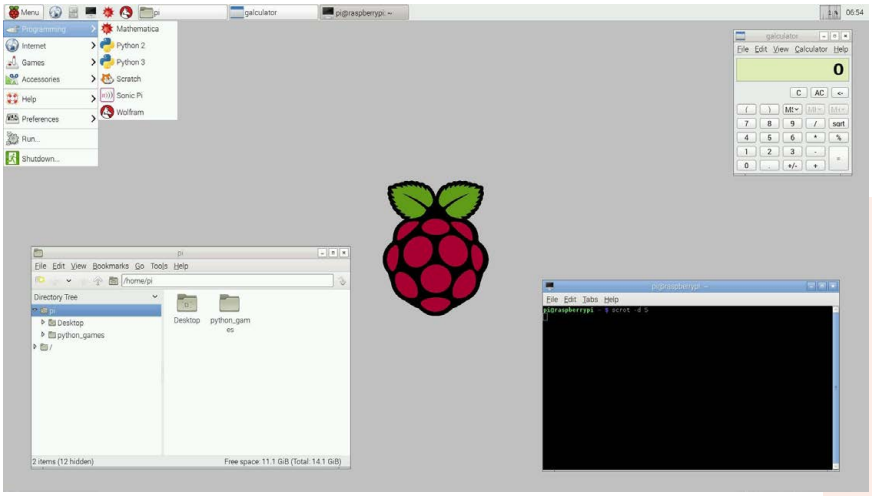
We bring you some optional updates to Raspberry Pi Desktop, which gains modern features while keeping its simplicity.

By **Simon Long**

What is currently known as the Raspberry Pi Desktop has been around for over a decade now. It started out as a mildly customised version of the stock LXDE desktop, and over the years has slowly evolved into something with almost entirely different underpinnings, based on Wayland and labwc. The look has been refreshed on several occasions, with new icons, themes, and desktop backgrounds, to the point where there is very little visually in common between the modern Raspberry Pi Desktop and the original LXDE.

But one thing which has never changed is the overall feel of the desktop, which, with its taskbar, status icons, and main menu launcher, is very reminiscent of the state of the art in desktop design from 30 years or so ago, with Windows 95 and Apple System 7. Not without good cause – both were good examples of an intuitive desktop interface, and they have the benefit of being very familiar to most users. The intention has always been to offer a desktop experience which most people can simply use without a learning curve, and I believe Raspberry Pi Desktop has offered that.

Desktop designs have continued to evolve in that time, and last year, I started to think that some of the features which are now commonplace in other desktops might be beneficial in ours. Since then, I have been working on a few ideas for how we might add these features to the desktop – not losing the fundamental simplicity of it, and not forcing change for



◀ Back in the day!



▲ The new icon dock

the sake of change on users who are happy with the way it is, but just providing some more options.

I do have to stress here that these new features are entirely optional; unless you choose to enable them, your desktop will continue to look and feel exactly the way it always has. These are purely intended to give users who want a more up-to-date desktop experience the option to have one if they want it.

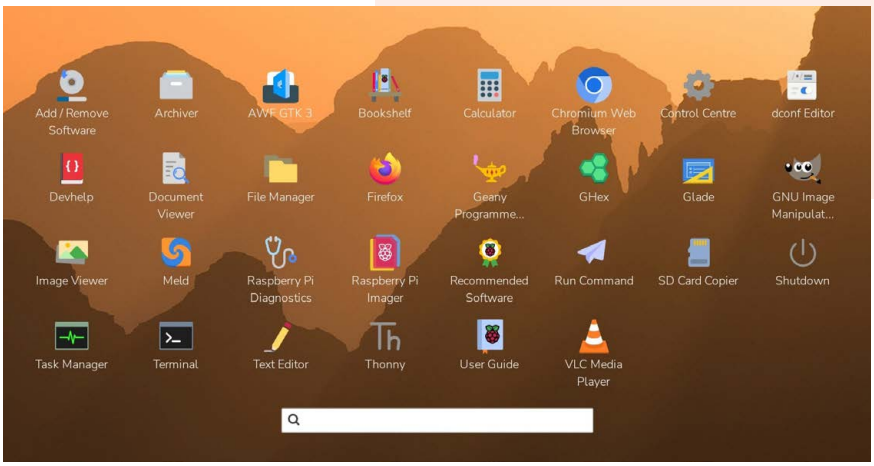
So, assuming you do fancy a change, what do you get?

The icon dock and launcher

The biggest change is the introduction of an icon dock, which can either replace the existing taskbar, or be used in addition to it. Two new widgets have been added which are intended to be used in the dock – one is a graphical application launcher, and the other an icon-based combined quick launcher and task list.

The Raspberry Pi icon for the graphical application launcher opens the launcher screen. (This is intended as a replacement for the main menu widget which has been the standard application launcher up until now.)

This shows all the application icons which would have been in the main menu. By default, icons are sorted in alphabetical order, but they can be dragged and dropped to rearrange them.



▲ Raspberry Pi Desktop's launcher screen

These new features are entirely optional

Single-clicking any icon launches the application; right-clicking (or a long press on a touchscreen) brings up a context-sensitive menu similar to that shown in the traditional main menu.

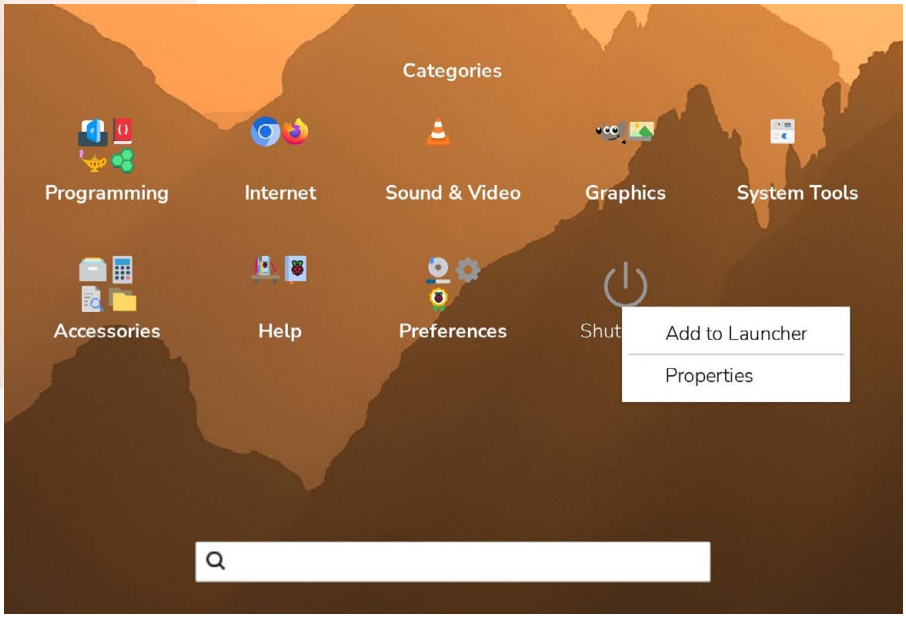
The search box can be used to filter the displayed applications, and the cursor keys and **ENTER** are used to navigate and select applications from the keyboard.

Right-clicking the raspberry icon and selecting the 'Configure Icon Menu Widget' option gives some configuration options for the launcher.

Enabling alphabetical sorting undoes any drag-and-drop rearrangement and restores the default order of icons.

Enabling menu categories makes the launcher hierarchical, with a top-level screen for application category icons which can then be clicked to enter individual launcher screens for each category – this is useful if you have a large number of applications installed. To help with remembering which applications are in which category, turning on composite category icons shows miniature versions of the first four application icons in each category in the top-level screen instead of





▲ The Configure Icon Menu

▶ The new launcher screen with menu categories enabled, showing application categories

the category icons themselves. (If menu categories is enabled, then drag-and-drop can no longer be used to rearrange the launchers; instead, the hierarchy and arrangement from the main menu is used, which can be edited using the Main Menu panel in Control Centre as before.)

By default, the icons are displayed overlaid on the current desktop background, but in the event that this looks too cluttered, a transparent or opaque overlay colour can be applied.

The new task list

The combined quick launcher and task list can display a set of the most commonly used application icons, and will by default show the same set of icons as the original quick launcher from the taskbar. New applications can be added by right-clicking an entry in the main menu or the graphical application launcher and choosing 'Add to Launcher' from the context-sensitive menu. As with the application launcher, icons in the quick launcher can be reorganised by dragging and dropping.

Clicking an icon launches the application. Running applications – whether launched from the quick launcher or elsewhere – are shown with a superimposed count of the application's open windows. If an application was already in the quick launcher, this count is superimposed on the existing icon; if the application was not in the launcher, its icon is added to the end and the count is superimposed on the new icon.

Clicking on an icon for a launched application brings all that application's windows to the foreground. If all the application's windows are already in the foreground, a new window is opened (if the application supports multiple windows).

Right-clicking an icon brings up a menu which shows all the application's open windows, allowing them to be individually brought to the foreground, and also offers options for maximising or hiding the application's windows.

Applications can be removed from the quick launcher by right-clicking the icon when the application is not running, and then choosing 'Remove from Launcher' from the context-sensitive menu.

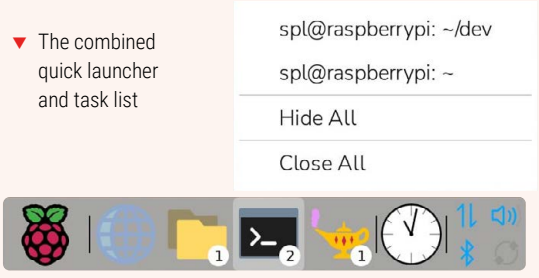
These two new widgets are intended to offer alternatives to the existing main menu and window list, and while they are intended for use in the new dock, they can also be used in the original taskbar if desired; likewise, the original main menu and window list widgets can be used in the dock.

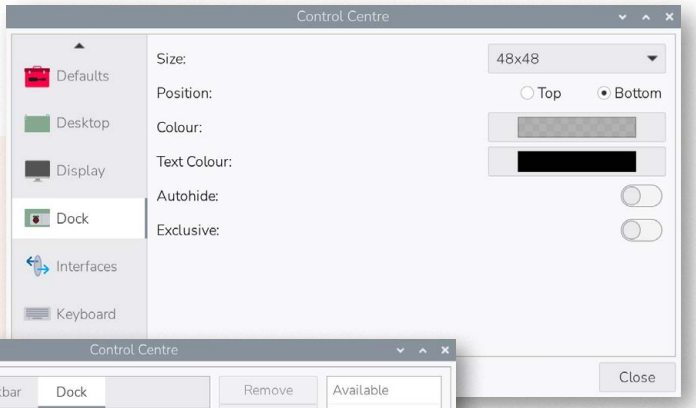
Customising the dock

The dock itself can be customised in the same way as the taskbar. A Dock page has been added to the Control Centre and offers options for colour, position, and icon size.

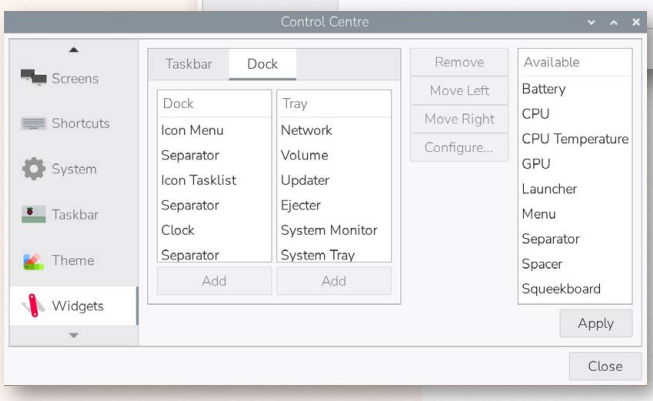
One often-requested feature which has been added is the ability for the taskbar

▼ The combined quick launcher and task list





▶ Control Centre is used to customise the dock



▲ Add widgets to the dock

▼ Running the taskbar and dock to display icons in the top-right of the screen

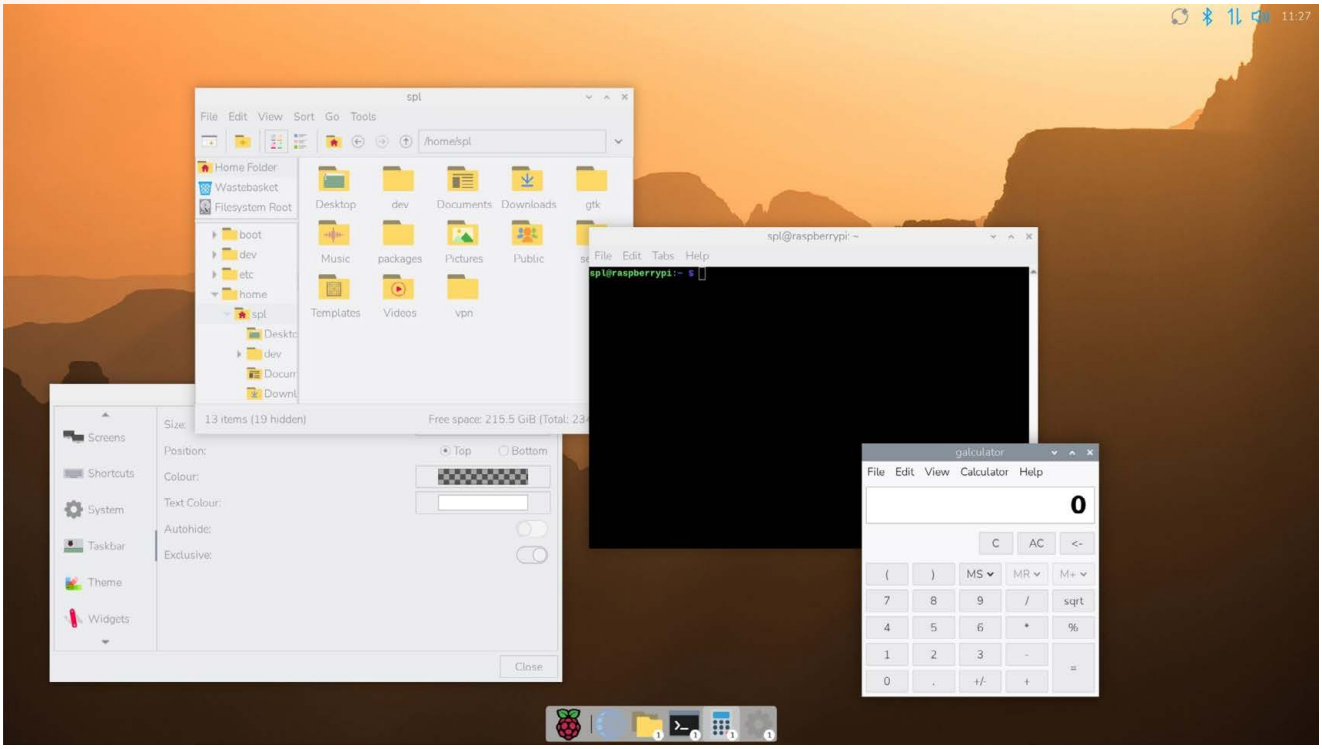
and the dock to automatically hide when not in use – if the 'Autohide' option is enabled for either, it will slide away when the mouse is not over it, and reappear when the mouse is moved to the edge of the screen where it has hidden. The 'Exclusive' option controls whether a maximised window covers the taskbar or dock; if exclusive is set for either, maximised windows will not overlap it. This can be useful if you want, for example, the clock to always be visible.

A new Widgets page has been added to the Control Centre which allows the plug-ins to be placed in either taskbar or dock, and arranged as desired.

The dock offers two locations for widgets. The left-hand side of the dock is the dock proper, while the right-hand side is given over to the 'tray'. This is intended to hold status icons, which are shown as two rows with the icons half the size of those in the main dock.

When icons are added to the tray, they are automatically split across the two rows such that the lengths of the top and bottom rows are kept as consistent as possible. If more control over the split is required, the 'tray split' widget can be added to the tray widgets – this imposes a hard line break between the widgets at this point.

In order for either the taskbar or the dock to be shown, either must contain at

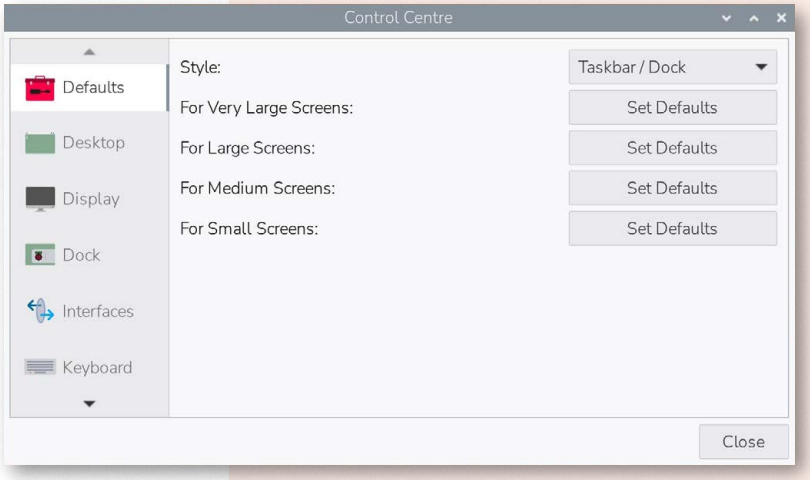


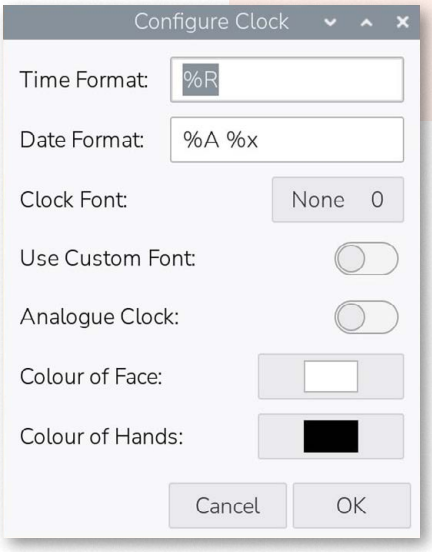
least one widget – if either contains no widgets, it is hidden.

Note that the taskbar and the dock can be used simultaneously – for example, if desired, the status icons can be displayed on the taskbar as they are now, with just the new launcher and task list in the dock. (This is how my own desktop is configured.)

I'd encourage you to play about with the Widgets page in Control Centre to come up with whatever arrangement of taskbar, dock, and widgets works best for you. To make it easy to try the new features, an option to switch between a few preset desktop styles has been added to the Defaults page in the Control Centre.

Setting the style to 'Dock' will remove all widgets from the taskbar, add the new launcher and task list to the dock, and add the status icons to the tray. Setting the style to 'Taskbar / Dock' will add just the launcher and task list to the dock, while leaving the status icons on the right of the taskbar. Or if you just want the desktop as it has always been, set the style to 'Taskbar', which will leave the existing main menu and task list on the left of the taskbar, with the status icons on the right.





▲ Preset desktop styles have been added to the Defaults page in Control Centre

◀ An analogue clock is now available

The biggest change is the introduction of an icon dock

Other changes

One other small change which you can see in some of the pictures above is the addition of an analogue clock mode to the clock widget – simply right-click the digital clock, choose 'Configure Clock Widget', and turn on the analogue clock mode. The colour of the face and the hands can also be customised.

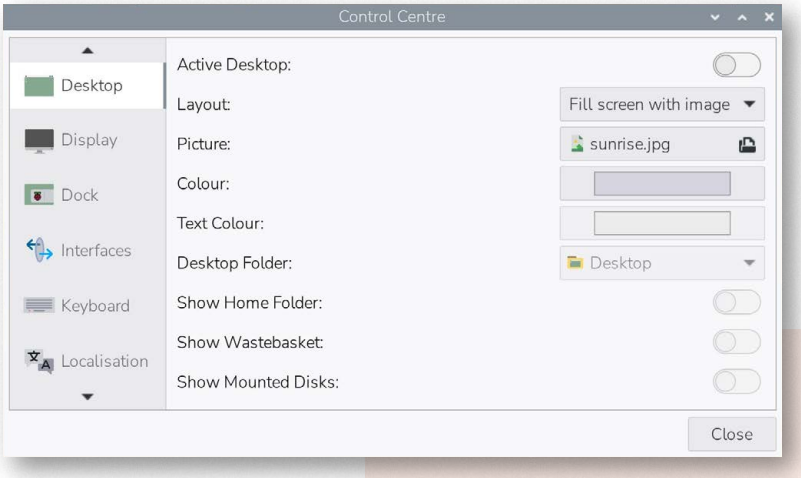
In addition to the dock changes, there is now the ability to use a more efficient mechanism for drawing the desktop background picture. This has always been drawn by pcmanfm, the file manager application – as this is the file manager, it allows icons to be placed on the desktop. But if desktop icons are not required, significant memory savings can be made by using the very lightweight swaybg program to draw the desktop.

This can be enabled in the Desktop page of the Control Centre – switching off 'Active Desktop' disables the drawing of the desktop by the file manager and instead uses swaybg to display the same picture. (The Wastebasket which is normally displayed on the desktop can then be found in the Places pane in any file manager window.)

We now recommend disabling the active desktop for any platform with less than 2GB of memory – this is checked at boot, and a notification is displayed suggesting that active desktop be disabled. Also, if you choose either of the new dock-based styles for the desktop in the Defaults page, active desktop is disabled by default, but it can be re-enabled from the Control Centre if desired.

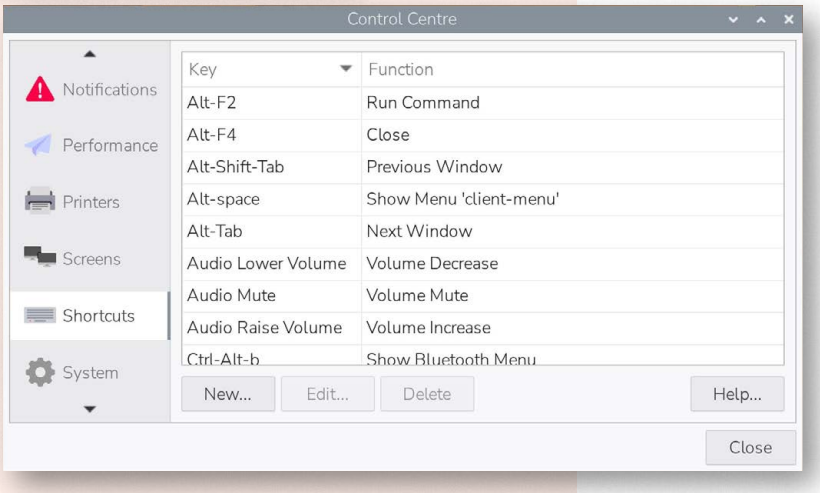
Note that the file manager was previously responsible for automounting removable drives; as this would no longer happen if the file manager was not running all the time, this functionality has now been moved into the ejector plug-in, which will now automount and display a notification when a drive is inserted.

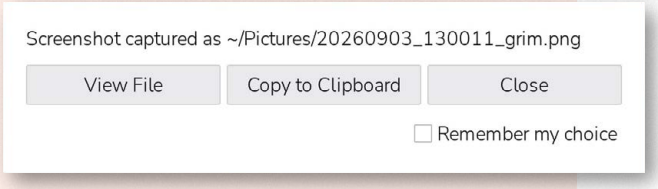
A new Shortcuts page has also been added to the Control Centre. This allows all system keyboard shortcuts to be viewed and edited, and new ones to be created if desired. (Note that this



◀ Disable Active Desktop for any platform with less than 2GB memory

▼ A new Shortcuts page has been added to the Control Centre





▲ A new screenshot dialog provides options when you take a snap

only covers general system shortcuts, not those which are assigned by individual applications.)

One final change which has been made is an enhancement to the creation of screenshots. Previously, pressing the **PrtScrn** key took a shot of the entire screen and saved it to the **Pictures** folder. Now, the **PrtScrn** key takes the screenshot and also opens a dialog offering the option to open the captured file in an image editor, or to copy it to the clipboard. (If you always want to do the same thing, just tick the 'Remember my choice' checkbox before pressing one of the buttons, and the prompt will not be shown in future.)

Holding down **ALT** when pressing **PrtScrn** does the same thing, but first displays a selection rectangle allowing you to select an area of the screen to be captured rather than the whole screen.

How to get it

We've released a new image with all these new features added. As I said at the start, these features are all optional – if you install the new image, the default appearance of the desktop will be the same as it has always been, with a taskbar and no dock; to try the dock, just launch the Control Centre, go to the Defaults page, and select either the 'Taskbar / Dock' or 'Dock' style.

Similarly, for existing images, simply update using either the updater widget or the usual terminal commands:

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

Again, installing the updates will not enable the dock; use the Control Centre as above to try it.

SUBSCRIBE TODAY FOR JUST £10

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

SUBSCRIBER BENEFITS

Free delivery
Get it fast and for free

Exclusive offers
Great gifts, offers, and discounts

Great savings
Save up to 37% compared to stores

SUBSCRIBE FOR £10

Free Pico 2 / Pico 2 W

3 issues of Raspberry Pi Official Magazine

£10 (UK only)

SUBSCRIBE FOR 6 MONTHS

Free Pico 2 / Pico 2 W

6 issues of Raspberry Pi Official Magazine

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



Subscribe by phone: +44 (0)1778 392472

Subscribe online: rpimag.co/subscribe

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

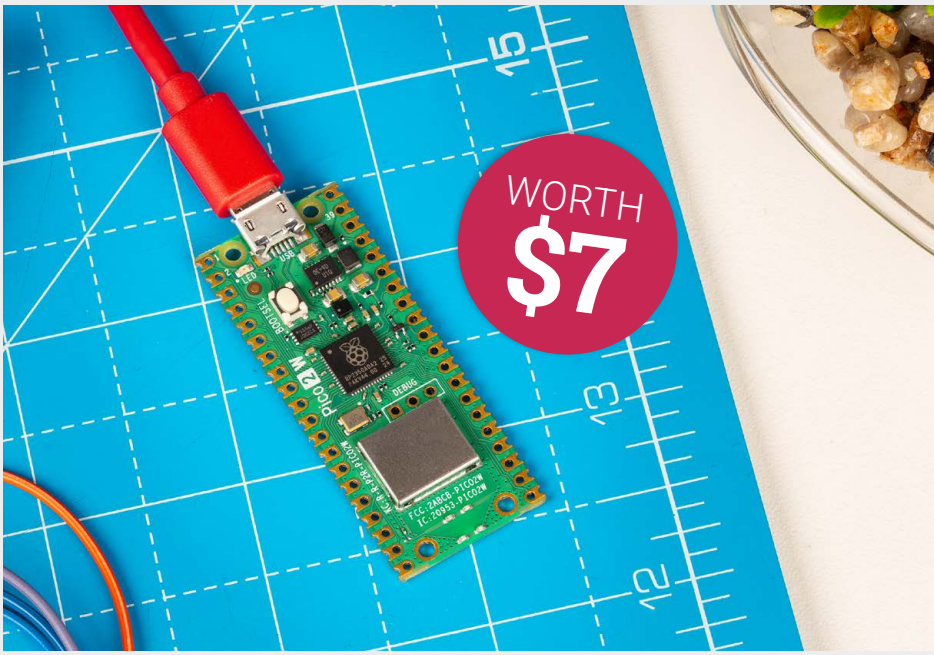
Subscribe for £10 is a UK-only offer. The subscription will renew at £15 every three months unless cancelled. A choice of free Raspberry Pi Pico 2 W or Pico 2 is included with all subscriptions.

SUBSCRIBE TODAY AND GET A

FREE Raspberry Pi Pico 2 W (or Pico 2)

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

- A brand new RP2350-based Raspberry Pi Pico 2 W / Pico 2 development board
- Learn to code with electronics and build your own projects
- Make your own home automation projects, handheld consoles, tiny robots, and much, much more



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



Buy now: **rpimag.co/subscribe**

SUBSCRIBE on app stores

From **£2.29**

Available on the App Store

GET IT ON Google Play

Recreate Apollo

This accurate Apollo DSKY replica is truly out of this world.

By **David Crookes**



Maker

Michael Newton

Michael is a commercial aircraft avionics engineer based in Calgary, Alberta, and if he's not on the flight line, he's learning and recreating historical aircraft and spacecraft hardware using modern technology and techniques.

recreateapollo.com

In 1966, three years before Neil Armstrong became the first human to set foot on the Moon, the Charles Stark Draper Laboratory at the Massachusetts Institute of Technology introduced the Apollo Guidance Computer (AGC). Installed on board each Apollo Command Module and Lunar Module, its power was on a par with the Apple II computer with less than 100kB of memory and a clock speed of 1.024MHz. Yet, as Michael Newton points out, it allowed astronauts to execute navigation routines, autopilot manoeuvres, burns, orbital rendezvous, and Earth entry from their spacecraft. It was also the first computer to be based on silicon integrated circuits.

Michael is an engineer who is passionate about building historical spacecraft hardware using modern technology and techniques and he's been leading an open source project called Recreate Apollo. It uses a Raspberry Pi computer running the Virtual AGC emulator to power a 1:1 scale 3D-printed DSKY – the Display and Keyboard used by Apollo astronauts. Specifically running the Apollo 13 Command Module software, it grants anyone the chance to get hands-on with a piece of computing and space history.

"My interest in the project began many years ago when, during the COVID-19 pandemic, a group of Apollo enthusiasts began translating the original MIT Instrumentation Lab engineering documents into modern-day CAD files with the eventual goal of 3D printing," Michael says. The documents had been made available following a public records request and they were archived by the Virtual AGC group.

"I lost touch with most of their progress as life carried on, but about a year ago I came across another maker named Eric who runs a blog called **mkme.org** and I saw that he had created a DSKY based off of the now nearly-completed drawings uploaded on GitHub. Unsure where he sourced his files, I went on the hunt for them myself. I found many in different GitHub repos or other open-source projects, but other parts needed some work to refine and some others still I had to model on my own. After a few weeks of work, I had a full set of .stp files ready to print that I was satisfied with. I purchased a 3D printer and began learning that world."

> *One of my biggest goals for this project was documentation*



Lift off

Michael began learning as he went along, identifying three main tasks. First, he would print, finish, and paint the case. Then he would organise and wire the hardware. He also needed to write the code and software. "I tackled each part of the project over a few months, in that order, taking my time and learning as I went along," he says. "But one of my biggest goals for this project was documentation. I wanted anyone who is interested to be able to build one of these, even those with no prior knowledge. My guide, available on GitHub, is the fruit of those labours."

Driven by his love for the Apollo mission and the AGC, he wanted to be as faithful to the original as possible. "Physically, the DSKY is about 90% authentic," he says. "From the outside, I strived for a perfect visual representation and I'm satisfied where it's at." Michael had to make adjustments internally, changing the mounting points for the microcontrollers and the Raspberry Pi computer.

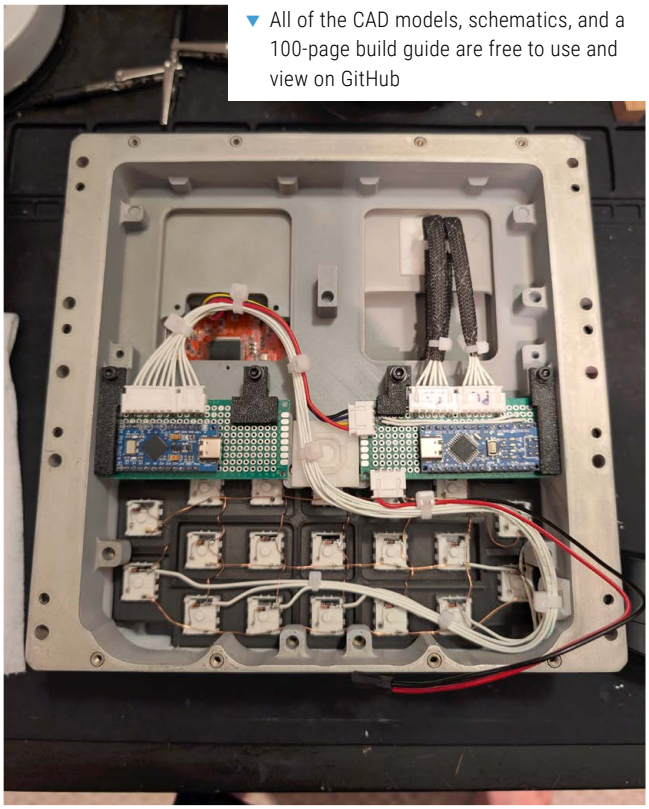
"I also needed to get a bit creative with the screen mounting, since the PCB that the screen mounts on is physically larger than the part designed to hold the original," he says. "I modelled a clip-style holder for the screen and made a few minuscule adjustments to get the fit down perfect. The keycaps were also made a bit differently than the originals, since the lettering was too small for me to print with great definition. Even though the font matches the historical one (yes, someone recreated the font!), the words are just a bit larger."

01. A Raspberry Pi computer runs all the software needed for the build while interfacing with the sensors and I/O

02. The case is designed to be as faithful as possible to the original and looks like it came straight from the spaceship



▼ Michael has built his DSKY as a prototype for an improved version, but all the printed parts are modelled on archived NASA drawings



▼ All of the CAD models, schematics, and a 100-page build guide are free to use and view on GitHub

Rocket science

A Raspberry Pi computer felt like a perfect fit. "It is nearly purpose-built for this project," Michael says. "It integrates through microcontrollers via USB, it doesn't need a lot of power, it has wireless LAN built in for SSH and remote applications, and it also runs cool. Raspberry Pi OS is very easy to install, too, and you can create a custom distribution to send an entire OS instance to other users, which makes that section of the build easy to copy."

More than that, however, Raspberry Pi easily runs yaAGC, VirtualAGC's computer program that replicates every characteristic of the real AGC. "This includes the unique asynchronous timing (interrupt) that saved Apollo 11 (1201/1202 alarm codes), the 15-bit-plus-one-parity word length, and the archaically slow 1.024MHz clock speed," Michael says.

The data used by the AGC is fundamentally different from how all modern computers are engineered. The individual 1s and 0s are arranged as a 15-bit data block with a single parity bit used for error checking. Michael says the emulation proved extremely difficult to achieve. But the programmers managed to emulate the AGC with the exact same logic, using the same timing, as the original.

"Fundamentally, the exact same constraints are preserved in this emulation because the timing of the AGC is kept intact and the memory that the computer reads is virtualisation of the real code that was woven into core rope ROM and magnetised in the RAM," Michael explains. "In fact, it emulates the real computer so well that, to me, VirtualAGC is not an emulator. It's a real Apollo Guidance Computer that borrows your processor."

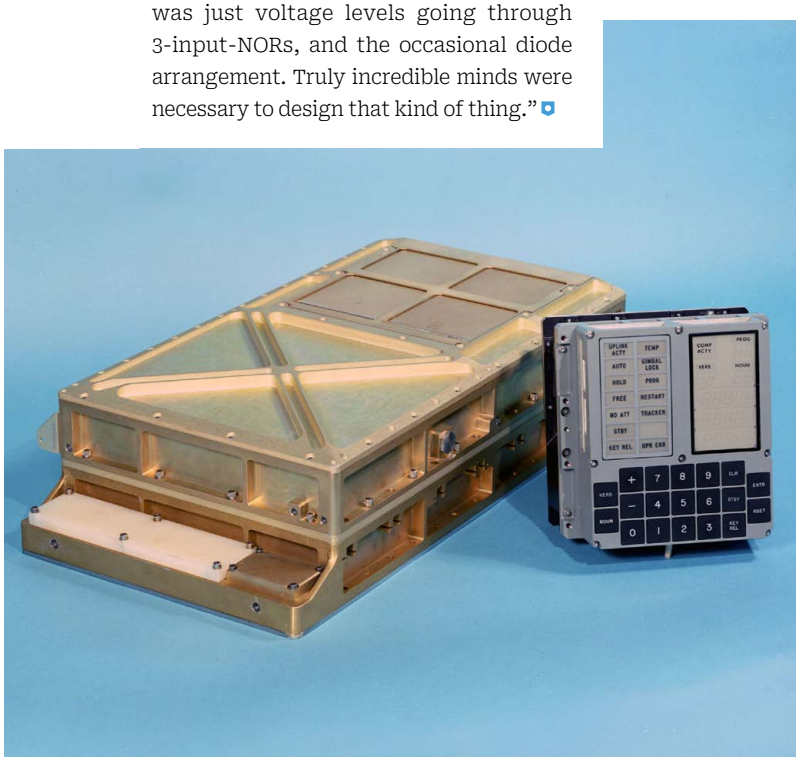
Quick FACTS

- It recreates the Apollo Guidance Computer's Display and Keyboard
- Modern electronics, including Raspberry Pi, substitute for the older parts
- There's a fully-working keypad that allows interaction with the software
- Arduino Pro Micros operate the annunciators, screen, and keyboard
- It would cost you about $685 to recreate

Great minds

The result is an amazing, fully usable device, one that allows users to interact using a keypad. For this, Michael used off-the-shelf mechanical keys with the highest actuation force and most solid 'click' he could find. "I settled on Kailh Box Navy," Michael says. "They feel and sound OK, and they are easy to get, but they are far from the real keypad." He's currently looking to create a much more premium DSKY with a full aluminium body, better annunciators, and maybe a real electroluminescent display. This, he says, would incorporate a better keypad.

As it is, however, the Recreate Apollo project has allowed Michael and others to learn much more about the AGC. He says the ingenuity of those behind the original computer is clear. "The memory alone was an ingenious design of magnetism and did you know that the Apollo Guidance Computer used only ONE type of logic gate: the 3-input NOR? All the mathematics, data movement, navigation, autopiloting, burns – everything, was just voltage levels going through 3-input-NORs, and the occasional diode arrangement. Truly incredible minds were necessary to design that kind of thing."



▼ This is the original Apollo Guidance Computer with the DSKY as used on Apollo spacecraft. Source: Wikipedia (**rpimag.co/wiki_agc**)

Building an AGC

1. The recreated AGC uses a custom-built power supply integrated into a mobile phone battery bank. This allows the Raspberry Pi computer to work completely independently. When powered, using a switch, Raspberry Pi boots and supplies power to two Arduino Pro Micros and the display.



2. Raspberry Pi also boots the program yaAGC (running Apollo 11's Command Module code) together with the Python scripts in a headless environment. Once the DSKY receives a 'PROG' alarm, the LEDs light up and the program is ready to go. The user is then able to perform a few demonstrations.





3. There is also an option to switch to a 'peripheral' mode by long-pressing one of the keys. This allows the DSKY to be used over a local network with Project Apollo – NASSP, an add-on for the Orbiter space flight simulator released on the PC, which simulates all of the Apollo missions.

Pupper v3

This is a dog that can teach robotics enthusiasts a trick or two.

By **David Crookes**



Maker

Paige Bailey

Paige is the AI Developer Relations Lead at Google DeepMind and she has worked on Google's most advanced generative models.

rpimag.co/pupperv3

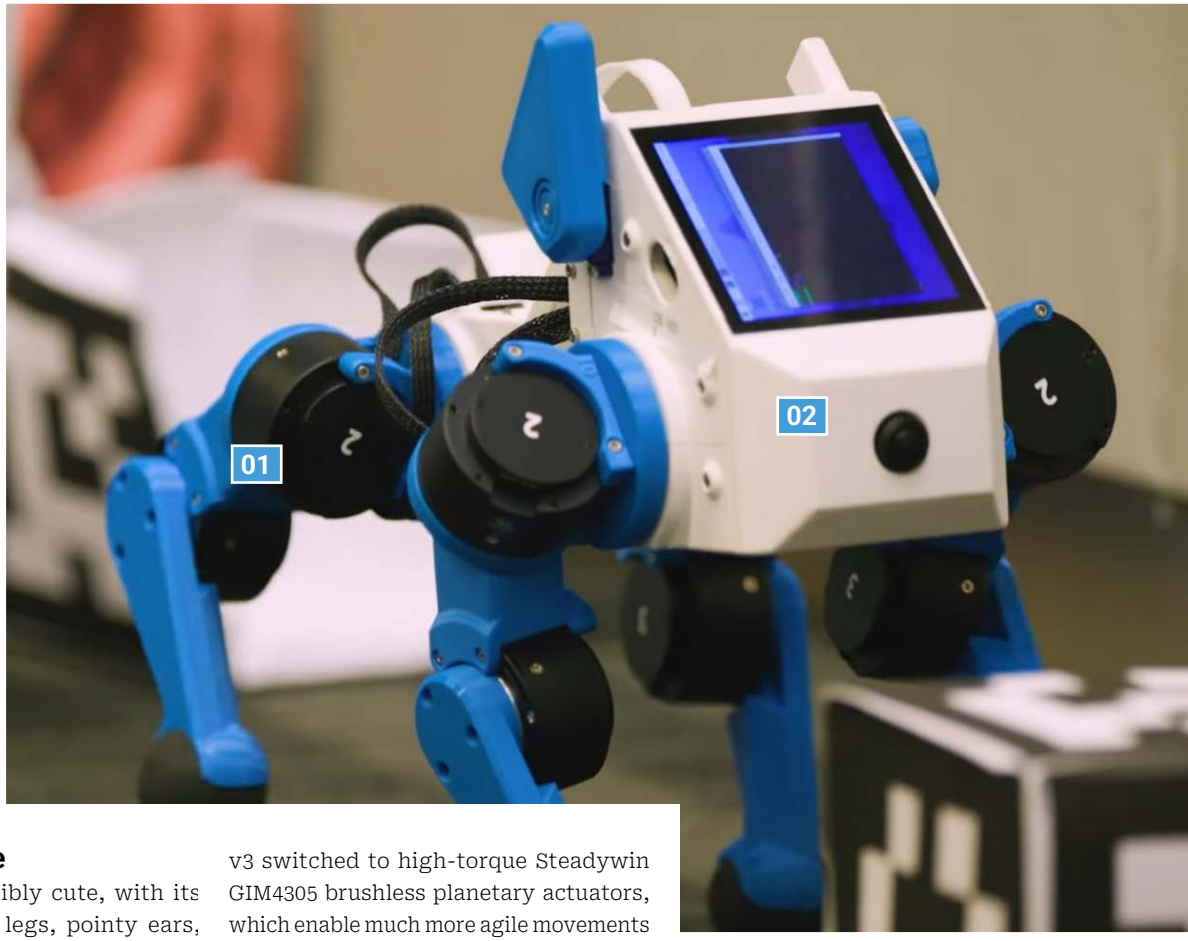
The Stanford Pupper has long been a source of inspiration for robot enthusiasts. Building on Stanford Student Robotics' open-source Doggo project developed in 2019, Pupper emerged as an accessible quadruped designed to teach students the fundamentals of robotics, from mechanical design and electronics to software and control.

Now on its third version, Pupper is staying true to its roots. Created by Stanford University's CS 123 team in collaboration with the non-profit Hands-on Robotics, the latest model is more agile and capable than ever before. Crucially, it also remains relatively inexpensive to build and, because it's been made with safety in mind, it can be used in the real world to make a big difference to people's lives.

"Traditional quadruped research robots like Boston Dynamics Spot or commercial industrial platforms cost tens of thousands of dollars and they are locked behind closed hardware and software, which make them inaccessible for most students and consumers," says Paige Bailey, AI Developer Relations Lead at Google DeepMind, another collaborator on the project.

"The Pupper project was started at Stanford to make agile, dynamic quadruped robots affordable and accessible for students, researchers, and hobbyists – to truly democratise robotics. The team also wanted to build an expressive, non-intimidating and safe companion. This inspired extensions like the Dr Pupper initiative with Stanford Medicine, which explores using friendly robotic pets to comfort children and paediatric patients in hospitals where live animals cannot go due to allergy or infection risks."

> *Pupper's paws are standard squash balls glued to its lower legs*



01. Pupper has been trained to walk over obstacles and interact using voice and motion

02. It senses its environment using cameras, microphones, and proprioception

Lead by example

Pupper looks incredibly cute, with its small body, nimble legs, pointy ears, and expressive eyes. The body and leg components are 3D-printed using PETG, which is perfect for withstanding movement and repeated loads. There are sensors including a Bosch Sensortec BNO086 six-degrees-of-freedom inertial measurement unit (IMU) which tracks acceleration and rotation across three axes to help the robot maintain its balance as it walks.

There's also a Sony IMX296 wide-angle/fisheye camera mounted on the robot's nose, a microphone, a 3W speaker, and a 4×4-inch LCD face display for emotions and status. But some significant changes have been made from versions one and two of Pupper. Inexpensive RC hobby servo motors and traditional inverse-kinematics gait planning were used in the first two versions. This kept the quadruped price down.

But since hobby servos lack the torque bandwidth, impact resistance, and compliance needed for dynamic motion, an alternative was used. "Pupper v3 switched to high-torque Steadywin GIM4305 brushless planetary actuators, which enable much more agile movements so Pupper can do a lot more," Paige says. "Traditional kinematics were also replaced by end-to-end neural network locomotion policies trained in GPU physics simulations."

Twelve of these actuators are used, offering three degrees of freedom per leg, and there are mini servos for expressive ear twitches. "Pupper's paws are standard squash balls glued to its lower legs, providing high-grip, shock-absorbing feet," Paige adds. And at the heart of it all is a Raspberry Pi 5 computer.

"The team opted for a Raspberry Pi 5 because it delivers substantial quad-core processing power and high-speed I/O (CAN bus, USB, I2C, SPI) at an affordable price," Paige explains. "It natively runs standard Linux (Ubuntu/Raspberry Pi OS) and the Robot Operating System 2 (ROS 2) stack, and it includes a high-speed PCIe interface, allowing optional hardware AI accelerators to be mounted directly on board."

A robot unleashed

Since large language models (LLMs) and vision-language models (VLMs) – such as Google DeepMind's Gemini, and OpenAI's and Anthropic's models – have advanced significantly in recent times, there was a further opportunity for improvement. "The project has expanded into a fully interactive agent capable of voice conversation, visual scene understanding, and autonomous following," Paige says.

In other words, users can interact with Pupper using voice and motion and they can teach it to do things. But to get to this stage, the team had to resolve a series of problems. In conversational robots, high latency makes interactions feel awkward. "The Pupper team solved this using Google DeepMind's Gemini Live API and OpenAI's GPT-Realtime API paired with Cartesia for ultra-low-latency voice synthesis, allowing near-instant speech generation," she adds.

The on-board microphone sits near cooling fans and motor drivers and the team found this could lead to sound pollution when sending voice commands to Pupper. Paige says this was mitigated with software noise filtering and prompt engineering to ignore ambient motor hum.

Handcrafting gaits and complex tricks (like dancing, shaking paws, swimming, or sitting) is also prone to failure. "The team used massively parallel physics simulators (such as Isaac Gym/MuJoCo), training reinforcement learning (RL) policies across thousands of virtual robots simultaneously – simulating three months of trial-and-error in just one hour before deploying to the real hardware," Paige continues. Slowly but surely, Pupper v3 began to take shape.



▶ Pupper can be used in medical situations to help keep patients calm and comforted

A great companion

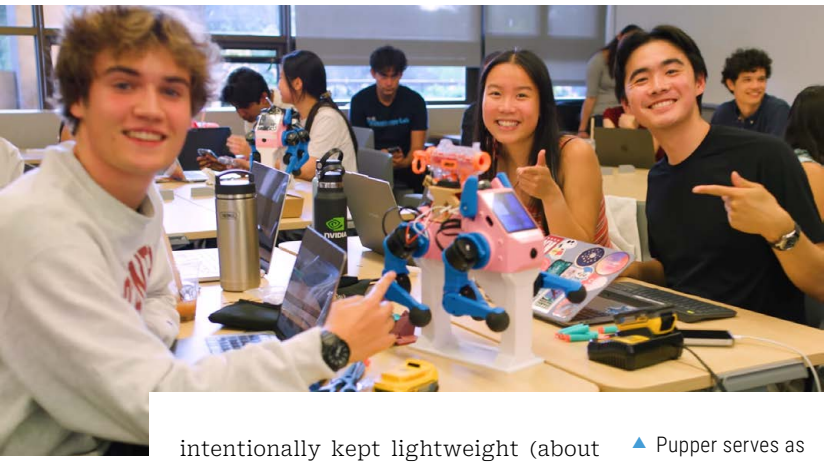
Under the hood, Pupper's software is divided into several layers. A Python-based AI layer handles high-level tasks: OpenAI Realtime is used for conversational logic/tool calling, Google Gemini Flash for visual perception, and Cartesia for text-to-speech and voice cloning. Robotics middleware, ROS 2 Humble, acts as the communication backbone. "It connects different modules – the camera feeds, AI agent, joystick teleoperation, kinematics – via standard publisher/subscriber topics and services," Paige says.

Pupper's walking behaviour is trained using GPU-accelerated RL environments that are exported into a lightweight format (ONNX or TorchScript runtimes on Raspberry Pi 5). At the lowest level, custom firmware communicates with the motor drivers over CAN bus, handling high-frequency field-oriented control, torque feedback, and safety limits (using C / C++ / Embedded CAN).

Since the team wanted to ensure the robot was safe to use, even by an inquisitive child, Pupper was assessed for its potential issues. Chief among them was the potential for little fingers to become pinched or bruised by the high-torque brushless motors or at the points between the leg linkages. "To mitigate these dangers, Pupper v3 was

▼ The robot has been designed to be safe and light, making it ideal for using around children





intentionally kept lightweight (about 3kg) while using soft squash-ball feet and integrating compliant motor control that 'gives' under external resistance," Paige says. Pupper also features software/hardware emergency stops.

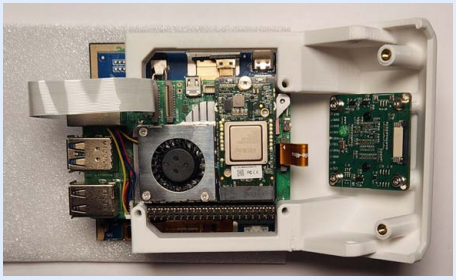
As such, Paige says Pupper v3 is ideal not only as a way of teaching students and makers embodied AI, dynamic locomotion, and Robot Operating System 2, but as a device that can be used in less technical settings. "Pupper's friendly, expressive appearance, small form factor, and voice-driven persona make it an ideal companion for environments where real pets aren't viable, such as hospital wards, nursing homes, or classroom environments," Paige says. "We're excited to share Pupper with the world!"

▲ Pupper serves as the core hands-on platform for Stanford's CS 123 robotics course

Quick FACTS

- Pupper has been designed to teach robotics and AI
- The robot costs less than $2000 to build
- Vision-Language Models are used for natural interactions
- It has a hi-res screen, expressive ears, and an agile body
- All of Pupper's code is open source, ready for download and modification

Training a robot dog



1. The fisheye nose camera captures the environment and the on-board microphone streams the user's voice commands. The multimodal AI (GPT-Realtime plus Vision AI) processes the request in Pupper's 'brain' – which is also known as a Raspberry Pi 5 computer.



2. If the user asks for an action (for example, "Pupster, shake my hand" or "walk over books"), the AI triggers an internal Tool Call instead of just replying with text. The Tool Call routes commands to the ROS 2 locomotion controller which controls Pupper's movements.



3. A pre-trained neural network reads the desired speed/trick as well as the on-board IMU balance data, computing the target joint angles in real time. Commands are transmitted over the CAN bus to the twelve brushless motor controllers to adjust torque and velocity.

Self-sufficient mobile weather station

A weather research station powered by the weather itself. **Rob Zwetsloot** takes a reading



Maker
Jack Young
A master's student studying mechanical engineering who enjoys making electromechanical devices with real-world uses.

As Andrew Scott reminds us in the movie *Pressure*, the weather is not boring, so we were quite interested to hear about this mobile weather lab that is also completely self-powered.

"The fundamental idea of a self-powering weather station was one proposed by my supervisor Dr Will Harrison for my year 3 dissertation project," creator Jack Young tells us. "But the idea of it being mobile and easily transported was one that came about when I was researching what was lacking in the current weather station market – there weren't any existing products that could provide hobbyists, students and researchers alike an easily accessible, mobile and cost-friendly device, whilst still producing a full dataset of local atmospheric conditions that could be used for data analysis."

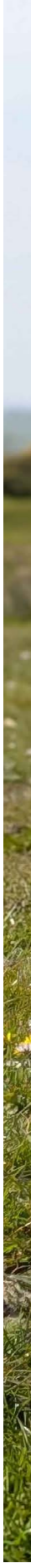
> *The idea of it being mobile and easily transported was one that came about when I was researching what was lacking in the current weather station market*

To keep the cost down, of course Jack eventually settled on a Raspberry Pi to help power it.

"After researching microcontrollers and small-scale computers, Raspberry Pi Zero W was the most cost-effective, low-power-consumption choice, and the impressively small form factor meant that the size of the weather station could be reduced by a large amount," Jack explains. "The 1GHz clock speed and in-built network capabilities also allowed for fast data collection and transfer rates to the external web server, resulting in a smooth user experience."

Everywhere you go

As well as your usual temperature and humidity sensor, along with an anemometer for wind speed recording, the solar panel used to power it allows for the station to also measure light intensity





01. A solar panel powers this mobile weather station, and takes light readings too

02. The kit is modular, with parts added and removed during transport



The enclosure had many iterations to get the size and weight right

in the location where it's set up. Of course, being used to measure weather also means it needs to survive said weather while using those sensors.

"To keep to an IP rating of 65, allowing for continuous usage in harsh weather conditions, all the internal electronics needed to be completely sealed from water," Jack says. "To achieve this, the enclosure of the device went through many iterations, and much time was spent tweaking the tolerances of its dimensions and printing parameters to ensure an airtight fit for the lid gaskets and cable grommets that housed the power and data cables. Planning the layout of the electronics and PCB was also a difficult process, having many iterations before finally being soldered, and at times I would only find out if there was a dodgy solder once the full device had been assembled! The most satisfying part of the build process was when Raspberry Pi was finally installed into its header, the power was turned on, and the blue LED fitted to the enclosure – indicating that the code was running smoothly – was blinking as intended."

Take the weather with you

Back in April, the full kit was tested in realistic conditions on the Gower Peninsula near Swansea.

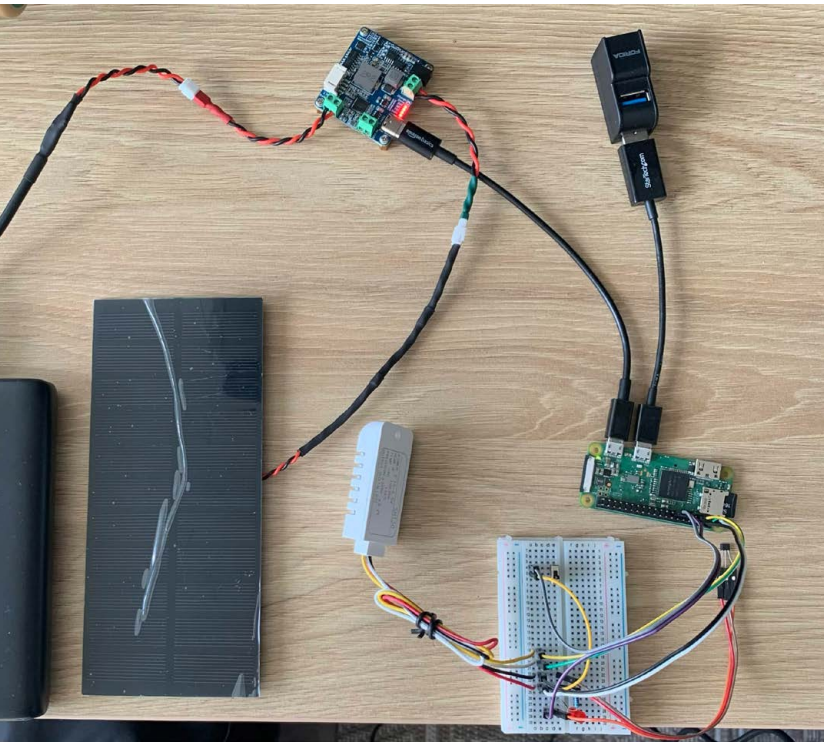
"The wind up on the open hill provided the anemometer with ample wind speed for testing the Hall effect sensor, measuring rpm via a magnet," Jack details. "It was found that at slower wind speeds the bearings had enough friction to stall the rotation and cause the device to read 0, so the bearings were lubricated and it performed much better. The temperature sensor consistently sampled at roughly 1–2°C above what was expected, and this was found to be due to the sensor head trapping heat, possibly because the sensor was producing its own heat via electrical resistance with no way for that heat to escape. Even up in the hills of the Gower [in Wales], the device successfully established a web server and stored the data it had sampled, so despite the few inevitable issues, I would say the field testing was successful!"

There are several improvements Jack wants to make for future iterations – fix the trapped heat issue, add an easier way to adjust the sampling rate, streamline the modular nature of the station, etc. – but his ultimate goal is to market it to the real world. Hopefully soon you too can do your own Raspberry Pi-powered mobile weather research.



Quick FACTS

- Backups are saved every two minutes
- You need to hold the button down for five seconds to shut the system down...
- ... then the power can be turned off
- It can be attached to standard scaffolding
- Data is safely stored on the microSD card until a network connection can be made



▲ The prototype circuit shows some of the sensors

▼ A CAD drawing of the final weather station



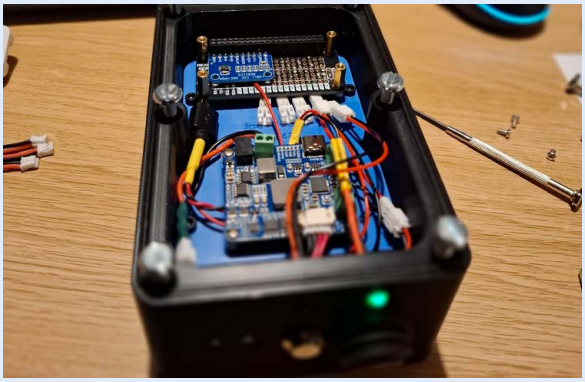
KEY:

1. Electronics Enclosure
2. Enclosure Lid & Slide Rail
3. Solar Panel Housing
4. Sensor Head & Anemometer Shaft Holder
5. Anemometer Cups
6. Curved Radius Backplate for Jubilee Clips

Deploy and measure



1. After finding a spot for your weather station, switch it on and a green LED will light on the side of the box. After a short wait, a blue LED will turn on to signify the station is ready.



2. Pressing a silver button begins gathering data from the sensors. The frequency of samples is indicated by the flashing of the blue LED and can be changed using remote access.

Weather Station 2217476
Home

Self-Sufficient Mobile Weather Station Dashboard

Live Last Sync:
Download CSV Report

Time	Temp (°C)	Hum (%)	Wind (km/h)	Volt
2026-05-06 02:33:57.123166	25.23	40.13	None	0.4
2026-05-06 02:33:56.123181	25.23	40.13	None	0.4

3. Data is transmitted to an external database over Wi-Fi and is viewable on a smartphone or computer on the same network.

D&D Book

A near magical book with ever-changing pages. **Rob Zwetsloot** gives it a read



Maker
Géza Csorba
A software engineer from Switzerland who describes himself as a curious person with many interests who tries many things.

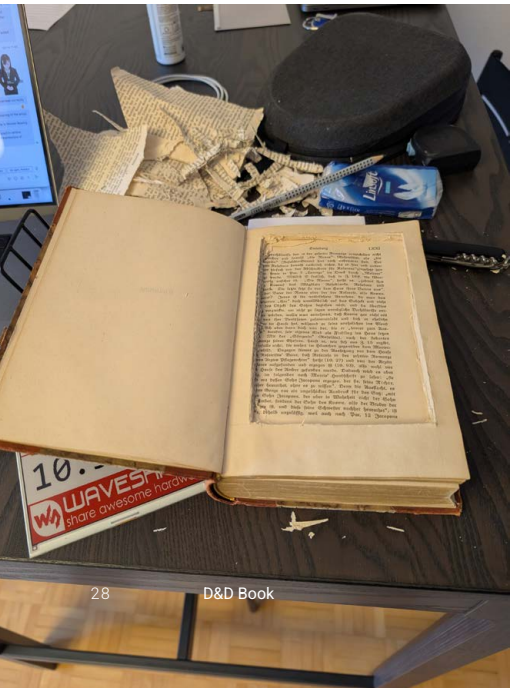


Table top role-playing games (TTRPGs) have never been bigger, and the biggest of them all is still Dungeons & Dragons (D&D), the epic fantasy RPG definitely not based on *Lord of the Rings*. They're halflings, after all. While you can still get massive tomes of knowledge to play the game, for a very long time you've been able to read the rules digitally too. What if you wanted a mix of both, making the physical process easier to transport? This is where Géza Csorba comes in.

"I [got a] Raspberry Pi 5 and some sensors, then I found an e-ink display online and couldn't resist the idea of making something out of it," Géza explains. "An e-book reader seemed like the most obvious option, but the display isn't very fast, and the ~20-second rendering time made me sad and almost made me give up. Luckily, we started playing a D&D game with some friends, and suddenly, the idea popped into my head."

Thus an ancient codex of digital knowledge was born.

Forgotten digital realms

Despite a Raspberry Pi 5 inspiring the build, it instead uses a Raspberry Pi Zero 2 W to display the text and receives button inputs in the book from a Raspberry Pi Pico acting as an HID (human input device), like a keyboard. There are four buttons for navigating the pages, the latter being handled by Zero 2 W.

"Raspberry Pi Zero 2 W runs an app written in Rust," Géza says. "The Rust app can use the C program that can render text and images on the display. The renderer is written in C because I reused some of the Waveshare sample code and, since C is not my favourite language, I kept it as a low-level program with some public API that can be used by other programs. There's a Node.js app that starts after booting the device if it's configured so, and hosts a website for uploading text and image files for the game."

> *The old book look and the rendering effect stunned my friends, so I'm a happy Dungeon Master*

There are some limits to displaying like this, some of which were set by Géza.

"Each page can have three paragraphs, and each paragraph can be text or an

01. Opening the ordinary-looking book, you're presented with a special 'magical tablet'

02. The tablet updates with the press of one of the four 'mystical stones' (buttons)



▲ A hole for the connector allows Raspberry Pi Zero 2 W to drive the display

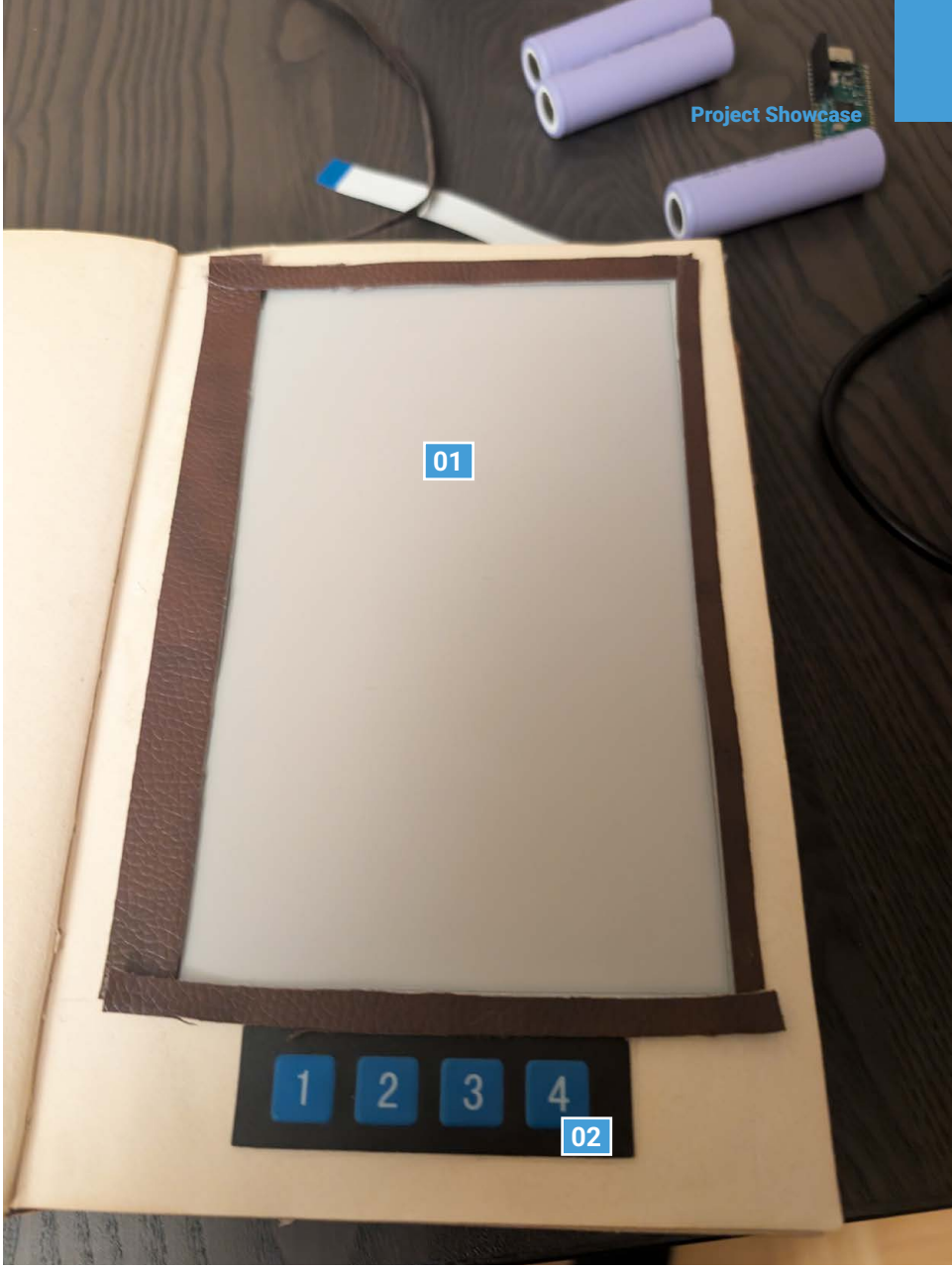
image," he tells us. "There is an option for full page rendering, as well. The images can have three colours only (black, white, red) because of the display. As of now, I manually use some dithering algorithm on the images that I upload. It's also a future update task to make them automatic, as there are CLIs that can do this after an upload."

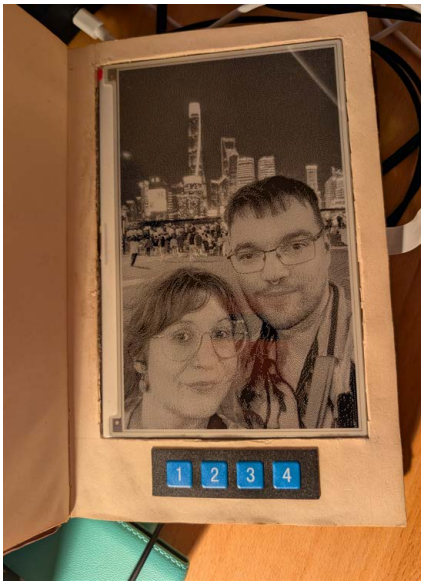
Reading with advantage

While Géza admits it's not perfect (he has to log in via Raspberry Pi Connect to shut it all down), and he would like to improve the rendering of the menu and images, it is serving its purpose.

"The old book look and the rendering effect stunned my friends, so I'm a happy Dungeon Master," Géza says. "A battery of 5500mAh is used to power the book, so I had no power issue during the game session."

We think the project deserves an Inspiration point.





◀ Dithering algorithms are applied to the images to improve how they display

Quick FACTS

- Géza got the book from a charity shop
- Preparing the book took a long time
- A leather border hides the screen's bezel
- There are many dithering algorithms for limited displays
- The buttons can plug into any USB port and work

Archive 88 project

A Raspberry Pi-based memory box displaying 50-year-old American football games brought incredible joy to a father and son, hears **Rosie Hattersley**





Maker

David Stein

David is a senior software engineer who enjoys using Raspberry Pi to "RoboCop" vintage items to repurpose old machines with new brains.

rpimag.co/memorybox

David Stein had long been fascinated with his father's early 1970s high school football career, once shelling out $200 a reel for the 16mm films held by the Heinz History Center in Pittsburgh to digitise its archives since acetate degradation had set in.

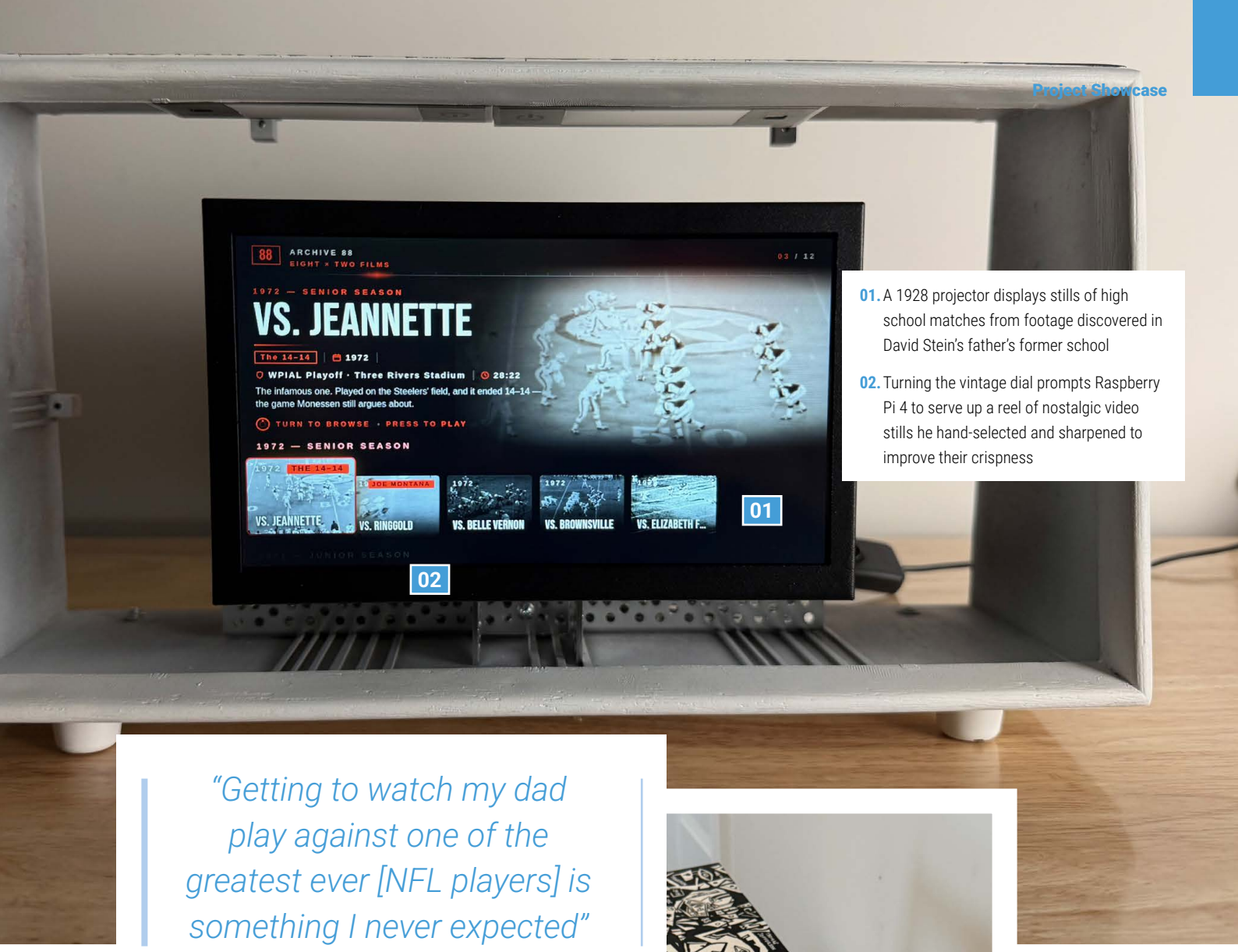
Having then discovered several more films of Monessen High School's football team matches in a storage room, senior software engineer David decided to find a way of digitising them on his own terms, for far less, in a bid to forestall the ravages of time and share some precious memories with his elderly father and his friends.

The scanner for the film collection David unearthed had a 1928 DeVry projector base, Raspberry Pi 5 brains, and a Raspberry Pi High Quality (HQ) Camera module with microscope lens to capture individual frames. The video records at 30fps while the film plays at 3fps, allowing for rich detail to be captured. An Arduino controls its NEMA stepper motors. The camera has manual exposure and focus and "rolls continuously and grabs a few shots of every film frame, then my Python script keeps the sharpest one," explains David. "Continuous motion is also gentler on fragile old film than a start/stop mechanism."

The published project (**rpimag.co/retro-football-pi**) was well received, with potential for a creative expansion. One of the Raspberry Pi team suggested David use his film scanner to turn his selected shows into slow-moving artwork along the same lines as a project we've covered before in which you can watch a frame of a festive film every day until Christmas (**rpimag.co/holiday-screen**).

Your move, creep

Encouraged by the results and how the project was received, David spent this summer creating a memory box as the direct descendant of the vintage scanner. He worked on the presentation of his original film scans, some of which he had shared on YouTube – "not necessarily the best method of displaying these films". Instead, he embarked on a process of "RoboCopping" an old Magnavox solid-state radio back to life, by converting it to play video content. Now, one machine gets the memories off the film; the other one gives them somewhere to live.

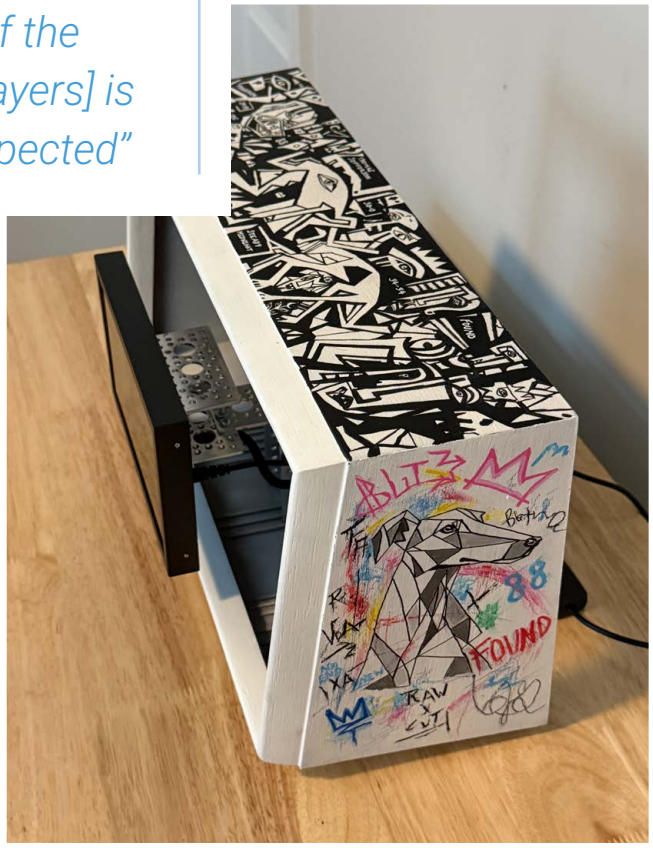


01. A 1928 projector displays stills of high school matches from footage discovered in David Stein's father's former school

02. Turning the vintage dial prompts Raspberry Pi 4 to serve up a reel of nostalgic video stills he hand-selected and sharpened to improve their crispness

> *"Getting to watch my dad play against one of the greatest ever [NFL players] is something I never expected"*

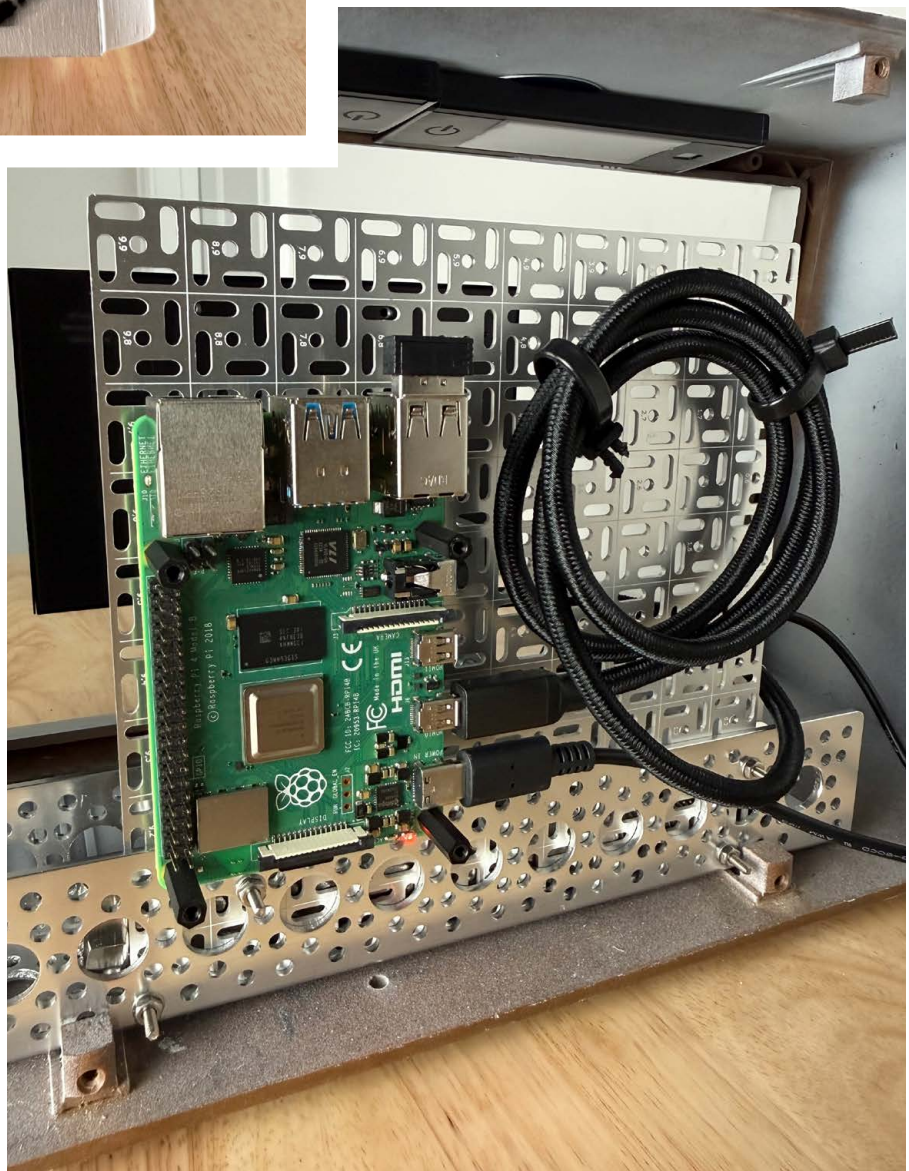
He rates Raspberry Pi as "the best tool out there for RoboCopping", due to its processing power, capability to control components, and ability to communicate with other hardware. Here, Raspberry Pi 4 acts as a video player, web server, displays the interface, and the GPIO pins react to the knob. All in all, it's "a computer quiet and cool enough to mount in a wooden box". The GPIO is crucial to the feel of the project since a physical vintage knob drives the interface – which means a rotary encoder and a video player are controlled by the same program. The screen displays 1080p video using a hardware H.264 codec. It plays silently with low power consumption. The program is set to read-only, which means precious footage can't be accidentally overwritten. All of this sealed the deal.



◀ David faithfully decorated the case with the memories of his father's school football years



Archive 88 refers to David's father's shirt number



Actobotics aluminium mounting brackets connect to the monitor and also attach it to the radio

In the trenches

David only kept the radio shell and one knob for the encoder, so he decided it would be simpler to not have to re-attach the face. Instead, he thought an 'open' concept with LEDs streaming from above the monitor would look cooler in dim lighting and would make the display appear to float. Online tutorials and AI templates allowed him to create Netflix-style navigational menus. Artwork for the case took several attempts, especially as David was determined to ensure the stencil of his father looked right, with references to games his dad had played and the catches he made.

The school friends' football careers overlapped those of a young Joe Montana. In fact, the film David digitised for this project features the future San Francisco 49ers superstar making his quarterback debut against Monessen High School, his father's team. "Getting to watch my dad play against one of the greatest ever [NFL players] is something I never expected," says David.

Quick FACTS

- Archive 88 references David's father's football shirt number
- Original reels from the Heinz History Center cost $200 a piece to digitise
- David built his own scanner based on a 1928 DeVry projector
- A Python script assigns a Laplacian variance sharpness value to each frame
- Only the sharpest shot per film frame is retained

Vivid memories

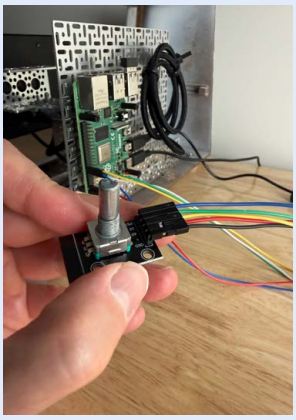
1. With an open shell of a vintage radio housing Raspberry Pi 4, David added an LED to create a floating effect, with film clips and previews chosen via a rotary dial.

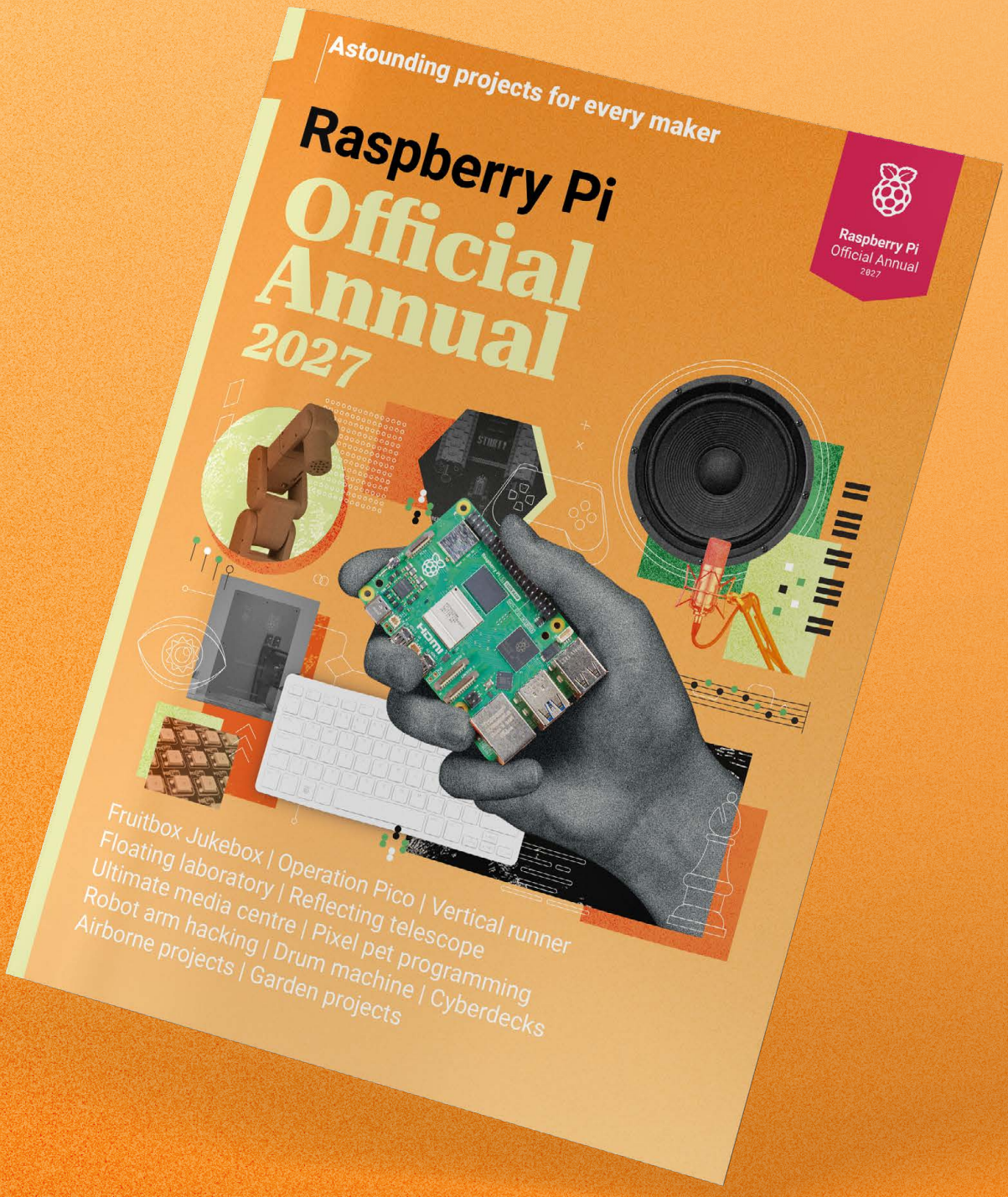




2. He set Raspberry Pi OS to automatically log in to the desktop, turn off screen blanking, and launch a **player.py** session. The display is a touchscreen, but a separate keyboard can also be used.

3. GPIO Zero reads the rotary encoder, mpv plays the video file, and there's a text file that provides a digital description of the football game being shown. A Python script runs a local version of Chromium to display poster art.




Astounding projects for every maker
Raspberry Pi
Official Annual
2027
Raspberry Pi
Official Annual
2027
Fruitbox Jukebox | Operation Pico | Vertical runner
Floating laboratory | Reflecting telescope | Pixel pet programming
Ultimate media centre | Pixel pet programming
Robot arm hacking | Drum machine | Cyberdecks
Airborne projects | Garden projects

Raspberry Pi Official Annual 2027

Dive into the world of Raspberry Pi with this huge book of tutorials, project showcases, guides, product reviews, and much more from the pages of *Raspberry Pi Official Magazine*.

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

Buy online
rpimag.co/annual27

NixieGraph

By **Björn Lindrooth**

rpimag.co/nixiegraph

We have a love/hate relationship with Nixie tubes. On the one hand, they look incredible. Nothing else has the same analogue, high-voltage feel. They look as though they should make a thunking noise every time a digit changes. They're right on the cusp of electronics and magic. The downside of Nixies is that they're old; there's a Czech company, Dalibor Farny, making new Nixies in 2026, but other than that you're looking at new old stock: tubes made in the 1970s, 1960s, or even the 1950s that have been sitting in a warehouse untouched until they're discovered and placed on the market. This makes them fragile and expensive compared to other display methods; then there's the fact that they operate at around 180V, so if you want to use Raspberry Pi, Pico, or any other microcontroller to control them, you're also going to need the appropriate power circuits, and the know-how to operate them safely. The other downside, compared with modern screens, is that each Nixie tube only displays one digit of information.

Björn Lindrooth has taken the Nixie and dragged it kicking and screaming into modernity with this Nixie-inspired LCD clock. Each digit is actually an LCD screen, each one in a custom glass tube, behind a custom-made grille, to enhance the Nixie look and feel. Other than that, the clock has one toggle switch and one rotary switch to control the menus; there's also a 3.5mm audio jack used to connect a Morse key – to transmit Morse code.

The screens are controlled by an ESP32-S3 and the displays are 1.9-inch ST7789 IPS LCDs. Yes, we know that's not Raspberry Pi, but Björn is working on a setup that should enable anyone to slot their own board into this device. And as the tubes are more than just digits, they can display any kind of data: PC performance stats, Telegram chats, the aforementioned Morse code input, weather forecasts, anything.



▼ Each tube has an RGB LED beneath it, to further emulate the Nixie glow


0930
SUN


5
DECODING
MORSE
HELLO
MORSE
Game 1
MORSE READ

WallCalToDo

By **Kevin Clayland**

rpimag.co/wallcaltodo

We'll let the maker of this calendar and to-do app display unit, Kevin Clayland, tell you all about it: "This was a way to organise my and my family's life. I am interviewing for positions as a product designer and thought it would be a great project to make something for myself as well as keep designing while I am in my job hunt. … The purpose, for me, was to sync with the native apps on my phone without creating a new calendar and to-do app that handles entries. All I had to do was default to Google Calendar in my [iOS] Calendar app and link my Microsoft account in the Reminders app. No new apps, easy syncing. There is a companion app where you can set a ton of settings and connect accounts (adding accounts only works on the physical Raspberry Pi, but everything else can be configured from a phone connection)."

"I went to Michaels Arts and Crafts to make a custom frame, but… the overhang on their frames was not large enough to hide the sides of the screen. I decided to go to Home Depot and make my own. I bought oak boards that had the correct depth to case-in the screen, [and] cut the edges to 45 degrees with my power saw (not exactly perfect angles, but close enough!). I then inlaid L-shaped trim pieces with wood glue and it held the screen. One side of the screen was thicker where the ribbon band was set, so I added padding on the other sides to make up for it so the screen had the same padding width on each side. I then added brackets on the inside to keep it sturdy. In the areas where the 45-degree angles weren't perfect when put up against each other, I mixed wood glue with sawdust to fill it in. Then I stained the wood to get the beautiful dark walnut colour it has. I laid in the screen and then added spare parts I had lying around to keep the screen in place and lock it into the frame. Unfortunately, the monitor didn't have a USB port to power Raspberry Pi, so [they both] had to be plugged into an extension cord that then comes out of the frame back."

▶ Some real woodworking skill went into this custom frame


11:15 pm
September 2026
SUN
MON
TUE
WED
THU
FRI
SAT
Wednesday, September 16th
To Do
Dentist appts for all
Repairs
Clean carpet
Water plants
Pay PSEG
69°

Boombox cyberdeck

By **High Tech \\ Low Life**

rpimag.co/boombox-cyberdeck



Electrical hardware from the 1980s and 1990s is absolutely brilliant for playing with. The enclosures are usually put together with screws rather than clips or glue, making them easy to disassemble and reassemble. And the natural march of miniaturisation means that if you replace old electronics with new, you're going to have a lot of extra space in the existing enclosure for more fun stuff.

Take this cyberdeck built into 1980s boombox television, for example. For starters, the old CRT screen has been replaced with a modern IPS display – this alone frees up a ton of space.

There's also a modern x86 computer with 16GB of RAM, a 256GB NVMe boot drive, and over 2TB of storage. The speakers have been updated to Faital Pro 3-inch 40W units. Every capacitor in the amplifier has been replaced, and there's stereo output from the computer. As for the original features, the original AM/FM radio works, as do the two tape decks, and there's a 3.5mm external input. Sharp eyes will have noticed that there's no keyboard attached; that's not a problem, since there's a separate Bluetooth module for connecting an external keyboard.



▲ Just the thing for watching *The Terminator*, an outlandish fiction in which tech goes rogue and kills us all

Warning!

CRT

Be careful with projects involving old television and CRT equipment. Opening up a CRT can be dangerous, risking electric shock even if the TV is not plugged in.

rpimag.co/crtsafety

Raspberry Pi Cinema Camera

By **menggu**

rpimag.co/cinema-camera

In writing this, we're trying very hard not to call the project 'CinePi'; that's the name of another, similar project which inspired this one. It's a cinema camera. We thought that any camera could be a cinema camera – we know that the zombie film *28 Years Later* was shot partly on iPhone. But when we dug a little deeper, we found that the filming crew weren't running around with phones in their hands. They used custom lenses and rigs, and put them into cages for protection and ease of handling.



This project uses Raspberry Pi hardware and is designed to be modular, with either a 3D-printed or aluminium body. It captures video in HD, FHD, 4K, and open gate (which captures everything going into the camera lens, rather than cropping it to a standard size).

There's an internal 720 × 720 pixel touch LCD preview monitor and an HDMI out for attaching the camera to external monitors; threaded mounts for professional camera add-ons; an internal battery; and a whole load of features that we don't understand but we know camera geeks will appreciate.

▼ It's a nice coincidence that Raspberry Pi boards are made in the same factory that produces cameras used for live sport broadcasts, costing upwards of £200,000



3D print

Achieve the Zen enlightenment that only comes with knowing you have 100% battery charge

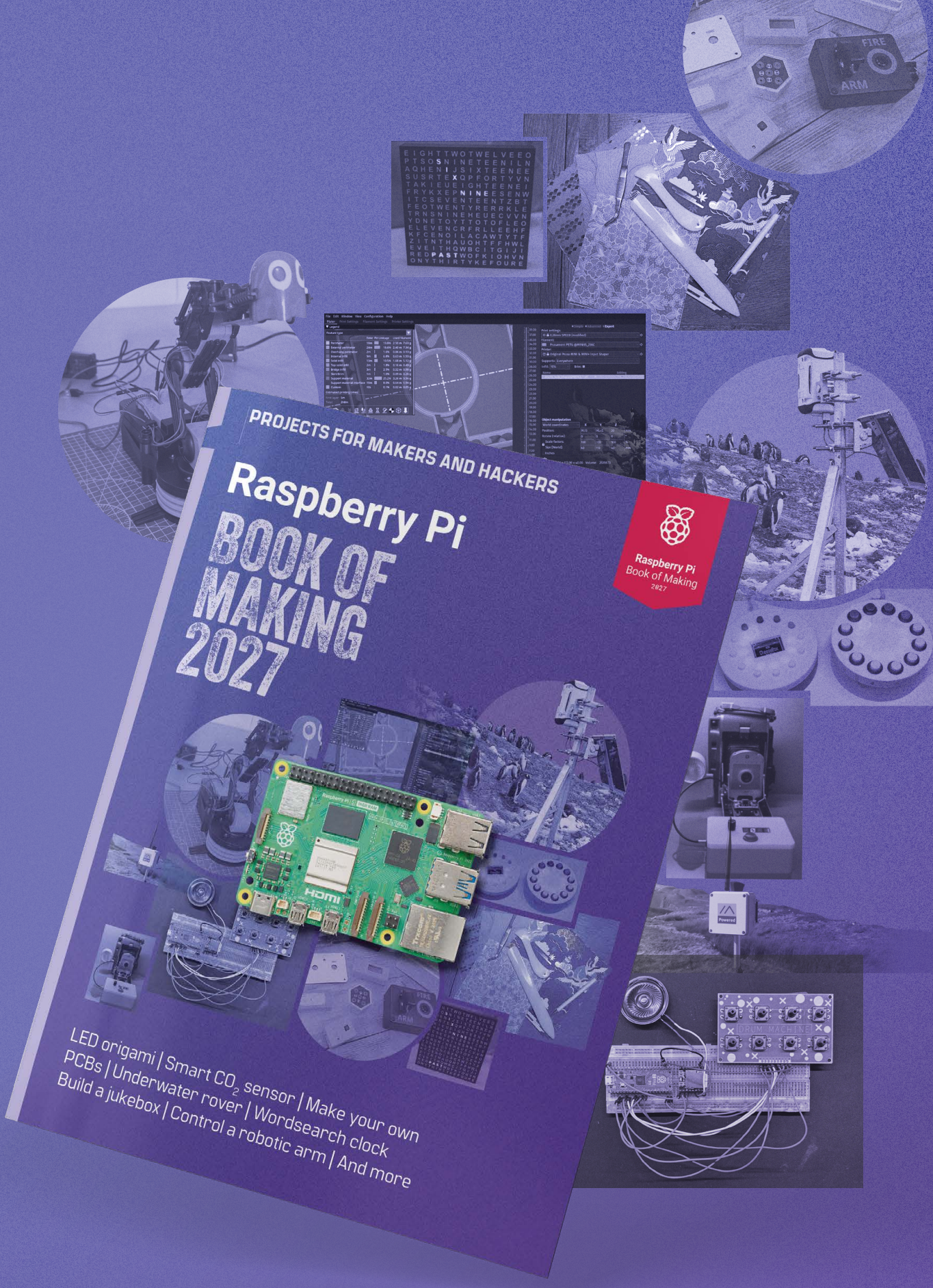
rpimag.co/bonsai-charger

Multifilament printing (also known as AMS, or automatic material system) has been a boon for FDM printing. Rather than printing in one colour, then painting a design on the surface of the printed object, anyone with the right kind of printer can now print objects that contain different colours. Imagine a 3D-printed pumpkin with black eyes, for example – with basic, one-colour printing you'd have to ensure in the design files that the black parts were separate shapes to the orange parts, with all the work involved in making sure that the parts matched and that you had the tolerances correct, followed by any sanding needed to ensure the parts fit. With a multifilament print, you just press print, and the slicing software decides what colour filament gets extruded when. It's like magic. It's also really slow and wasteful, as each colour change has to be purged from the extruder before the new filament can be deposited.

This bonsai garden, which houses a MagSafe charger for Apple iPhone, looks like it was printed using AMS technology, but its maker, Adrian Stollenwerk, decided instead to make it out of discrete parts – the garden, torii, lantern, stepping stones, and tree are all separate parts and fit together without glue. The design is modular, so if you splash out on an Apple Watch or a set of AirPods, or you need an additional charger for a second phone, you can print the parts to house their respective chargers, and there's a version that includes a night light too, which lights up the base. The whole thing took roughly 27 hours of printing on a Bambu Lab P2S.






PROJECTS FOR MAKERS AND HACKERS
Raspberry Pi
BOOK OF MAKING 2027
Raspberry Pi
Book of Making
2027
LED origami | Smart CO_2 sensor | Make your own PCBs | Underwater rover | Wordsearch clock
Build a jukebox | Control a robotic arm | And more

Raspberry Pi

BOOK OF MAKING 2027

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

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

Buy online
rpimag.co/bookofmaking2027


WEDNESDAY
AUG 12
10:44
AM
Drizzle
THU
FRI
SAT
SUN
Weather
HAT
24

Incredible weather projects

Come rain or shine, your Raspberry Pi can help you make your own weather predictions.

By **David Crookes**

Weather forecasters and climate scientists have been warning the world about a strengthening El Niño along with changing ocean temperatures and wind patterns in the central-eastern equatorial Pacific. This could lead to wetter weather and stormier conditions in the UK during autumn and winter, providing a perfect opportunity to engage in citizen science and better understand what is happening.

You're already well equipped to get started. A Raspberry Pi computer can be at the heart of a host of weather-related projects, allowing you to monitor rain, wind, and heat and even geological events such as earthquakes. In later tutorials, we'll explore attaching sensors to Raspberry Pi microcontroller boards, and how to build your own weather satellite receiving station. But first, let's take a look through plenty of incredible projects to watch our wild weather.



Warning!

Batteries and working outdoors

Some projects mentioned in this feature use lithium batteries for remote functionality. Be careful when charging and handling lithium batteries. When working with Raspberry Pi outdoors, make sure you use effective cases to contain electrical equipment. Projects mentioned in this feature involve working at height in the great outdoors, and potentially handling electrical equipment during storms. Be mindful of the risks involved.

rpimag.co/power-bank-safety

rpimag.co/outdoor-safety

Measure the weather

Create your own weather station and start collecting data with these builds

Air Quality Weather Station

rpimag.co/air-quality-station

As well as collecting local weather data using an anemometer to measure wind speed and wind pressure, Kutluhan Akar's wallet-friendly weather station has a BMP180 sensor to measure barometric pressure changes, temperature, and altitude. Not only that, it tracks ozone concentration via an I2C ozone sensor and, by combining its readings with online data in a TinyML neural network model, it's capable of delivering an AI-driven air quality forecast that the maker says is 92% accurate. What's more, the device has been designed to endure harsh weather conditions, which makes it suitable for monitoring a storm. As a finishing touch, there's a little 3D-printed windmill, which looks pretty cool.

The device has been designed to endure harsh weather conditions



▶ Sensors gather data and screens mounted on the unit are used to display it

Off-Grid Weather Station

rpimag.co/off-grid-weather

Vinnie Moscaritolo's weather station uses a compact all-in-one Ecowitt WS90 outdoor sensor mounted on a PVC pipe to monitor temperature, humidity, wind speed and direction, solar radiation, UV, and rainfall. He likes it because there are no moving cups or spoons to worry about and nothing to clog; but, more than that, he loves the fact his setup doesn't rely on Wi-Fi and cloud services. Instead, it transmits raw RF packets of data that can be picked up by a Raspberry Pi computer connected to an inexpensive RTL-SDR, a software-defined radio device. The 915MHz radio transmissions can then be decoded into clean JSON packets using the rtl_433 project (UK and EU readers should use the 868MHz variant). Once the information has been made accessible through a local JSON API, it can be displayed on a dashboard.



▲ The Ecowitt WS90 all-in-one weather station tracks seven different weather metrics

Laundry-saving Rain Detector

rpimag.co/rain-detector

Simple to build and capable of alerting users to the first drops of rain, this precipitation detector is assembled inside an inexpensive airtight box. Developed by PJ Evans, it includes a Raspberry Pi Zero W, a couple of rain sensor boards with a controller, a small breadboard, a small USB power bank, and a handful of jumper cables. When raindrops land on the sensor, it shorts the connection and a small current is converted into a digital signal that's detected by Raspberry Pi. It then uses the Pushover (**pushover.net**) notification system to display an alert on your phone.

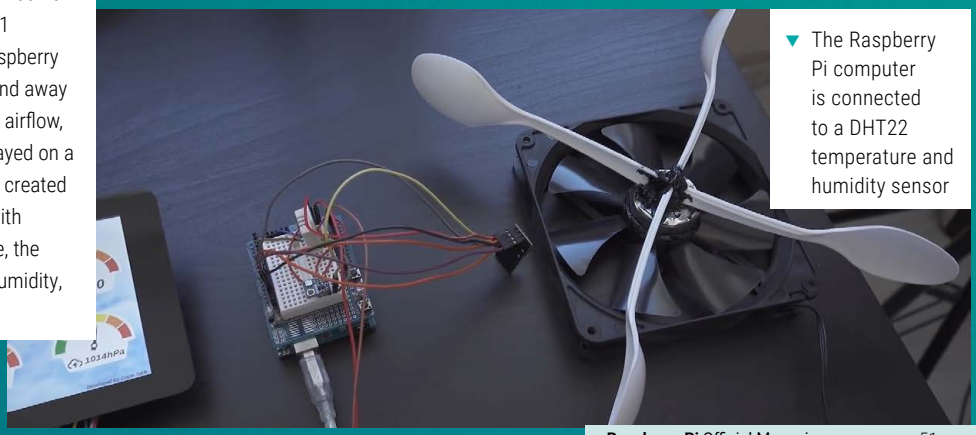


◀ Make sure your container is sufficiently sealed, including around the wire hole, to prevent water seeping in

Plastic Spoon Weather Station

rpimag.co/spoon-weather

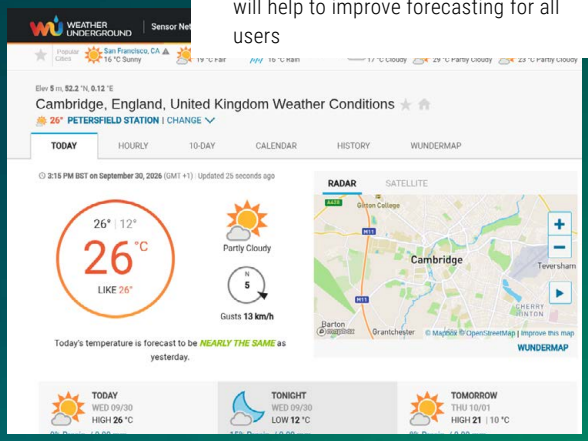
Çağan Çelik has put an old computer case fan to great use. By attaching four plastic spoons, he's made a very effective anemometer with an Arduino Uno collecting its readings and an nRF24L01 transceiver providing a wireless link to a Raspberry Pi 4 computer. Placed as high as possible and away from surrounding features to ensure a clear airflow, the anemometer provides data that is displayed on a 7-inch touchscreen running a user interface created using the GUI framework Tkinter included with Python's standard library. For good measure, the weather station also senses temperature, humidity, and air pressure.



▼ The Raspberry Pi computer is connected to a DHT22 temperature and humidity sensor

Display your forecast

Discover when you need to take your umbrella with you using these forecasting projects



▼ Sharing data to Weather Underground will help to improve forecasting for all users

WeatherMan

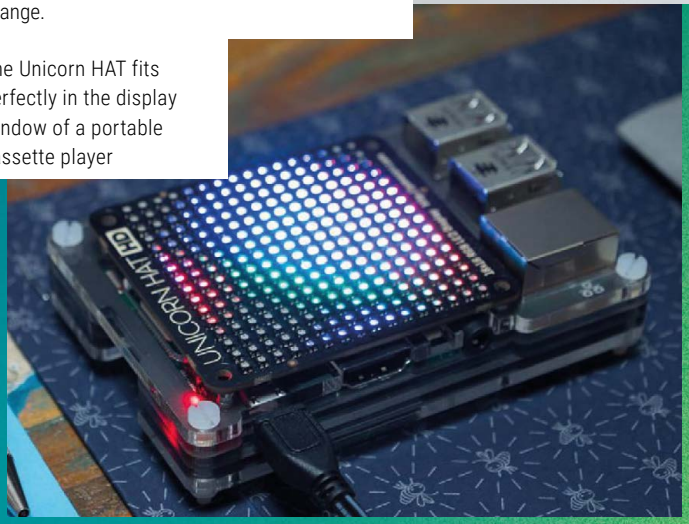
rpimag.co/weather-man-pi

Rather than have a fixed weather station, Martin Mander's project takes rain-watching on the road by repurposing an old portable cassette player. Rather than measuring the conditions itself, WeatherMan retrieves data from the Dark Sky Weather API before turning it into a colourful display using a Pimoroni Unicorn HAT. Powered directly from a Raspberry Pi Zero W computer, the HAT features 64 individually controlled RGB LEDs. They're used to animate the current weather conditions, scroll the temperature across the display, and show a graph indicating the likelihood of rain. A servo is also used to jiggle the headphones whenever the weather is about to change.

▼ This project repurposes an old Hitachi portable cassette player, but any model can be used



▼ The Unicorn HAT fits perfectly in the display window of a portable cassette player

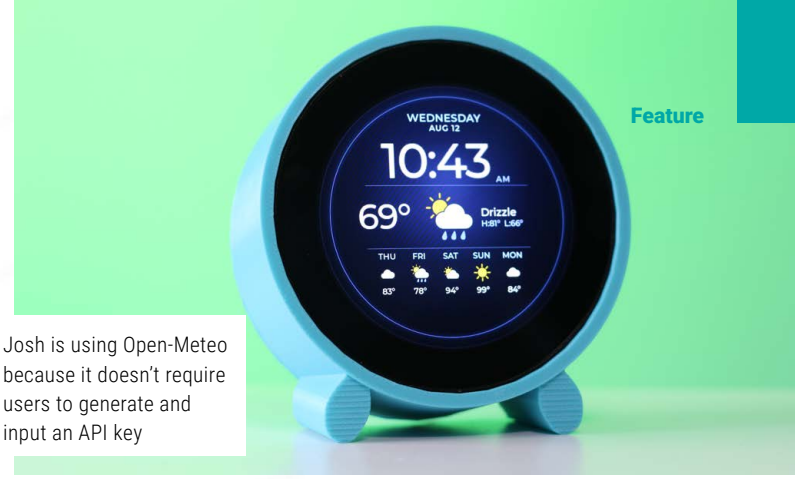


Connect with others

If you create a personal weather station, you may want to consider sharing the data you collect. Weather Underground (**wunderground.com**) is a community of like-minded souls operating more than 250,000 weather stations across the globe. By sharing information from environmental sensors, you will be contributing to a huge databank that is used for more accurate, hyperlocal forecasting.

The website suggests the best ways to install a personal weather station because where it is sited is crucial. A weather station needs to be out in the open with no obstructions, around 10 metres above the ground if you're monitoring wind speed and direction, and at least 5 metres above whatever surface it's mounted on if you're only monitoring the temperature. It should also be at least 15 metres away from the nearest tree or body of water.

Once you're satisfied your weather station is providing accurate data, you can register your device and obtain a station ID. You can find out more about uploading weather data to Weather Underground by going to **rpimag.co/weather-upload**.



▶ Josh is using Open-Meteo because it doesn't require users to generate and input an API key

Clock Weather Display

rpimag.co/round-weather-display

This stylish round smart clock by Josh Hendrickson provides the perfect way for weather-watchers to wake. It pulls real-time weather data from Open-Meteo and it shows the forecast for the current day and the week ahead, giving you the heads-up should any adverse conditions be heading your way. Powered by a Raspberry Pi Zero 2 W running Raspberry Pi OS 64-bit Trixie, it displays data on a Waveshare round HDMI touchscreen surrounded by a 3D-printed case. You could adapt it to highlight changing conditions and unsettled weather by displaying key storm indicators such as wind speed, gusts, direction, precipitation, and atmospheric pressure.

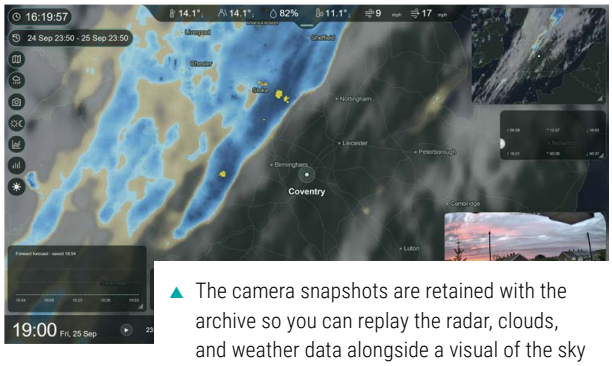
Pi Rain Radar

rpimag.co/rain-radar

Alex Soul's Pi Rain Radar is designed to sit by the door, allowing users to quickly view the movement and intensity of approaching rain and weather systems. Always on, it runs as a Docker application on a Raspberry Pi 4 computer and displays animated radar imagery from RainViewer, with Rainbow Weather available as an alternative provider. With no interaction needed for everyday use, it can replay up to six hours of radar while displaying temperature, wind speed, gusts, wind direction, and pressure. It can also capture camera snapshots alongside its archive. Users can interpret what is happening outside and make their own immediate forecast.

▶ Pi Rain Radar tracks showers and rain bands and lets the users determine the likelihood of needing an umbrella





▲ The camera snapshots are retained with the archive so you can replay the radar, clouds, and weather data alongside a visual of the sky

StormShell

rpimag.co/stormshell

Short on time but still want to get stuck into a weather project? StormShell is a quick project that uses ASCII art to display data sourced from Open-Meteo. Running as a full-screen kiosk on Raspberry Pi, it shows the current temperature, humidity, and wind conditions alongside hand-drawn ASCII weather animations for everything from sunshine and clouds to snow and fog. It also displays air quality, atmospheric pressure, and the current moon phase, as well as an analogue clock. It makes for a surprisingly comprehensive dashboard that, while distinctly old-school in appearance, is very easy to comprehend.

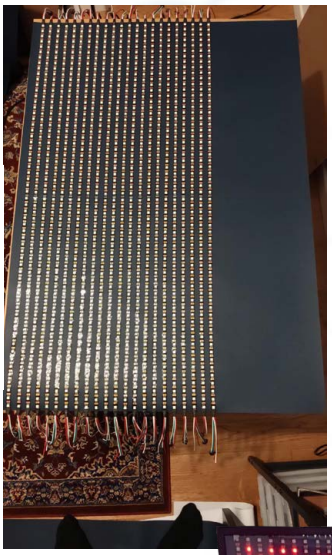


▼ StormShell works with city names and postcodes across the world

Creative weather projects

See the weather differently with these stunning ways to present data

Take two LED strips and have them change colour depending on the weather



▶ The board is made up of LED strips spaced a centimetre apart

LEDWeatherBoard

rpimag.co/ledweatherboard

Who would want to look out of the window to check the weather with this eye-catching, LED-festooned beast on the wall? Made up of 1920 RGB LEDs split across 32 LED strips, the huge metre-high board by Diego Braga displays a 24-hour forecast, split into four parts, showing the minimum and maximum temperature, the time, and a weather icon that best represents the predicted condition (from sun to cloud to rain). The weather is pulled from an API every ten minutes and it's processed via a Raspberry Pi computer. The information refreshes with the aid of an eye-catching animation.



◀ The maker was aiming for an 8-bit style – simple but effective

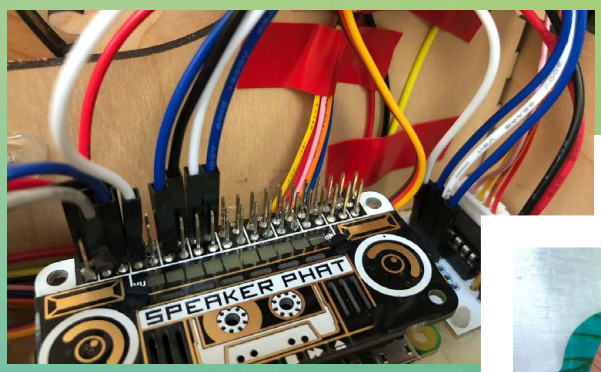
Cloudy-A

rpimag.co/cloudy-a

It's a simple concept but very effective: take two LED strips and have them change colour depending on the weather. By inserting the strips inside a recycled five-litre plastic bottle covered with cotton wool and drawing on local data from Weather Underground, the project produces an amazing, animated cloud – an eye-catching way to determine if it's going to be sunny, snowy, rainy, cloudy, or stormy. It's a great project to occupy young minds and while this particular version was based on a Raspberry Pi 3 computer, it would work just as well with a Raspberry Pi Zero.

▶ A script for the light animations allows the cloud to burst into life





A chime is used to indicate low and high tides via a Speaker pHAT connected to a Raspberry Pi Zero



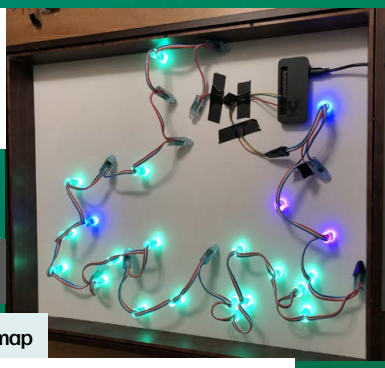
A Trinket M0 microcontroller controls the NeoPixel LEDs

Tide Clock Weather Thing

rpimag.co/weather-tide-thing

Although a lot of weather projects visualise data on a screen, Fin Hopkins's marvellous creation uses a series of physical wheels, pointers, and LEDs. Constructed using medium cherry and maple plywood, a large wheel shows the current weather using a series of familiar icons while a finger in the centre points to the current temperature. There are LEDs indicating the likelihood of rain and a set of 24 lights giving an hourly forecast (blue for rain, yellow for sunny, white for cloudy, and green for windy), with the data sourced from the Dark Sky API. Apple has since shut down Dark Sky, so a good drop-in replacement would be Pirate Weather (**pirateweather.net**). In addition, a micro-servo and gears show when the twice-daily low tides are due to occur using data from the NOAA's Tides and Currents website. It's a true work of art.

Philip pulls his data from **aviationweather.gov**, but you can find alternative sources



METAR Map

rpimag.co/metar-map

Pilots use coded weather reports called METARs (METeorological Aerodrome Reports) to check the wind, cloud, temperature, pressure, and visibility around aerodromes. With Philip Rueker's project, it's possible to grab such data and visualise the weather across different locations using a set of colourful LEDs. They'll turn green if the weather conditions are favourable and red if they are not. The lights will blink to indicate high winds and turn white if there's a lightning storm in the area. Built around a Raspberry Pi Zero W at its heart, the maker says it could be adapted to map the weather across villages, towns, and cities; so long as you find a suitable online source.

You can easily adapt this project for your local area



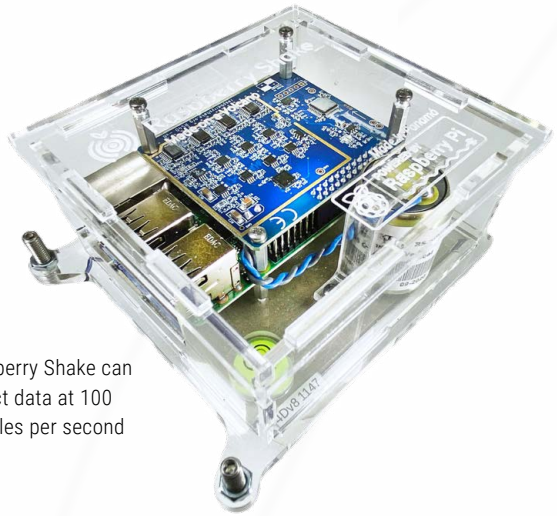
Explore our planet

Boom, boom, shake the world with these environmental experiments

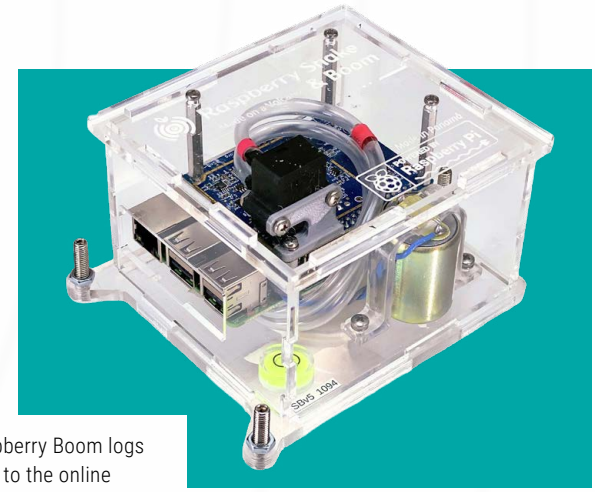
Raspberry Shake

rpimag.co/shake

Raspberry Shake is a relatively low-cost seismograph that plugs into a Raspberry Pi computer. Its professional-grade seismic sensors measure Earth's motion and they are sensitive enough to detect small local tremors and large events across the world. In fact, Raspberry Shake is capable of picking up the ground motion generated by a magnitude-6 earthquake on the other side of the globe. For that reason, it's become incredibly popular and there's a smartphone app connecting to Raspberry Shake seismographs around the world. Simply connect Raspberry Shake to Raspberry Pi, wire up a geophone, install the Raspshake SD image, and start sharing your data.



▶ Raspberry Shake can collect data at 100 samples per second



▶ Raspberry Boom logs data to the online Station View app: **rpimag.co/shakestations**

Raspberry Boom

rpimag.co/shake-boom

Created by the brains behind Raspberry Shake, Raspberry Boom detects infrasound: acoustic waves with frequencies below 20Hz, which is below the lower limit of human hearing. It measures what is happening in the atmosphere around and above the Shake device. This makes it possible to detect and monitor the atmospheric signatures associated with tornadoes, severe storms, and lightning while also picking up on infrasound generated by events such as volcanic eruptions, avalanches, and landslides. It can even measure meteors colliding with the atmosphere.

Sixfab

Complete Edge AI Hardware Stack for Raspberry Pi

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

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

START HERE



SIXFAB AI HAT+ FOR RASPBERRY PI 5

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

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

EXPAND



SIXFAB EDGE AI EXPANSION BOARD

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

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

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

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

DEPLOY



SIXFAB ALPON X5 AI

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

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

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

SIXFAB × ULTRALYTICS ACCELERATION PATH

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

Raspberry Pi Design Partner

Intelligented by DEEPX

ultralytics | Sixfab

Scan to browse and pre-order

Contact: hello@sixfab.com

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

Experiment with weather sensors

Communicate with a wide range of external hardware



Maker

Ben Everard

Ben is slowly becoming a cliché as he moves away from fast-paced, ever-changing world of tech to the gentle embrace of woodworking.

Youngwoodworkers.co.uk

YOU'LL NEED

- Solderless breadboard **rpimag.co/breadboard**
- BME280 sensor **rpimag.co/bme280**
- 10 × Pin-to-pin jumper wires **rpimag.co/p2p-jumpers**
- PMS5003 air-quality sensor **rpimag.co/pms5003**
- Soldering iron and solder (to attach BME280 pins) **rpimag.co/soldering-iron**

In our book, *Python Projects for Raspberry Pi* (rpimag.co/python-projects), we explore how to connect LEDs and buttons. These are basic components that your Raspberry Pi computer or Pico can control by turning a GPIO pin on or off, or read from by interrogating the current state of a GPIO pin. These kinds of components are relatively straightforward. However, many electronics modules and sensors are more complex.

Python or MicroPython

We're going to make a weather station. At its heart, it will be a collection of sensors hooked up to a controller. We'll use it to read temperature, pressure, humidity, and air quality.

Throughout our *Python Projects* book, we use MicroPython (on Raspberry Pi Pico) and Python (on Raspberry Pi computers). While MicroPython has support for a lot of devices, the best set of modules for using hardware with Python is the CircuitPython Library Bundle. Even though we're not using CircuitPython, we can still use its libraries. The underlying Python language is the same; they just have a slightly different way of interacting with the hardware. To bridge the gap between CircuitPython libraries and MicroPython or standard Python, we use a compatibility layer called Blinka. Blinka allows CircuitPython modules to run on both the Raspberry Pi and the Pico, making hardware integration easier.

In the book, we connect the same hardware to both Raspberry Pi and Pico to demonstrate the differences between these two platforms. In this guide, however, we're just using Pico.

Pico: Weather station

For building a Pico-based weather station, we'd recommend Raspberry Pi Pico 2 W for two reasons: the extra flash storage on Pico 2 W compared to Pico W means we can save more data; and the on-board Wi-Fi means that we can obtain the current time from the internet.

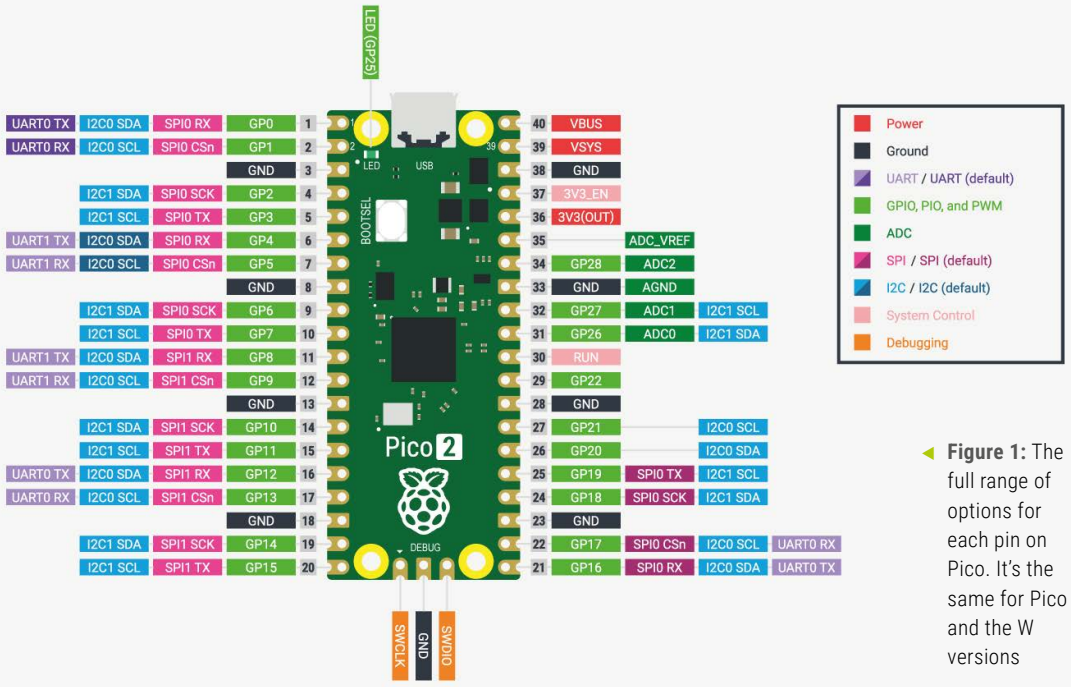


Figure 1: The full range of options for each pin on Pico. It's the same for Pico 2 and the W versions



Warning!

Hot solder

Soldering irons get very hot, and stay hot for a long time after they're unplugged. Make sure that you put the iron in the stand when you're not using it and don't touch the metal parts – even after it's unplugged.

rpimag.co/soldering

We're going to need some sensors to collect weather data. The first we'll use is the BME280. The sensor itself is a little tricky to work with because of its size and form factor, so we'll work with a module that breaks out the connections and adds a little supporting hardware. There are lots of different versions of these modules available. We'll be using one from Adafruit, but others should work provided they break out the necessary pins.

At its heart, it will be a collection of sensors hooked up to a controller. We'll use it to read temperature, pressure, humidity, and air quality

The BME280 can communicate over both SPI and I2C, so we'll look at both. The pins on the BME280 module are labelled a bit unusually because they can be used for either SPI or I2C:

- Serial Data In (SDI) is either the MOSI pin in SPI or the data pin in I2C
- Serial Data Out (SDO) is the MISO pin in SPI and unused in I2C
- SPI Clock (SCK) is the clock pin in both SPI and I2C

On Pico, only certain pins can be used for certain functions. Look at **Figure 1** for the full range of options. You'll see that there are I2C0, I2C1, SPI0, and SPI1. This is because there are two peripherals for each protocol, so you can have two I2C buses and two SPI buses.

Python Projects for Raspberry Pi

This tutorial is taken from Ben Everard's book, *Python Projects for Raspberry Pi*. In the book, you'll find an expanded description of the parts featured here, and information on how to hook them up. You'll also discover how to use this project with a Raspberry Pi board as well as Pico. It's a great book and we suggest you pick up a copy.



rpimag.co/python-projects

Make it permanent

All our wiring diagrams show the components on breadboards. This is so you can try different configurations of components. In this guide, we look at connecting the BME280 with I2C and SPI on Raspberry Pi Pico.

However, if you want to make a weather station for long-term use, a breadboard isn't a great option. On Pico, you'd want to solder wires directly between the BME280 and Pico's GPIO pins. (On Raspberry Pi, you can use something like a protoboard HAT that slots onto the GPIO pins and has some prototyping space to solder up your circuit.)

When configuring a peripheral, you can assign any compatible pin that supports the desired function, but all pins used for a bus must belong to the same numbered peripheral (e.g., all pins on I2C0, or all on SPI1).

The labels on the Pico pinout diagram don't follow the usual naming conventions for SPI. There are Tx (transmit) and Rx (receive) labels rather than MOSI and MISO. This is because Pico can be either a master or slave device. In the usual case, where Pico is the master, the Tx pin is MOSI and Rx is MISO.

Pico: Install Blinka software

Now let's set up the software. We'll install two Adafruit modules, Blinka and PlatformDetect. You can download these from the following links:

- **rpimag.co/blinka**
- **rpimag.co/ada-ppd**

Download the Blinka source zip file from the release page, then unzip it somewhere on your computer. This is the full set of code for every board supported by the module. Most of this isn't needed, so we'll get rid of the bits for other platforms. The important bits are in the **Adafruit_Blinka_<version>/src** folder. Inside this, there are two folders that we're going to clean up:

- **adafruit_blinka/microcontroller** (not to be confused with the microcontroller folder in the top level of the **src** directory itself).
- **adafruit_blinka/board**

In these two folders, you can delete everything except:

- **adafruit_blinka/microcontroller/generic_micropython** folder and the **adafruit_blinka/microcontroller/rp2040** folder
- **adafruit_blinka/board/raspberrypi/pico.py**

Once you've removed those, you can upload the rest to Pico. Open Thonny and make sure you're in the Regular UI mode. If you see the text 'Switch to regular mode' in the upper right of the Thonny window, click it, then quit and restart Thonny to enable the new mode.

If you haven't done so already, install the latest version of MicroPython on your Pico 2 W (**rpimag.co/micropython**).

Click the View menu in Thonny and make sure that Files is checked. This will open a section with two file panes: one showing the files on your computer and one showing those on Pico.

On Pico, we want to make a **/lib** folder that will store our modules. This isn't strictly necessary but does help keep things a bit tidier in cases like this where we have a lot of modules. Right-click in the Pico files pane, then select New directory and name it **lib** (it has to be exactly **lib** or MicroPython won't search it for modules). Once you've created **lib**, double-click it to change into this directory.

Now, on the computer files pane, navigate to where you unzipped Blinka and then into the **src** subdirectory. Select everything in there (you can click on the top item, then hold down **SHIFT** and click on the bottom item), then right-click to open the context menu and select 'Upload to /lib'.

That's Blinka set up; we now need PlatformDetect. Fortunately, this is a bit simpler. Just unzip the directory; then, in the Thonny pane, navigate to the **Adafruit_Python_PlatformDetect-<version>** folder and inside that, select **adafruit_platformdetect**, then right-click and select 'Upload to /lib'.

In the REPL (Shell pane), you can check everything's worked by typing:

```
>>> import board
```

If that works, then you've installed Blinka correctly. If you get any errors, you can go back and try and fix them. The full guide for installing Blinka on MicroPython can be found at **rpimag.co/cplibsonmppico**.

We have one final piece of setup. Although we've got Blinka up and running, we still need our BME280 library. This is part of the CircuitPython Library Bundle – a single download where you'll find modules for almost all common hardware. Head to **circuitpython.org/libraries** and download the latest Python Source Bundle (the bundles above it, labelled as 'Bundle for version Y.x', won't work as they are compiled for CircuitPython, not MicroPython). Unzip this and use Thonny to copy the **adafruit_bme280** folder to the **/lib** folder on Pico. Do the same with the **adafruit_bus_device** folder.

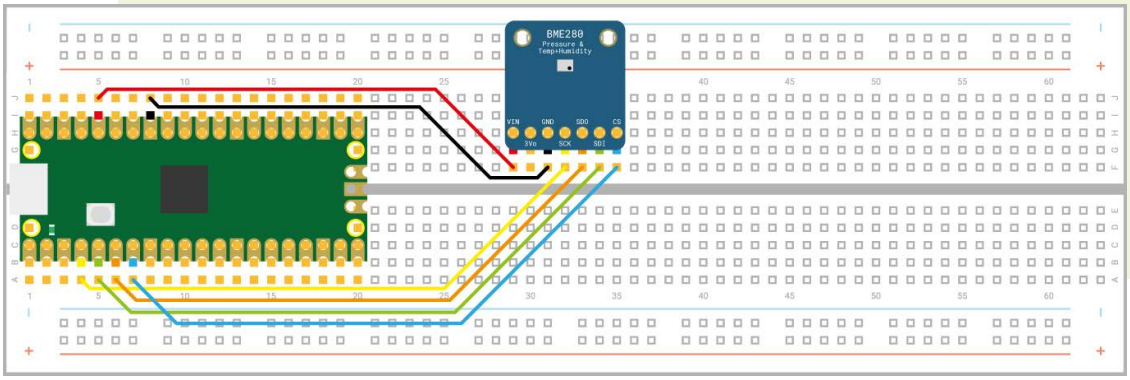


Figure 2: Connecting the BME280 to Raspberry Pi Pico over SPI

Pico: Read BME280 values (SPI)

We'll wire our BME280 up to SPI0 using GPIOs 2–5. The full wiring diagram is given in **Figure 2**. Although you need more pins for SPI, you have separate lines for each direction so you can send and receive data faster.

The table below shows all the connections that you will need to make.

We're now ready to read values from the sensor. You can find this code in the GitHub repository (**rpimag.co/pythonpi-resources**) in the **ch03** subdirectory as **ch03-bme280-spi.py**.

As you can see, we just need to set up an `SPI` object and then use this to create a `bme280` object which has values for temperature, humidity, and pressure in Celsius, relative humidity (RH) percentages, and hectopascals (hPa) respectively.

Pico	BME280	Notes
3V3	Vin	Don't use the BME280's 3Vo pin. It's a 3V output you can use if you've connected Vin to a 5V power source.
GND	GND	
GP2	SCK	SPI Clock
GP3	SDI	MOSI
GP4	SDO	MISO
GP5	CS	

ch03-bme280-spi.py

> Language: **MicroPython**

DOWNLOAD THE FULL CODE: rpimag.co/pythonpi-resources

```
001. import board
002. import busio
003. import time
004. import digitalio
005.
006. from adafruit_bme280 import basic as
     adafruit_bme280
007.
008. spi = busio.SPI(board.GP2, MISO=board.GP4,
     MOSI=board.GP3)
009. cs = digitalio.DigitalInOut(board.GP5)
010. bme280 =
     adafruit_bme280.Adafruit_BME280_SPI(spi, cs)
011.
012. while True:
013.     print("Temperature: %0.1f C" %
     bme280.temperature)
014.     print("Humidity: %0.1f %%" %
     bme280.humidity)
015.     print("Pressure: %0.1f hPa" %
     bme280.pressure)
016.     time.sleep(1)
```

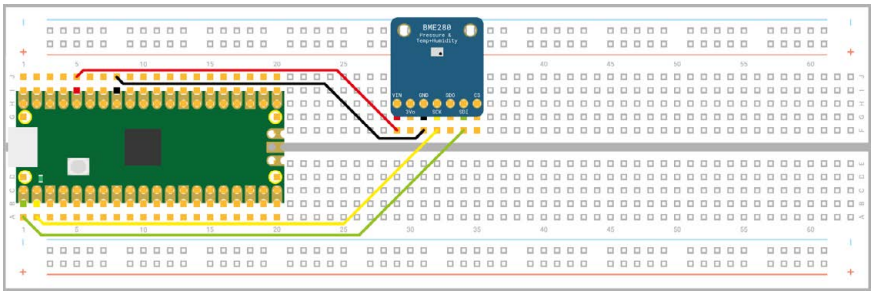


Figure 3: Connecting the BME280 to Raspberry Pi Pico with I2C

> *The same data must be transferred whatever protocol we use*

Pico: Read BME280 values (I2C)

Let's now take a look at wiring up the BME280 as an I2C sensor (**Figure 3**). With fewer wires, I2C wiring can be a bit tidier than SPI. Leave the power and ground connections as they are, remove the Chip Select (CS) connection and SDO (MISO) connections, and change the following connections as shown:

Pico	BME280	Notes
GP0	SDI	SDI may be labelled as SDA on some boards
GP1	SCK	May be labelled as SCK on some boards

You can find the code to read the sensor over I2C in the **ch03-bme280-i2c.py** file.

The code, as you can see, is almost identical. The only difference is that we create an `I2C` object rather than an `SPI` object. The reason that it's so similar is that there's not much difference between them. SPI and I2C refer to the way data is transferred, and this is handled by the `busio` module.

When we use the sensor, the same data must be transferred whatever protocol we use, and since `busio` abstracts away the underlying protocol, we can access the sensor in the same way regardless of whether we use I2C or SPI.

While there can be more subtle reasons to choose one protocol over the other, the big advantage of SPI is speed and the big advantage of I2C is the simplicity of the wiring.

Since speed is almost irrelevant to our weather station, we might as well keep the wiring simple and use I2C. This will be even more of an advantage if you want to add extra sensors, since that would just mean connecting more up to the same I2C bus.

Scanning I2C

You can scan an I2C bus to find out what (if any) devices are connected to it. This can be a very useful step in ensuring that your circuit is wired up correctly and everything is working.

You can use standard MicroPython classes for scanning. In this example (**i2c_scan.py**) we're using the built-in machine.I2C class rather than Adafruit's busio.I2C class:

ch03-bme280-i2c.py

DOWNLOAD THE FULL CODE: rpimag.co/pythonpi-resources

> Language: **MicroPython**

```
001. import board
002. import busio
003. import time
004.
005. from adafruit_bme280 import basic as
     adafruit_bme280
006.
007. i2c = busio.I2C(board.GP1, board.GP0)
008. bme280 =
     adafruit_bme280.Adafruit_BME280_I2C(i2c)
009.
010. while True:
011.     print("\nTemperature: %0.1f C" %
     bme280.temperature)
012.     print("Humidity: %0.1f %%" %
     bme280.humidity)
013.     print("Pressure: %0.1f hPa" %
     bme280.pressure)
014.     time.sleep(1)
```

Pin Definition



Connector Definition

PIN1	VCC	Positive power 5V
PIN2	GND	Negative power
PIN3	SET	Set pin/TTL level@3.3V, high level or suspending is normal working status, while low level is sleeping mode.
PIN4	RX	Serial port receiving pin/TTL level@3.3V
PIN5	TX	Serial port sending pin/TTL level@3.3V
PIN6	RESET	Module reset signal/TTL level@3.3V, low reset.
PIN7/8	NC	

◀ **Figure 4:** Your first port of call with new hardware should be the manufacturer's datasheet

Pico	PMS5003
VSYS	VCC
GND	GND
GP12	RX
GP13	TX

Pico: Air quality sensor (UART)

Now let's turn our attention to UART. To refresh your memory, UART – or Universal Asynchronous Receive and Transmit – is a protocol that (unlike SPI and I2C) runs just between two devices. There is one data line in each direction and no clock. To wire it up, you just take the transmit (Tx) connection from one device and connect it to the receive (Rx) connection on the other, and vice versa.

The specific UART device we're going to look at is the PMS5003 air quality sensor. This sensor from Plantower will detect and output the number of fine particulates in the air, categorised in PM1, PM2.5, and PM10 (particles under 1, 2.5, and 10 micrometres across, respectively).

There are modules that we can use to work with the PMS5003, but this time we're not going to use them because we're going to use it as an example for how to work directly with hardware.

When working with new hardware, the first thing you need is a datasheet. This is a document (usually a PDF) that tells you the key information for how to use the hardware.

You should be able to get the datasheet from either the part manufacturer's website or the reseller you bought the product from. For example, you can get the PMS5003 datasheet from Adafruit at **rpimag.co/pms5003datasheet**.

There are a few key bits that we need from the datasheet. The first is the physical pinout. There aren't any labels on the PMS5003, so without that pinout, we don't know what to connect to where.

Figure 4 shows a screenshot of the datasheet with the connections. This also tells us the voltage of the signals. We only need the power, ground, Rx, and Tx connections.

i2c_scan.py

DOWNLOAD THE FULL CODE:

rpimag.co/pythonpi-resources

> Language: **MicroPython**

```
001. import machine
002. i2c = machine.I2C(0, scl=machine.Pin(1),
     sda=machine.Pin(0))
003.
004. print('Scanning i2c bus...')
005. devices = i2c.scan()
006.
007. if len(devices) == 0:
008.     print("No i2c devices found!")
009. else:
010.     print(
     f"Found {len(devices)} i2c devices.")
011.
012.     for device in devices:
013.         print(f"Decimal addr: {device} | Hex
     addr: {hex(device)}")
```

Actually making the connections can be a bit tricky because the connector on the PMS5003 has too small a pitch to easily connect headers. The sensor should come with a cable, and we cut this and soldered jumper wires onto the connections we needed – as shown in the table above.

Now we've wired the sensor, we need to communicate with it. Again, we can go back to the datasheet.

From this, the first key thing we learn is that there are two modes: active and passive. In active mode, the sensor will keep sending readings; in passive mode, it will only send readings when the microcontroller requests them. We want to use passive mode because otherwise we have to keep reading the UART constantly to avoid buffers filling up and potentially getting stale data when we want it.

Appendix II of the datasheet gives the procedure for setting passive mode. We need to send 7 bytes. The first two are `0x42` and `0x4d` for any command. The command definition table then gives the next three as `0xe1`, `0x00`, and `0x00`. The final two bytes are a sum of all the previous bytes.

We can create all this with the following code:

```
mode_passive = b'\xe1\x00\x00'
start_of_frame = bytearray(b'\x42\x4d')
set_passive_command = bytearray()
set_passive_command.extend(start_of_frame)
set_passive_command.extend(mode_passive)
set_passive_command.extend(
    sum(set_passive_command).to_bytes(2, "big"))
```

The `big` parameter in `to_bytes` is to specify big-endian. At a hardware level, numbers can be stored either left to right or right to left. In this case, the most significant bit is sent first, and this is known as big-endian (as opposed to little-endian where the least significant bit is sent first).

It's set up as a module so you can call it from other programs

We can use the same tables to create the data string for the read command, but this time the command is `0xe2`, `0x00`, and `0x00`. This doesn't run the command – it just creates the byte string that we need to run the command.

That's how we need to format the data for sending. Let's now look at how to interpret the data we get back.

Again, there is a table in the datasheet. This states that there are two start bytes, then 15 chunks of data, each composed of two bytes. Python has the `struct.unpack` method for converting a buffer into a tuple using a particular formatting. The format string starts with a character defining the endian setting, and then a series of characters that each denote a data type. In our case, it's `">BBHHHHHHHHHHHHHHH"`. `B` is an unsigned char (that is, a single-byte positive number) and `H` is an unsigned short (a two-byte positive integer).

The MicroPython code to read the sensor resides in the file **ch03-pms5003.py**. It's set up as a module so you can call it from other programs. There are two functions, `available()` and `get_ticks()`, that make it possible for this module to run on both the Pico (under MicroPython) and a Raspberry Pi computer (under CPython). If we're running on MicroPython, it will call a MicroPython-specific function, otherwise it will call the CPython version:

- `time.ticks_ms` doesn't exist in CPython, so instead we use `time.monotonic` and this returns a result in fractional seconds rather than milliseconds
- The CPython serial port doesn't have the `any()` method, so we use the `in_waiting` value
- In CPython, `struct.unpack` requires the exact number of bytes, so we have to trim off any other data with `[:32]` (we ignore a response to the initial command to put it in passive mode). MicroPython doesn't require the exact number of bytes, but it doesn't hurt to include it.

This loops every five seconds. Each time, it sends the read command, then waits a short while for the sensor to respond. It then reads data in, one byte at a time, giving the sensor time to send each byte. Finally, it unpacks the data and returns the relevant data.

This is fairly easy to integrate with our code to read the BME280 sensor, and we'll look at combining everything shortly, but before we get to that, let's work out how to actually do something with the data we're getting.

Pico: Saving data

Now we've got our data, we need to do something with it, and the simplest option is to store it for later analysis. We're going to save it to Pico's internal storage as a Comma-Separated Values (CSV) file. This is a plain text file which contains a table of data with a new-line character after each row, and a comma between each value. You can view and analyse CSV files in most spreadsheet applications.

To make each reading meaningful, we also need to record when it was taken; for this, we're going to use the Pico's Real-Time Counter. This is known as the RTC, but RTC can also stand for real-time clock and the two are a little bit different. Pico's internal RTC can keep track of time while it's powered up and running. A real-time clock typically has a battery and keeps track of time even when powered off.

The Pico's RTC will help us timestamp our readings, but it needs to know the correct time first. There are a few ways to set it. We could:

- Set the time from a computer. For example, when we connect with Thonny, it sets Pico's RTC. We can also set it using `mpremote` with: `mpremote rtc set`
- Add an external real-time clock to hold the time
- Get the time from the internet

All of these have different advantages. The first – getting the time from a computer – works well for testing things out, but it's not a great solution for long-term use. We could add a real-time clock, but that adds hardware and expense. Most places have Wi-Fi, so the easiest and cheapest option is to use this to grab the time and set the RTC. We only have to do this when we first power up Pico, and the RTC will keep track of time from then on.

We can use a module called `simple_ntp` that you can download from **rpimag.co/simplentp**. Copy the **simple_ntp.py** file to the **/lib** folder on Pico. You'll need to use the module we created earlier to read the PMS5003, so you'll also need to upload **ch03_pms5003.py** to the **/lib** folder.

Near the start of the **ch03-pico-full.py** code (lines 15–32), there's some boilerplate Wi-Fi connection code that is just the standard way of connecting MicroPython to a wireless network.

When connecting to a wireless network, you generally need a password, and this is something that you want to keep secure. At the same time, it can be helpful to share your code. For example, you might choose to upload it to GitHub, or you might post it on a forum asking for help. One option is to simply delete any sensitive information like passwords when you do this, but there's a pretty good chance you'll make a mistake at some point. A more secure option is to keep any sensitive data in a separate file, not our main program. This way, you can share the main program without having to remember to remove any of the sensitive data.

We'll do this with a file called **secrets.py**. We can put anything in there that we don't want to make public, and that way we can share our main code however we like without having to remove passwords, API keys, or anything else that we don't want to announce to the public.

The **secrets.py** file contains a dictionary called `secrets` that looks like:

```
secrets = {
    'ssid' : '<your ssid>',
    'password' : '<your password>'
}
```

If there's a problem connecting to Wi-Fi, the main program will raise an exception which will stop it running. Once we're connected, we set the time:

```
simple_ntp.set_time()
```

Hexadecimal

When working with hardware – or really any time we want to work with bytes – it's common to give data in hexadecimal format. In this, there are 16 characters, 0–9 then A–F. Each character then contains four bits of information (since the largest 4-bit number is 15 (in decimal) or F in hexadecimal. One byte (eight binary bits) of information is then exactly two hexadecimal characters. In Python (and a lot of other languages), hexadecimal numbers are indicated by a leading 0X. The X isn't case-sensitive.

The other part of the program involves writing to the file. Our CSV file is a simple text file with one line for each reading. The only slight complication is that we want the first line to be the column headers.

Since we don't want to delete old data, we first check to see if the file exists by trying to open it. If it doesn't exist, MicroPython will raise an `OSError`. Normally, errors will stop your program running, but since we enclosed this in a `try:` block, we can catch the error, so the `except OSError:` block will run if an `OSError` occurs within the `try` block. In our case, the `except` block will run if the file doesn't exist, and so we create the file, write the first line, then close it.

Throughout this program we'll keep adding data to the same file, but each time we'll open it fresh and close it when we're finished. This is because our hardware is designed to be powered down at any point, so by closing it, we'll ensure that the data is written properly to storage and not held in a cache.

To open the file from Thonny, right-click its name (**data.csv**) in the list of files on your Pico. Click 'Configure .csv files' from the menu that appears. Choose Open in Thonny's text editor (Thonny doesn't support opening Pico files in the default system app). After that, you can double-click a CSV file to open it.

Strange things can happen when you fill up every last bit of space in storage, so before writing data, we're checking that there's at least 1kB of space left.

When we write our data to the file, we've set it up a little differently to before. Firstly, we've used a `with` block. The advantage of this is that as soon as you end the `with` block, the file is closed. Secondly, we've used the file mode `a` for append, so data is always written to the end of the file rather than overwriting data from the start.

Flash storage lifespan

No storage medium lasts forever, and the flash on Pico is no exception. Eventually, it will wear out. The flash on Pico is rated for at least 100,000 write cycles per sector and has 1000 sectors.

The file system underneath MicroPython will attempt to 'wear level' the flash by distributing the writes around the memory, thereby preventing one part of the flash wearing out too quickly if one file is updated frequently. It's hard to say exactly how well this will all work in practice, but writing data every ten minutes should ensure the flash lasts quite a few years. The flash will fill up long before then, so this is assuming that you regularly delete the **data.csv** file to let it fill up again.

If you're concerned about the lifespan of the flash storage, you could add an SD card and write data there, but given that an SD card is more expensive than a new Pico 2 W, we're not sure this makes financial sense unless you want the extra storage space.

The other part of the program involves writing to the file

ch03-pico-full.py

DOWNLOAD THE FULL CODE:

rpimag.co/pythonpi-resources

> Language: **MicroPython**

```
import board
import busio
import time
import os
import simple_ntp
from adafruit_bme280 import basic as adafruit_bme280
from secrets import secrets
import network
from machine import UART, Pin
from ch03_pms5003 import init_pms5003, read_pms5003

uart = UART(0, 9600, tx=Pin(12), rx=Pin(13))
init_pms5003(uart)

wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect(secrets['ssid'], secrets['password'])

max_wait = 10
while max_wait > 0:
    if wlan.status() < 0 or wlan.status() >= 3:
        break
    max_wait -= 1
    print('Waiting for connection...')
    time.sleep(1)

if wlan.status() != 3:
    raise RuntimeError('Network connection failed.')
else:
    print('Connected.')
    status = wlan.ifconfig()
    print('IP = ' + status[0])

simple_ntp.set_time()

print("Time set.")

i2c = busio.I2C(board.GP1, board.GP0)
bme280 = adafruit_bme280.Adafruit_BME280_I2C(i2c)

# Create CSV file if it doesn't already exist
try:
    f = open("data.csv", "r")
except OSError:  # open failed
    f = open("data.csv", "w")

f.write("Time,Temperature,Humidity, \
        Pressure,PM1,PM2.5,PM10\n")
f.close()

rtc=machine.RTC()

while True:
    # Check free space
    stat = os.statvfs("/")
    freespace = stat[0] * stat[3]
    print("Free Space: ", freespace)
    if freespace > 1000: # At least 1Kb left.
        with open('data.csv', 'a') as file:
            timestamp = rtc.datetime()
            pm1, pm25, pm10 = read_pms5003(uart)
            timestring = "%04d-%02d-%02d \
     %02d:%02d:%02d" % (timestamp[0:3] + \
                    timestamp[4:7])
            file.write(
                    timestring + ","
                    + str(bme280.temperature)
                    + "," + str(bme280.humidity)
                    + "," + str(bme280.pressure)
                    + "," + str(pm1)
                    + "," + str(pm25)
                    + "," + str(pm10)
                    +"\n")
            print("Wrote data.")

    time.sleep(10)
```

Custard Cream Server

A Raspberry Pi-hosted image storage service which you can run on your desk and program in JavaScript



Maker

Rob Miles

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

robmiles.com

YOU'LL NEED

- Raspberry Pi 4 or Raspberry Pi 5
- microSD card (between 16GB and 64GB will work fine)
- An external hard drive or the Raspberry Pi SSD Kit

The Custard Cream Server was created to complement the Custard Cream Camera. Images taken with the camera can be uploaded for the whole world to see. In this article, we'll discover how the server is built and the technology it uses. You can find all the code for the server along with instructions for deploying it at **rpimag.co/custard-cream-server**.

Custard Cream Camera

Figure 1 shows the Custard Cream Camera. You can find out all about it in the article in issue 170 (**rpimag.co/170**). After you have taken a picture with the camera and edited it, you can publish it on the Custard Cream Server.

Figure 2 shows the camera display just after a picture has been published. It shows a three-word phrase that identifies the image on the server. You can browse to **custardcream.cam** and enter the phrase to locate the image.

Figure 3 shows the homepage for the server. It provides a space for you to enter the phrase. When you click the 'Find picture', the server opens a page which finds the picture and displays it.

Figure 4 shows the picture as displayed on the site. You can click the Home button to return to the homepage. The Slide Show button displays a slideshow of related pictures.

Figure 5 shows the slideshow view. Pictures are presented as prints falling on an infinitely long tabletop. The server provides the images and a JavaScript program running inside the browser performs the animation. Now that we know what the web pages do, let's investigate how they are fetched by the browser.

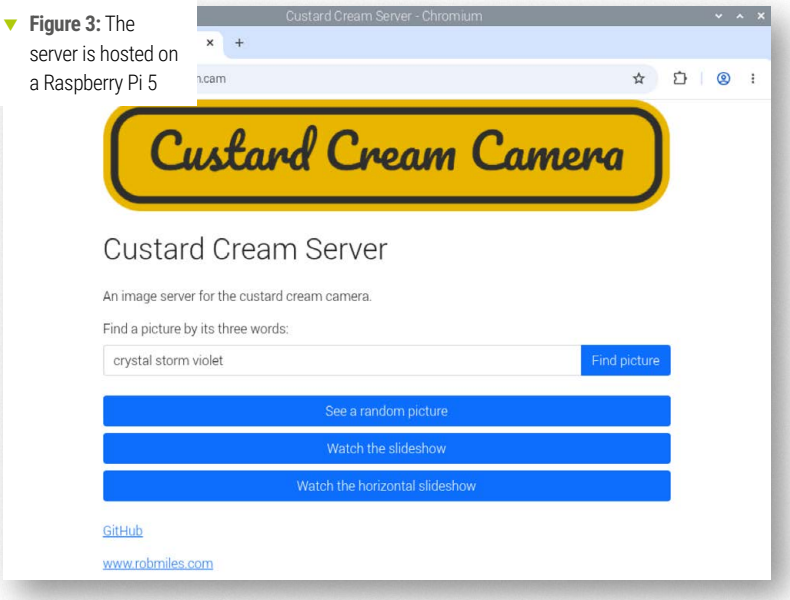
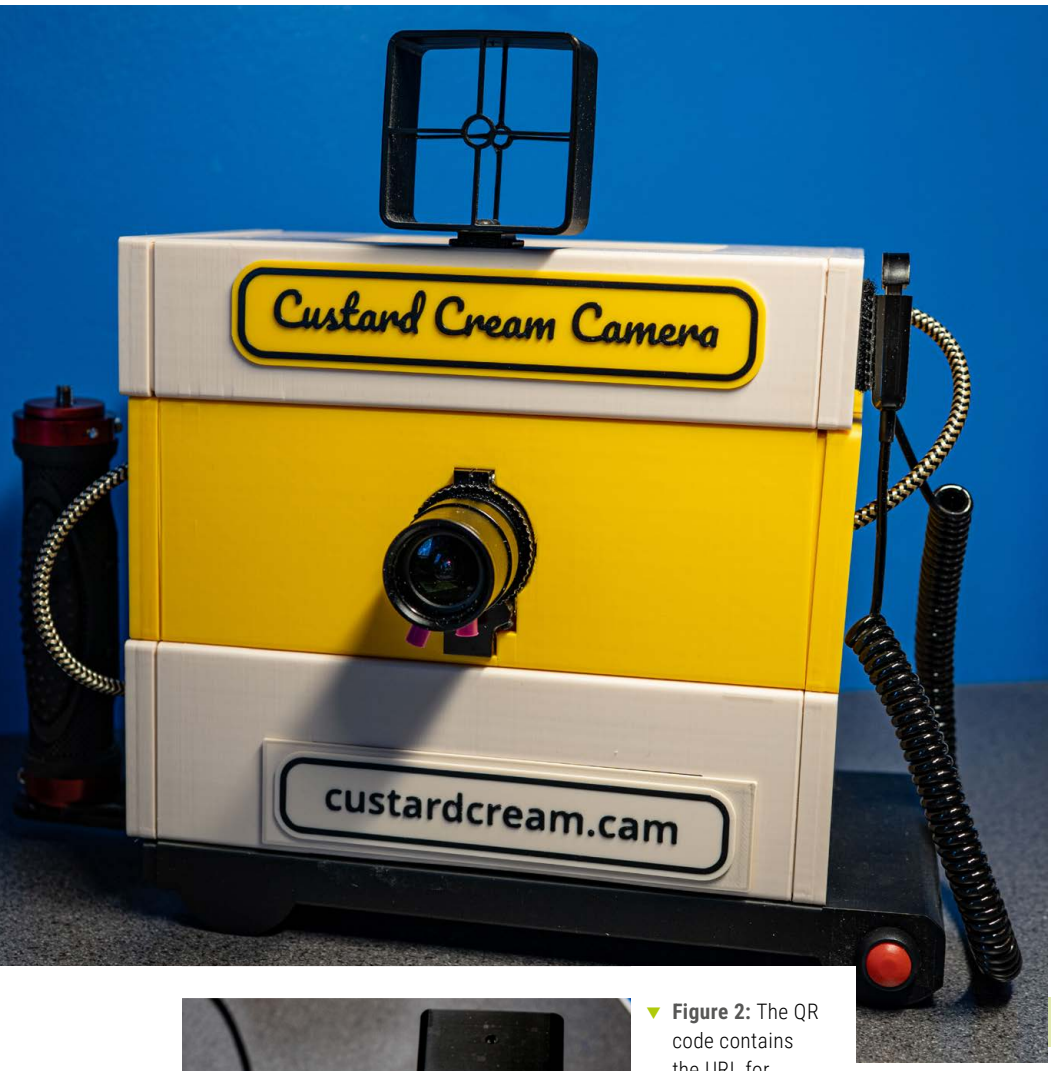


Figure 3: The server is hosted on a Raspberry Pi 5



◀ **Figure 1:** The curly cable connects a microphone which enables voice-controlled picture edits



▼ **Figure 2:** The QR code contains the URL for the picture on the server

Images taken with the camera can be uploaded for the whole world to see

How it works – the website

When you open a website, the browser (perhaps Chrome, Safari, Edge, or Firefox) extracts the name of the host (**custardcream.cam**) from the request and asks the Domain Name System (DNS) for the Internet Protocol (IP) address of the server device for that host. This is the same as looking up someone's name in a telephone directory and getting the number of their telephone, except that nobody does that any more.

The browser uses the IP address to contact the server machine on the internet and send it a Hyper Text Transfer Protocol (HTTP) message. The message starts with the word GET, which is how a browser asks a server for a page.

```
GET / HTTP/1.0
```

The `GET` request is followed by two things: the address on the site of the page to get, and the version of the HTTP being used for the conversation. The address `/` means the resource at the very top of the site, which is usually the 'index' page. The Custard Cream Server sends the page you can see in **Figure 3**.

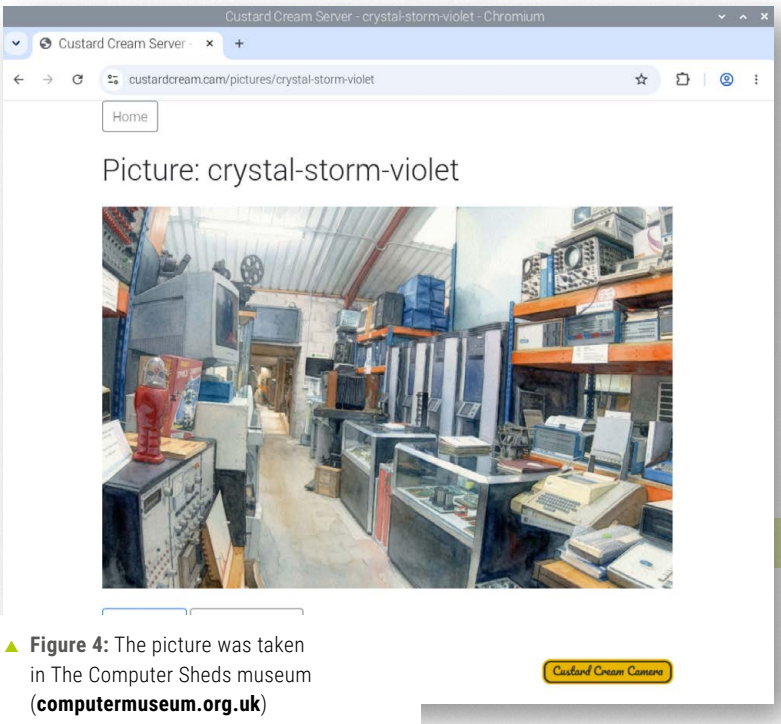
QUICK TIP

If the browser wants to send something to a server (for example upload a picture) it would use a `PUT` message.

The server replies to the request with a response which begins with a header:

```
HTTP/1.1 200 OK
```

The header contains the protocol used for the response (in this case, `HTTP/1.1`), a status code (in this case, `200`) and a reason phrase (in this case, `OK`).



▲ **Figure 4:** The picture was taken in The Computer Sheds museum (**computermuseum.org.uk**)

After the header, the server will send the type it'll be sending:

```
Content-Type: text/html
```

The type above tells the browser that it is about to receive an HTML document. This is like you calling your mum and asking her to get you a drink. You address the "Please could you get me a drink" request to mum and what comes back is either "No – Get it yourself", or "OK – Here is a coffee, white with sugar". It's important to understand the difference between Hyper Text Transfer Protocol (how you ask and mum responds) and Hyper Text Markup Language (one of different drinks that mum could get you).

Now we know how the basics, we can see that a server is a program that accepts HTTP requests and then sends responses, some of which are formatted with HTML. We could sit down and write such program from scratch. That's what Tim Berners-Lee, the inventor of the World Wide Web, did in 1990. However, there are now much easier ways to build web servers.

Taking the Express

The Custard Cream Server program uses a framework called Express (**expressjs.com**) which runs JavaScript code in response to HTTP requests and uses EJS (Embedded JavaScript, **ejs.co**) middleware HTML template pages to construct responses. This sounds horribly complicated, but boils down to solving two problems: we want something that can run our code in response to web requests, and we want something that lets us customise web page templates with relevant data.

QUICK TIP

HTTP status code 404 is the infamous 'page not found' error.

The web server code runs as a JavaScript program inside the Node.js runtime environment and uses the Node package manager (npm) to import the Express and EJS frameworks. There are instructions for setting this up on the article repository: **rpimag.co/custard-cream-server**. When the Custard Cream Server starts running, it creates an app using the Express package:

```
const express = require('express');
```

This statement uses the JavaScript `require` mechanism to load the `express` package. This package returns a factory function (i.e. a function that makes a JavaScript object) which will make an Express application object. The factory function is used to make an app.

```
const app = express();
```

The code above creates a server application and sets the variable `app` to refer to it. Now that we have our server, we can add some routes to it.

Rooting for routes

If you take a careful look at the top of **Figure 4**, you will see the Uniform Resource Locator (URL) which is being used to access the picture:

```
https://custardcream.cam/pictures/crystal-storm-
violet
```

The first component of the URL is the scheme being used to access the resource (in this case, we will be sending HTTP requests). The second component is the host (in this case, we are talking to **custardcream.cam**). The third component is the path which is passed to the server. A path is broken into segments, separated by the '/' character. The first path segment is **pictures** (we want to look in the pictures storage part of the server) and the second path segment is **crystal-storm-violet**. This is the three-word phrase of the picture you want to view.

The Express framework can bind JavaScript code to a path. When you navigate to **https://custardcream.cam/pictures/xxx** (i.e. the **pictures** path segment followed by any other address segments), a JavaScript function can be made to run to deliver

the response. We bind the function to the address using a router object which the server loads when it starts.

```
const pictures = require('./pages/pictures');
```

The statement above sets the variable `pictures` to refer to the item exported by the JavaScript source **./pages/pictures.js**. You can find the source code file in the repository for the project. When something is loaded using `require`, the code in the source file is executed. The file **./pages/pictures.js** contains code that creates a router object (we will see how in a moment) and the very last statement of this file exports the router object (which it has called `router`):

```
module.exports = router;
```

JavaScript uses `require` and `exports` as way of linking a source file (something which exports a thing to be used) and a user program (something which requires that library). You can think of a module that exports something as a toolmaker. You need a tool to do a particular job, so you run a program to make that tool by using `require`. The program runs and creates the tool you need (in this case, the router object). Then it uses `export` to deliver the object so that it can be picked up by your code. Once the server has the router exported by pictures, it can connect that to the pictures path segment:

```
app.use('/pictures', pictures);
```

This tells the app implementing the web server that if it sees anything on the **/pictures** path, it must show it to the `pictures` router. There are also routers for all the other paths available from the server, but let's look at how the `pictures` router works.

Getting in the picture

You can find this code in the file **/pages/pictures.js** in the source repository. The purpose of this code is to create and configure a router for use by another program. The router is based on one provided by the Express framework. The `Router` factory function code is held in the `express` package, so the first thing the router does is use `require` to get a reference to `express`:

```
const express = require('express');
```

The statement before loads the express package and creates a variable called `express` which refers to it. We can now create a `Router`:

```
const router = express.Router();
```

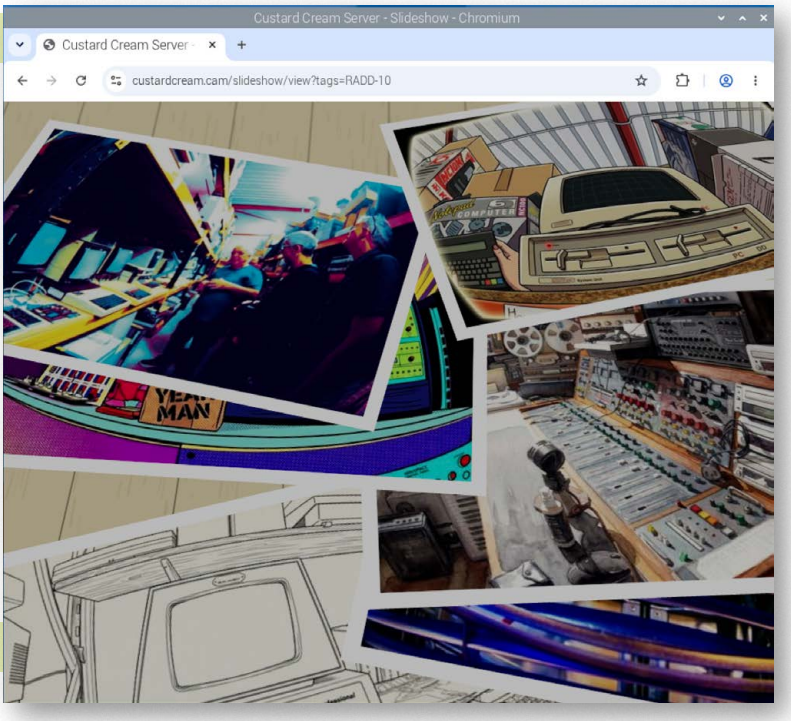
The statement above sets the variable `router` to refer to a new router which was created by the `Router()` factory function.

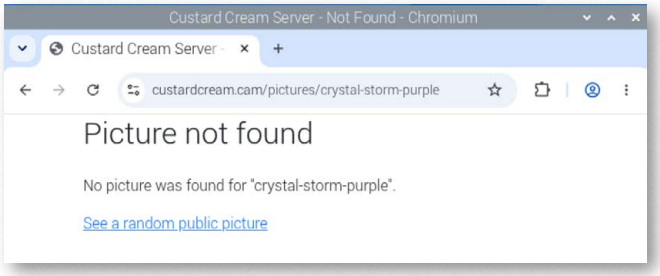
Now we are going to specify the behaviours in the router we have just created. If got a new butler, you'd have to spend a while telling them what to do. You'd say things like "If the gardener calls round asking for the plant cuttings, could you get them from the greenhouse? If the gardener asks for the lawnmower, could you get it from the garage?" Your butler will accept requests, decode them, and then act appropriately. This is just what we are doing with the router. The router contains a function called `get` which we can use to specify a function to run when a particular route is requested by the browser.

QUICK TIP

A router may contain multiple calls of `get` to set up different routes.

Figure 5: The pictures were generated using image processing on the Custard Cream Camera





▶ **Figure 6:** As a consolation prize, you are offered the chance to look at a random picture

The server has bound the `pictures` router to the **pictures** address segment. So, in the `pictures` router code, we are matching the address segment that comes after the **pictures** segment in the URL. This is the three-word phrase we are going to find the picture for. Let's have a look at the start of the call of `get` that sets up this route:

```
router.get('/:phrase', async (req, res) => {
```

The line above is the most complicated piece of code in this entire article. The first part is easy enough: `router.get(` says that we are making a call to the `get` function on the `router` object. The string `'/:phrase'` is passed as an argument into the call of `get` and says: "match any path segment and allow the handler function to refer to it as 'phrase'." When the function runs, it will get hold of the phrase and use it to find the required picture.

The second argument that is passed into `get` is an asynchronous (`async`) and anonymous `(=>)` function that we are going to create to handle the request. Asynchronous means "don't wait for this function to complete, let it run in the background", and anonymous means "the body of the function is right here, it is not a named function located somewhere else". The anonymous function is given two arguments when it is called by the router in response to a server request: `req` and `res`.

The `req` argument describes the request (i.e. what has been asked for) and the `res` argument is used to deliver the result. The code that follows the header is the code that runs to find the picture and display it as the web page shown in **Figure 4**. It's probably worth taking a step back here and considering what we are trying to do, so we can be clear about everything going forwards. The starting point was the URL that is used to locate a picture:

```
https://custardcream.cam/pictures/crystal-storm-
violet
```

We've created a router called `pictures` and bound it to the address element **pictures** in the URL. When the server gets a request for the **pictures** address, it will send it to that router. We are using the `get` function in the router to tell the router about the code to perform when an address below **pictures** is hit. We tell `get` things by passing arguments to the call. The first argument defines the address to match (in this case the phrase). The second argument defines the code that will be performed when the address is hit. We are now writing that code. The first thing the code must do is find the picture we want to display.

```
const picture =
    await Picture.findOne(
                { phrase: req.params.phrase });
```

This code above is the first statement in the function that runs to handle a picture request. It is the second most complicated piece of code in this article. It gets the `phrase` out of the `params` to the request (remember that `req` is a parameter to the call of the handler function that tells it about the request) and uses `findOne` to make a query of the `Picture` database which holds all the picture details. It sets the variable `picture` to whatever the database returns. If the phrase is found, the matching picture record will be returned and assigned to the `picture` variable. If the phrase is not found, the value of `picture` is set to `null`. This would happen if we tried to access a picture that isn't there:

```
https://custardcream.cam/pictures/crystal-storm-
purple
```

The above link has the wrong colour (we should have said violet). If we use this address, we get a special message in the browser: **Figure 6** shows the 'Picture not found' page you see if you get a phrase wrong. We are very close to understanding just how this is generated. Let's look at the next part of the picture route handler and discover how it detects a failed request and displays this.

```
if (picture === null) {
```

This statement tests the value of `picture`. The code in the block controlled by the test will run if `picture` is `null` (i.e. there was no picture matching the phrase). This must create a 'Picture not found' response. Now let's look at how it does it.

Breaking bad news

This part of the handler for picture display has the job of telling the user that things did not go well when they asked for a picture with an invalid phrase. We start by setting an appropriate response code. The following statement sets the status code of the response to the value 404. In the HTTP standard, this means 'Not found'.

```
    res.status(404);
```

Now we send the 'not found' web page. The application contains definition files for all of pages that it sends back to the user. You can find them in the **/views** folder in the application repository. They have the **.ejs** extension to indicate that they are for use by the embedded JavaScript (EJS) package. We are going to use **pictureNotFound.ejs** to break the bad news.

```
    res.render('pictureNotFound.ejs',
        { phrase: req.params.phrase });
```

The `render` method is given the name of the web page to render, along with information to be put into the page text. Most of the page content will be the same every time it is sent, but it needs to include the phrase so that this can be displayed as part of the "No picture was found for this phrase" message. Let's have a look at the **pictureNotFound.ejs** file:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <title>Custard Cream Server - Not Found</
title>
</head>

<body>
    <div class="container mb-4">
        <h1 class="mb-4">Picture not found</h1>
        <p>No picture was found for "<%= phrase
%>".</p>
        <p><a href="/random">See a random public
picture</a></p>
    </div>
</body>

</html>
```

If you compare the text above with the page in **Figure 6**, you'll get a feel for how HTML works. The `<title>` element sets the title of the page in the browser. This is followed by a `<div>` container element that contains a heading `<h1>` and two paragraphs `<p>`. The first displays the "No picture was found for" message. The interesting thing about this line is the `<%= phrase %>` part. Remember that when we called `render`, we passed in a tiny object which contained a property called `phrase`. This is where the value of `phrase` is passed from the JavaScript program directly into the text of the web page.

When `render` builds the response to be sent to the browser, it looks for the sequence `<%=` in the text and then regards what it finds after that as the name of a JavaScript value to be inserted into the web page. When render sees the `%>` sequence, it goes back to just passing the HTML document to the output. This is how 'crystal-storm-purple' gets into the web page shown in **Figure 6**. This part of the process is performed by the EJS (embedded JavaScript) package. We call EJS middleware as it fits within an existing framework. Now that we've handled a failed request, let's take a look at how a successful one works.

QUICK TIP

You can embed JavaScript code (rather than just variable values) into a web page. This can make the page rather hard to read (you have HTML and JavaScript together), but it is super-powerful.

Back to the happy path

```
router.get('/:phrase', async (req, res) => {
    const picture =
      await Picture.findOne(
                { phrase: req.params.phrase });

    if (picture === null) {
        res.status(404).render(
            'pictureNotFound.ejs',
          { phrase: req.params.phrase });
        return;
    }

    res.render('picture.ejs', { picture });
});
```

The code above shows the complete `get` function. We can see the code for the 'unhappy path' where the picture value is found to be null. Right at the bottom of the code, we can see the statement that completes the 'happy path' where the picture is found:

```
    res.render('picture.ejs', { picture });
```

The statement above renders the EJS page and passes it the entire `picture` object obtained from the database. The page can be populated with data from the object. Let's have a look at the part of the EJS page which draws the picture information you can see in **Figure 4**.

```
<h1 class="mb-4">Picture: <%= picture.phrase
%></h1>

<img src="/pictures/<%= picture.filename %>"
alt="<%= picture.phrase %>" class="img-fluid
mb-4">

<p class="mb-4">
    <% if (picture.tags) { %>
        <a href="/slideshow/view?tags=<%=
encodeURIComponent(picture.tags) %>"
       class="btn btn-outline-primary">Slide
Show</a>
    <% } %>
    <a href="/random" class="btn btn-outline-
secondary">Random picture</a>
</p>
```

The code above contains JavaScript alongside the properties of the picture that is being displayed. The JavaScript code makes a decision to display a slideshow button if the picture has any tags. The user clicks the button if they want to see a slideshow with 'matching' tags, i.e. viewing a picture with the `CustardCreamCamera` tag gives you the option to see a slideshow of all the pictures with that tag. The sequence `<%` marks the start of the JavaScript and `%>` marks the end. The JavaScript can use the value of `picture` (the data about the picture being displayed) because this was passed into the call of the `render` function.

If this seems rather hard to understand, the best way to get to grips with it is to focus on the problem that is being solved. We want to make a web page which contains a link to the image file and description information about the page. We also want the page to display a 'Slide Show' button if the page has any tags assigned (so that we can make a slideshow out of pictures with matching tags). We do this by putting active code inside the markup information for the page. When the page is rendered, the code runs and can make some parts of the page appear conditionally. This is super-powerful. It is a very neat way to make web pages with active elements.

Scratching the surface

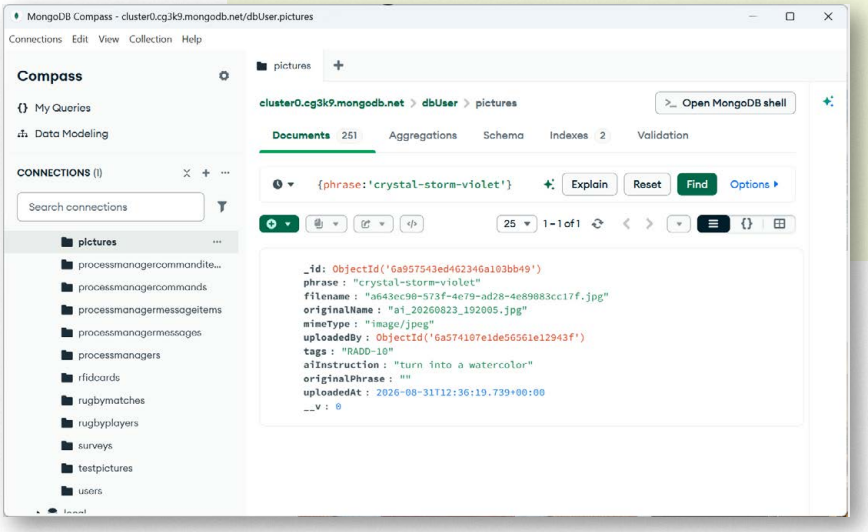
This article only scratches the surface what the Custard Cream Server does. However, it has hopefully removed some of the mystery of what happens behind your browser when you surf the web. Many sites feed address segment values into databases to find information and then present it, and now you know how this can be made to work.

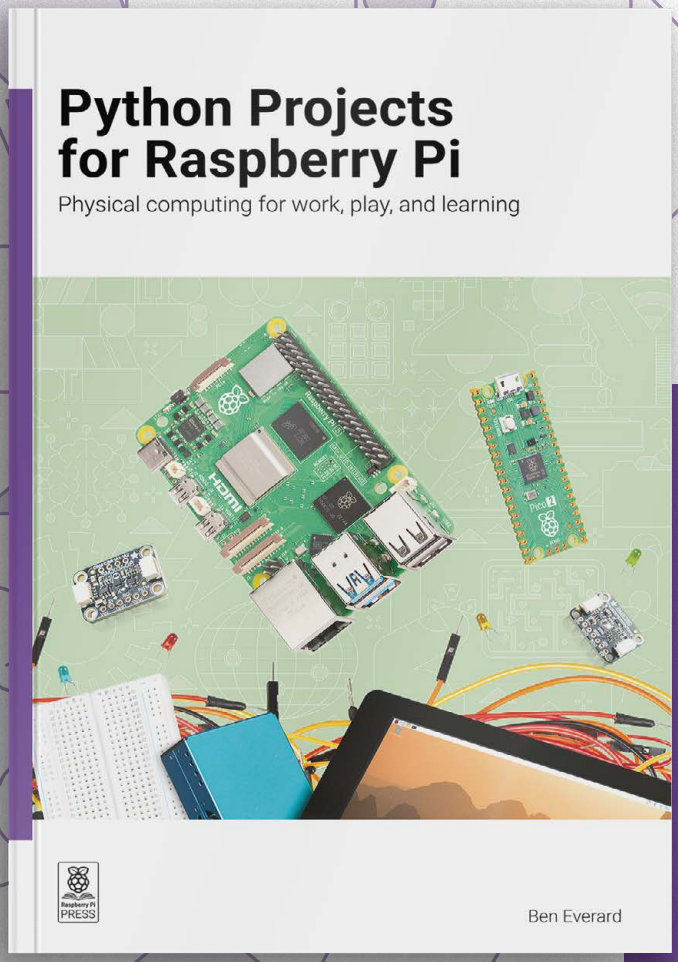
Why use a database?

You might wonder why we need a database for the Custard Cream Server. If we matched each picture file name to the phrase used to locate it (for example, we had a picture file called **crystal-storm-violet.jpg**), the server would not need to look up the names of pictures. It could just load the file by name. However, using a database gives us a lot of flexibility. It allows us to store metadata about the pictures. Metadata is 'data about data'. The Custard Cream Server uses tag metadata to group pictures together. Pictures taken at a particular event can be tagged so they can be found later. The database also records any AI edit instructions used on a picture. The server uses a MongoDB database which is accessed in JavaScript via the Mongoose package.

Below is the data entry for a picture. It contains a file name field which is where the picture is stored. You can see the tags assigned to the picture, and the AI instruction that was used to create it. Later versions of the server might add and use more metadata. One of the wonderful things about MongoDB is that it is very easy to add new fields to an existing database. Note that the image is not stored in the database. Databases are useful for storing structured data, but large objects like images are best stored in files.

- The content of a database entry is defined by a JavaScript class you can find in **schemas/pictures.js**





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

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

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

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

Part 03

Doll's house details

Pottery on a tiny scale, and a teensy TV that works!



Maker

Nicola King

Nicola is a freelance writer and sub-editor. This month there's definitely an autumnal feel in the air in this part of the world.

@holtonhandmade

In this final instalment of our doll's house project, we're going to craft a couple of tiny accessories for the house using air dry clay. We would not call ourselves full-fledged miniaturists at this point, as our attention to precise and intricate detail is maybe a little lacking in some areas, but with a little more practice, who knows? In addition to tiny crafting (be warned: requiring a bang-up-to-date eyewear prescription), we will be adding some screen-based tech to our tiny living room. So, first, let's get to slippery grips with our clay.

Step up to the plate

The tiny clay accessories we are going to make include a plate and a pot. Sounds straightforward, and it is to a certain extent, but it's slightly complicated by the fact that these are going to be very small, and therefore a little fiddly.

First of all, we took some terracotta clay, worked it with our hands until it was more malleable, and then rolled it out. Do remember that as you are working with air dry clay, it immediately starts drying slowly when exposed to the air. Your working time is therefore not short but it is limited, so have some 'slip' on hand. Slip is used to 'glue' pieces of clay together, but is also useful when clay starts to dry out, just to add some moisture again. (It's easy to make a few hours in advance, by taking a tablespoon of water and adding some small paper-thin pieces of clay, ideally equal amounts of clay and water, to it. It will go milky in colour, and after a few hours the clay will absorb the water. You should end up with a thin, smooth consistency and this will make perfect slip.)

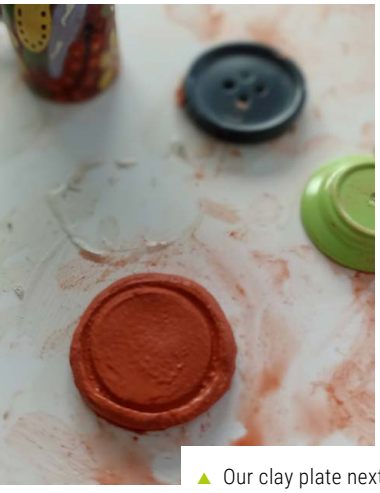
QUICK TIP:

To avoid clay sticking to your fingers, rub a little hand cream or lotion on your hands before getting underway.

▼ This miniature TV really works. Powered by Pico, it receives a video stream over the wireless network



We're going to craft a couple of tiny accessories for the house using air dry clay

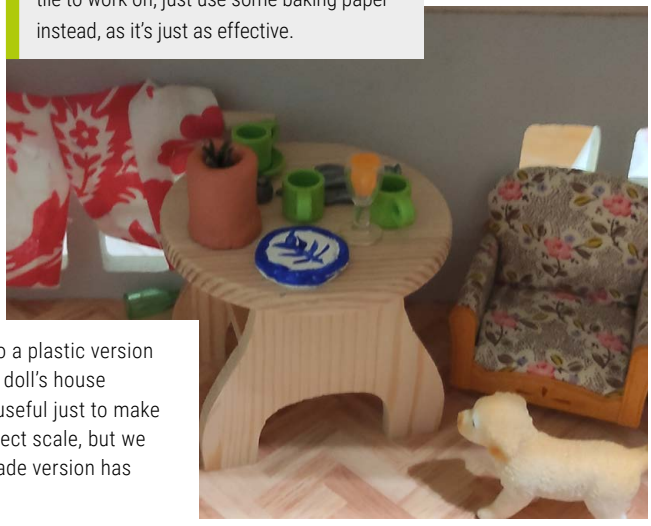


▲ Our clay plate next to a plastic version we found in a box of doll's house accessories. It was useful just to make sure we had the correct scale, but we do think the homemade version has more character!

QUICK TIP:

If you don't already have a craft mat or an old tile to work on, just use some baking paper instead, as it's just as effective.

▼ Our clay plate and plant pot on the table in the house



YOU'LL NEED:

- Your doll's house
- Some air dry clay (we used terracotta-coloured clay; plain white is a good option)
- A plate 'mould'; e.g. an old button
- Water dish (for your slip)
- Sculpting tools
- Surface to work on; e.g. an old bathroom or kitchen tile
- Clay roller (anything cylindrical will suffice; e.g. a glass bottle)
- Something to cut the clay; e.g. a knife, or even a plastic ruler
- Acrylic paints (or something to colour your finished item)
- Fine-tipped paintbrush
- Sealant (e.g. water-based varnish)
- Raspberry Pi Pico W or 2 W (with headers)
- Pico Display Pack (**rpimag.co/picodp28**) or similar
- A computer to stream a video file



◀ A collection of items used in this project. Keep all opened clay in an airtight bag and it will last you for months

▼ Using a sculpting tool to hollow out our tiny pot. This is real pottery on a miniature scale!



QUICK TIP:

As soon as you are finished, store scrap clay immediately in an airtight container. Ours has been stored like this for many months and is still usable.

Clued up on clay

We've chosen to use air dry clay in this tutorial and there are many reasons why we've selected this medium. It's a water-based modelling material that does not require a kiln, oven, or any other specialised equipment for it to dry – it simply hardens naturally at room temperature. (Polymer, earthenware, stoneware clays and the kind of clays used by potters, for example, do require some kind of oven or kiln to harden them.) When left to dry, the moisture from the air dry clay evaporates into the surrounding air. Now, this hardening is not instant, so you will usually have to wait a couple of days for the full hardening process to be complete. This is a point to bear in mind, and also consider that we don't want our clay to dry too fast as that can cause cracking (as the piece will naturally shrink a tiny bit as it hardens).

Other benefits of air dry clay include the fact that most types are non-toxic, it's relatively inexpensive, and it's versatile and easy to use. One thing we've noticed is that it is worth investing in a decent brand of clay. We've used the well-known and readily available DAS clay (**hobbycraft.co.uk/brand/das**). We wouldn't normally be fussy over particular brands, but these are packets that we opened literally months ago, kept completely airtight, and are still highly malleable and usable. We also bought a cheaper craft store brand at the same time and, after retrieval from its airtight bag, it didn't perform as well as its counterparts and had to be binned, so do bear that tip in mind.

Once our clay was rolled out around 4mm in thickness, we took an old button and cut around it to make the basic plate shape. We then indented the centre to give it more of a plate-like form. Basically anything circular that is smaller than the button we used will work here. It was then that we used a little slip just to smooth the edges and then we left it to dry for about 24 hours. Once dry, we got our acrylic paints out, along with a very fine brush, and attempted some artwork. Once again, leave to dry and then you can use a varnish to seal and protect it.

Gone to pot

Now, when it came to making our pot, the terracotta clay that we were using was perfect: it was the exact 'look' we wanted and it meant that we didn't need to paint the finished item, just seal it. We rolled our clay this time into a small tube and cut off a length about 3 cm. It's then a question of forming a pot-like shape, based on however you want to tackle it and whatever shape you want to achieve. We indented one end of our tube shape, and then began to gently hollow out a pot-like dip with our sculpting tools. Once happy with it, again use some slip to smooth down edges, and then leave it to dry. Paint/decorate if you wish, when dry, and seal as before. Once your pot is complete, you could fill it with some coffee grounds for a soil-like effect, and mould and paint some leaves from your clay and then glue them in – we used the green leaf top from a tiny garden tomato and it looks like a cactus in our pot.

TV tech

To keep our doll's house inhabitants entertained, we opted to give them a TV. The easiest method is to stick a mini 'flatscreen' display on a wall. While you could use one powered by a Raspberry Pi Zero, we opted for a Pico-based one, Pimoroni's Display Pack, since we already had the 2.8-inch version in our stash.

One issue is that Pico lacks the required processing power and RAM to decode and play video by itself. You could just show a series of still images. Inspired by Alan Edwardes's example (**rpimag.co/picovideostream**), however, we opted to stream a video from a remote computer (you could use a Raspberry Pi). We fine-tuned his Pico MicroPython script, as shown in the **tvstream.py** listing (overleaf).

You'll need to insert your own Wi-Fi SSID and password on lines 9 and 10, and the IP address of the computer which will be streaming the video on line 41. You may also need to use a different country code on line 7. Before running the script, start your video file streaming on the computer with the following terminal command:

```
  $ ffmpeg -re -i <video.mp4> -vf
"scale=320:240,fps=10:round=up" -f rawvideo
-pix_fmt rgb8 tcp://0.0.0.0:2000\?listen
```

...replacing `<video.mp4>` with the path to your chosen video file. Now run your Pico script (you can save it as **main.py** so it'll run automatically on power-up). Within a few seconds, the video should start playing on its screen. The frame rate is a little low and there's no sound, but it works! We attached our mini 'TV' to a wall of the house with some sticky tack, but you could give it a proper surround or put it in a 3D printed case. We powered Pico via USB, but you could do it via its VSYS and GND pins.

That concludes our little series on crafting and tech on a tiny scale. As always, we hope this has ignited some ideas if you are interested in doing something similar. This is one of those projects that you can just keep adding to and tinkering with, and can truly make it your own. Make sure you share any doll's house creativity and tech that you create with us here at the magazine.

QUICK TIP:

Don't overdo the use of water, or slip, when working with air dry clay, as you need very little. Too much can make the project soggy and increase drying time.

3D-print some diminutive furnishings

One standout option with regard to furniture sourcing for any doll's house is the 3D printing rabbit hole. If you don't have the time to craft something yourself, even though that is a hugely rewarding and satisfying process, then 3D printing might be an alternative or complementary route to consider. It's certainly an avenue that others have already traversed, and you can also find people online willing to 3D-print items for you. Whatever tiny home décor or furnishing item you are contemplating, you can almost guarantee that someone, somewhere has already designed and 3D-printed it. Whether you have your own 3D printer or not, here are a few examples to inspire:

1. Maker Martin Hardi has designed and 3D-printed everything from dressers to microwaves to doll's houses for the doll's house (!) – **rpimag.co/tinydollhouse**.

2. Take a look at this site for details of many minis including the kitchen sink, but these makes are far from dull as dishwater – **rpimag.co/printablesdollfurniture**.

3. Finally, this website is an inspiring go-to for plenty of infinitesimal ideas – **rpimag.co/pinterestdollfurniture**.

tvstream.py

> Language: **MicroPython**

DOWNLOAD THE FULL CODE:

rpimag.co/github

```
import network
import picographics
import rp2
import socket
import time

rp2.country("GB")

ssid = "<Wi-Fi SSID>"
psk = "<Wi-Fi Password>"

# Connect to Wi-Fi
wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect(ssid, psk)

print("Connecting to Wi-Fi...")
while not wlan.isconnected():
  time.sleep(0.5)
print("Connected! IP:", wlan.ifconfig()[0])

# Initialize display with RGB332
display = picographics.PicoGraphics(
    display=picographics.DISPLAY_PICO_DISPLAY_2,
    pen_type=picographics.PEN_RGB332,
)

# Turn on the backlight
display.set_backlight(1.0)

width, height = display.get_bounds()
print(f"Display resolution initialized as: {width}x{height}")

# Create a backing bytearray and map it to the framebuffer
frame_size = width * height
fb_data = bytearray(frame_size)
display.set_framebuffer(fb_data)
mv = memoryview(fb_data)

# Connect to server socket
sockaddr = socket.getaddrinfo("<IP address of computer>", 2000)[0][-1]
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

print("Connecting to streaming server...")
sock.connect(sockaddr)
print("Streaming active!")

# Streaming loop
try:
  while True:
    bytes_read = 0
    while bytes_read < frame_size:
      chunk = sock.readinto(mv[bytes_read:])
      if not chunk:
        raise OSError("Connection closed by server (EOF)")
      bytes_read += chunk

    # Push the buffer to the screen
    display.update()

except Exception as e:
  print("Error:", e)
  sock.close()
```

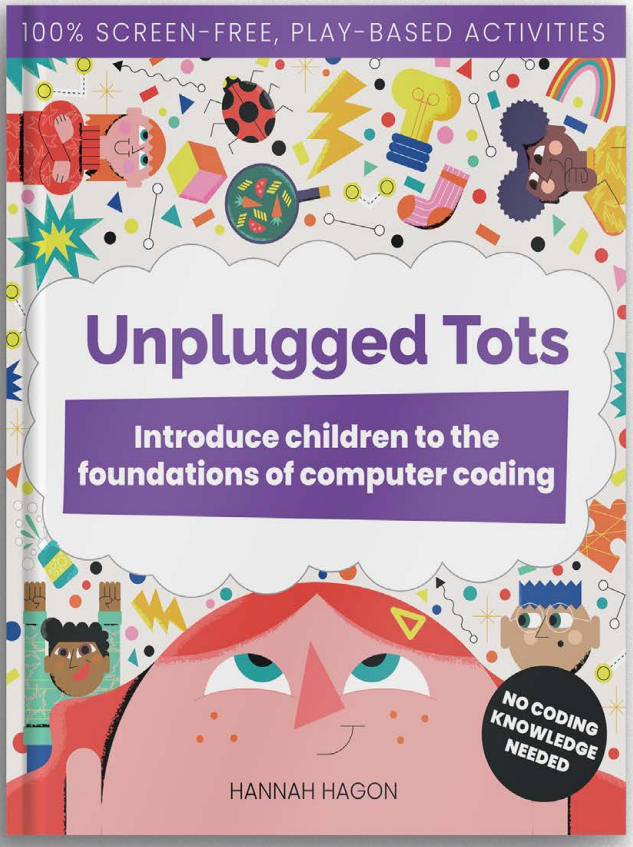
QUICK TIP:

You can add a little PVA glue to your slip to help strengthen small, detailed items.



We found that rolling clay is a bit like rolling pastry. We used a button as a mould for our plate

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



rpimag.co/unpluggedtots

Control GPIO with Needle

Turn text input into actions with Needle, a 14MB function-calling LLM



Makers

Roman Shemet
Founder of Cactus, quant, economist, and product leader.

Henry Ndubuaku
Founder of Cactus, author of Maths, CS & AI Compendium.

rpi.co/needle

YOU'LL NEED

- Raspberry Pi 5 **rpimag.co/raspberrypi5**
- Active Cooler **rpimag.co/activecooler**
- Raspberry Pi Camera Module (optional) **rpimag.co/camera3**

We type 'Turn the LED on' and, 78 milliseconds later, the LED on our Raspberry Pi 5 comes on. That's thanks to Needle 2, running on the CPU alone – without a dedicated AI HAT.

Let's start with what Needle is not. It is not a chatbot. Instead, the team at Cactus Compute (**cactuscompute.com**) trained it for one task only: reliable, structured on-device actions.

You declare Python functions, then Needle picks which one to call and fills in the arguments. We used it to write local notes, query `vcgencmd` (for system info), interact with the LEDs, and more on a CPU-only Raspberry Pi 5.

Needle setup

The setup is simple. First let's open a terminal window and create a virtual environment so we can `pip install` the cactus-needle package:

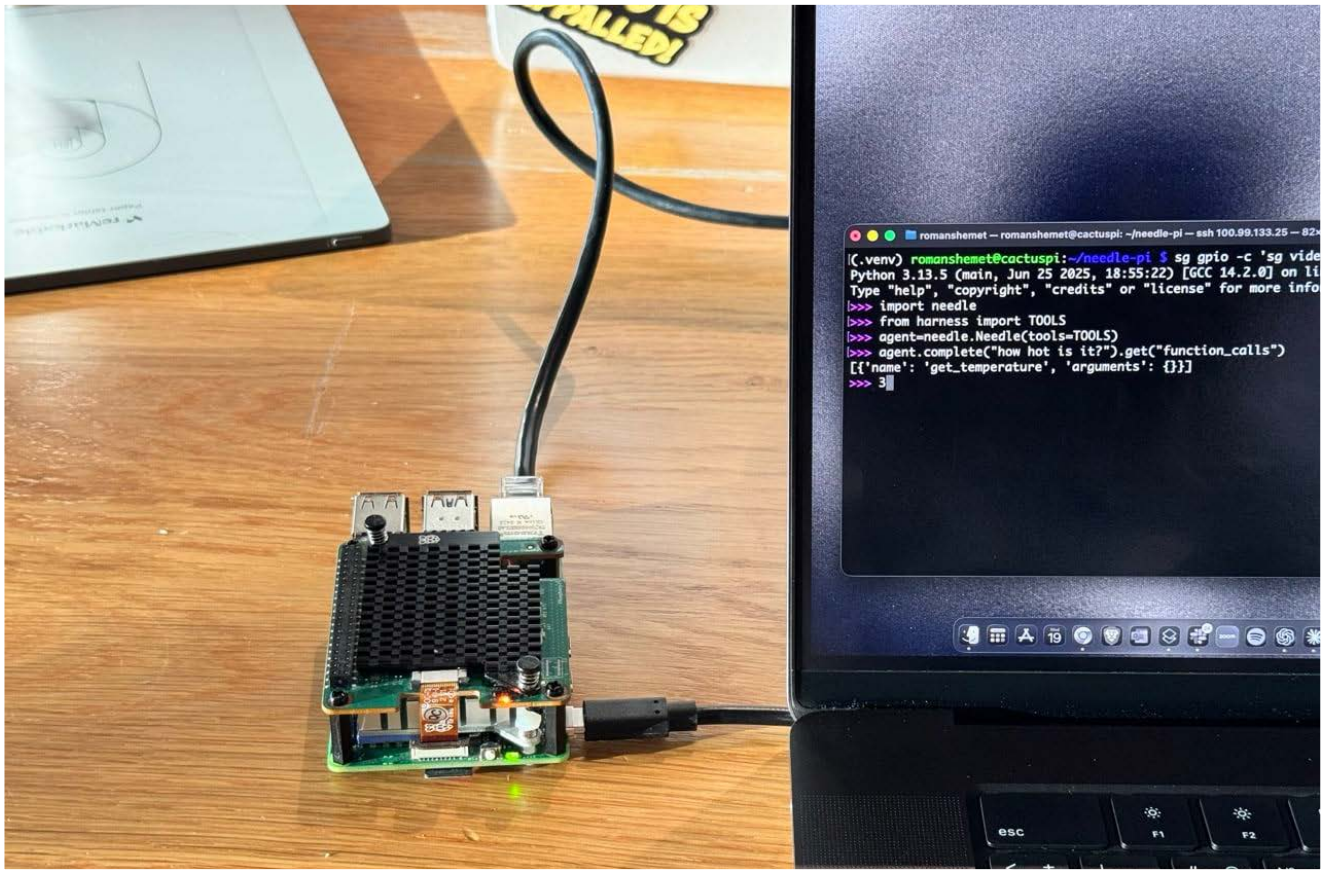
```
$ python3 -m venv needle-env
$ source needle-env/bin/activate
$ python -m pip install cactus-needle
```

Thonny setup

We're going to write our code in Thonny, but first we need to connect it to the virtual environment we just created.

Open Thonny and click **/usr/bin/python3** in the bottom right of the interface. Now choose 'Configure interpreter'. Click the ... icon next to 'Python executable' and use the 'Open file' window to choose **needle-env/bin/activate** and click OK and OK again.

After the first download, Needle runs locally without a cloud API or network connection. This first example needs no extra



▲ Checking temperature and blinking an LED with Raspberry Pi hardware and Needle

hardware. Enter the Python code from **needle_test.py** (shown overleaf) into Thonny.

The `Needle` class accepts functions decorated with `@needle.tool`. The decorator uses each function's name, docstring, and type annotations to build its tool schema. In `save_note`, the annotation tells Needle that `text` must be a string.

Click Run to run the code. Inside `run()`, the model first produces this tool selection (output trimmed for clarity):

> *After the first download, Needle runs locally without a cloud API or network connection*

```
{
  "type": "call",
  "function_calls": [
    {
      "name": "save_note",
      "arguments": { "text": "the cooler is
working" }
    }
  ]
}
```

`run()` executes the function and includes its return value in `response["results"]`. You can view the results in the shell:

```
>>> response["results"]
```

```
[
  {
    "text": "the cooler is working",
    "path": "needle-notes.txt"
  }
]
```

In this run, the initial `complete()` inside `run()` selected the function and produced its arguments in 107ms.

We can now ask questions directly into the shell:

```
>>> agent.reset()
>>> response = agent.run("How hot is this
Raspberry Pi?")
```

Needle calls:

```
{
  "type": "call",
  "function_calls": [
    {"name": "get_temperature", "arguments": {}}
  ]
}
```

`run()` executes `vcgencmd` and includes the reading in its results:

```
>>> response["results"]
{ "temperature_c": 49.9 }
```

Controlling hardware

We also declared `set_led(on)`, `blink_led(times)`, and `take_photo()` via `rpicam-still` to test tool selection. Needle chooses the tool; ordinary Python code decides what happens next. You can similarly decorate any Python function that has a clear name, type annotations, and description with `@needle.tool`.

Needle is deliberately narrow. Ask "What is the capital of France?" and it returns `{ "function_calls": [] }` in 92ms because none of the tools we gave it can answer that question. For an action model, refusing an unrelated request is the correct response.

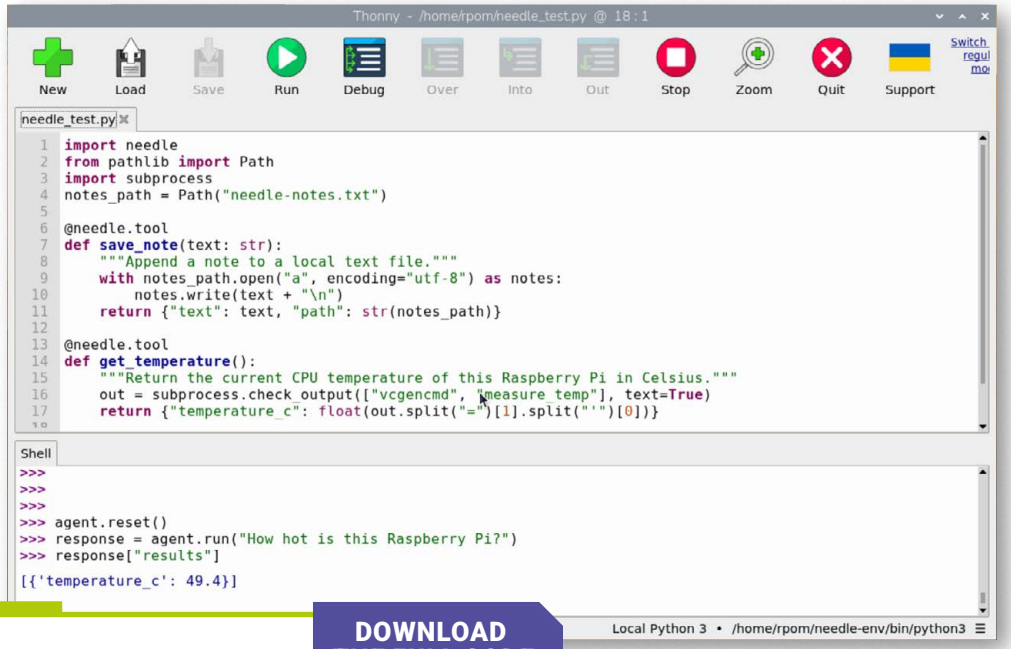
Needle's native session stays around 28MB. In this Python demo, the complete process peaked between 43MB and 46.4MB, including the interpreter.

A few more runs are shown in the table below. These examples were run on a Raspberry Pi 5, 8GB, Raspberry Pi OS, CPU only, cactus-needle 2.0.7. Prefill and decode are Needle's session counters. Each latency in the Time taken column is wall-clock time for one `complete()` call, before the selected Python function runs.

If you already write GPIO Zero functions, you can decorate them and pass them in. The same goes for other application logic. Needle can also be fine-tuned locally on a laptop for a particular set of tools.

You'll find the weights at **rpimag.co/needle2** and code at **rpimag.co/needlegithub**. Both are released under Apache 2.0.

Prompt	Call	Prefill (tok/s)	Decode (tok/s)	Time taken (ms)
Turn the LED on.	`set_led`	488	296	78
How hot is this Raspberry Pi?	`get_temperature`	487	303	149
Blink the LED 2 times.	`blink_led`	475	314	83
Take a photo.	`take_photo`	461	248	76
Save a note that says the cooler is working.	`save_note`	475	305	107
What is the capital of France?	(none)	470	297	92



- Asking Python code questions with an LLM in Thonny's shell

needle_test.py

> Language: **Python**

DOWNLOAD THE FULL CODE:

rpimag.co/github

```
001. import needle
002. from pathlib import Path
003. import subprocess
004. notes_path = Path("needle-notes.txt")
005.
006. @needle.tool
007. def save_note(text: str):
008.     """Append a note to a local text file."""
009.     with notes_path.open("a", encoding="utf-8") as notes:
010.         notes.write(text + "\n")
011.     return {"text": text, "path": str(notes_path)}
012.
013. @needle.tool
014. def get_temperature():
015.     """Return the current CPU temperature of this Raspberry Pi
     in Celsius."""
016.     out = subprocess.check_output(["vcgencmd", "measure_temp"],
     text=True)
017.     return {
     "temperature_c": float(out.split("=")[1].split("'")[0])}
018.
019. agent = needle.Needle(tools=[save_note, get_temperature])
020. response = agent.run("Save a note that says the cooler is
     working.")
```

Emulate Apple II

Recreate the classic 8-bit computer on Raspberry Pi



Maker
Brian Jepson
Publishing director for Raspberry Pi Press, and expert on classic computers of the 1980s.

rpimag.co



Warning!
Video game copyright
Video game files are protected by copyright law. Be sure to use ROM files that have been released with the owner's blessing, or modern homebrew games designed to be shared. There are lots of legal options.
rpimag.co/legalroms

The Apple II was not the first personal or home computer, but it was part of the iconic 'trinity' of computers that came out in 1977: Apple II, Commodore PET, and TRS-80. The Apple II wasn't as capable a gaming computer as some of the other 8-bit systems of the era, but it was designed with games in mind. In designing the Apple II, Steve Wozniak took inspiration from his work on the Breakout arcade game during his time at Atari. It's why the Apple II has colour graphics, support for paddle controllers, and even audio.

Apple II audio was quite limited: anything more complex than a beep was quite CPU-intensive. Add-on audio boards, such as the Mockingboard, became popular and enjoyed support from a number of game developers.

The Apple II included a version of the BASIC language that only supported integer numeric data types: by eschewing floating-point numbers, Wozniak's Integer BASIC, as it was called, processed numbers extremely fast. This made it less than ideal for certain programming tasks, but it was fantastic for video games. Integer BASIC was supplied in the system ROM image of the Apple II, while a more full-featured BASIC, Applesoft BASIC, was originally supplied on cassette tape (it would be baked into the system ROM beginning with the Apple II Plus, though).

In designing the Apple II, Steve Wozniak took inspiration from his work on the Breakout arcade game for Atari

Install an Apple II emulator

For Apple II emulation on Raspberry Pi, we are using the LinApple emulator (**rpimag.co/linapple**). It's a fork of the AppleWin emulator which was originally created by Mike O'Brien. David Baucum (maxolasersquad) and several other

contributors got it running on Linux. You may also want to have a look at audetto's fork of the AppleWin emulator (**rpimag.co/audetto**). Because audetto's fork is regularly synchronised with the original AppleWin GitHub repository, it tracks AppleWin features more closely.

LinApple can emulate Apple II sound cards such as the Mockingboard (enabled by default in the configuration file). LinApple is not available through the Raspberry Pi OS apt repository, so you'll need to compile it yourself. Fortunately, it's quite easy to do this. First, open a Terminal window. This is where you'll install dependencies, download the source code, and compile, then install it.

In the Terminal window, run these commands to update your operating system packages and to install the dependencies needed to compile LinApple. You may be prompted for your password when you run the commands that use sudo.

```
$ sudo apt update
$ sudo apt -y full-upgrade
$ sudo apt -y install git g++ cmake
libzip-dev libsdl3-dev libsdl3-image-dev
libcurl4-openssl-dev zlib1g-dev imagemagick
```

Now you're ready to clone the emulator repository. If you'd prefer not to check the source code out into the root of your home directory, create a directory and change into it before you clone it. For example, `mkdir ~/src` would create a **src** directory under your home directory, and `cd ~/src` would change into it. You can clone the repository with this command:

```
$ git clone
https://github.com/linappleii/linapple.git
```

After you've cloned it, `cd` into the **linapple** folder and build the project:

```
$ cd linapple
$ cmake -B build -DBUILD_TESTING=OFF
$ cmake --build build -j4
```

The first `cmake` command prepares the source code, and the second compiles it. The `-j4` argument tells cmake to use as many as four CPU cores. Once it's compiled successfully, install it under the **/usr/local** folder with:

```
$ sudo cmake --install build
```



▲ **Figure 1:** LinApple is up and running

Now you can run the emulator from any Terminal window with the command `linapple`. But before you do that, let's customise some options. First, copy the default configuration file and sample disk image to your home directory. You'll need to create a couple of directories before you copy files into them:

```
$ mkdir -p ~/.config/linapple/
$ cp /usr/local/etc/linapple/linapple.conf
~/.config/linapple/
$ mkdir -p ~/.linapple/disks/
$ cp /usr/local/share/linapple/Master.dsk
~/.linapple/disks/
```

Next, edit your configuration file with your favourite editor, for example using `nano ~/.config/linapple/linapple.conf`, and make these changes:

1. Find the entry that reads `Joystick 0 = 2` and change `2` to `4`. This lets you use your mouse as a joystick, which you'll either love or hate (give it a try in Brickout later in the article!). The configuration file describes the other options you can choose.
2. Locate the `Screen factor = 1.0` entry and change `1.0` to `2.5`. This will make the emulator window a bit bigger.

Save the config file and exit your editor. If you're using the nano editor, press **CTRL+X**, answer **Y** for yes, then press **ENTER** to confirm the file name. Let's test LinApple and make sure everything is working. In the Terminal window, run the following command:

```
$ linapple --autoboot --d1
~/.linapple/disks/Master.dsk
```

LinApple should open and display the screen shown in **Figure 1**. If you find that the emulator has taken over control of your mouse, press **F1** to release it (this will display the help

screen, but you can press **F1** again to turn it off). You can toggle full-screen mode with the **F6** key.

Here's a little BASIC program you can try running. You can either type it into the emulator, or you can find the program in the **ch03/AppleIIBasicPrograms.dsk** file in the GitHub repository (**rpimag.co/retro-guide-git**). To use the disk image, launch the emulator with `linapple --autoboot --d1 AppleIIBasicPrograms.dsk`, type `LOAD BLOCKMOVE` at the] prompt and press **ENTER**, then type `RUN` (and press **ENTER**).

```
  10 HGR
  20 X = 140: Y = 80: D = 3: REM INITIALISE
  30 HCOLOR = 3: GOSUB 1000: REM DRAW CUBE
  40 REM WAIT FOR AND READ KEYPRESS
  50 K = PEEK (- 16384): IF K < 128 THEN 40
  60 REM CLEAR STROBE
  70 POKE - 16368, 0: K$ = CHR$ (K - 128)
  80 HCOLOR = 0: GOSUB 1000: REM ERASE CUBE
  90 IF K$ = "W" THEN Y = Y - D
  100 IF K$ = "S" THEN Y = Y + D
  110 IF K$ = "A" THEN X = X - D
  120 IF K$ = "D" THEN X = X + D
  130 IF X < 0 THEN X = 0
  140 IF X > 272 THEN X = 272
  150 IF Y < 0 THEN Y = 0
  160 IF Y > 184 THEN Y = 184
  170 GOTO 30 : REM BACK TO TOP OF LOOP
  1000 REM DRAW CUBE IN CURRENT HCOLOR
  1010 HPLOT X, Y TO X + 8, Y TO X + 8, Y + 8 TO
X, Y + 8 TO X, Y
  1020 RETURN
```

This will draw a small square on screen. Use the **W**, **A**, **S**, and **D** keys to move it around the screen. The program is quite simple. It first defines the starting X and Y position, as well as the number of pixels the square should move each time (the variable `D`). After that, it sets the drawing colour to 3 and calls the subroutine to draw the cube. The program then waits for a key press, and when it has a key press, it clears the *keyboard strobe*, which is a bit of hardware that indicates when a key press has been read.

After that, the program converts the integer key press value (`K`) into a string (`K$`) that contains the character typed. From there, it updates the X and Y position, makes sure the square hasn't run off-screen, and returns to the top of the loop (line 30).

After you type in the program, type `RUN` followed by **ENTER**. Try moving the square with the **WASD** keys, and press **CTRL+C** when you're done to exit the program. **Figure 2** shows the program in action. Notice that the vertical and horizontal lines are different colours, and that the vertical lines change colour as you move. This is due to the way Apple II graphics work: the Apple II only generates a monochrome display signal, but it uses some signalling tricks to generate colour artefacts. Game programmers could manipulate these artefacts to create colour combinations they wanted.

The Raspberry Pi Guide to Retro Gaming

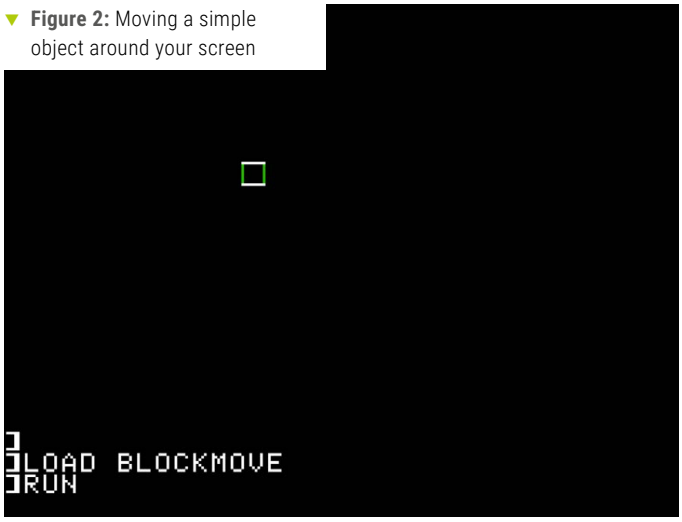
This tutorial is part of a series of emulation guides in *The Raspberry Pi Guide to Retro Gaming*:
rpimag.co/retroguide



After you run a graphical program, you'll only have a small text area below the graphics mode portion of the emulator screen. You can type the `TEXT` command and press **ENTER** to return to text-only mode. You'll also find these commands helpful:

1. `LIST` – Lists all the lines of the current BASIC program.
2. `RUN` – Runs the current program.
3. `NEW` – Erases the current program and lets you start afresh.
4. `CATALOG` – Lists all files on the disk image.
5. `SAVE NAME` – Saves the current BASIC program under whatever file name you specify.
6. `LOAD NAME` – Loads the program with the specified file name.
7. `CTRL+C` – Use this key combination to stop a running BASIC program.

▼ **Figure 2:** Moving a simple object around your screen



Apart from `CATALOG`, these commands only apply to BASIC programs. Many games are binary executable files that don't include BASIC source.

You can find some fun type-in programs in Lee Fastenau's collection of short programs: **apple2programs.com/programs**. Lee's programs are intentionally dense, as most are designed to be entered on one or two lines of code. So, take your time typing them in!

Game spotlight: Little Brick Out

The **rpimag.co/a2intbasic** GitHub repository includes the assembly language source for Integer BASIC, along with some ready-to-run disk images on the repository's release page. The disk image includes the unofficial Breakout clone that Woz created for the Apple II, Little Brick Out.

Head on over to the repository's Releases page (**rpimag.co/integerbasic**) and look for the file named **intbasic_system.po**. Download it to your Raspberry Pi and quit LinApple if it's already running. Launch LinApple with:

```
$ linapple --autoboot --d1
~/Downloads/intbasic_system.po
```

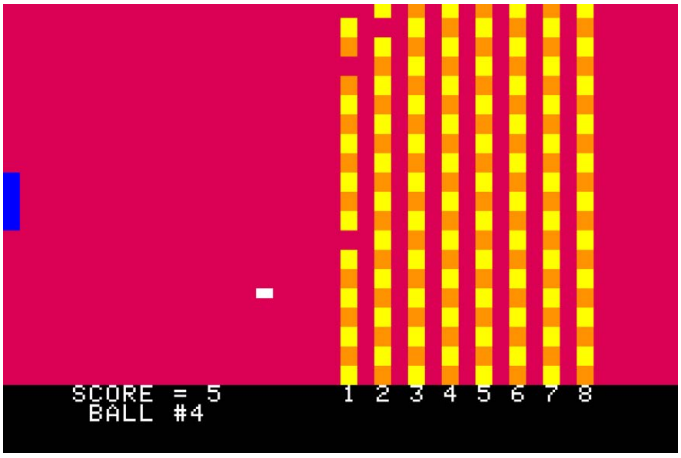
After the emulator is up and running, type `CATALOG` and press **ENTER**. Make sure you see a file named **WOZ.BREAKOUT**. To run it, type `RUN WOZ.BREAKOUT` and press **ENTER**. The program pauses a bit at startup, then asks your name and whether you want the standard colours. Once the game starts, you'll notice that it's been turned on its side, a subtle twist on the original game. If you configured your mouse as a joystick as suggested earlier, you can move the mouse to control the on-screen paddle. **Figure 3** shows the game in action.

More great games

There is no shortage of great games for the Apple II. Eamon is a text-based role-playing game system that is, and always has been, in the public domain. Its creator, Donald Brown, launched it on the Apple II initially, but it was ported to many other systems. The game's interface is similar to that of interactive fiction like Colossal Cave Adventure or Zork, but it also has role-playing elements such as character statistics, spells, and to-hit probabilities. Members of the Eamon community have created hundreds of adventures, which are available from the Eamon Adventurer's Guild at **rpimag.co/eamonlist**. You can download any disk image by right-clicking a link in the Download DOS 3.3 column and saving it to your Raspberry Pi. Load it as you would any other emulator disk image. We suggest you start out with the original adventure, Main Hall & Beginners Cave.

> *There is no shortage of great games for the Apple II*

There are also a number of new games that have been made for the Apple II. Froggo (**rpimag.co/froggo**) from Laser Mego is a great retro-style game inspired by Frogger and Crossy Road. Crow Cousins offer utilities, tools, and some fun games for Apple II (**rpimag.co/crowcousins**), including The Orion Trail, a space-based reinterpretation of the classic Oregon Trail. It's also worth keeping an eye on Indie Retro News, which occasionally features modern Apple II games: **rpimag.co/indieretroA2**.



▲ **Figure 3:** Playing a piece of computing history!

LinApple's in-game menu

You don't have to quit and restart LinApple each time you want to load a new disk image. Press **F3** to bring up the floppy disk chooser and navigate the text-based file system menu to wherever you've saved a disk image. Select the image, press **ENTER**, then press **CTRL+F2** to reboot the emulator and load the disk.

Raspberry Pi operating systems

Transform your Raspberry Pi experience with a different OS

By **Phil King**

While Raspberry Pi OS is a great all-round operating system that's tailor-made for our favourite single-board computer, there's no shortage of alternatives available. It's also very easy to try them out, simply by swapping out the microSD card (or USB stick) for one with a new OS written to it and rebooting. In this feature, we take a look at some of the best OSes out there, suitable for work, play, the home, and other uses.

> *It's very easy to try them out, simply by swapping out the microSD card*

Work

The best OSes for productivity and coding

Ubuntu Desktop

Available: Raspberry Pi Imager, ubuntu.com
Works on: Raspberry Pi 4/400, 5/500/500+ with 4GB+
Best for: General-purpose desktop Linux

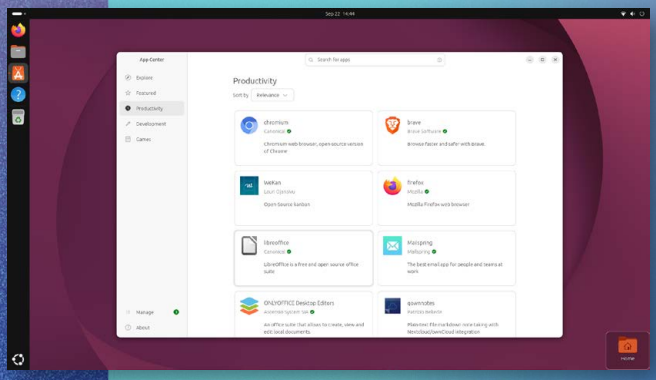
One of the most popular flavours of Linux, Ubuntu is used on millions of PCs around the world. Based on Debian, like Raspberry Pi OS, it benefits from a huge software library.

While the pre-installed applications are sparse, with only the Firefox web browser and a few utilities, you can install more via the App Center. Check its Productivity section for the excellent LibreOffice suite, Mailspring or Thunderbird email client, GNOME Calendar, Evince document viewer, WeKan kanban board, among many others. You'll also find useful coding tools etc. in the Development section.

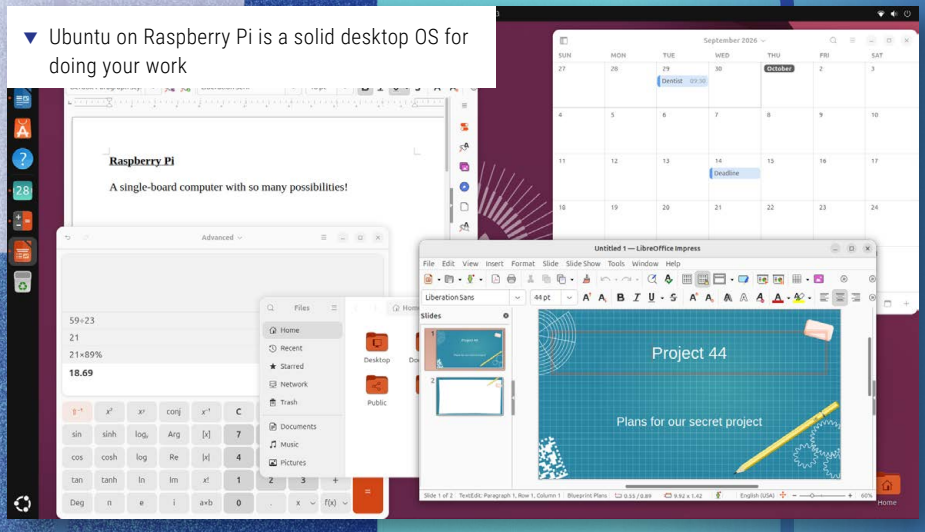
Alternatively, it can be a lot faster to install apps directly via the Terminal, using snap, flatpak, or apt – the last of which can be used to install the lgpio Python library to access Raspberry Pi's GPIO pins.

The GNOME (v49) desktop interface is slick, with lots of appearance settings to tweak it, but you always have the option of installing and switching to a different one like Kubuntu or Lubuntu if you prefer.

▼ You can use the App Center to install productivity and development applications



▼ Ubuntu on Raspberry Pi is a solid desktop OS for doing your work



Manjaro

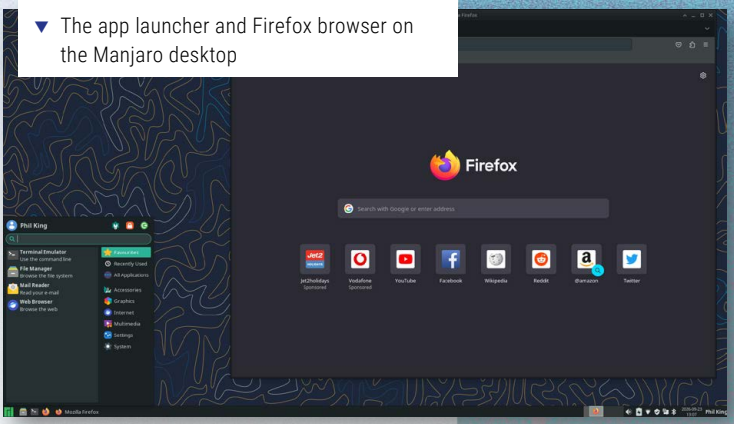
Available: manjaro.org
Works on: Raspberry Pi 3B/3B+, 4/400, Zero 2 W
Best for: Bleeding-edge software

Manjaro brings the power of the Arch Linux ecosystem to Raspberry Pi while making it a lot easier to set up and use. Since it receives rolling releases, it's always kept up to date without needing a full OS reinstall.

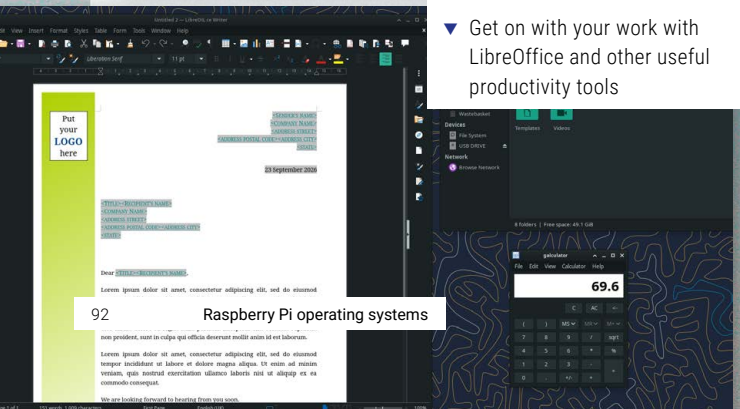
One downside is that Manjaro is not yet available for Raspberry Pi 5. Still, there's a choice of six editions to download from the website with different desktop environments: GNOME, KDE Plasma, MATE, Minimal, SwayWM, and Xfce.

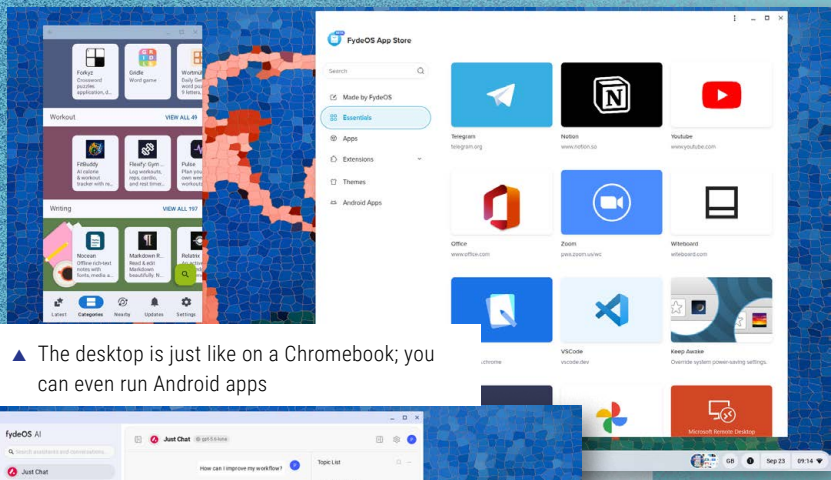
Upon bootup, you'll find a small selection of applications and tools, but you can easily install others – such as the LibreOffice suite – using the Add/Remove Software app. Or do so in the Terminal using the Pacman package manager (with the pamac command) or Yay. The latter gives you access to the Arch User Repository (AUR), a vast, community-driven collection of PKGBUILD scripts – recipes for simpler software installation.

▼ The app launcher and Firefox browser on the Manjaro desktop

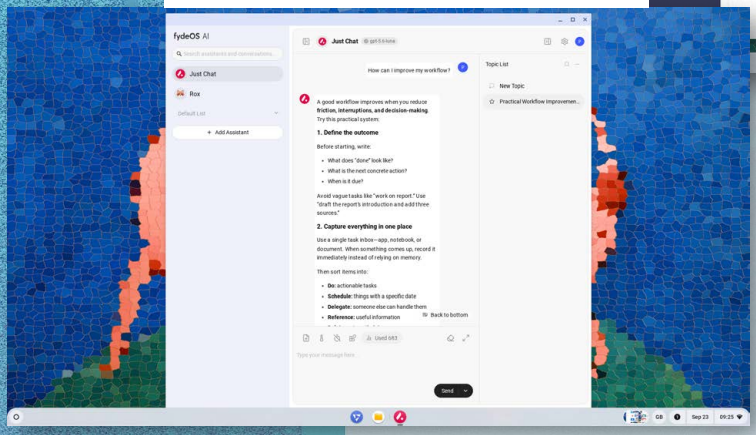


▼ Get on with your work with LibreOffice and other useful productivity tools





▲ The desktop is just like on a Chromebook; you can even run Android apps



▲ FydeOS even includes an AI assistant to chat with

FydeOS

Available: fydeos.io
Works on: Raspberry Pi 4/400, 5/500/500+ with 4GB+
Best for: Chromebook-style cloud computing

FydeOS is designed to bring a Chromebook-style experience to any computer, including Raspberry Pi. You're strongly recommended to install it to an SSD, though, as it tends to lag and crash a lot on a microSD card.

Upon bootup, a welcome wizard takes you through a few settings before prompting you to sign in – with a new FydeOS account or a Google one for syncing files between devices using browser-based apps, although you can switch to a more limited local account instead.

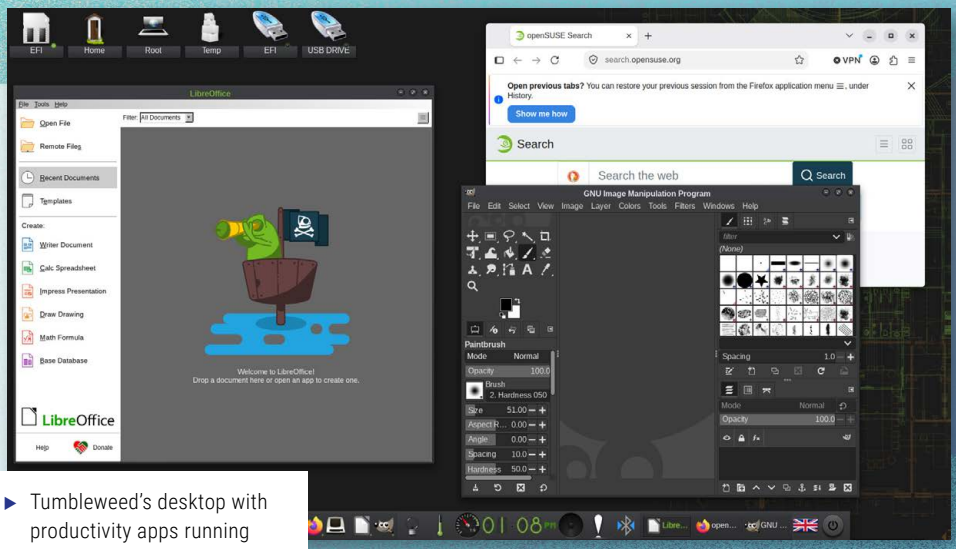
The desktop interface is just like on a Chromebook, with the calendar and settings at the bottom right and the app launcher at the bottom left – click it to reveal installed apps, including an AI assistant. More browser-based apps/extensions can be added via the FydeOS Store. It's even possible to run Android apps using the F-Droid app. You can also set up a Linux development environment.

openSUSE Tumbleweed

Available: opensuse.org
Works on: Raspberry Pi 4/400, 5/500/500+
Best for: Latest software and bulletproof stability

Tumbleweed is a rolling-release distro by openSUSE. If something goes awry, you can just roll back to a previous 'snapshot' of the system.

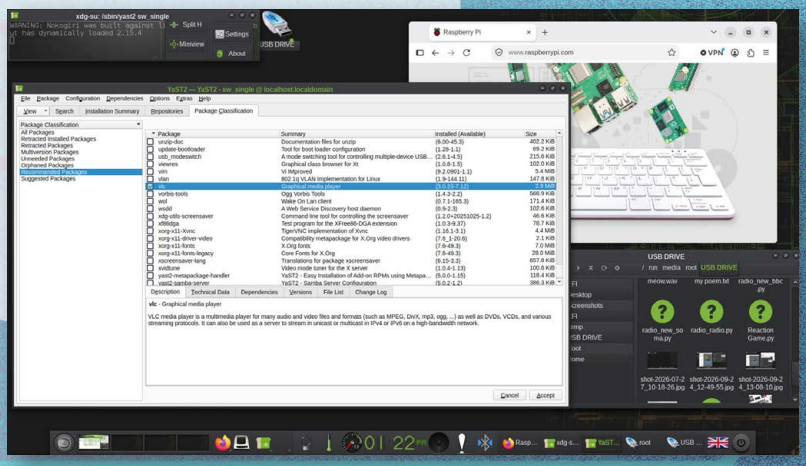
To get started, visit the downloads page at **rpimag.co/tumbleweed-dl**, search for 'raspberrypi.aarch64', and download a Tumbleweed .raw.xz file – numerous versions are available, with different desktops (or none).



▶ Tumbleweed's desktop with productivity apps running

Upon bootup, you're presented with a bootloader menu where you can choose a snapshot. You get a choice of a few settings before the login window appears: use the default 'root' username and 'linux' password (you should change it on use).

A lot of apps available will be familiar. You can use the YaST GUI package manager to install software, including lots of productivity and coding tools. Or use zypper to install packages in the Terminology terminal. We found we needed to use the terminal (with `sudo nmtui`) to open a network manager to connect to Wi-Fi.



▼ Use YaST to install new software, or add and update it via the Terminology terminal

More work options

DietPi: A slimmed-down OS that's ideal for running lightweight, optimised servers and projects.

Bass OS: Android based (but Google free) and modular, it features hundreds of add-ons.

Armbian: Lightweight Arm-based version of Debian that can run on an array of single-board computers.

Alpine: Uses BusyBox instead of GNU/Linux utilities and is so tiny, it can run entirely in RAM.

Twister OS: Easily switch desktops and use emulators to run selected Windows apps.

Ultramarine: Brings the Fedora Linux ecosystem to Raspberry Pi, if you can manage to get it working.

Play

All work and no play? Let's fix that...

LibreELEC

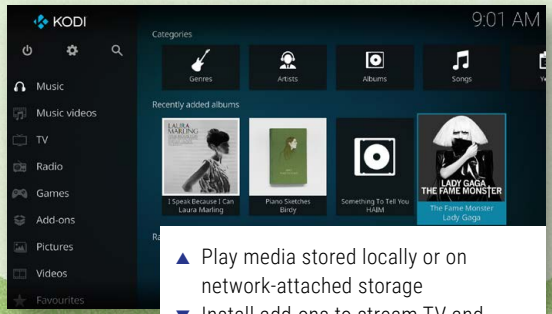
Available: Raspberry Pi Imager, libreelec.tv
Works on: Raspberry Pi 2, 3B/3B+, 4/400, 5/500/500+, Zero 2 W
Best for: Dedicated media playback

Based around the open-source Kodi software, the lightweight LibreELEC strips away all background OS bloat, allowing it to boot faster, use less RAM, and dedicate all hardware resources strictly to media playback.

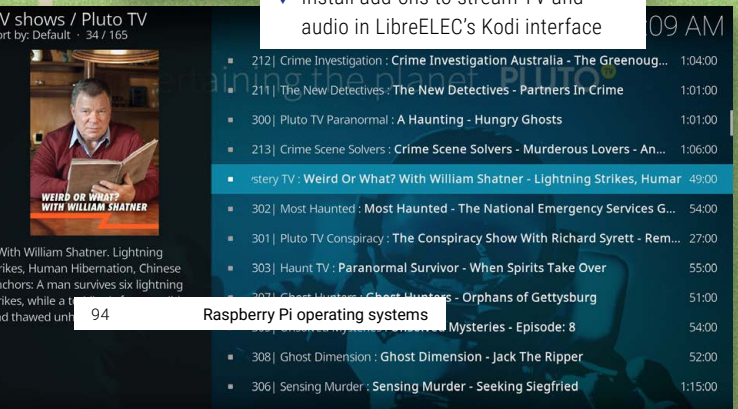
A setup wizard helps you to connect to your Wi-Fi network and enable SSH remote access and Samba sharing. You're then in the standard Kodi user interface – which you can change to a different skin if you like.

Alongside the usual Kodi add-ons, there are dedicated LibreELEC ones including various decoders, tools, PVR clients, and retro game emulators. There's also a handy LibreELEC settings menu for managing updates etc.

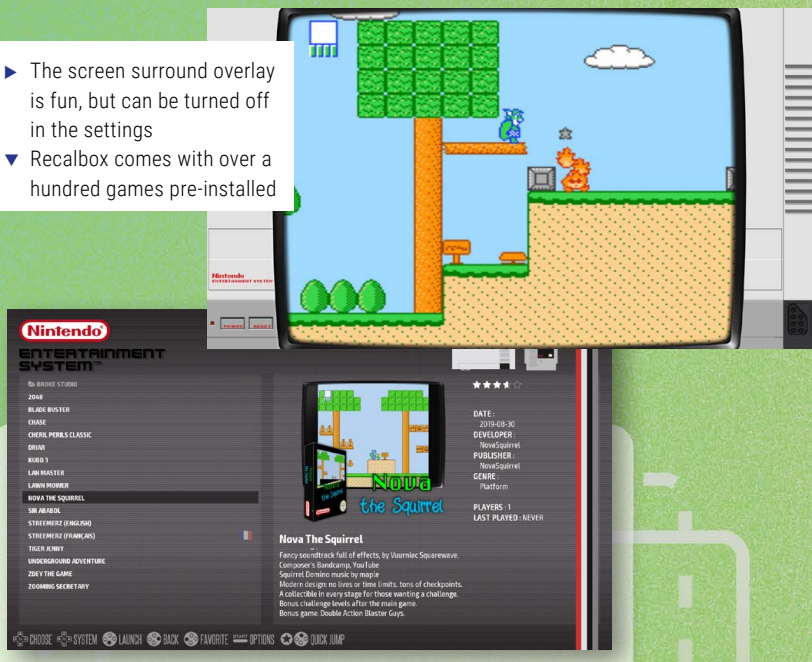
For the ultimate media centre experience, team LibreELEC with a specialist media case for your Raspberry Pi and an IR remote.



▲ Play media stored locally or on network-attached storage
▼ Install add-ons to stream TV and audio in LibreELEC's Kodi interface



▶ The screen surround overlay is fun, but can be turned off in the settings
▼ Recalbox comes with over a hundred games pre-installed



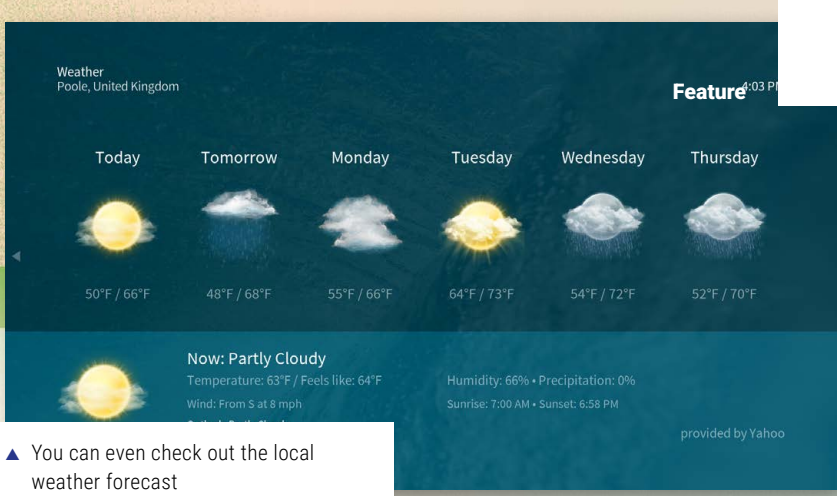
Recalbox

Available: Raspberry Pi Imager, recalbox.com
Works on: Raspberry Pi 1, 2, 3B/3B+, 4/400, 5/500/500+, Zero/Zero 2 W
Best for: Retro gaming and more

There are several ways to enjoy retro gaming, but one of the most accomplished is Recalbox. You can get it for any Raspberry Pi model, although 32-bit ones are limited to an earlier OS version. Also, the more powerful the computer, the more recent games systems it can emulate.

It works out of the box with minimal setup. Most game controllers are recognised automatically. Recalbox even comes with over 100 homebrew and indie games pre-installed. You can add new ones to the relevant system folders via USB or over the network. There's even a web interface to monitor and configure the system.

A nice bonus is that Recalbox also has Kodi pre-installed, so it doubles as a media centre to play video and audio content.



▲ You can even check out the local weather forecast

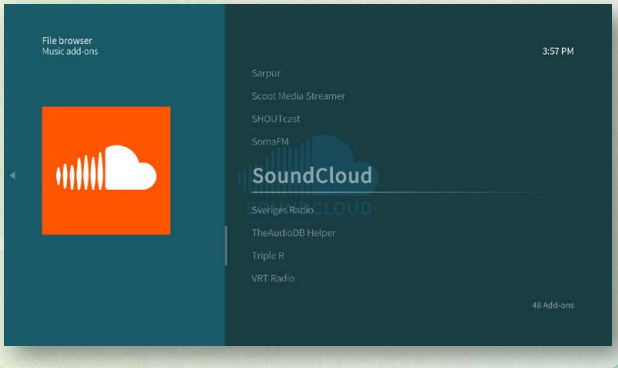
OSMC

Available: Raspberry Pi Imager, osmc.tv
Works on: Raspberry Pi 1, 2, 3B/3B+, 4/400, Zero/Zero 2 W
Best for: Media playback and more

Built on top of Debian, OSMC (Open Source Media Center) is a more flexible option than LibreELEC. You can access the Linux command line and use the standard apt package manager to install extra software.

By default, it boots into the standard Kodi interface, as used in LibreELEC. So you can play local video/audio files, along with those shared via NFS/SMB on network-attached storage, and install (unofficial) add-ons to stream content from various services.

One drawback is that while OSMC supports most Raspberry Pi models, it's not yet available for Raspberry Pi 5. So if you have one, for now you'd be better off using Raspberry Pi OS and installing Kodi on top of that.



◀ There's a wide range of add-ons to choose from to use various services

Volumio

Available: Raspberry Pi Imager, volumio.com
Works on: Raspberry Pi 1, 2, 3B/3B+, 4/400, Zero/Zero 2 W
Best for: Hi-fi music playback

Running headlessly, Volumio turns your Raspberry Pi into a high-end music player and streamer – especially if you equip it with a DAC board to output ultra-high-quality (up to 24-bit) audio to a hi-fi system.

Upon first boot, it creates a Wi-Fi hotspot so you can connect from another device to set everything up. You can then log into its slick web interface – or use the Volumio Android or iOS app – to control everything.

As well as playing music stored locally, you can connect it to shared network drives. Plug-ins enable you to stream music from services such as Spotify, YouTube Music, Pandora, SoundCloud, and Bandcamp. You can also choose from a wide range of internet radio stations.

Upgrading to a Premium account unlocks extra features such as casting to and grouping playback devices/speakers, and the 'play here' function.

More play options

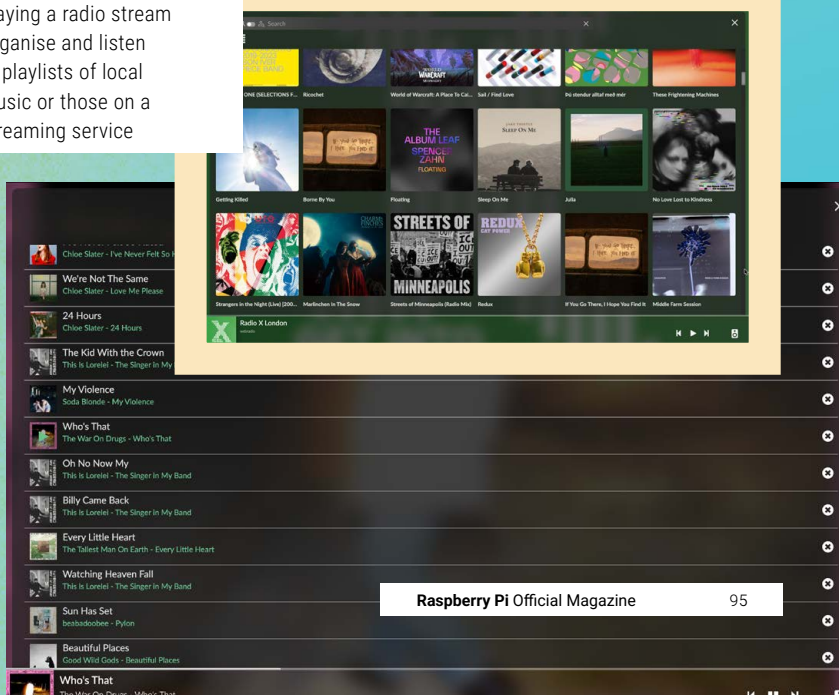
RetroPie: It can emulate a broad array of classic consoles and computers, while offering advanced settings for fine-tuning your gaming.

Batocera.linux: A cross-platform games emulation OS that supports a wide range of systems and has a handy web manager.

DOSbian: It boots straight into a DOS operating system, enabling vintage PC emulation – it can also run Windows 3.1, 95, and 98.

RISC OS Pi: Relive the 1990s heyday of the Acorn Archimedes. Check out our tutorial series starting in issue 153 (**rpimag.co/153**).

▶ Browsing a selection of albums on Spotify while playing a radio stream
▼ Organise and listen to playlists of local music or those on a streaming service



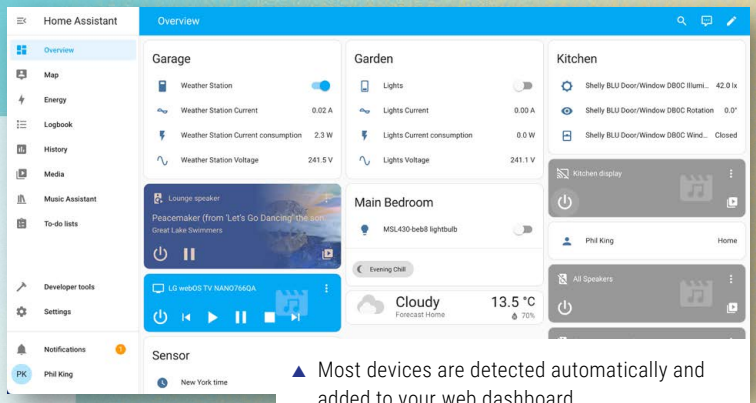
Home

Operating systems for a smarter home

Home Assistant

Available: Raspberry Pi Imager, home-assistant.io
Works on: Raspberry Pi 4/400, 5/500/500+
Best for: Running a smart home

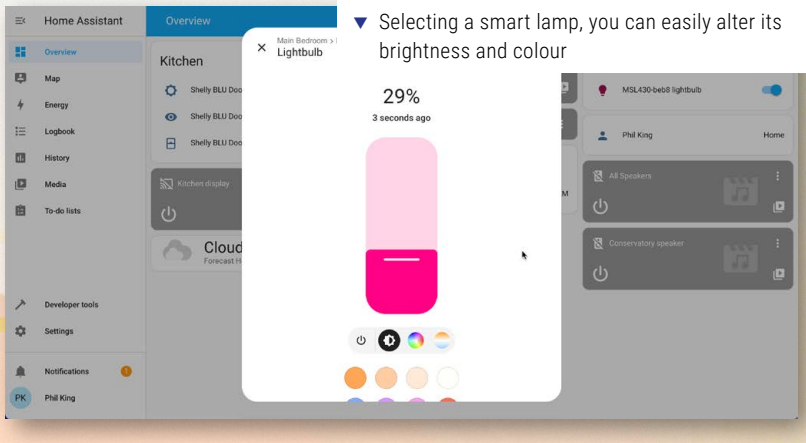
If you want your Raspberry Pi to monitor and control smart home devices, the Home Assistant operating system is the most popular way. This free, open-source home automation platform is easy to get started with, packed with features, privacy focused (it stores your data locally), and has a strong community creating a vast array of custom add-ons.



▲ Most devices are detected automatically and added to your web dashboard

While the OS can run on Raspberry Pi 3 and Zero 2 W, a Raspberry Pi 4 or 5 with at least 2GB of RAM is recommended for smooth performance. Installing it and setting it up is a simple process, as covered in our issue 154 feature (**rpimag.co/154**). It supports a wide range of smart devices and will detect most of them on your network automatically; others will work by adding the required 'integrations'.

You can then view all your smart home details on a web dashboard and set up automations and 'scenes' (to set a state for a group of devices). It's also possible to create custom scripts, as covered in Ben Everard's tutorial in issue 167 (**rpimag.co/167**) in which he turned a Pico W into a smart sensor.



▼ Selecting a smart lamp, you can easily alter its brightness and colour

More home options

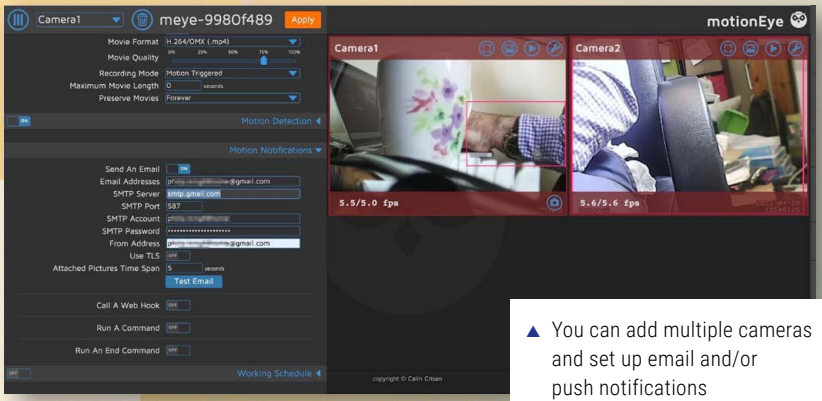
Gladys Assistant: An alternative to Home Assistant and openHAB, this smart home hub is privacy focused and offers a streamlined user experience.

Homebridge: Turns Raspberry Pi into a smart home bridge to integrate non-HomeKit compatible smart devices into the Apple Home app.

OpenCCU: This one turns Raspberry Pi into a local, cloud-free smart home central control unit for eQ-3's Homematic and Homematic IP hardware.

nymea: A privacy-focused OS that turns Raspberry Pi into a dedicated local automation server that requires no internet access.

evcc: An extensible, open-source home energy management application designed primarily to control electric vehicle (EV) smart charging.



▲ You can add multiple cameras and set up email and/or push notifications

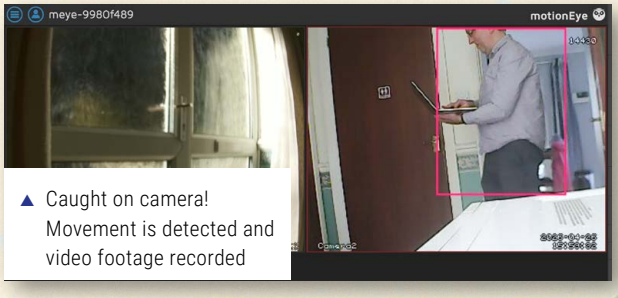
motionEyeOS

Available: motioneyeos.com
Works on: Raspberry Pi 3B/3B+, 4/400, Zero/Zero 2 W
Best for: Video surveillance

As well as making your home smarter, Raspberry Pi can help make it more secure with motionEyeOS. This free and open-source video surveillance OS enables you to turn one or more Raspberry Pi computers and cameras into a cost-effective CCTV system that's highly customisable. You can use official Raspberry Pi Camera Modules, USB webcams, and IP cameras – or a combination of them.

Before booting up the OS, you can preconfigure the settings for your wireless network – or just plug it into your router with an Ethernet cable (recommended). Once it's up and running, you can use another computer to log into the motionEyeOS web interface to configure everything, including adding one or more cameras and adjusting their settings.

The OS uses motion detection based on changes in the camera views and can send you alerts via email. It will also record video of the event, stored locally or in the cloud. By setting up port forwarding in your router settings, you can access the system from anywhere in the world to keep an eye on things at home.



▲ Caught on camera! Movement is detected and video footage recorded

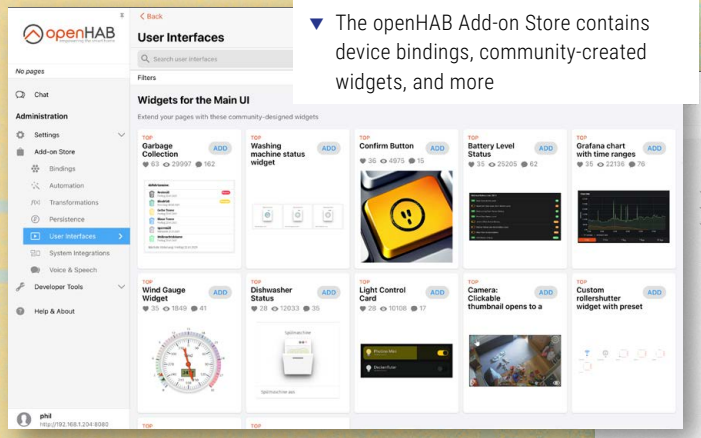
openHABian

Available: Raspberry Pi Imager, openhab.org
Works on: Raspberry Pi 4/400, 5/500/500+
Best for: Complex smart home setups

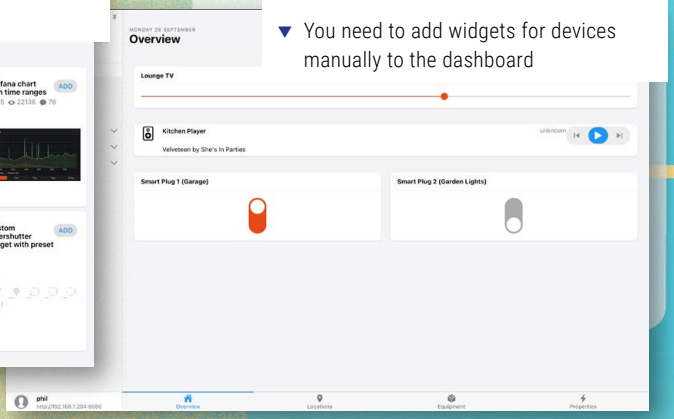
The main alternative to Home Assistant for a smart home setup is openHAB. It tends to be popular with users who have a sophisticated smart home setup and want finer control and the ability to run more complex tasks. It's not as beginner-friendly as Home Assistant and you do need to be prepared to learn the ropes to figure out how its powerful rules engine works. On the plus side, it does offer great stability and long-term reliability.

The openHABian OS is a Debian-based way to set up your openHAB with Raspberry Pi. Like Home Assistant, while there's a version that will run on 32-bit models, you'll get a smoother experience on a Raspberry Pi 4 or 5 with at least 2GB of RAM.

As soon as it's up and running, you can view its web-based user interface, where you're prompted to create your user account and install recommended add-ons such as software bindings to integrate devices. It takes a while to set everything up. Once you get the hang of it, you can try creating your own advanced scripts using Blockly or text-based programming.



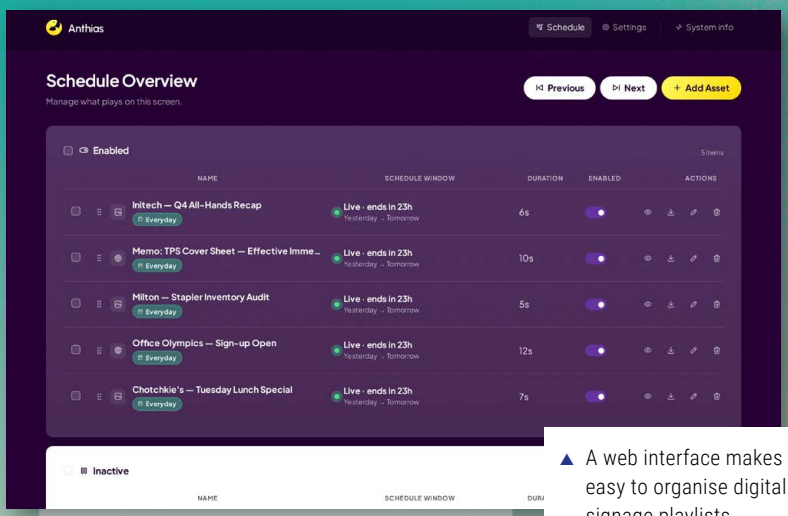
▼ The openHAB Add-on Store contains device bindings, community-created widgets, and more



▼ You need to add widgets for devices manually to the dashboard

Miscellaneous

Special-purpose operating systems



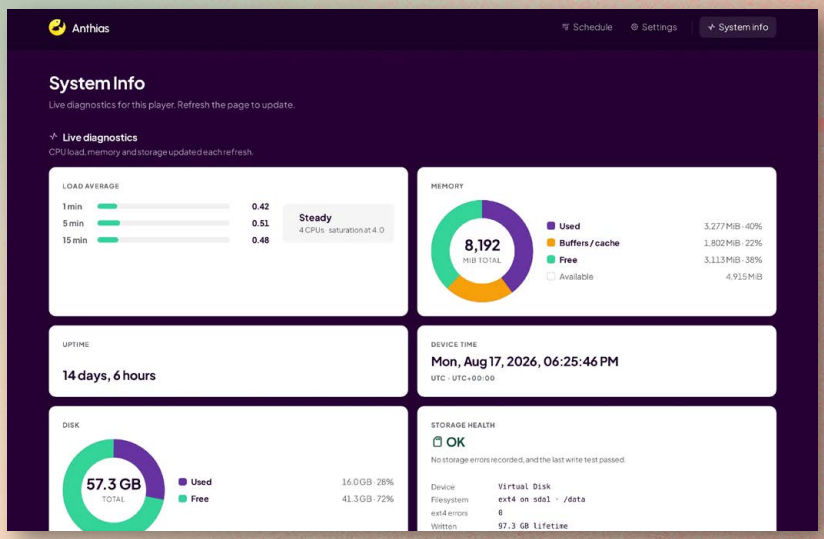
▲ A web interface makes it easy to organise digital signage playlists

▼ View the live diagnostics for your player

Anthias

Available: Raspberry Pi Imager, anthias.screenly.io
Works on: Raspberry Pi 2, 3B/3B+, 4/400, 5/500/500+, Zero/Zero 2 W
Best for: Digital signage

Created by the people behind the commercial digital signage platform Screenly, Anthias is free to use and ideal for a single-screen solution. Everything is stored locally in Raspberry Pi and you can manage the system via a web dashboard on any device on the network, so you can schedule what is shown on the connected screen using playlists.



Other options

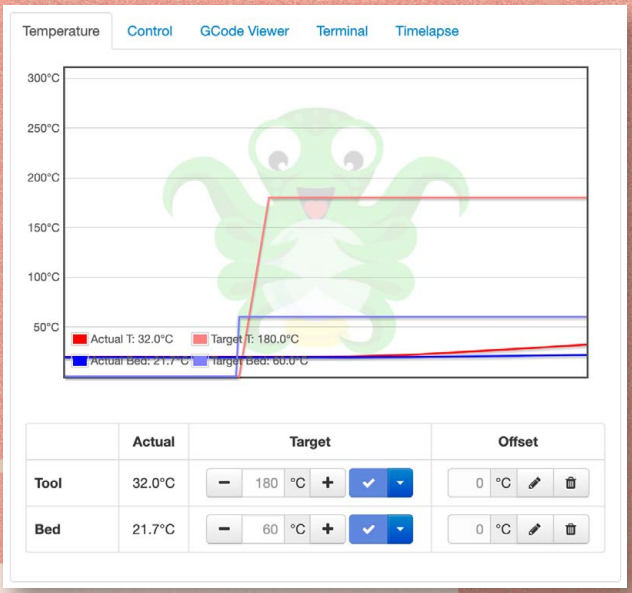
Clea OS: Designed for Raspberry Pi-based industrial devices, it comes in 'Embedded' (full OS) and 'Things' (minimal, for IoT nodes) versions.

LoLaOS: Stream demanding Windows applications or heavy AAA games directly to Raspberry Pi from a remote host PC or cloud workspace.

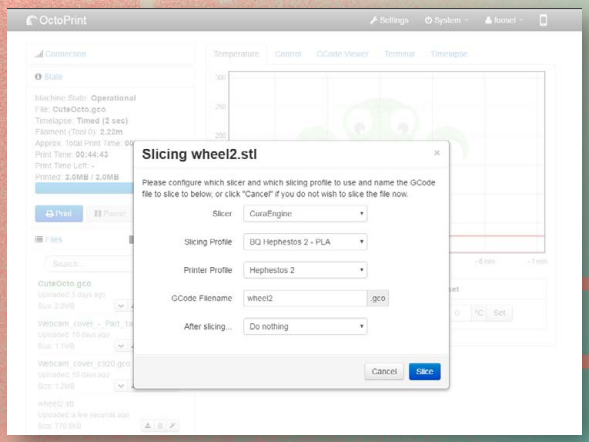
Falcon Player: Control complex light shows using addressable RGB LEDs. Ideal for holiday lights or landscape and ambience lighting.

SatNOGS: Link your radio ground station to the global SatNOGS network so that you can track low-Earth Orbit satellites and download their data.

eBlockerOS: An alternative to Pi-Hole, it can be used to block internet adverts, trackers, and malware for every device in your home.



◀ Monitor your 3D printer and upload G-code files



▲ You can choose a slicer and format to prepare your files for printing

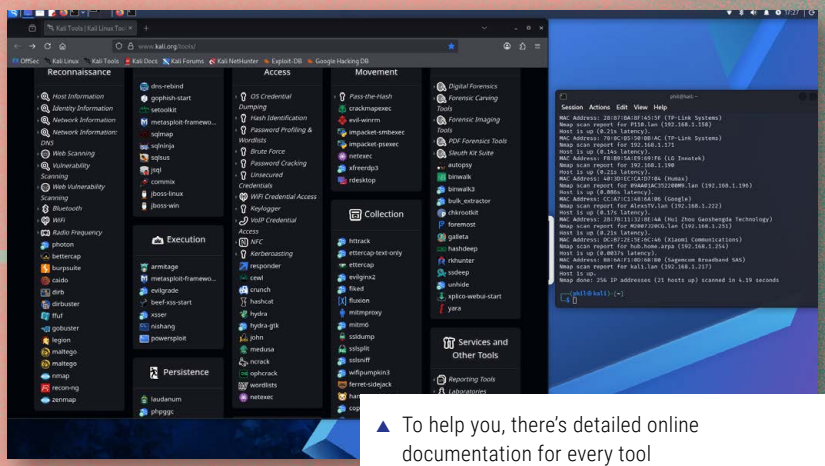
OctoPi

Available: Raspberry Pi Imager, octoprint.org
Works on: Raspberry Pi 1, 2, 3B/3B+, 4/400, 5/500/500+, Zero/Zero 2W
Best for: 3D printer control

The OctoPi OS comes with the OctoPrint software pre-installed, enabling you to turn a USB-connected standard 3D printer into a network-connected machine for wireless control and monitoring from a web interface on any device. You can even use a Raspberry Pi Camera Module or USB webcam to get a live view of what the printer is doing.



▲ Find Kali's many tools within the app menu's various categories



▲ To help you, there's detailed online documentation for every tool

Kali

Available: Raspberry Pi Imager, kali.org
Works on: Raspberry Pi 2, 3B/3B+, 4/400, 5/500/500+, Zero 2 W
Best for: Security testing

Based on Debian, this specialist OS turns your Raspberry Pi into a portable low-power penetration testing and ethical hacking rig, enabling you to tackle a variety of cybersecurity tasks. It comes with a huge array of pre-installed tools such as Nmap (for network mapping), Wireshark (data packet analysis), Metasploit Network (exploitation testing), and Burp Suite (web app testing).

ONLY THE **BEST**

Retro gaming kit

Turn your Raspberry Pi into a classic games machine

By **Phil King**

Following issue 169's cover feature on retro gaming (rpimag.co/169) and the launch of a brand new retro gaming book by Raspberry Pi (covered here), it's an ideal time to revisit the topic.

One of the most popular uses for a Raspberry Pi computer, it makes use of the latter's low cost, power, and ability to run emulation software to replicate a wide range of classic consoles and computers: from NES to PlayStation 2, and ZX Spectrum to Amiga. It's easy to get started by installing a specialist emulation OS such as Recalbox, RetroPie, Lakka, or Batocera.

While you can enjoy Raspberry Pi retro gaming with a standard keyboard and monitor setup, there are many ways to achieve a more authentic experience so you can pretend you're living in the past! Here, we take a look at a wireless gamepad, classic console-style case, a couple of ways to output analogue video (including one for Pico), and a mini arcade cabinet kit for two players. Insert coins to continue play...

Picade MAX

Pimoroni | £495 / $549 | pimoroni.com



▲ Having grown in size, this Picade model has controls for two players

These bartop arcade cabinet kits seem to be getting ever larger. First there was the original Picade with an 8-inch screen, then a 10-inch version, and now we have this super-size 19-inch MAX model – back in stock in the Pimoroni store – which adds controls for a second player.

The IPS display has a retro-friendly 5:4 aspect ratio and 1280 × 1024 resolution, with vibrant colour and wide viewing angles. The robust arcade joysticks and buttons are connected to an RP2040-powered USB controller board, resulting in very low latency for the inputs. A separate audio board (also powered by RP2040) outputs stereo sound to the cabinet speakers and, at last, there's a physical rotary volume control.

If you want to go further, you can add NVMe SSD storage, swap out the standard buttons for RGB LED-illuminated ones, and even add a Galactic Unicorn for marquee lighting. The controller and audio boards are also available separately, so you could use them (or the original Picade X HAT) as the basis for a custom-built cabinet.



Warning!

Legal ROMs

Please be cautious when finding ROMs for your retro setup that you are not downloading from a site hosting software illegally. There are lots of legal ROMs available, including many modern 'homebrew' titles developed for classic hardware.

rpimag.co/legalroms

Verdict

A bigger bartop arcade cabinet with plenty of upgrades.

The Raspberry Pi Guide to Retro Gaming

Raspberry Pi | £19.99 / $24.99 | raspberrypi.com

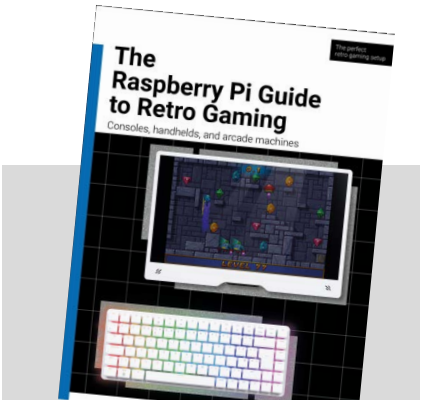
The easiest way to get started with retro gaming on your Raspberry Pi is by bagging a copy of this book. As well as a step-by-step guide to quickly create your own Raspberry Pi-based home console setup, it features advanced tips and tricks to help you perfect the emulation experience with hardware upscalers, dedicated video converters, and even vintage CRT displays for the ultimate in authenticity. The book also explains where to find legal ROMs, how to play your old floppies and CDs, and how to get games running from online stores.

In-depth tutorials cover building the arcade machines of your dreams, including a pocket handheld for gaming on the go, and emulating 8-, 16- and 32-bit classic computers such as the Apple II, Atari 400/800, BBC Micro, ZX Spectrum, C64, Amiga, Atari ST, and IBM PC.

For those readers who don't just want to play, there's a chapter on classic 8-bit BASIC programming, including creating a full ZX Spectrum game – now that takes me back!

Verdict

A comprehensive guide to retro gaming on Raspberry Pi.



▲ The official guide to Raspberry Pi retro gaming covers everything in detail

RGB Dual 2

Recalbox | From €72 / £62 / $83 | recalbox.com



Verdict

The easiest way to output analogue video to a TV or monitor.

◀ Enjoy retro gaming on an old TV for the authentic emulated experience

As well as providing a free and open source emulation system for Raspberry Pi, the Recalbox project has an online store in which it sells various bits of retro-related hardware. One of these is the RGB Dual 2, a Raspberry Pi 5-compatible board with analogue video outputs to connect it up to an old TV or monitor.

Fitting on Raspberry Pi 5's GPIO header, and matching the SBC's dimensions, it features four outputs: SCART, Composite RCA, VGA, and a 3.5mm Audio/Composite jack. So your connection options are well covered and it supports 4:3 and 16:9 TVs. Four DIP switches enable you to alter settings such as forcing 50Hz mode and enabling 31kHz/Multisync. It also has a built-in I2S digital to analogue converter (DAC) for high-quality analogue audio.

You can buy the RGB Dual 2 board on its own or in a kit with various extras such as a RecalTower stacking case. Either way, an Active Cooler is recommended to keep Raspberry Pi chilled as you enjoy pixel-perfect video.

Raspberry Pi Compatible Wireless Gamepad / Controller

The Pi Hut | £14 / $19 | thepihut.com



For console gaming, you'll want a suitable controller, whether it be a gamepad or arcade-style joystick. There are countless options available, in a wide variety of designs, typically based on classic console controllers.

Most of them can be plugged directly into a standard USB-A port. This one, however, is wireless (powered by three AAA batteries). Instead of using Bluetooth like some other gamepads, it has a USB receiver dongle with which it communicates via 2.4GHz wireless – no extra setup required. As with any other controller, you may need to map the buttons in your chosen emulation software.

Resembling a classic PlayStation DualShock gamepad, it features an Analog button to press to enable full analogue thumbstick control (otherwise they just acts as buttons). There's also a Turbo button for rapid fire – great for shoot-'em-ups. There's no noticeable wireless lag, either.

Verdict

A good value-for-money wireless controller.

▲ This wireless controller can also be used for robotics projects

Game 5Pi Retro Gaming Case for Raspberry Pi 5

52Pi / The Pi Hut | £17 / $23 | 52pi.com / thepihut.com

Verdict

As well as looking cool, this classic case has excellent cooling.



▲ Feel the eighties nostalgia with this classic console case

When playing emulated games, you might want to add a vintage-style case to enhance the illusion that you're back in the eighties, nineties, or whenever. There are many cases available resembling classic consoles such as the PlayStation, Dreamcast, SNES / Super Nintendo, N64, and Game Boy. This one harks even further back with its boxy NES-like design, albeit a lot smaller than the real thing.

To help prevent your Raspberry Pi 5 overheating, it features excellent cooling with two fans: one mounted on Raspberry Pi with thermal pads and a second 'exhaust fan' blowing air through the vents in the case.

The two-part case is easy to assemble and there are cutouts for micro HDMI and USB-C power ports, and the microSD card slot. A hinged flap can be lifted to reveal the USB and Ethernet ports. Unlike some cases we've seen, the two buttons on the front are non-functional, but there is a tiny one at the rear to press Raspberry Pi 5's own power button.

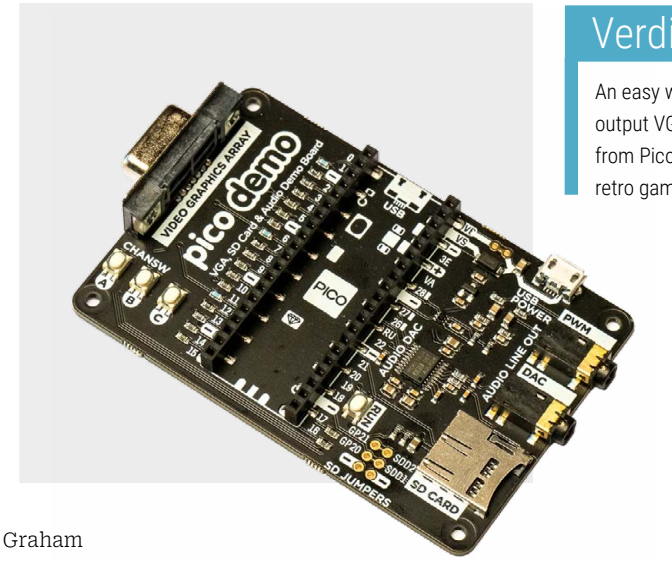
Pico VGA Demo Base

Pimoroni | £20 / $22 | pimoroni.com

While not as powerful as a Raspberry Pi computer, a Pico can emulate many classic 8-bit consoles and computers. The biggest issue is outputting video and audio. You could opt for Pimoroni's PicoVision (**rpimag.co/picovision**), which outputs hi-res video via an HDMI port. The downside is that it won't run standard Pico-based emulators unless you adapt them to use its hardware.

The Pico VGA Demo Base is a simpler option for Pico-powered emulation on a VGA monitor. Compatible with Pico and Pico 2 models, it has its own micro USB power port (freeing up Pico's own for USB devices), a microSD card slot for storage, and two 3.5mm audio jacks (for DAC and PWM).

You can flash the RP2040 version of the classic *Doom* (**rpimag.co/rp2040doom**) created by Raspberry Pi's own Graham Sanderson. Or use one of the Pico-based system emulators available – some, like pico-zxspectrum even have precompiled UF2 files for the VGA Demo Base; others can be compiled with an option to use the board's pin mapping etc.



▲ Yes, even a Pico can emulate classic computers and consoles

Verdict

An easy way to output VGA video from Pico for retro gaming.

8BITDO RETRO MECHANICAL KEYBOARD

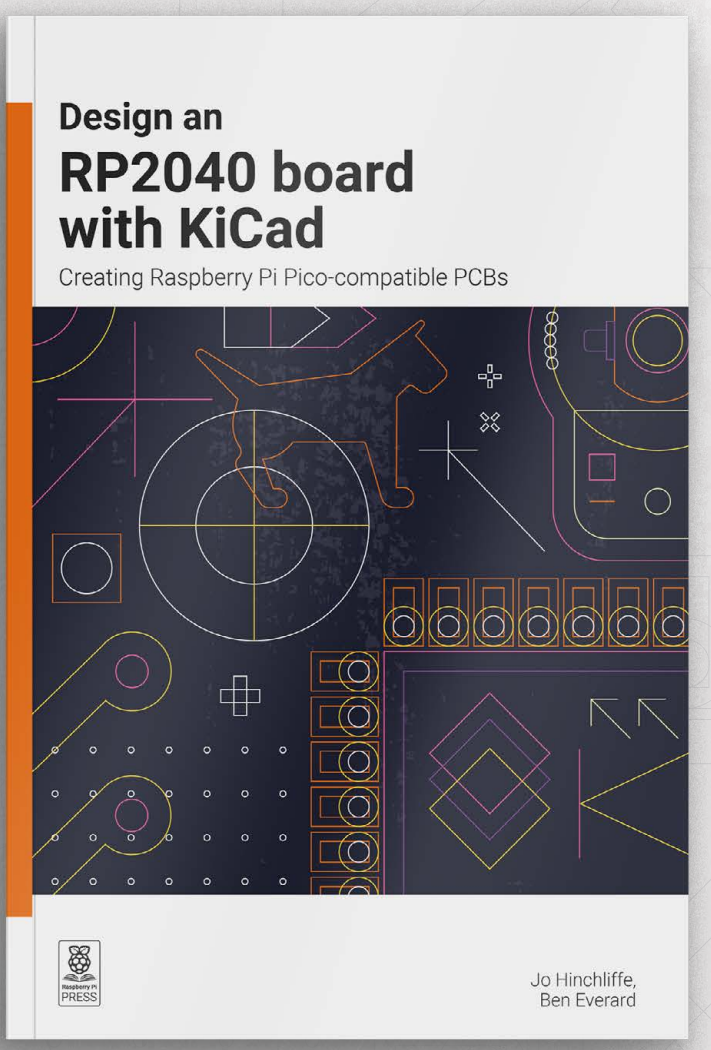
8BitDo / The Pi Hut | £90 / $121 | 8bitdo.com / thepihut.com

When emulating classic computers, it's worth connecting a mechanical keyboard with old-fashioned clicky-clack keys. This one features an 87-key layout with Kailh Box White switches V2 for tactile typing. It comes in four retro colour schemes: C64, N, Fami, and M.





▶ This keyboard comes with a couple of giant buttons



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

Meshstick

Ready-to-deploy LoRa adapter that turns Raspberry Pi into a Meshtastic-connected device. By **Rob Zwetsloot**

Elecrow **rpimag.co/meshstick** £18/$23

SPECS

DIMENSIONS:
88 × 30 × 18mm (excluding antenna)

WEIGHT:
36.6g (including antenna)

FREQUENCY:
EU 868MHz / US 915MHz

Verdict

A little tricky to set up for Raspberry Pi, but once it's running it's a great way to make a Raspberry Pi an extra node in your Meshtastic network.

8/10

There are various ways to add LoRa to a Raspberry Pi or even Raspberry Pi Pico, but not many are easier than with this Meshstick from Elecrow. It's just about ready out of the box (you just need to screw on the included antenna) and then can be plugged into a Raspberry Pi (but not Raspberry Pi Pico in this case) to be used as a LoRa node with the right Meshtastic software – in this case MeshtasticD, which can be installed using instructions from Meshtastic: **rpimag.co/meshd**.

Cross-compatibility

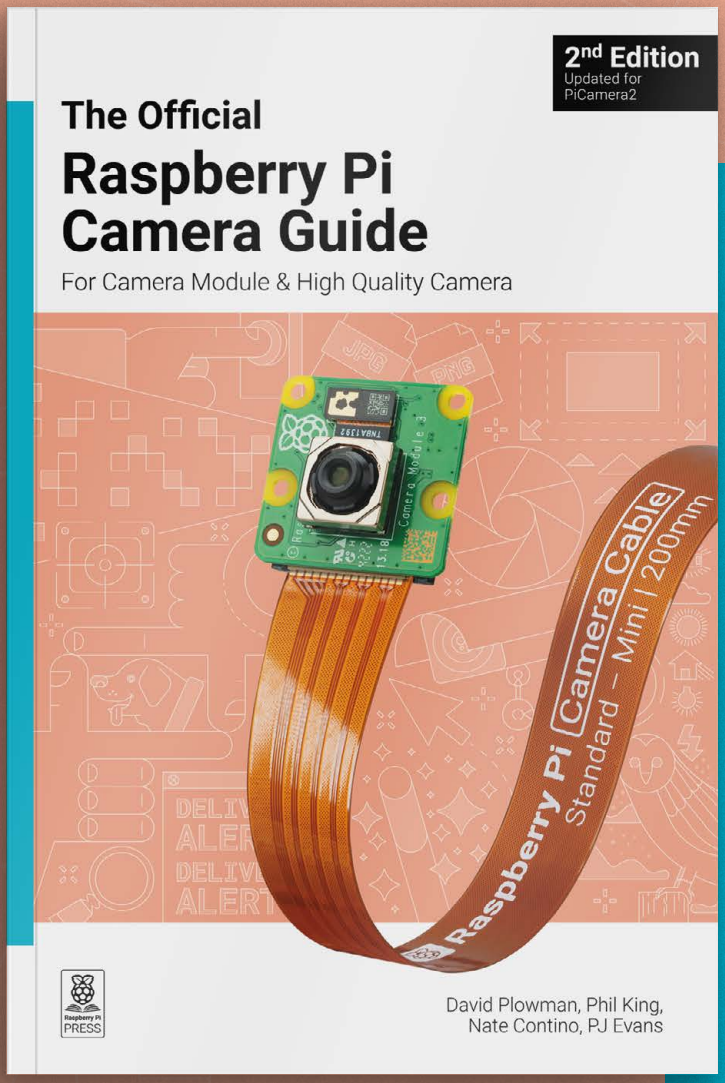
Meshstick also works on other Linux computers and also Macs, with drivers currently underway for Windows devices. Once the Meshtastic server is running, you can use it via specialist libraries and the web client (**client.meshtastic.org**) with the right configuration. This requires a more advanced level of knowledge to get working and isn't just a piece of software you can install and immediately get it configured. However, a piece of kit like this is designed for the kind of user who would need to do the advanced configuration to add a node to their network.

Configuration

The whole device is as configurable as you'd expect of a Meshtastic device, and the range with the high-gain antenna is nice too. There are no surprises with the Meshstick: it just works as you'd expect it to, while being presented in a nice package at a decent price.

▼ The antenna can be easily removed and reattached for transport, but the device has a low profile otherwise





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

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

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



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

From the practicalities of everyday life to creative endeavours, Raspberry Pi power is behind hundreds of products, learns **Rosie Hattersley**

Who knew that the humble launderette is having a moment as a social space to hang out with friends? According to Korea Laundry, the washateria is an increasingly desirable destination, with lifestyle branding alongside turn-up-and-wash convenience. Another company mixing things up its own way is Sweden's Elk Audio, which lets audiophiles have free rein over almost every aspect of their mixes, including almost every hardware element – Raspberry Pi 4 and Compute Modules among them. Yet more choice comes from Handlet, which gives you the option to run one of its Raspberry Pi-based tablets off a built-in battery and also offers customisations galore.

These three examples illustrate the flexibility of Raspberry Pi hardware and how readily Raspberry Pi has been adopted by both industrial and consumer clients, as well as how the market for high-quality Raspberry Pi products has gone global.

This issue, our showcase of Powered by Raspberry Pi products features innovations from Stockholm to Sydney, South Korea to Slovakia. Although prototyping and iterating can be relatively cost-effective, once you have a marketable product, there are compliance testing hoops to jump through. Raspberry Pi introduced Powered by Raspberry Pi accreditation to help manufacturers with this process, standardising compliance regulations and testing procedures for every Raspberry Pi. This info can be found at the Production Information Portal (**pip.raspberrypi.com**). Accreditation is also a visual and visible mark of quality and rigour that helps manufacturers stand out from other brands and helps give customers confidence.

Horace 2 photo booth

France | rpimag.co/horace-photobooth



The Horace photo booth is an excellent addition to any party or social occasion. Featuring two Raspberry Pi High Quality Cameras, the booth comes in 1.5m- and 85cm-tall versions, meaning there are options for adults and smaller photo fans, as well as wheelchair-using partygoers. Raspberry Pi 5 and Node-RED control everything inside the booth and provide a live 2K video stream of what the cameras can see. The photo booth's dye-sublimation printer produces Instagramable prints in less than 30 seconds. The whole setup weighs 35kg and can be folded flat for easy transportation. It has been incredibly well received in France, where Horace is based and where customers can hire him for special events along with a fun selection of photo backdrops.

Elk Audio

Sweden | rpimag.co/elk-audio



Stockholm-based Elk Audio describes Raspberry Pi as "the go-to platform for prototyping and experimentation". Thanks! We are also rather taken with the firm's 8GB Raspberry Pi 4-based audio endeavours. Raspberry Pi 4 was chosen for its quad-core processing power, efficiency, and scalability. Elk's low-latency DSP (digital signal processing) platform offers multichannel audio inputs and outputs, synthesis engines, samples, and effects. Audio nerds have the option of accessing the source code and playing around with absolutely everything, or choosing a reference design such as Nina, a twelve-voice motorised analogue polysynth. Elk offers a Raspberry Pi development board, Compute Module system on module, and the Stomp dev kit, giving audiophiles complete freedom over how to explore sound.

YUMA-X-FCC

Germany | rpimag.co/yuma-products



YUMA Engineering specialises in embedded technology, particularly aerospace computing. The company's precise engineering and creative thinking result in mission-ready hardware and software that can be used despite real-world uncertainties. Its YUMA-X-FCC airborne Linux-based computing platform features highly embedded, scalable AI and machine learning that's controlled, of course, by Raspberry Pi. The non-TSO (technical standard order) avionics platform is ideal for rapid prototyping. It features dual u-blox GPS with dead reckoning, 3D accelerometer, magnetometer, and gyroscope, and a barometer, along with a 5G/LTE modem, wireless LAN and two Gigabit Ethernet ports. This can be used alongside other smart components as an autopilot, for real-time moving digital maps, as a flight data recorder, jet engine recorder, or visual navigation and landing aid.

Handlet

Finland | rpimag.co/handlet



Handlet is a 5-inch battery-powered Raspberry Pi computer. As you may guess, the Assembled model works straight out of the box, but there is another option, the build-it-yourself Hobbyist version (**rpimag.co/handlet-5**) Equipped with Raspberry Pi 4 with 2GB RAM, the base version is priced competitively enough to attract schools and hobbyists keen to use it for mobile projects. Handlet includes a ruggedised VESA-mountable case with easy access to the microSD card slot, speaker, and 40-pin GPIO port. Customisation options include a choice of touchscreen covers and cases to provide protection, as well as speaker and buttons that boost the Handlet's entertainment credentials.

Korea Laundry Company Beebot

South Korea | rpimag.co/beebot



Beebot OS is an intelligent management and payment system for unattended kiosks such as self-service laundries. It has become synonymous with its origins in the Korea Laundry Co. and its hundreds of WashEnjoy lounges. The complete management system – based on Raspberry Pi and IoT capabilities – toggles each washing machine on and off, recognises and updates members' cards, takes one-off payments, and manages account top-ups. Raspberry Pi tracks each laundry site's performance figures, providing details to Korea Laundry's centralised network about each commercial washer's and dryer's operational status. Screens embedded in each kiosk can display offers and marketing information, providing added value via the Beebot 360 Manager software. This aspect is becoming increasingly important as there is built-in downtime at launderettes and Korea Laundry boasts a revisit rate of at least 80% at what it terms "a unique lifestyle space" with leisure and marketing potential.

System-IS STL.one

Slovakia | rpimag.co/stl-one

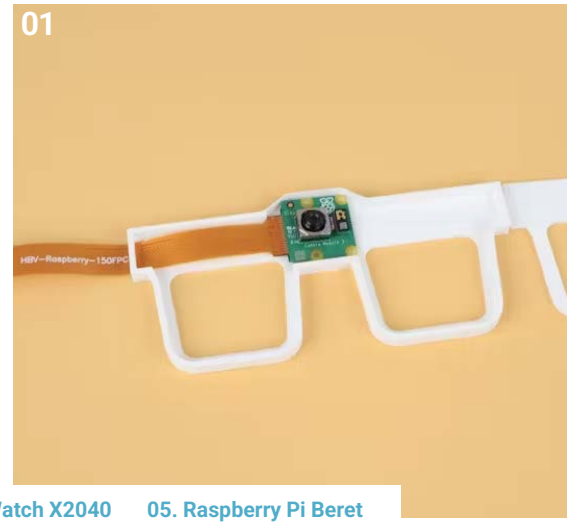


Showing up is half the battle for some of us! If it's your role to keep tabs on everyone's comings and goings, STL.one makes things much simpler. The check-in terminal is a 240 × 165mm, 8-inch touchscreen with a built-in card reader, running your favourite embedded computer and Fenix remote management software. Colleagues log themselves as present at meetings, events, and in specific rooms or zones using attendance cards or tags. A webcam checks that individuals are registering themselves as present, ensuring accurate attendance figures. The system not only monitors and summarises attendance; it also prepares a monthly report for managers.

10 amazing:

Raspberry Pi wearables

Put on your Raspberry Pi, there's lots of world out there



Raspberry Pi can be anything, and while it may not work well as a suit jacket to that upcoming wedding, you could probably use it in a suit jacket to make it just that extra bit special. Here are some ways people have been adding Raspberry Pi to their wardrobe to make their life to do just that.

01. Sign language translation glasses

Signal your understanding

rpimag.co/signglasses

Using computer vision to read the signer's hands, Raspberry Pi can then translate the signs and play audio of what they're saying on a speaker.

02. Raspberry Pi Zero wearable deck

Wrist-borne cyberdeck

rpimag.co/wearabledeck

A very functional computer on your wrist is the dream of cyberpunks everywhere, and it's very doable with a project like this.

03. Raspberry Pi Zero Pip-Boy

V.A.T.S. not included

rpimag.co/zeropipboy

We've seen many types of Raspberry Pi Pip-Boys over the years because, frankly, the basic idea is very cool. This one was made from scrap, so it's very economical too.

04. ChronoWatch X2040

Classic wearable tech

rpimag.co/chronowatch

What is a watch if not a piece of wearable tech? This is a simple digital watch powered by an RP2040 chip that we think looks very nice.

05. Raspberry Pi Beret

Princetastic pun

rpimag.co/picoberet

Need to get around a crowded convention hall safely? This Raspberry Pi Pico-powered headwear includes indicators and brake lights you control with your phone.

06. Constellation LED Prom Dress

Glow up

rpimag.co/promdress

This wonderful dress lights up with patterns from RGB LEDs sewn into the fabric. It was created in only a few hours too, but thanks to Pico, that's no problem.

07. Smart glove

It's so bad

rpimag.co/smartglove

This glove has in-built AI tech that can recognise gestures to be used with IoT devices – and also control an RC car.

08. Flight watch

Single-function tech

rpimag.co/flightwatch

This watch doesn't connect to your phone, or track your heart rate. It doesn't even tell the time. It will let you know what plane is flying overhead, though.

09. Cyber Glasses

Illuminated monocle

rpimag.co/cyberglasses

These fun glasses aren't very smart, but they are cool – they include a servo and ring of LEDs just in case you need to light up some fine soldering you're looking at.

10. PiGlass V2

Augmented reality

rpimag.co/piglass2

While smart glasses seem to be making a (controversial) comeback, for a while they went away, so folks started making their own to make up for it.


02


03
FALLOUT Pip Boy 2000 mk V


04


06

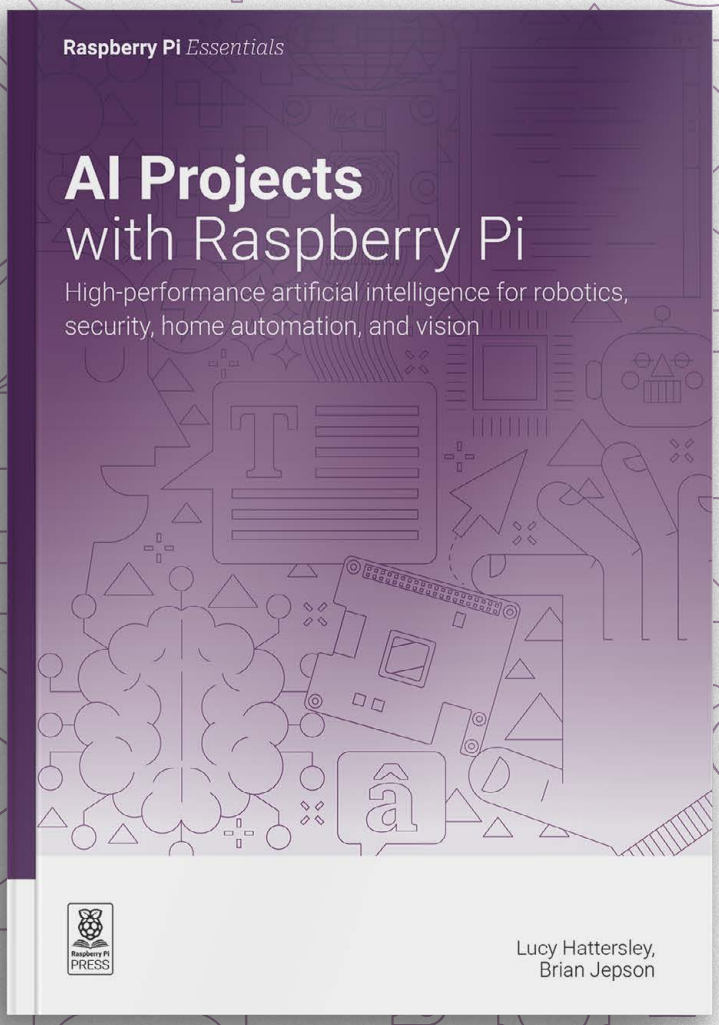

05


07


09


10


08
EZY94MG
FNC → BRS
EASYJET
GPS


Raspberry Pi Essentials
AI Projects
with Raspberry Pi
High-performance artificial intelligence for robotics, security, home automation, and vision
Raspberry Pi PRESS
Lucy Hattersley,
Brian Jepson

AI Projects
with Raspberry Pi

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

With this essential guide, you'll learn how to create all kinds of AI-powered projects on Raspberry Pi!
Combining Raspberry Pi hardware with artificial intelligence gives you new ways to interact with the world: turn your surroundings into a playing field with computer vision models, predict and adjust to changes in environmental conditions, and engage with people in natural ways.

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

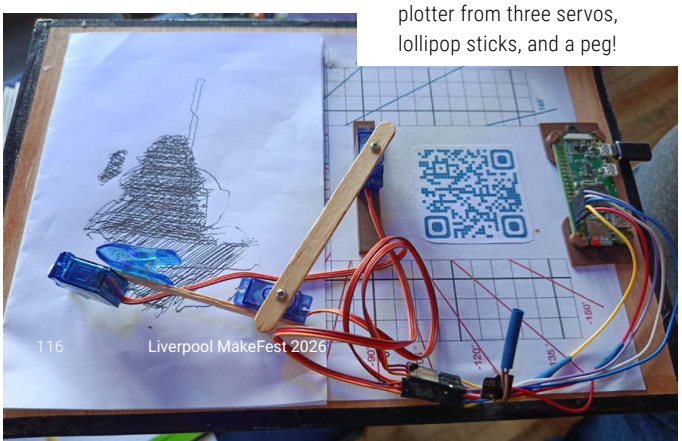
BUY ONLINE: **rpimag.co/aiprojectsbook**

Liverpool MakeFest 2026

The UK's largest free maker event celebrates creativity, arts, science, and technology. By **Jo Hinchliffe**







▲ The three floors of MakeFest ensured that nowhere was too packed, despite being a busy day

◀ Members of Wirral Repair Cafe had a room set up not only to promote repair cafes, but to do actual repairs!

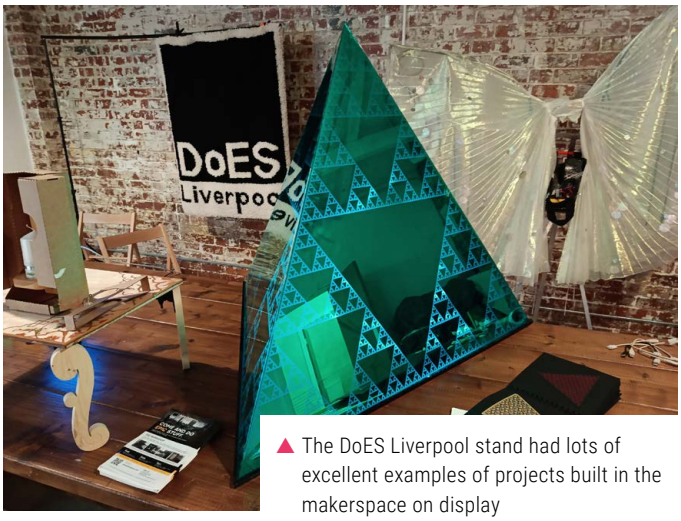
▼ A Brachiograph was spotted; this is an excellent Raspberry Pi Zero project to make a pen plotter from three servos, lollipop sticks, and a peg!

Sunday 13 September saw the return of Liverpool MakeFest. Traditionally, MakeFest has taken place at Liverpool Central Library, but this year it was located in the lovely Tapestry building, which is also home to DoES Liverpool, a fine example of a community makerspace with co-working and events.

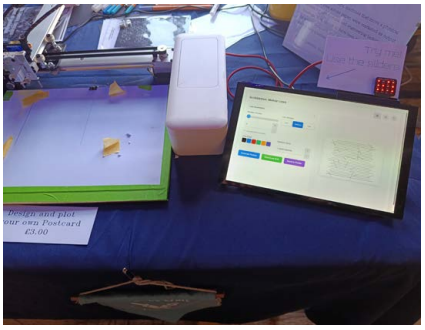
MakeFest was across three floors, each of which had plenty to offer. The ground floor of the building has previously been used as an infinity backdrop to photograph cars. This made it great for photographs, but somewhat tricky to know at any given time where the walls were! The author of this article was on this floor show-and-telling the (recently seen in this magazine) FoamO, the RP2040-powered 4-axis hot wire CNC foam cutter. Also spotted on this floor were the amazing Lego constructions from Share Our Bricks and there was even a repair room operating as a pop-up repair cafe by the lovely Wirral Repair Cafe team. DoES Liverpool had their own stall on this level and it was chock-full of amazing things.

It's always fun to spot Raspberry Pi hardware being used in projects. We spotted the fantastic Museum in a Box project on the DoES Liverpool stand. This project has a box containing a Raspberry Pi; if you place an object or a card on top of the box, it gives an audio description of the item. Combined with 3D scanning and printing, this means that you can have a museum of objects that are replicas of large museum items (think statue of David) and people can handle the items as well as learn about them from the audio stories. Packing a museum experience into a small box, it really does what it says on the tin.

The middle floor is where DoES is situated and throughout the day, attendees could jump onto a repeating tour of the DoES space. A great way to introduce new people to the amazing space chock full of fantastic tools, equipment, and community



▲ The DoES Liverpool stand had lots of excellent examples of projects built in the makerspace on display

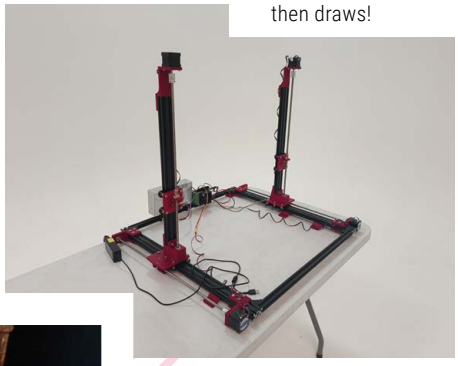


◀ A fabulous activity where a Raspberry Pi and a touchscreen act as a connected kiosk for people to design their own postcard, which an attached pen plotter then draws!

knowledge. As a large and well-established UK makerspace, DoES Liverpool has a great selection of machines and tooling. CNC routers, digital embroidery, and even a snazzy UV printer sit alongside the more usual but equally fabulous 3D printers and laser cutters. Also on the second floor there was an event space. Laid out theatre style, this space had a timetable of events, talks, and demonstrations throughout the MakeFest day.

The top floor of the Tapestry building offered a large space with lots of stands from makers and artists and more. It was a great cross-section of people and projects. There was representation of lots of communities, projects, and techniques: amateur radio, 3D scanning, pen plotting, synthesizers, live drawing, lace and bobbin crafts, CAD, CAM, game design, Godot, printmaking, and more.

It was an excellent day and the new venue worked really well with a steady flow of visitors throughout the day. The tireless MakeFest team and volunteers worked hard to make sure both visitors and makers had a great day, and we did. We look forward to being back in Liverpool for the next MakeFest.





▲ As seen recently in these pages, the RP2040-powered FoamO 4-axis hot-wire CNC foam cutter was on display

◀ An amateur radio group had a messaging system set up with some brilliant vintage tech

> *It's always fun to spot Raspberry Pi hardware being used in projects*

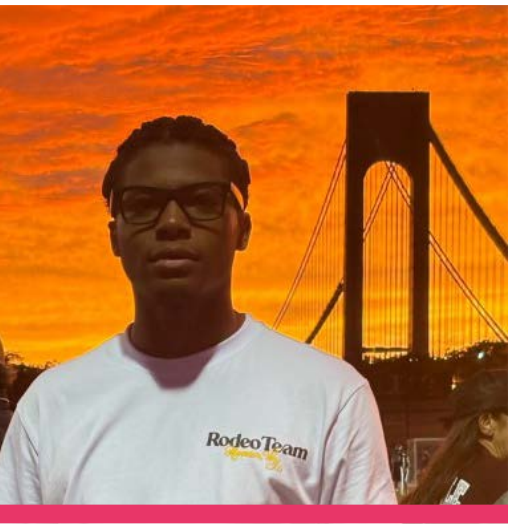
We spotted a great system using a Raspberry Pi and an attached touchscreen monitor as the interface for a pen plotting machine. The interface allowed people to generate their own designs that the attached pen plotter could then plot for them, a brilliant activity from Forge and Craft. Whilst visiting that stall, we also took a look at their Brachiograph, a fun low-cost plotting device made from a Raspberry Pi Zero, three servos, hot glue, lollipop sticks, and a peg.



▲ Liverpool Synth Group, who regularly use DoES as a meet-up venue, had a large display of synthesizer equipment set up for people to play with



▲ The Museum in a Box: a collection of objects can be placed on the box and the story of the object is then told



Favor Brunner

A mechatronics engineer who finds building a form of artistic expression

Name Favor Brunner
Occupation Student
Community role Online personality
URL rpimag.co/yard-mon-robotics

Everyone has a different story, even if there are some familiar trappings for makers such as a love for Lego, which Favor Brunner also had.

"My history with making started when I was around seven years old," he says. "I loved playing with Lego, but eventually I wanted the things I built to move and interact with the real world in more intricate ways. My uncle, who was a software engineer, noticed that interest and would bring over Raspberry Pi [boards] and other things I could build with. I thought it was the coolest thing."

Different backgrounds mean that not everyone can just go from there though.

"My mom wanted to nurture that love, so she enrolled me in an elementary school robotics programme called Brooklyn Robot Foundry," Favor continues. "She was really instrumental in making that happen. When the programme was too expensive, she bartered a piece of her own original artwork so I could have the opportunity to study something I still love doing today. So I guess you could say my history with making started with a sacrifice."

When did you first discover Raspberry Pi?

I first learned about Raspberry Pi around third grade. My uncle saw that I was interested in making things and brought home a Kano kit for me. To give credit where it's due, he did the assembly because I was still a little too young and impatient to put it all together myself!

But being able to run Minecraft and play games on a computer that had come into our home as a kit felt incredible. It gave me a sense of agency: a computer was something I could start to understand, build with, and eventually make my own things on.

I was pretty terrible at video games back then – and, honestly, I still am – but that computer gave me a way to make memories with my cousins.

How did your robotics channel start?

My robotics channel started just over a year ago. I was building things with spare parts from my summer programme and began posting videos because I wanted to share the cool stuff I was making.

Things really changed when I built a deodorant dispenser for people with Parkinson's disease. By God's grace, that project got a lot of traction and opened the door for me to crowdfund future builds I cared deeply about, including more accessibility technology and other ideas I was excited to bring to life.

What started as me sharing my projects became a way to keep building and connect with people who cared about what I was trying to do.

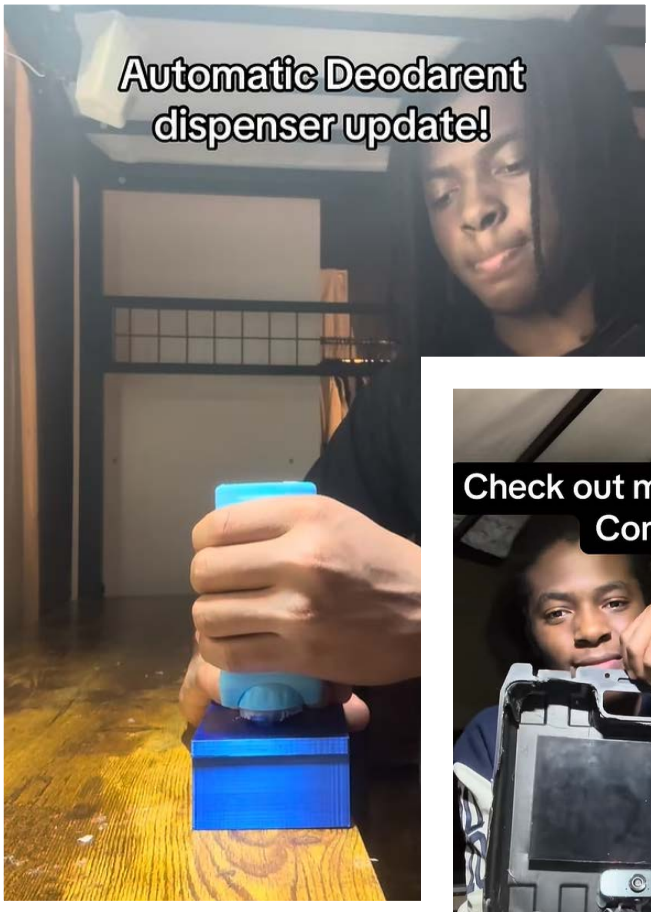
What are some of your favourite things you've made with Raspberry Pi?

[If] I had to pick a couple, they would probably be Run It Back NYC and my AI chatbot, Baymax.

Run It Back NYC is a transcriber I built for those MTA [Metropolitan Transportation Authority, New York's public transport agency – Ed] announcements that nobody can understand. It uses a Raspberry Pi and a Touch Display 2 to transcribe announcements, translate them into different languages, and read them aloud so people can understand what's being said.

That project has a special place in my heart because, during the summer, I ride the subway almost every day, and I can barely ever understand the announcements. It was exciting to work on a problem I experience myself while also thinking about how the solution could help other riders. The inspiration was basically, "Wait, what did they just say?" and then figuring out how to answer that question.

Another favourite is an AI chatbot that ran locally on my Raspberry Pi. I named it Baymax after one of the greatest robots of all time. It acted as a Mandarin tutor and let me practise my Chinese without paying for a tutoring subscription. It also gave me a chance to experiment with local AI first-hand, which made it both a useful tool and a really fun learning experience.





- This machine twists stick deodorants for folks with fine motor control issues, such as Parkinson's
- Favor has also made a classic Raspberry Pi cyberdeck project

SEE MORE OF FAVOR

You can find Favor online on TikTok (**@yard.mon.robotics**) and YouTube here: **rpimag.co/favor-yt**

What started as me sharing my projects became a way to keep building and connect with people

Maker Monday

Amazing projects direct from social media!

Every Monday, we ask the question: have you made something with a Raspberry Pi over the weekend? Every Monday, our followers send us amazing photos and videos of the things they've made.

Follow along to #MakerMonday each week over on our various social media platforms!





01. Air-gapped systems are very secure as they don't directly interface beyond their network – **rpimag.co/171mm1**

02. We like the pixellated LCARS display from *Star Trek: The Next Generation* – **rpimag.co/171mm2**

03. This fire engine looks great with its subtle weathering – **rpimag.co/171mm3**

04. Rocketry was a classic cover for HackSpace Magazine! We love it - **rpimag.co/171mm4**

05. We talked a lot about kiosks and digital displays last issue! – **rpimag.co/171mm5**

06. Pico projects are built for transferring to custom circuits – **rpimag.co/171mm6**

07. The WonderSwan is a semi-obscure 1990s handheld console, so we love this – **rpimag.co/171mm7**

08. Oh, this is a work of art – **rpimag.co/171mm8**



Vincent Willcox
@talktech@hachyderm.io

@rpimag - I've been working on a Home Display (kiosk, canvas etc) - blog.talktech.info/post/homeca...



TalkTech.info · Sep 26

HomeCanvas: My Raspberry Pi Home Kiosk

HomeCanvas has grown from a photo kiosk into a Raspberry Pi home hub for Immic...

05

DanInvents
@DanInvents

04

This weekend I flew a fiberglass water rocket featuring an RP2040-based flight computer.

The plan is to publish a full video in the coming days explaing the science and engineering behind it.

Here's is a sample of some of those flights
youtube.com/shorts/RtBMNpQ...



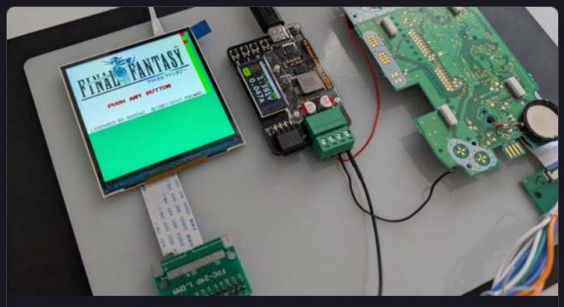
DerTobi
@DerTobi@mastodon.social

@rpimag I wrote drivers for the RP2350 that capture WonderSwan and WonderSwan Color display data entirely using PIO and DMA, without CPU involvement.

The drivers should be adaptable to other platforms as well. I wrote a blog post about the display signals, the capture process, and some of the challenges involved in capturing the WonderSwan Color's multiplexed RGB444 data.

#makermonday #rp2350 #pipico

embedded-ideas.de/posts/260910...



embedded-ideas.de · Sep 10

Capturing WonderSwan and WonderSwan Color display data

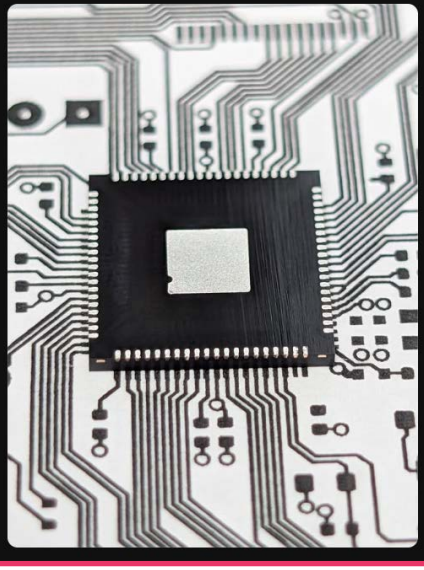
A short introduction to the display signals on the WonderSwan and WonderSwan Col...

07

john_newall 15/09/2026

06

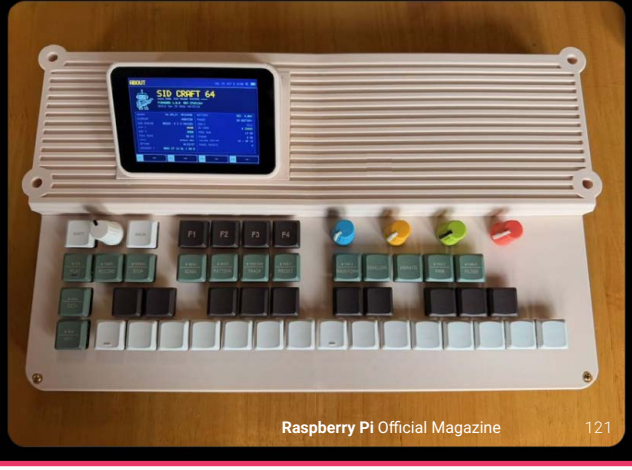
Just finished checking my RP2350 footprint before getting a PCB fabricated. I'm attempting to migrate a Pico project onto a custom PCB.



steddy
@steddyman

08

I'm making a C64 dual SID based synth using the Raspberry Pico 2354B that is running an Open Source firmware I am building #MakerMonday



ODIE V3

AI-powered robo-dog, courtesy of Raspberry Pi 5

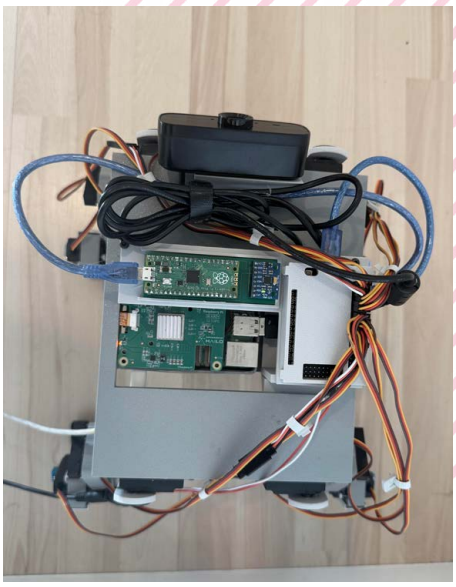
We got an email from 13-year-old Malhar about his quadrupedal robot project called ODIE which has a Hailo-8L NPU on board too.

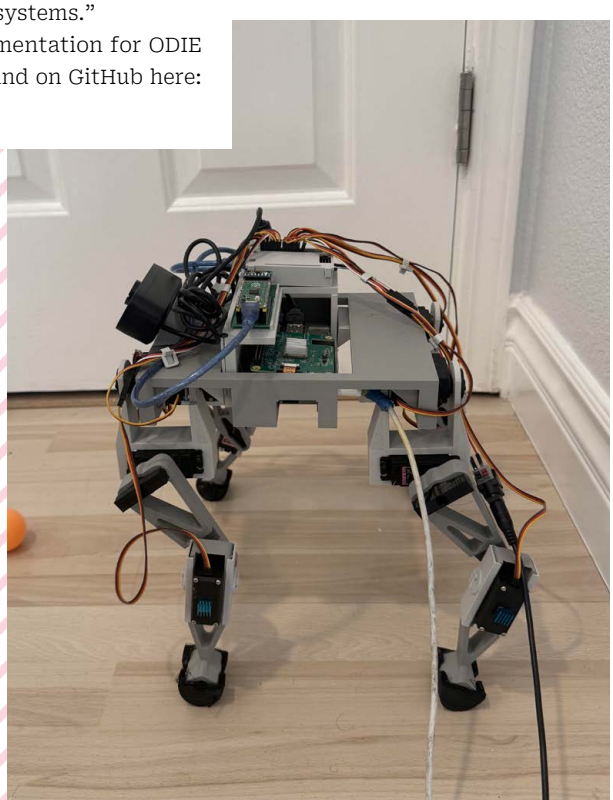
"ODIE V3 is a 12-DOF quadruped robot powered by Raspberry Pi 5 with three servos per leg," Malhar tells us. "It demonstrates physics-driven locomotion using MuJoCo simulation that directly translates to real servo commands – no manual tuning needed. The total build cost is $400–500, making it accessible for hobbyists and students interested in robotics and control systems."

There's full documentation for ODIE too, which you can find on GitHub here: **rpimag.co/odiev3**.



◀ ODIE uses a variety of off-the-shelf parts and 3D printed bits





▲ Raspberry Pi 5 and a Pico are used to get the robot working

◀ ODIE's square shape might make it tricky to realise this is the side of it

Crowdfund this!

Crowdfunding campaigns to keep an eye on

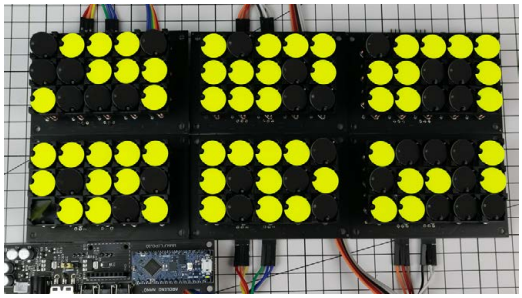
Hacki Flex



This simple-looking cable has an RP2350 (or ESP32-S3) microcontroller on board that makes it a powerful cybersecurity tool. It's described as a "smart device is designed to teach, test, and explore ethical hacking – openly, responsibly, and locally".

▶ **kck.st/3U0pvpW**

FLIPDOT



This is a modular 3 × 5 flip-dot matrix display, which can be controlled via Raspberry Pi or one of several microcontrollers. You can change the colour of the dots if you like, and the whole thing looks better with multiple displays connected together.

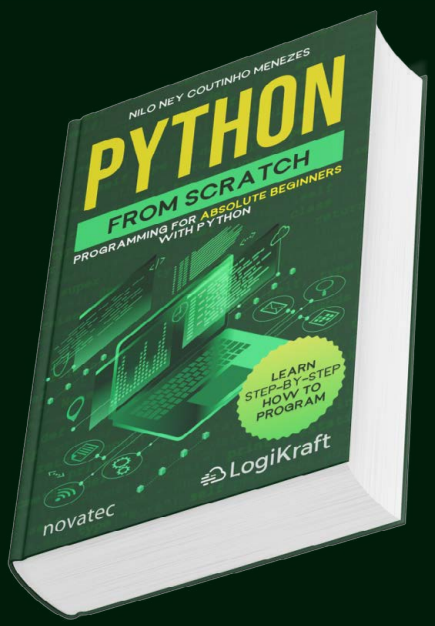
▶ **rpimag.co/flipks**

YOUR RASPBERRY PI RUNS PYTHON. DO YOU?

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

No prior coding experience needed

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



PYTHONFROMSCRATCH.COM

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

Your Letters

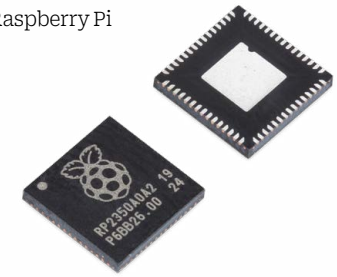
RP2350 Security

I read with interest (though not exactly 100% comprehension) about a security breach on the RP2350 chip used in Raspberry Pi Pico 2, which lets hackers get round the secure boot security and potentially load their own, malicious code. Obviously it's not that big a deal if hackers take over my Raspberry Pi Pico 2 greenhouse irrigation system, as the worst that can happen is they overwater my tomatoes, but might this be significant for other uses, and are you doing anything to fix the breach?

Adam, via email

The breach you're talking about is utterly, utterly fascinating, and at the same time is vanishingly unlikely to place your tomatoes under threat. To begin at the beginning, when RP2350 was released, Raspberry Pi set up a hacking challenge, with $20,000 in prizes up for grabs for anyone who could sidestep the security measure in the chip. This was a great success: four hacking teams found security flaws, and thanks to their efforts the next version (known in chip terms as the next 'stepping') is now considerably more secure.

The most recent breach requires physical access to the chip; it also requires about $250,000 worth of specialised equipment, and the expertise to use that equipment, so it's not something that we expect to be repeatable on any sort of scale. It's a lab-based exercise, and the fact that it's been done out in the open means that we can do something about it to make the next revisions of Raspberry Pi hardware even more secure.



▶ The Ledger Donjon team used a highly focused infrared laser beam to alter the contents of specific control registers in the RP2350 chip, to enable debug mode



▲ Sensea – underwater photography using Raspberry Pi

Sensea

I take my hat off to the team behind the Sensea underwater camera. It looks so cheap and cheerful, but I bet it works just as well as products that cost loads more.

Mark, via email

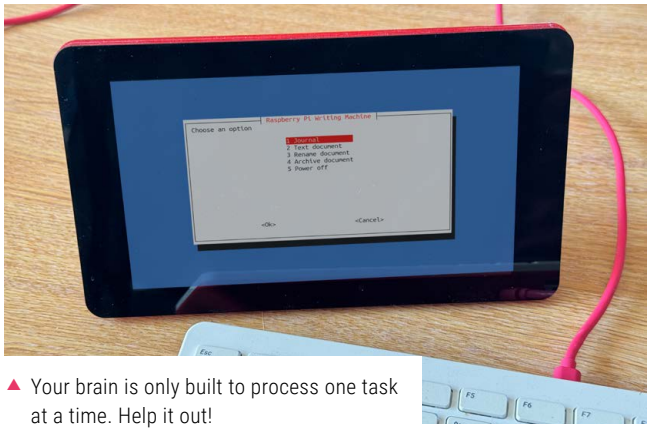
Sensea was built by a young team aged 10-13, which fills us with hope for the future. It is much like the underwater rover featured in this magazine recently, built by Jo Hinchliffe: cheaply, out of parts that anyone can get hold of. PVC pipes and fittings are used by plumbers for a good reason: they're waterproof, easy to work with, and cheap. Tech giants have conditioned us all to believe that devices come in shiny aluminium enclosures, but it ain't so.

Writing machine

Thanks for the tutorial on making a distraction-free writing machine [in *RPOM* issue 170]. I've been finding myself pulled into the internet more and more when I'm trying to get work done. It's fine to break off from writing every now and then to do some fact-checking, but the internet is a rabbit hole and it's so easy to get sucked in and waste the day. I find I get much more work done if I use a pen and paper, just because it's not linked to social media. I needed an excuse to get the new touchscreen, and now I've got one!

Sarah, via email

To maximise your productivity, it helps to set up your working environment so that it's just another tool for your task. Anything that's not useful is in the way. Your workbench should be clear apart from whatever you're working on and whatever tools you're using. So it comes as a surprise that we work all day on computers that are filled with, and have access to, so much stuff that we just don't need. Building your own writer deck can solve this problem, and it gives you a good reason to play with new technology too.



▲ Your brain is only built to process one task at a time. Help it out!

Contact us!

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

USA SPECIAL!

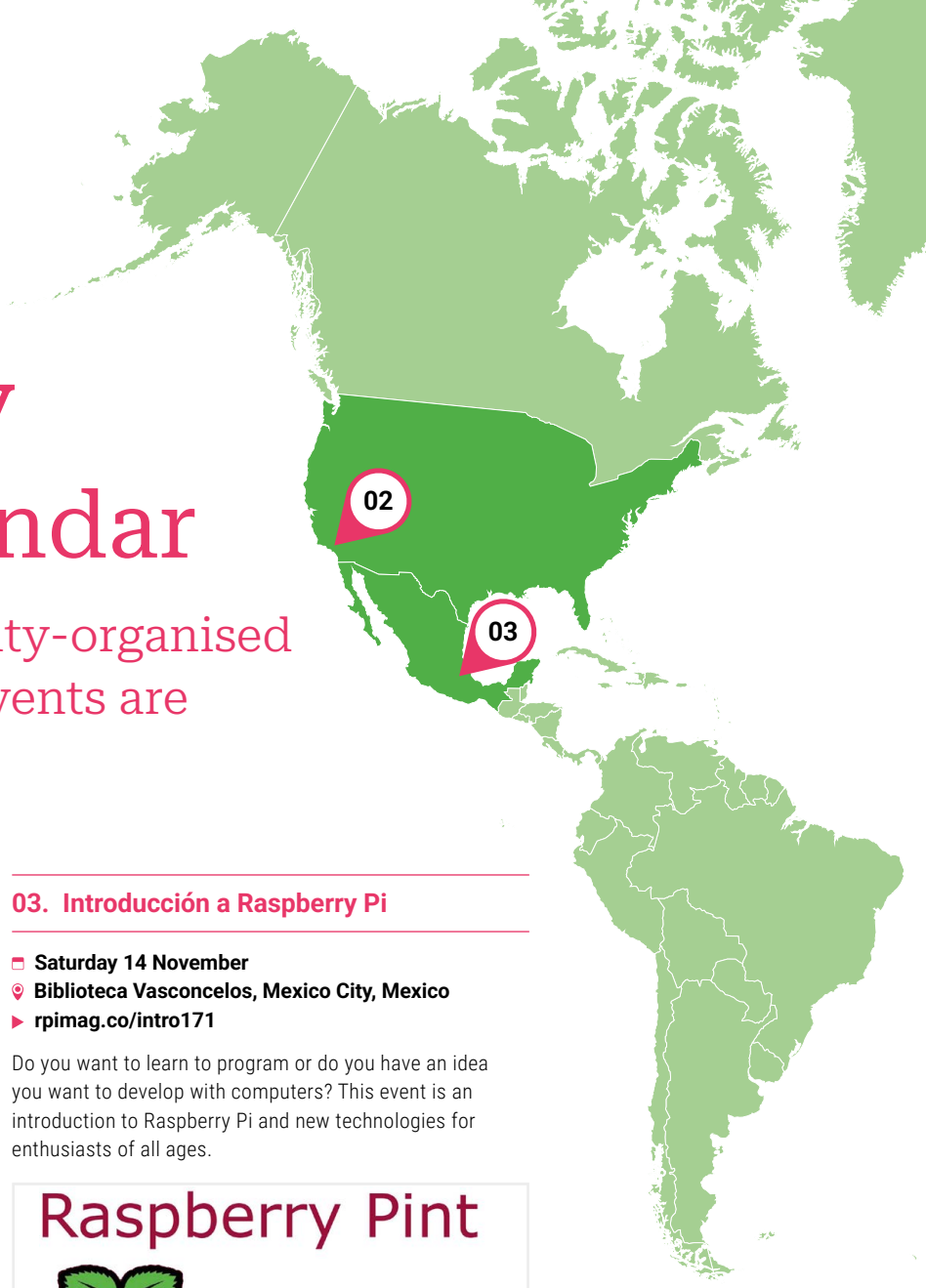
6 ISSUES FOR $43



Subscribe online:
rpimag.co/subscribe

Continuous credit card orders will auto-renew at the same price unless cancelled.
A choice of free Pico 2 W or Pico 2 is included with all subscriptions. This is a limited offer.
Not included with renewals. Offer subject to change or withdrawal at any time.

Community Events Calendar

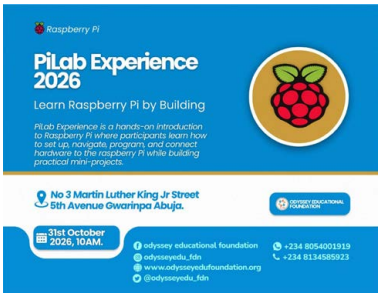


Find out what community-organised Raspberry Pi-themed events are happening near you...

01. PiLab Experience

- **Saturday 31 October**
- **Developer Fab Lab, Abuja, Nigeria**
- **rpimag.co/pilab26**

PiLab Experience is a hands-on introduction to Raspberry Pi where participants learn how to set up, navigate, program, and connect hardware to the Raspberry Pi while building practical mini-projects.



02. Riverside Raspberry Pi Meetup

- **Monday 9 November**
- **3600 Lime Street, Riverside, CA, USA**
- **rpimag.co/rrpm171**

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

03. Introducción a Raspberry Pi

- **Saturday 14 November**
- **Biblioteca Vasconcelos, Mexico City, Mexico**
- **rpimag.co/intro171**

Do you want to learn to program or do you have an idea you want to develop with computers? This event is an introduction to Raspberry Pi and new technologies for enthusiasts of all ages.



04. November Raspberry Pint

- **Tuesday 24 November**
- **Online**
- **rpimag.co/pint171**

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



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

05. EdgeTech+2026

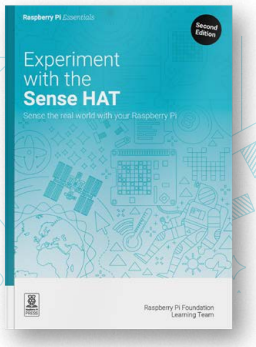
Official Raspberry Pi Event

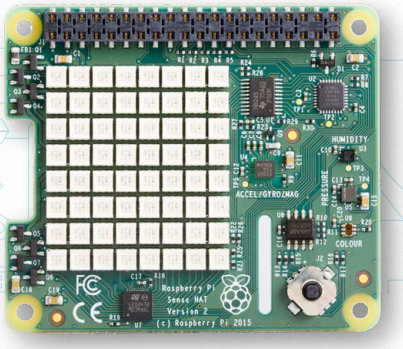
Wednesday 18 November to Friday 20 November
Pacifico Yokohama National Convention Hall, Yokohama, Japan
rpimag.co/edgetech26

The Raspberry Pi team is delighted to be exhibiting alongside Mechatrax and KSY at EdgeTech+ in Yokohama, Japan, this November. There, you'll be able to meet the team and experience the full range of technology, including Raspberry Pi single-board computers (up to Raspberry Pi 500+), the Raspberry Pi Pico family, and RP2350-based solutions, as well as AI products, cameras, and the latest industrial device: Compute Module 5.

Win 1 of 5 Sense HAT boards & Experiment with the Sense HAT books

Used for extraterrestrial experiments on board the International Space Station, the Raspberry Pi Sense HAT enables you to monitor pressure, humidity, temperature, colour, orientation, and movement, and show data on its bright 8 × 8 RGB LED matrix. Our official *Experiment with the Sense HAT* book explains how you can use it to make your projects and dreams a reality.





Head here to enter:

rpimag.co/win

Learn more:

rpimag.co/sensehatbook

Terms & Conditions

Competition opens on **21 October 2026** and closes on **19 November 2026**. Prize is offered to participants worldwide aged 13 or over, except employees of Raspberry Pi Ltd, the prize supplier, their families, or friends. Winners will be notified by email no more than 30 days after the competition closes. By entering the competition, the winner consents to any publicity generated from the competition, in print and online. Participants agree to receive occasional newsletters from Raspberry Pi Official magazine. We don't like spam: participants' details will remain strictly confidential and won't be shared with third parties. Prizes are non-negotiable and no cash alternative will be offered. Winners will be contacted by email to arrange delivery. Any winners who have not responded 60 days after the initial email is sent will have their prize revoked. This promotion is in no way sponsored, endorsed or administered by, or associated with, Instagram, Facebook, Twitter (X) or any other companies used to promote the service.

Official Magazine
#172

Portable Pico Programming

Take your programming project on the move

On sale 19 Nov

PLUS!

Ask a Raspberry Pi engineer

Build active websites

Train a robot dog



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, Ben Everard,
Rosemary Hattersley, Jo Hinchliffe, Nicola King,
Phil King, Rob Miles, Simon Long

Publishing

Publishing Director
Brian Jepson
brian.jepson@raspberrypi.com

Director of Communications
Helen Lynn

CEO
Eben Upton

Distribution

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

Subscriptions

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

This magazine is printed on paper sourced from sustainable forests and the printer operates an environmental management system which has been assessed as conforming to ISO 14001.
Raspberry Pi Official Magazine is published by Raspberry Pi Ltd, 194 Cambridge Science Park, Milton Road, Cambridge, England, CB4 0AB. The publisher, editor, and contributors accept no responsibility in respect of any omissions or errors relating to goods, products, or services referred to or advertised in the magazine. Except where otherwise noted, content in this magazine is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported (CC BY-NC-SA 3.0).

ISSN: 2977-4403 (Print)
ISSN: 2977-4411 (Digital)



3D printing and DIY

Owning a Raspberry Pi and a 3D printer has opened up a raft of new jobs for **Andrew Gregory** to do around the house

This month I've completed two repair jobs in the house: the first, a hole in the ceiling that I lived with for almost two years before I finally got round to fixing it. Safe to say that plastering is not my forte. The second job was a broken roller blind. A small piece of plastic had developed a small crack, which broke and stopped me closing the blinds. So I fired up my Raspberry Pi 500+ and started designing a replacement part. And then I stopped. 3D design is hard.

But after a few days with the problem percolating at the back of my mind, all became clear: if I couldn't do it the hard way, I'd do it the easy way. Instead of recreating the exact shape of the roller, I'd make the minimum viable roller.

> *Forget measure twice, cut once – my new mantra is 'measure once, print as many times as necessary'*

This made designing the thing in FreeCAD theoretically very easy. There was trial, and there was error, but I eventually realised that a cylinder is just a circle that's been extruded into three dimensions. An axle hole to accept a square rod is just a square, extruded into negative space to make a void (or a pocket, in CAD terminology) running all the way through the cylinder.

To hold the two ends of the ribbon that tilts the blind, I'd need some way of attaching the ribbon. I opted for a pair of cylindrical pockets, running all the way through the main cylinder close to its circumference, and two rectangular pockets joining these up with the outside edge of the cylinder. Forget measure twice, cut once – my new mantra is 'measure once, print as many times as necessary'. What I ended up with was rough, simple, and most importantly, it works.

K.I.S.S.

So how was I, with no 3D design experience and an entry-level 3D printer, able to simplify a part designed by professionals and made in a factory on an injection moulding machine set up to make that one specific part? When you consider the gap in expertise and resources, it shouldn't be possible. The answer is that the professionals are working under a set of constrictions that I just don't know about. That same part might have had to fit in a few different-sized assemblies; it might have had a minimum weight for packaging purposes; it might have had to fit several different thicknesses of fabric. Or perhaps it had to go through an automated final assembly process, in contrast with my replacement piece, which even though I'd designed it (ahem) perfectly, was still a bit fiddly to fit into the final build.

None of these considerations apply to us, the layman, the next time we need to fix or make something. It doesn't matter how simple or basic a homemade project is; if it works, it works.

Andrew Gregory – Author
Andrew is shocked that drinking lots of water and going to bed early really is as good as all the well-being gurus say it is.

rpimag.co

HiPi.io

HighPi 5S

The new case from the HiPi team



Reliable case for complex projects:

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

Customization options for volume orders:

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

Available at these great Pi stores:

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

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

PiKVM

Remote control **redefined**

Manage your servers or PCs remotely!

Powered by Raspberry Pi

PiKVM V4 Mini

Small, cost-effective, and powerful!

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

PiKVM V4 Plus

The most feature-rich edition

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



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

Available at the main Raspberry Pi resellers

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

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

ROBU.IN Your ideas, Our Parts · 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: