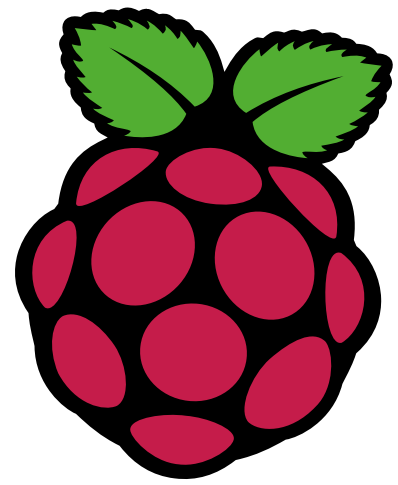


BUY IN PRINT **WORLDWIDE** MAGPI.CC/STORE

The *MagPi*



Issue 113 | January 2022 | magpi.cc

The official Raspberry Pi magazine

RASPBERRY PI

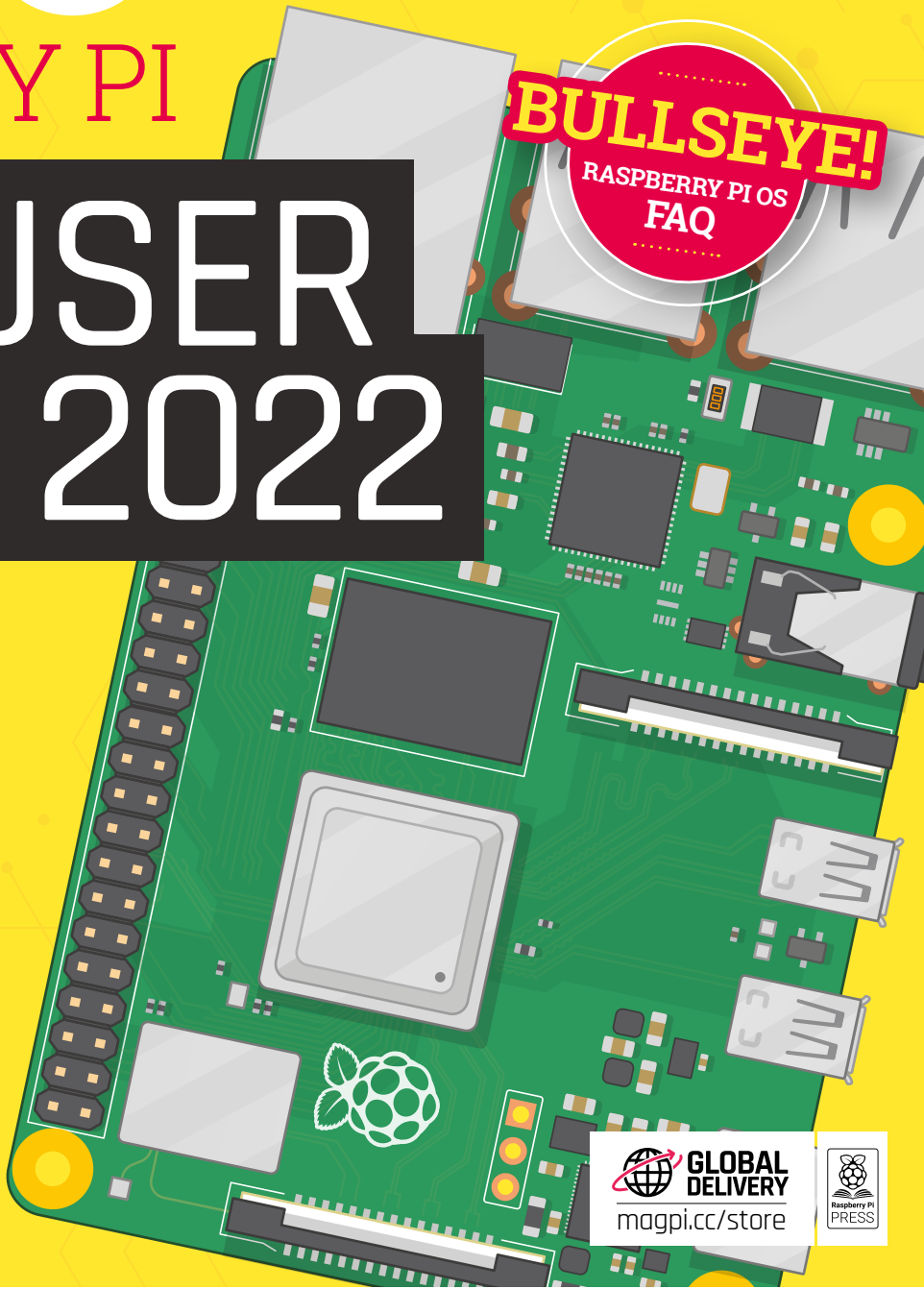
NEW USER GUIDE 2022

- > Installing the OS
- > Discover new features
- > Starter electronics

+ START CODING
WITH SCRATCH

+ BUILD A
BINARY CLOCK

BULLSEYE!
RASPBERRY PI OS
FAQ



 **GLOBAL DELIVERY**
magpi.cc/store



RASPBERRY PI **GAMING SPECIAL!**

100,000+ NEW PRODUCTS ADDED IN PAST 90 DAYS

Enabling the World's Ideas[®]

DIGIKEY.CO.UK

10.1 Million+ Products Online
2,000+ Industry-Leading Suppliers
100% Franchised Distributor



*A shipping charge of £12.00 will be billed on all orders of less than £33.00. A shipping charge of \$18.00 USD will be billed on all orders of less than \$50.00 USD. All orders are shipped via UPS, Federal Express, or DHL for delivery within 1-3 days (dependent on final destination). No handling fees. All prices are in British pound sterling or United States dollar. Digi-Key is a franchised distributor for all supplier partners. New products added daily. Digi-Key and Digi-Key Electronics are registered trademarks of Digi-Key Electronics in the U.S. and other countries. © 2022 Digi-Key Electronics, 701 Brooks Ave. South, Thief River Falls, MN 56701, USA



ECIA MEMBER
Supporting The Authorized Channel

WELCOME

to *The MagPi* 113

I still remember picking up my first Raspberry Pi computer like it was yesterday. I built a small two-wheeled robot and got all the likes on Facebook. One thing led to another, and here I am editing this incredible magazine.

Everybody has to start somewhere, and this month we've put together a Get Started with Raspberry Pi feature (page 30). This article helps newcomers put together a Raspberry Pi, install the Raspberry Pi OS operating system, and discover basic electronics with GPIO. It's perfect for Raspberry Pi newbies.

We've also got the best Starter Kits (page 82) and some great beginner-friendly tutorials. Whether it's creating Pong controllers (page 40), building a binary clock (page 58), or using sensors to build your own Weather Station (page 50), there are some great ideas here for Raspberry Pi fans.

Rob has also put together a great feature on Gaming with Raspberry Pi (page 68).

Not retro gaming, but the full-on, modern video games you can play with an amazing \$35 computer.

Raspberry Pi comes from humble beginnings, but every month it gets better. We can't wait to see what Raspberry Pi does in 2022!

Lucy Hattersley Editor



EDITOR Lucy Hattersley

Lucy is editor of *The MagPi* and her "wheelie great" robot is still standing proud on a shelf behind her.

magpi.cc

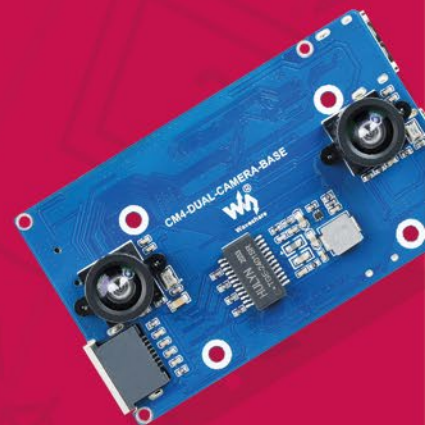




BASE BOARDS & MINI-COMPUTERS

Designed for Raspberry Pi Compute Module 4

- * PoE/IoT/Dual Gigabit ETH Base Boards for CM4
- * Various Mini-Computers Based on CM4



More details, please check Waveshare website:
<https://waveshare.com/cm4>

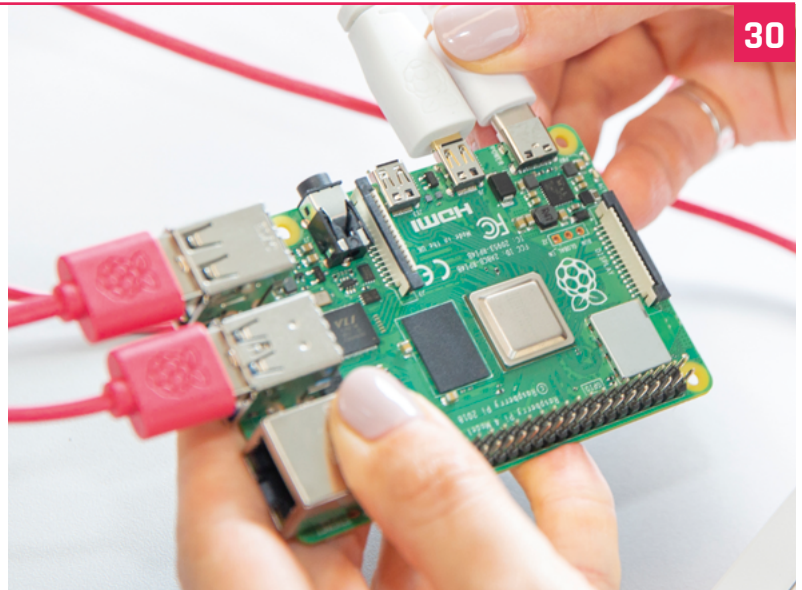


Contents

► Issue 113 ► January 2022

Cover Feature

30 Raspberry Pi new user guide 2022



30

Regulars

- 92** Your Letters
- 97** Next Month
- 98** The Final Word

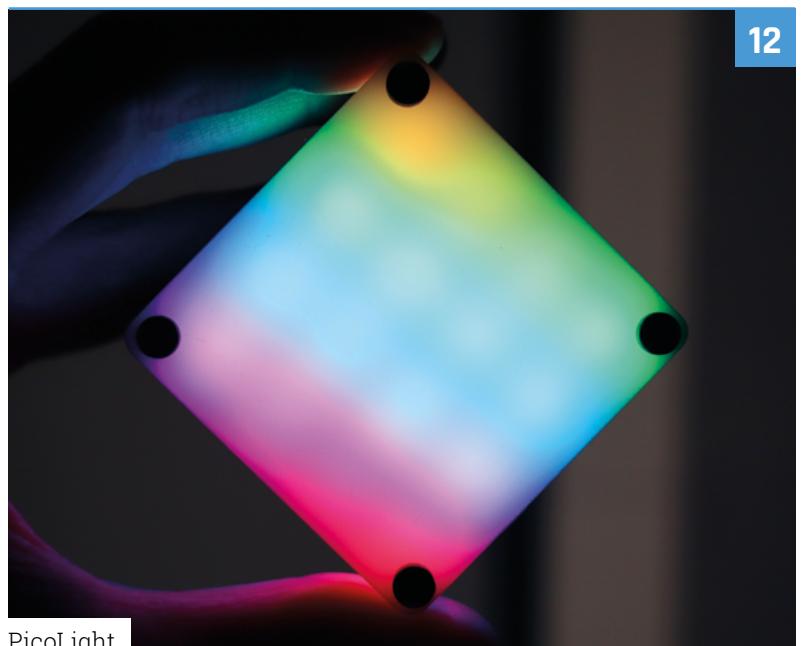
Project Showcases

- 08** Old School Minitel Laptop
- 12** PicoLight
- 14** Big Boxes
- 18** Virtual birthday cake
- 22** Environmental cargo logger
- 24** Super 8 Kamera



08

Old School Minitel Laptop



12

PicoLight

DISCLAIMER: Some of the tools and techniques shown in The MagPi 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 (Trading) Ltd does not accept responsibility for any injuries, damage to equipment, or costs incurred from projects, tutorials or suggestions in The MagPi magazine. Laws and regulations covering many of the topics in The MagPi 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 The MagPi magazine may go beyond. It is your responsibility to understand the manufacturer's limits.

Tutorials

- 40 LEGO® Pong
- 50 Sensory World - part 3
- 54 Composite video on Zero 2 W
- 58 Binary Clock
- 62 Pico Musical Cheese Box

The Big Feature



Raspberry Pi Gaming

Reviews

- 78 Argon IR Remote
- 80 DACBerry 400 S
- 82 10 Amazing Starter Kits
- 84 Scratch resources

Community

- 86 Stewart Watkiss interview
- 88 This Month in Raspberry Pi



LEGO Pong



Binary Clock



DACBerry 400 S



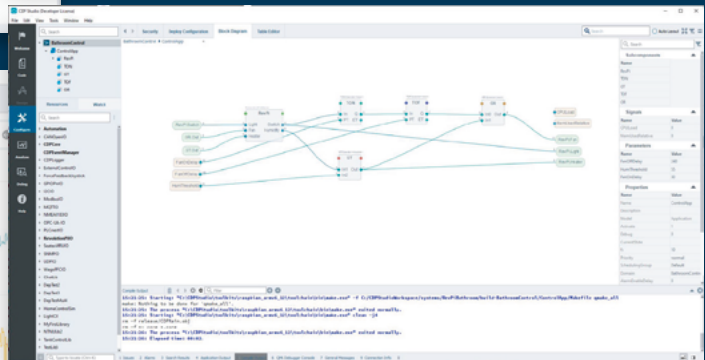
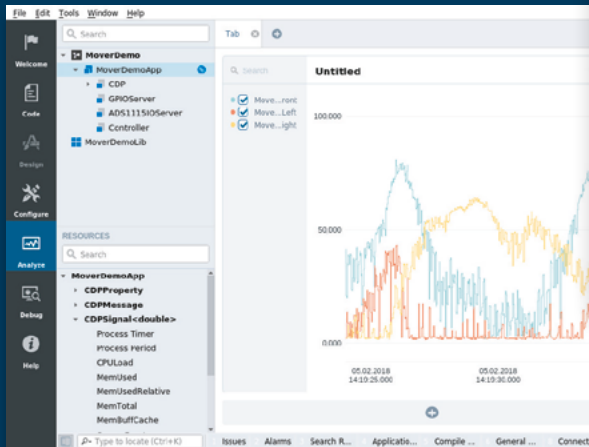
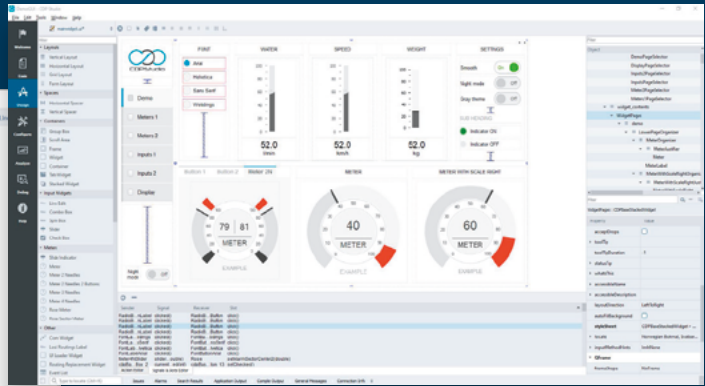
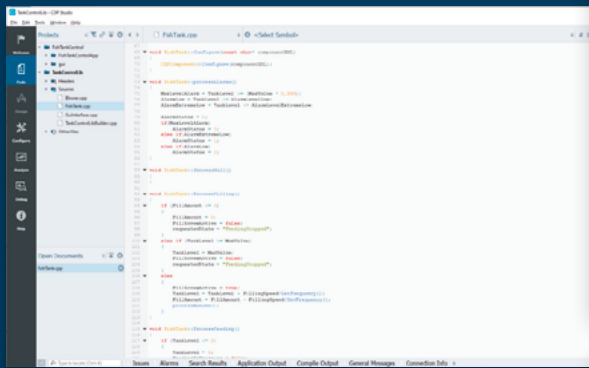
Stewart Watkiss interview

WIN
1 OF 5

PICOSYSTEM CONSOLES!

95

Code



Configure

Analyze

PROFESSIONAL CONTROL SYSTEM DEVELOPMENT TOOL

Home projects made easy.

CDP Studio, a great software development tool for your home projects. Build systems for Raspberry Pi, use C++ or NoCode programming, open source libraries, out of the box support for GPIO, I2C, MQTT, OPC UA and more. Create beautiful user interfaces. Built for industrial control system development, **FREE for home projects.**

cdpstudio.com

Tel: +47 990 80 900 • info@cdpotech.com

CDP Technologies AS // Hundsværgata 8, 6008 Ålesund, Norway



Old School Minitel Laptop

From Minitel to a near-mini laptop, Gautier Serodon has repurposed a terminal once used for France's innovative online service in the 1980s, as **David Crookes** explains



Gautier Serodon

Gautier is an engineering student, as well as an electronic and DIY enthusiast. He enjoys retro tech, vintage cars, watchmaking, 3D printing, and board games.

magpi.cc/minitel

Prior to the development of the World Wide Web, France had a hugely popular telecommunications service called Minitel.

It allowed the country's citizens to book train tickets, check their electronic mail, search the telephone directory, and access online banking among other things, attracting an estimated 25 million users and offering around 26,000 services at its peak.

Launched in 1982 and remaining in use for exactly 30 years, it was far ahead of its time. Anyone who wanted a terminal to connect to Minitel could get one for free from what became France Télécom, and this led to 9 million sets being installed in homes by 1999. But since Minitel closed, many have ended up being sold. "It's easy to find a terminal on sale for below €10," says French maker Gautier Serodon.

Having snapped one up himself at a garage sale, the 25-year-old decided to bring it back to life. "I love retro tech and I wanted to revive my Minitel with today's technology," he says. He decided a Raspberry Pi 3B computer would enable him to do just that. "It's affordable, compact, and internet-compatible," he explains. And the plan? To convert it into a battery-powered laptop so he could take notes during his engineering school classes.

Key to authenticity

Gautier bagged himself a Minitel 1B terminal, made in 1982 by Telic Alcatel. He stripped it of most of its parts, including the CRT display, but decided he wanted to at least retain use of the terminal's original – and satisfyingly clicky – AZERTY keyboard.

He also kept the power button and power indicator. "I wanted to keep as much of this



▲ It may not be the sleekest of 'laptops', but his charmingly bulky piece of retro kit is still portable

charming old technology as possible," he tells us. "The power button and the LED played an important role in maintaining the aesthetics of the Minitel."

The screen was replaced by a 10.4-inch LCD panel which came complete with a driver board and this was fixed within the Minitel casing using four 3D-printed parts, one for each corner. The biggest challenge, however, involved retrofitting the old Minitel keyboard so that it would simulate a USB HID keyboard that Raspberry Pi could work with.

Quick FACTS

- ▶ Minitels were first used to find phone numbers
- ▶ Thousands more services were later added
- ▶ The World Wide Web rendered Minitel obsolete
- ▶ A small number of QWERTY terminals were made
- ▶ This project is 100 percent open-source

The project runs Raspberry Pi OS, with the config.txt file having been edited to configure the screen size

The Minitel terminals cost about 1000 francs to make, but they were distributed free of charge. Users would pay about 60 francs per hour to use it

Gautier needed to determine the combinations for each key. "I would press a key, measure the Ohm with a multimeter, and note the combination, repeating for all keys, so I could write the Arduino code," he says

The original power button was retained, with this light indicating the system is turned on

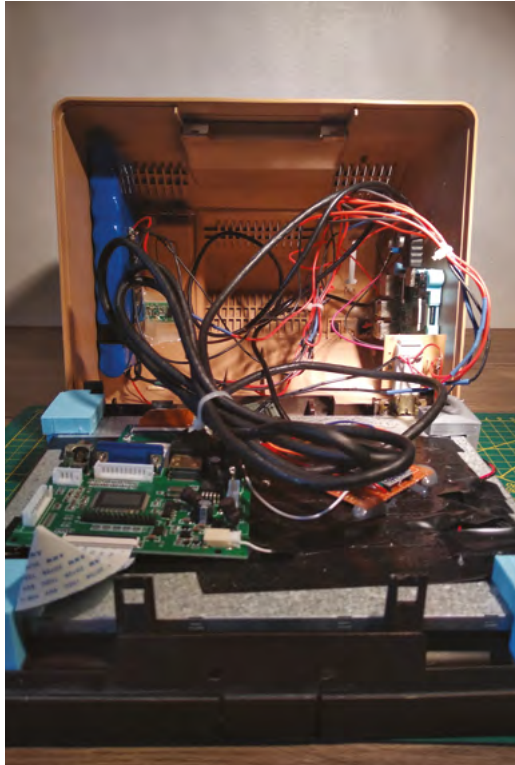


Warning!
CRT

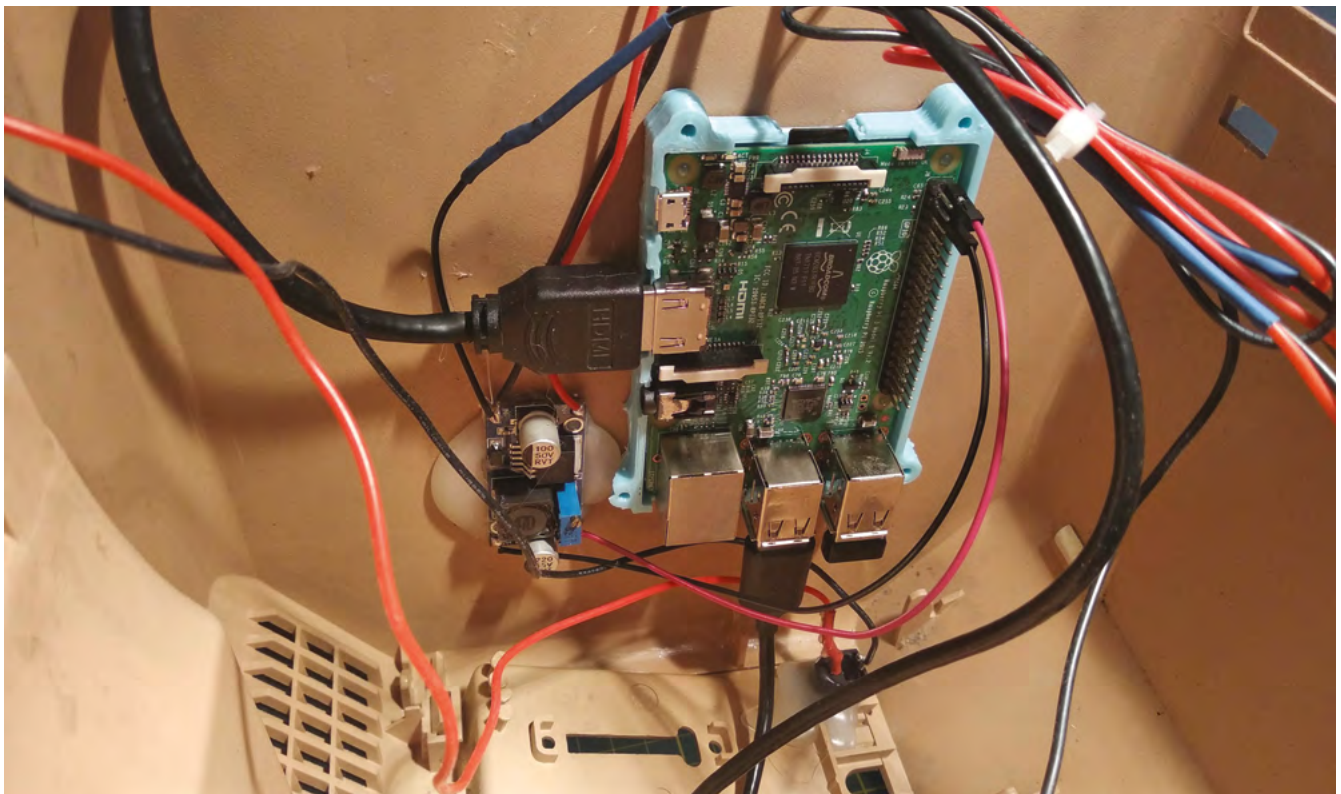
Be careful with projects involving old television and CRT equipment. Opening up a CRT can be dangerous, risking electric shock even if the TV is not plugged in.

magpi.cc/crt

- ▶ Here you can see the internal wiring, connecting the screen (laid flat) to Raspberry Pi fixed to the right of the case
- ▶ An HDMI cable connects Raspberry Pi to the display. You could easily upgrade to Raspberry Pi 4



▶ Given it dates back to 1982, Gautier's Minitel terminal - bought from a garage sale - is in surprisingly good condition




```

// 5 4 3 2 15 14 13 9
/* 6 */({'c','D','Q','S','r','l',' ',' '},
/* 7 */({'L','F','Y','A',' ','r','s',' '},
/* 17 */({'K','G','Z','T',' ','a','c',' '},
/* 16 */({'J','H','E','R',' ',' ','g',' '},
/* 12 */({'V','C',' ','U','e','r','s',' '},
/* 11 */({'B','X',' ','I','*','4','7','1'}),
/* 10 */({'P','W',' ','N','O','5','8','2'}),
/* 8 */({'M',' ','?','O','#','9','6','3'}});

```

“It's possible to plug an Ethernet cable directly into Raspberry Pi too”

“The Minitel keyboard is a matrix with a 17-wire output cable,” Gautier explains. “When you retrofit a matrix keyboard, the goal is to find the wires of the columns and the wires of the rows – in this case, there are eight rows and eight columns, making a keyboard with a maximum of 64 keys.”

Looking rosy

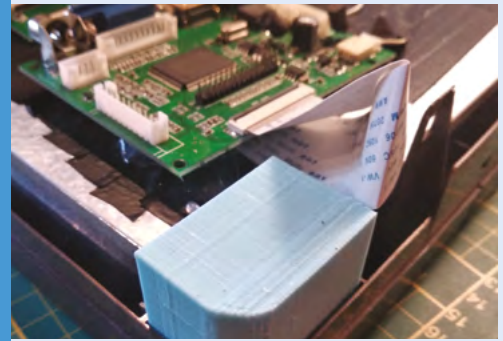
Gautier used an Arduino Pro Micro which has an ATmega32U4 microcontroller on board. “This is the easiest way to emulate a keyboard,” he says. “Other boards such as the Arduino Nano don't have the same microcontroller, so it's way more difficult to simulate a keyboard without modifying the bootloader and other things.”

From this point, the main work involved fixing a Raspberry Pi in place and connecting it to the keyboard, a DC step-down converter, and the power switch. “I placed Raspberry Pi right behind the Minitel's hatch [located behind the screen] so that it is easy to plug in a dongle for a wireless mouse,” Gautier says. “It's possible to plug an Ethernet cable directly into Raspberry Pi too.”

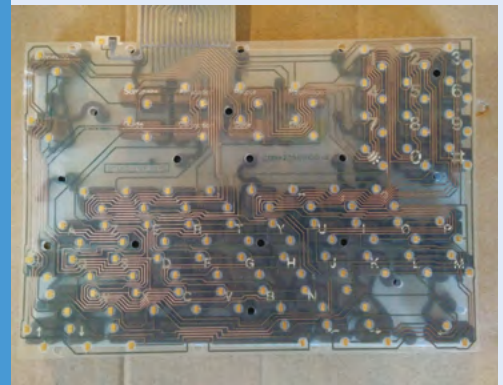
All in all, it makes for a very neat setup, one that is powered by a 12V Li-ion rechargeable battery for portability. Gautier doesn't attempt to emulate the UI of an old Minitel system – “I'm simply running Raspberry Pi OS,” he says – but it still gives a flavour of what it was like to use the system. “It's just a shame I never had the chance to use a Minitel for its original applications,” he laments. [M](#)

▲ This is the keyboard matrix Gautier used, with eight columns and eight rows

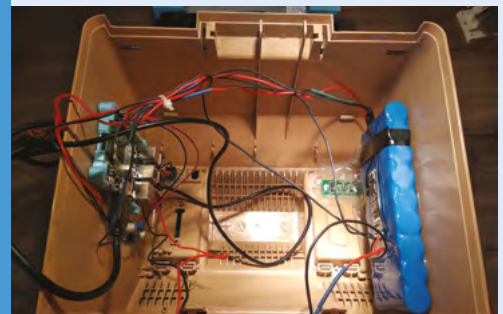
Mini-guide to revitalising a Minitel



01 The bulk of the electronics are behind the Minitel's screen and need to be removed. A new screen is fixed using printed parts. STL files of these are provided by Gautier.



02 A Minitel contains a simple matrix keyboard. The matrix is constantly scanned, allowing it to detect when a key is pressed and a circuit is closed. It must be mapped.



03 Now the case can be kitted out with Raspberry Pi (in a 3D-printed case), a battery, a light indicator, and power supplies. Hot glue is used to affix the components.

PicoLight

If you need an adjustable light, this neat little project might illuminate the way. **Nicola King** admires a glowing make



Alexandra Covor

MAKER

A maker and embedded engineer from Bucharest, Romania. She loves playing with microcontrollers and blinky LEDs, as well as combining beauty and functionality when designing PCBs.

zalmotek.com

As active members of the open-source hardware community, Bucharest-based makers **Alexandra Covor**, **Constantin Craciun**, and **Mihnea Stoica** regularly post project tutorials for the wider community. Impressive images of their projects are essential in order to ignite interest. However, finding the right light to shine on their endeavours was becoming a challenge, so like all good makers, the team decided to make their own.

PicoLight is the result – a Raspberry Pi Pico-based, minimalist adjustable light which is perfect for low-light photography, as project lead Alexandra shares: “Studio lights are great, but they are often too big and too bright for our tiny [hardware] projects, so this is where PicoLight comes in, being ideally sized.”

Small but mighty

Alexandra describes PicoLight’s user experience as “pretty simple”. It’s made up of two fully customised PCBs and an acrylic panel sandwiched together. One PCB is a LiPo battery shield including a charging circuit and a power switch, while the other PCB is a 4×4 NeoPixel matrix with a Raspberry Pi Pico soldered onto the back. It also features a rotary encoder with a push-button, “used to switch between the functioning modes: colour adjustment, brightness adjustment, and rainbow mode, in which the LEDs cycle through all the colours.”



Raspberry Pi Pico is surface-mounted on the rear of the NeoPixel matrix PCB



The 4×4 NeoPixel matrix in rainbow cycle mode

The team initially connected an external battery to Pico’s micro USB port, but Alexandra says that they soon realised that portability was key, “so we ended up designing our own battery shield as well.”

“Some of our photographer friends asked for PicoLights to use in their shoots”

All of this has resulted in a very neat-looking and compact end product. As the PicoLight measures just 52×52mm, they were a little concerned about how they would fit all of the features on it, but “then it hit us that we can actually build vertically, and add a new layer for every new feature we want to integrate, such as the battery charging module,” says Mihnea.

They’ve found it a useful tool on a day-to-day basis and while PicoLight’s main purpose is adding a “dash of colour to macro photography... we’ve also used it to add a rather nice touch to video calls,” Mihnea adds.

In the limelight

The response to PicoLight received so far has been hugely encouraging. “PicoLight ended up being much more popular than we’ve expected,” says



The rear PCB is a LiPo battery shield with a charging circuit and power switch

The main PCB hosts the 4x4 NeoPixel matrix and a Pico soldered onto the back

Diffused by an acrylic panel, the light from the NeoPixels is used to illuminate photo shoots

Quick FACTS

- ▶ PicoLight took three months to develop
- ▶ Components include a Raspberry Pi Pico along with...
- ▶ ...a custom acrylic panel, and rotary encoder with push-button
- ▶ The software run by Pico is written in CircuitPython
- ▶ See some of the team's projects at magpi.cc/zalmotek

◀ The NeoPixel PCB is sandwiched between an acrylic diffuser panel and the LiPo battery shield


Alexandra. “Some of our photographer friends asked for PicoLights to use in their shoots.”

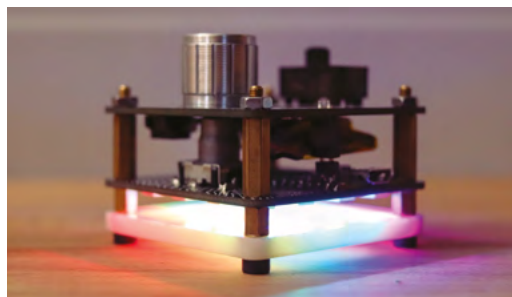
While considering a few improvements, including the possibility of using the RP2040 chip on a custom board rather than the entire Pico, Constantin tells us that since the feedback has been so positive, and everybody asked where they can buy it, they are pondering crowdfunding the project to bring it to mass production.

The team are well-accustomed to using Raspberry Pi boards in their projects. “We have worked a lot with Raspberry Pi development boards for various IoT projects,” says Constantin, “like automated points of sale, magic mirrors, and automated crypto trading bots to name a few, and we are excited about exploring the Pico microcontroller range.”

They have also, generously, made this project open-source, as Constantin explains: “We foresee using the Raspberry Pi variants in lots of future projects since they offer so much flexibility and

the support of the community is so good. We’re also actively giving back whenever we can by open-sourcing our [Raspberry] Pi-based projects.”

If you’d like to try and make your own PicoLight, step-by-step instructions are available on Hackster (magpi.cc/picolightguide), and Mihnea thinks that any intermediate-level maker with a hot plate or DIY reflow oven would be able to achieve this make, as the right tools “make short work of this project.” 



Big Boxes



Rich Thanki

MAKER

The possibilities of wireless connectivity led community-minded Rich, a self-confessed electronics tinkerer, to co-found a non-profit helping others get online.

janga.la

Social enterprise Jangala provides internet access to communities affected by disasters. **Rosie Hattersley** hears their big idea

For communities affected by man-made or natural disasters, web access provides a critical lifeline.

Not-for-profit organisation Jangala provides internet access for displaced people. The organisation was born out of its founders' involvement in humanitarian relief efforts. These included the Amatrice earthquake in central Italy in 2016 and supporting refugees living in Camp de la Lande, Calais.

Jangala's ruggedised Big Boxes provide web access in challenging environments and, with a recent Raspberry Pi upgrade, are now capable of offering connectivity to multiple users via almost any wireless means. Their mission is to connect

every school, clinic, and community resilience project worldwide. In 2019, the project won a Tech4Good Africa award, having supplied internet access to 50 students at a refugee camp in Kenya, enabling them to complete their diplomas.

The Big Easy

Jangala co-founder and head of technology Rich Thanki has been fiddling with gadgetry and taking electronics apart "as long as I can remember". A huge fan of wireless technology, Rich worked on projects with Microsoft bringing connectivity to people in sub-Saharan Africa before founding Jangala with Samson Rinaldi, who managed the CAD and hardware design aspects.

Each Big Box is a briefcase-sized device, designed to be simple to set up and take advantage of an available wireless connection to offer WiFi access to hundreds of users at a time. "Scalable high-quality WiFi without the need for costly technical expertise" characterises Jangala's single-box approach to providing internet access when and where it may be needed.

This is the first of Jangala's projects to use Raspberry Pi products. "I think the first of many," says Rich.

Jangala's updated purple box is now faster, more capable, and more rugged, the team explains, with the wireless antennas having been moved inside the box to protect them from the elements, plus twin modems that allow Big Box to connect to networks ranging from 2G networks to 5G. It was largely this aspect that prompted the switch to Raspberry Pi since it can handle the gigabit speeds that 5G networks offer.

▼ Each box provides secure wireless access, weighs just 1.9 kg, and fits in a small backpack



Quick **FACTS**

- ▶ Jangala has set up 20 Big Box installations so far
- ▶ They provide web access to more than 25,000 users
- ▶ Jangala's Get Box provided UK users with web access during lockdown
- ▶ Each Big Box has a 4G global modem
- ▶ It can also connect via satellite

The ruggedised purple boxes are briefcase sized and stuffed with enough electronics to connect hundreds of users wirelessly

Big Boxes support gigabit wireless and can be booted and updated remotely

Upgrading from a previous system to Raspberry Pi resulted in far more power, the need for a cooling system, and the ability to connect to anything from 2G to 5G



- ▲ Deployments include refugee camps such as this one for exiled Rohingyas
Credit: DFID, Rakhine camp, Myanmar
- ▼ Delivery of Big Boxes at Kakuma refugee camp, Kenya, where they will help support women and disadvantaged youths gain education and employment

Other components were sourced from “absolutely all over the place”, but the use of OpenWRT and the support of the Linux community were absolutely crucial. Rich also credits amazing manufacturers in China who designed the incredibly useful things that can’t be found anywhere else, “everything from the perfect boost converter to little waterproof slots for SIM cards”.

“ Without the stable base provided by Raspberry Pi, things would've gotten hairy quickly ”

Tech for good

“The role of Raspberry Pi in Big Box is essentially to run the show,” says Rich. “It’s really the brains and brawn of the system. We are able to control and see Big Boxes in the field by talking to Raspberry Pi. It routes every bit of traffic, which is no easy feat. We’ve a unique software setup to ensure security and it’s remotely updatable, so we have to use U-Boot as a second-stage bootloader. Without the stable base provided by Raspberry Pi, things would’ve gotten hairy quickly.”



Rich says working with Raspberry Pi 4 has been great because of the community, really accessible documentation, and standards that are open-source. “It’s made a huge difference for us. We wouldn’t have been able to do this on any other device,” he notes.

Although Big Box is Jāngala’s first Raspberry Pi experience, it won’t be their last. “Our next two projects will tackle other aspects of infrastructure which are really important in humanitarian settings, mainly power and edge computing. In both these cases, Raspberry Pi provides things that could easily be at the heart of these projects.”

For anyone else planning on designing products for social good, Rich says, “Focus on what you have to do to get the design into the hands of people and making a difference as soon as possible. This might run against the grain of people who want to be perfect and neat. Don’t let the perfect be the enemy of the good.” [M](#)

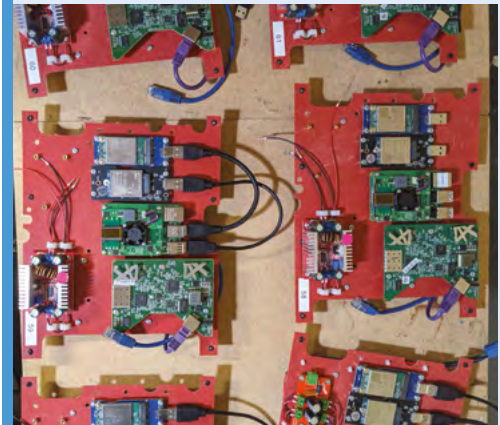
▲ Big Box and Lenovo devices support online education courses in Kalobeyei settlement, Kenya

▲ The boxes provide internet access to aid workers, refugees, and remote communities such as this one in Kenya

Big Box basics



01 Each Big Box provides internet access for more than 100 users, with additional connections added via extra radio controllers. Raspberry Pi provides the “brains” of each box.



02 The setup includes a PoE (Power over Ethernet) switch, a 4G global modem, and wireless access point, plus bespoke, highly efficient power management.



03 Due to the demanding locations in which Big Boxes are deployed, the ruggedised boxes need to be fully waterproof, including their connectors and ‘glands’.

Virtual birthday cake

This virtual birthday cake hides a delicious surprise, although **Rob Zwetsloot** had to settle with the Minecraft-powered part



Stephen Thompson

A research engineer at the Wellcome/ EPSRC Centre for Interventional and Surgical Sciences, and Code Club mentor.

magpi.cc/sthompson

While some (us) might argue that the best part of a birthday cake is the cake itself, there's definitely something to be said for an interactive model cake that does stuff in **Minecraft**. Also, there is a real cake underneath it all after you've finished with the virtual stuff.

"The virtual cake consists of a looping video of a cake created on Minecraft with nine candles that can be blown out by spinning miniature wind turbines on a matching physical model cake," Stephen Thompson, the creator of this cake, tells us. "Once all the candles are blown out, the birthday child can lift the model cake to reveal an edible cake which is mercifully free of children's saliva."

Sounds a bit more tasty and clean when you put it that way.

"I have always enjoyed making my children's birthday cakes, with a firm belief that form and function are as important as flavour," Stephen

explains. "In 2021, I was faced with the problem of not being able to blow out candles on the cake... A virtual cake controlled by miniature wind turbines seemed the obvious solution. I'd recently started developing web applications for research and education in surgery, so it made sense to reuse these skills for the virtual cake."

Virtualising baked goods

While something like this could be in some way achieved in Minecraft Raspberry Pi Edition, that was not why a Raspberry Pi was used.

"Raspberry Pi provided an easy way to interface with the analogue-to-digital converters needed to read the wind turbine voltages," Stephen says. "Along with sufficient computing power to serve a web application from within the cake itself, using Raspberry Pi enabled me use the same software architecture (Python, Flask, HTML) as I have used professionally. Sourcing the right HAT to measure nine voltages was very easy."

The virtual cake instead uses looping video overlaid with candle GIFs served via a web app in a clever way, as Stephen explains:

"The application uses Flask to serve a web page which any device on the local network can access... a JavaScript loop is used to repeatedly send a POST request to the web server asking for the voltage from one of nine analogue-to-digital converters (the ADC board can support up to 16 candles). The code to interface with the ADCs and reply to the POST requests is written in Python and utilises the Flask and gpiozero libraries. All the code is available on GitHub (magpi.cc/sthompson)."



▶ The building of the cake is a serious and exact procedure.



This cake model hides a delicious real cake underneath it

These little turbines will spin when blown upon, which is sensed by the system

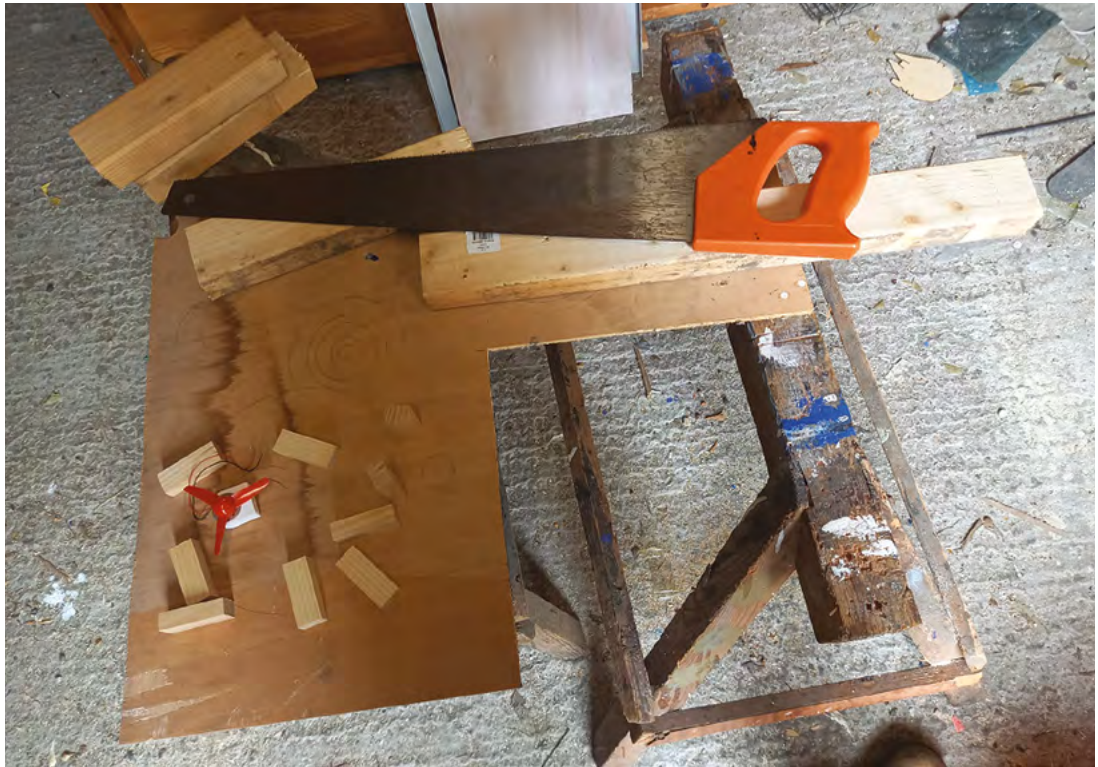
The digital candles are blown out by blowing on the turbines

Quick FACTS

- ▶ The model cake was built from spare pieces of wood
- ▶ Stephen's son, Samuel, built the Minecraft version
- ▶ Stephen's work helps develop augmented reality software for surgeons...
- ▶ ... that aims to better utilise open-source software to create open and reproducible science
- ▶ The wind turbines are actually mini generators for education



- ◀ The guests quickly figured out what they had to do to blow out the candles



▶ Sawing the wood for the model and the mini wind turbines

▶ There is space under the model cake for the electronics and the real cake



“ I was slightly nervous that the children would find it too abstract, but they worked it out pretty quickly ”

Proof in the dessert

Despite the best efforts of developers, sometimes an interface is not always clear to the user – especially when they’ve just turned nine.

“I was slightly nervous that the children would find it too abstract, but they worked it out pretty quickly,” Stephen mentions. “They saw the connection between the plastic wind turbines on the cake and the candles on the screen, and worked out that you could turn the on-screen candles off by blowing on the wind turbines. There was much laughter and smiles. The children were also very pleased to discover that there was actual cake underneath it all.”

If only there were cake underneath every interactive Raspberry Pi project. [M](#)



- ▲ A small mess of wires ready to be hooked up to Raspberry Pi
- ▶ The model was made from scratch



Cause and effect



- 01** Blowing on the wind turbines generates a small amount of voltage, enough to be detected as a change using an ADC.



- 02** A script that continually checks the voltage of each turbine written in JavaScript will get a response from Raspberry Pi via Python and Flask.



- 03** If voltage is detected from a turbine, it means it's being blown on, and the candles are 'blown out' in Minecraft by changing the animation that is being presented.

Environmental cargo logger

Keeping sweet potatoes sweet using data from a Raspberry Pi data collector. **Rob Zwetsloot** checks the results



Thinus Prinsloo

MAKER

A Regional Manager for an IT company who doesn't have to code any more but still enjoys doing it.

Getting sweet potatoes from South Africa to the Netherlands takes a while. It's a long way, and one farm wanted to figure out what the potatoes went through on their 40-or-so day journey. This is where Thinus Prinsloo came in.

"I was approached [by Zilverstroom Farms] to come up with a solution to monitor a cargo of sweet potatoes on their sea voyage to the Netherlands," Thinus tells us. "They wanted a visual record of the inside of a cargo bay for the duration of the sea voyage. The idea was to put a logger in a bin with sweet potatoes and log the environment in the bin for 40+ days, taking daily photos of the cargo. Once

the logger arrived in the Netherlands, the data were collected from the USB drive and sent back to Zilverstroom Farms."

For this project's requirements, a Raspberry Pi Zero became the obvious choice for Thinus.

"There are numerous commercial environment loggers on the market to monitor cargo, but the client also wanted to have a visual record of the whole sea voyage. The requirements for the logger system were: to be powered by battery, low power consumption, low component costs, to allow the use of custom programming code, [and] consist of easy-to-come-by, off-the-shelf components. They also wanted flexibility – the ability to reuse the system components for other projects in the future. The sensor readings and photos would also be stored on a USB flash drive for easy collection... Raspberry Pi Zero checked all the boxes for the above requirements."

Potato stats

Running continuously for 40 days is no real problem for a Linux-powered box like Raspberry Pi – there are several out there that have logged over 1000 days' uptime – however, there's no handy plug socket for Raspberry Pi to use here.

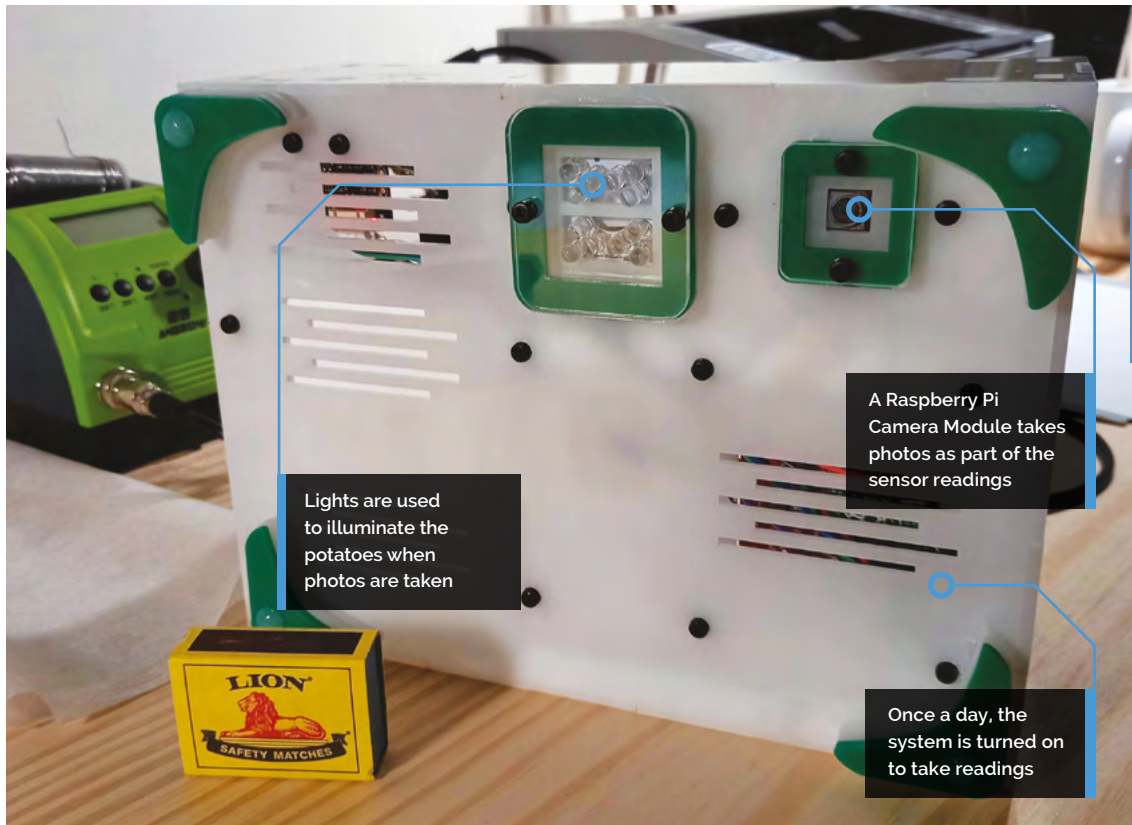
To conserve as much power as possible, a PiJuice HAT with a power supply and a real-time clock was used to power on the system once a day for just five minutes to do its job.

"Once Raspberry Pi was switched on, the temperature, humidity, and atmospheric pressure readings were taken and written to the USB flash drive," Thinus explains. "As a failsafe, the data is also written to the SD card of Raspberry Pi."

Thinus describes it working as follows. During the start-up process, crontab events at reboot are scheduled to:



▶ The components are all spread out inside the container

**Quick FACTS**

- ▶ Thanks to a mobile battery, the logger can operate for 50 days
- ▶ Otherwise, with just PiJuice, it runs for up to 30 days
- ▶ Pi Juice will only wait for ten minutes for Raspberry Pi Zero to turn on
- ▶ An enclosure was laser-cut for the project
- ▶ Some people call sweet potatoes yams, but they are not true yams



▲ Photos show how the inside of the container looks during the journey



◀ The potatoes and logger are ready for the long journey

- Take a photo after 45 seconds using a Bash script to:
 - Switch on the bright LED and take a photo with the Pi camera and save it to the SD card
 - Switch off the bright LED and switch on the infrared LED
 - Take an infrared photo with the camera and save it to the SD card
 - Copy the photos to the USB drive
- Read the sensor data after 30 seconds with a Python script:
 - Write the measurements for the temperature, humidity, and atmospheric pressure to a log file on the SD card
 - Copy the log file to the flash drive
- Set the PiJuice shutdown trigger for shutting down using a Python script.

Tuber analysis

Using results from the logger, the farm has put measures in place to reduce transit time for their sweet potatoes. The data also revealed some interesting stats, such as the big temperature shifts mainly occurring on the 1000 km journey from the farm to the harbour, and the unloading of the cargo at the destination. The potatoes also noticeably shrank over the course of the journey.

“It worked very well, and for longer than anticipated,” Thinus mentions. [M](#)

Super 8 Kamera

Switching film for Raspberry Pi-based digital capture transformed one creator's videography, as he relates to **Rosie Hattersley**



Bennet Fischer

MAKER

With a PhD in robotics and a day job working on mapping and camera-based assistance, Bennet's an accomplished coder and keen upcycler.

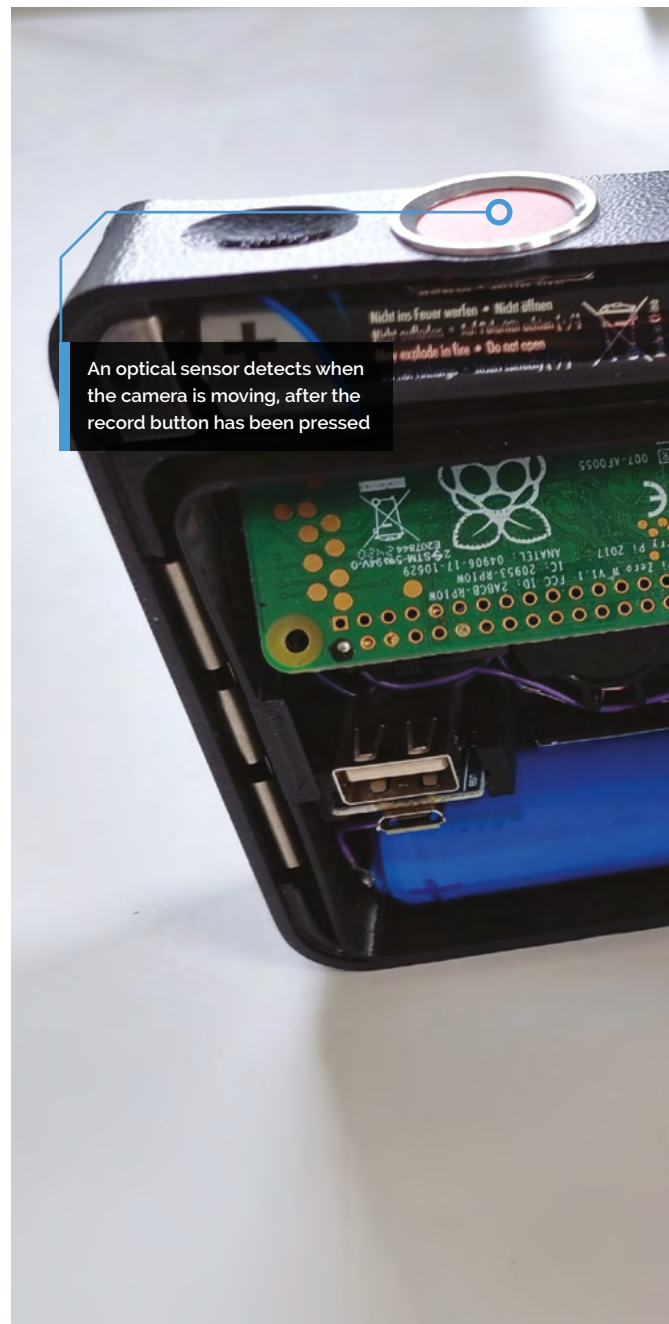
github.com/befinitiv

Over just a few short years in the early 2000s, there was an abrupt switch from film to digital photography and video, but there are plenty of analogue cameras and video cameras still around, and some that even earlier recording methods, such as 8mm cassette. Photography and film enthusiast Bennet Fischer spied an opportunity to retrofit his old film cameras and camcorders and go digital using Raspberry Pi, resulting in the Super 8 Kamera,

“An electronic film cassette that can be plugged into any Super 8 camera and make it usable again”

one of several upcycling projects he's undertaken. Modernising retro tech items is satisfying, he explains, “not only because it makes them useful again, but also because of the unique experience”, citing his enjoyment of programming Python web applications on a 286 laptop, or live-streaming full HD video on a 50-year old SLR camera.

Bennet is lucky enough to have a job that involves imagination, invention, and problem-solving. An engineer and professor of robotics, his role involves work developing mapping and camera-based assistance programs. In his free



An optical sensor detects when the camera is moving, after the record button has been pressed

time, he focuses on building projects that spark his curiosity. “Raspberry Pi is a perfect platform for such projects because it eliminates most burdens that come with building embedded devices,” he says, citing the “excellent” Raspberry Pi and open-source software communities.

Mechanical marvels

Super 8 cameras are “mechanical marvels that are destined to sit unused in drawers simply because you cannot buy the film cassettes any more,” he opines. And if you could, he calculates three



Raspberry Pi Zero W and a 5MP Raspberry Pi Camera replace the original film cartridge

Analogue video tape is hard to come by, prompting Bennet to find a way of bringing the Super 8 camera back into use

Quick FACTS

- ▶ Bennet enjoys finding new ways of using old technology
- ▶ The Super 8's original battery had leaked badly
- ▶ He had to cut out and replace most of the original wiring
- ▶ The camera attracted attention because it was quite noisy
- ▶ Few people recognised what the Super 8 camera was

◀ Replacing the obsolete 8 mm recording medium results in instantly viewable 30 fps video capture



Movie making makeover



01 This project uses Raspberry Pi Zero W, a Raspberry Pi Camera, and Bennet's adapted version of the picamera API code. Download the code from magpi.cc/campy, and the CAD files from magpi.cc/digital8cad. You'll also need an external power bank.



02 Trim down your Raspberry Pi Camera to the bare sensor chip. Be aware that cutting away its plastic parts with a craft knife requires patience, precision, and care.



03 See the instructions at magpi.cc/super8video for details of how to complete the setup, including the photoresistor and motion-detecting LED to recognise when the spindle moves.



▲ With no post-processing, the results have a similar patina to decades-old photos

minutes of footage would cost around €60 to develop. Troubled by this wasted potential, Bennet decided to create “an electronic film cassette that can be plugged into any Super 8 camera and make it usable again.” He used Raspberry Pi Zero W with a 5MP camera mounted on a flex circuit board. He chose Raspberry Pi for the Zero W’s compact size and low price, plus the “excellent software support of camera and video encoder hardware.” He was also attracted by the possibility of adding wireless live video streaming.

Another important factor was that Bennet had recently put together a very similar project for an SLR still camera – involving fitting a Raspberry Pi Zero W completely inside the camera – which had proved popular (see magpi.cc/digitalfilmcart). He hoped to use the same idea for the Super 8 camera. He was able to source the hardware including a photoresistor and motion-detecting LED from a supplier, Riechel, for around €50, and wrote code



for the Super 8 Kamera project using the custom Python program and the picamera API. This program monitors the light detector for changes (such as when the user presses the camera's record button) and starts recording and streaming at the same time. While Bennet's code is specific to the Super 8 video camera, both the project idea and the picamera API can be adapted for use with other camera and video camera models. See magpi.cc/bullseyecamera.

Fantastic facsimile

Bennet spent around ten hours designing and 3D-printing the hardware, assembling the electronics, and writing the software. Fitting everything together so it fitted within the original

Super 8 camera chassis involved three design versions. "It was quite hard to make the hardware fit into the cartridge. The camera placement and its cable proved to be an especially challenging origami riddle," he comments. The results, however, were more than worthwhile: "It surprised me how much the digital footage looks like the original Super 8 movies. I did not implement any post-processing to make it look more like the original film, but still it looks remarkably similar. This gives the movies generated by my system a lot of character and a certain patina." [M](#)

▲ Bennet has also successfully retrofitted an SLR film camera using Raspberry Pi Zero W

SUBSCRIBE TODAY FROM ONLY £5

SAVE UP TO 35%



Subscriber Benefits

- ▶ **FREE Delivery**
Get it fast and for FREE
- ▶ **Exclusive Offers**
Great gifts, offers, and discounts
- ▶ **Great Savings**
Save up to 35% compared to stores

Rolling Monthly Subscription

- ▶ Low monthly cost (from £5)
- ▶ Cancel at any time
- ▶ Free delivery to your door
- ▶ Available worldwide

Subscribe for 12 Months

£55 (UK) £90 (USA)
£80 (EU) £90 (Rest of World)

Free Raspberry Pi Zero 2 W with 12 Month upfront subscription only (no Raspberry Pi Zero 2 W with Rolling Monthly Subscription)

📞 Subscribe by phone: **01293 312193**

📧 Subscribe online: **magpi.cc/subscribe**

✉ Email: **magpi@subscriptionhelpline.co.uk**

JOIN FOR 12 MONTHS AND GET A

FREE Raspberry Pi Zero 2 W

WITH YOUR FIRST
12-MONTH SUBSCRIPTIONSubscribe in print
today and get a
FREE computer!WORTH
\$15

- ▶ A full Raspberry Pi desktop computer
- ▶ Learn to code and build your own projects
- ▶ Make your own retro games console, media player, magic mirror, and much, much more



This is a limited offer. Not included with renewals. Offer subject to change or withdrawal at any time.



Buy now: magpi.cc/subscribe



SUBSCRIBE
on app stores

From **£2.29**

Available on the
App Store

GET IT ON
Google Play

GET STARTED WITH

RASPBERRY PI

Set up your Raspberry Pi computer with Raspberry Pi OS, the latest version of the operating system and discover all the new features. By **Phil King**

Whichever model of Raspberry Pi you have, it is part of the most creative computer family on Earth. With a Raspberry Pi, you can hack, make, and build all kinds of different things. It could be a digital camera, a retro games console, or a home media centre. You could even control a sensor on board the International Space Station if you submit an entry for the Astro Pi missions (astro-pi.org).

In this guide, we show you how to get started with Raspberry Pi using the new Debian 'Bullseye' edition of Raspberry Pi OS, as well as connecting and controlling some basic electronics.



USING RASPBERRY PI OS (BULLSEYE)

Explore the latest version of the default operating system

To make a Raspberry Pi work, you'll need to install an operating system. Unless your Raspberry Pi came with one already preloaded onto a microSD card, you'll have to write the OS to the card.

While other operating systems are available for Raspberry Pi (check out our feature in *The MagPi* #111, magpi.cc/111), the official one is Raspberry Pi OS, which has recently been updated to the 'Bullseye' version of Debian Linux.

It's easy to install Raspberry Pi OS onto a microSD card using the Raspberry Pi Imager tool – download it on another computer (Windows,

Mac, or Linux) from magpi.cc/imager. See the 'Installing Raspberry Pi OS' box for details.

Upon inserting the microSD card with the OS on it into your Raspberry Pi's slot and powering up, it will first expand the file system before booting to the Raspberry Pi OS desktop. The Welcome to Raspberry Pi wizard will take you through configuration options including language and time zone, prompt you to change the default password, ask whether all of the taskbar fits on the screen, and prompt you to connect to your wireless router by entering its password.

You'll also be asked if you want to check for and install any software updates, which you can do or skip for now. With setup complete, you'll be prompted to hit Restart to reboot your Raspberry Pi. This time it'll boot straight to the desktop, without the wizard, and be ready to use.

Explore the desktop

Like most operating systems, the standard version of Raspberry Pi OS comes with a desktop interface that you can navigate with a connected mouse.

The default web browser in Raspberry Pi OS is Chromium, although you can install others such as Firefox ESR, Midori, Vivaldi, and Puffin.

Chromium is the basis for Google Chrome, so you may well find its user interface very familiar. There's an Omnibox where you can enter web

addresses or search terms. The default search engine is DuckDuckGo, which is focused on preserving the privacy of searchers, but can be changed in the Settings (after clicking on the three dots icon in the top right).

One drawback is that you can't sync your Google account in Chromium to use the same bookmarks and settings that you have in Chrome; Firefox is an alternative browser that enables cross-platform syncing. Other than that, most features

are present, including the ability to add extensions from the Chrome Web Store and also to group tabs together. You can also install web apps for

some sites such as YouTube, by clicking the option that appears on the right side of the Omnibox.

Install extra software

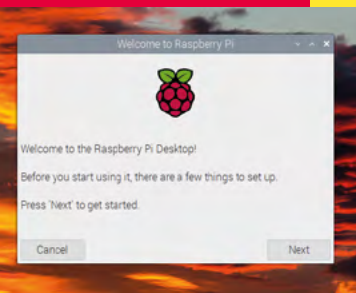
The standard Raspberry Pi OS only comes with a handful of core applications pre-installed – although there is a 'Full' version of the OS supplied with a lot more software (find it in Raspberry Pi Imager, under 'Raspberry Pi OS (other)').

It's simple to install any extra software you want, however. By far the easiest way is to use the Recommended Software tool (Menu > Preferences > Recommended Software). You can then browse a range of applications; to install one, simply tick its box and click Apply.

Applications in Recommended Software include the Claws email client and the LibreOffice productivity suite. The latter features six applications: the Writer word processor, Calc spreadsheet, Impress presentation, Draw diagrams, Base database, and Math formula editor. It can load/save Microsoft Office documents too.

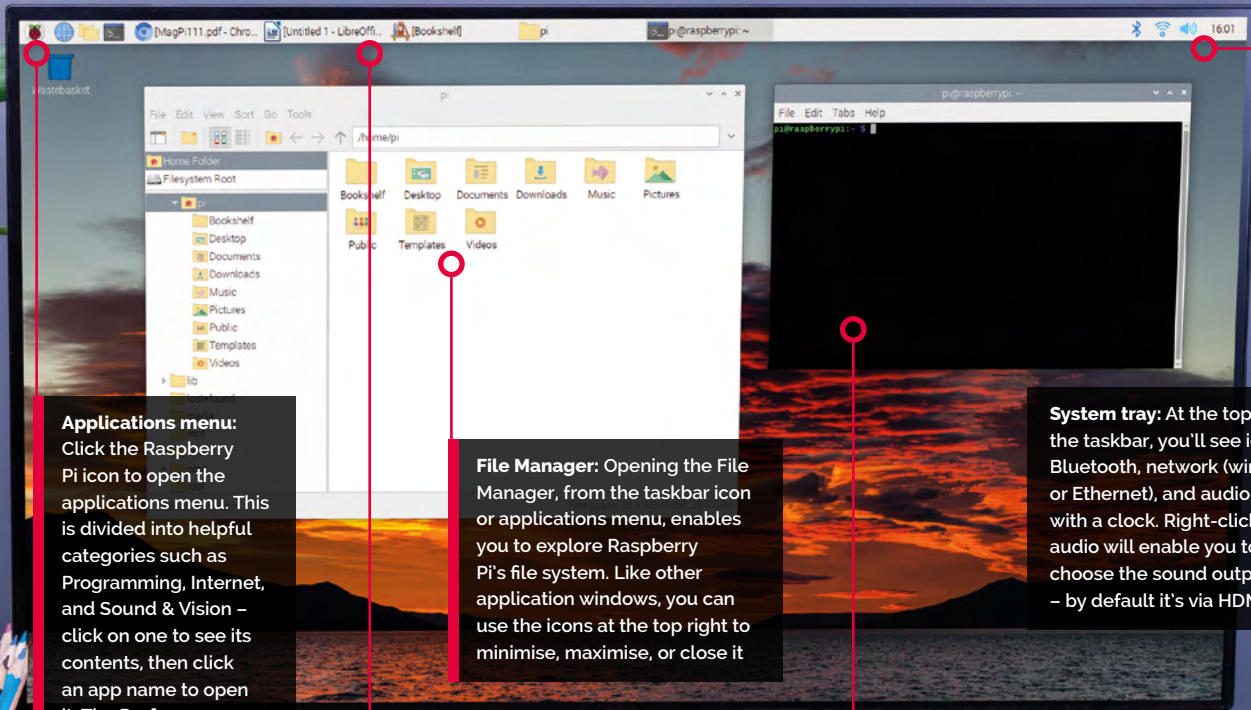
If you can't find what you need in Recommended Software, you will be able to install additional software packages using the Add/Remove Software tool, or by entering commands (such as `sudo apt install` and `sudo pip install`) in a Terminal window.

“ You'll need to install an operating system ”



▲ Upon first booting up Raspberry Pi OS, a welcome wizard will guide you through some configuration options

GET TO KNOW RASPBERRY PI OS' INTERFACE



Applications menu: Click the Raspberry Pi icon to open the applications menu. This is divided into helpful categories such as Programming, Internet, and Sound & Vision – click on one to see its contents, then click an app name to open it. The Preferences category is home to some useful tools

File Manager: Opening the File Manager, from the taskbar icon or applications menu, enables you to explore Raspberry Pi's file system. Like other application windows, you can use the icons at the top right to minimise, maximise, or close it

System tray: At the top right of the taskbar, you'll see icons for Bluetooth, network (wireless or Ethernet), and audio, along with a clock. Right-clicking audio will enable you to choose the sound output used – by default it's via HDMI

Taskbar: Shown at the top of the screen by default, the taskbar shows any open applications, including any that have been minimised. Right-clicking an empty part of the taskbar brings up the Panel Preferences dialog where you can alter its appearance and configuration

Terminal: Opening a Terminal window enables you to enter Linux commands. You may need to do this occasionally to install certain software packages

Installing Raspberry Pi OS

To install Raspberry Pi OS (or upgrade from an earlier version of Raspberry Pi OS to the latest 'Bullseye' edition), you will need to install a fresh version of the OS to your microSD card. Typically you'll do this using Raspberry Pi Imager (magpi.cc/imager) on a Windows, Apple Mac, or another Linux computer (including another Raspberry Pi). Open Raspberry Pi Imager on your other computer

and insert your microSD card (using a USB adapter if needed). Click 'Choose OS' and select 'Raspberry Pi OS (32-bit)'.

Now click 'Choose Storage' and select your inserted microSD card (which may well be labelled as 'Generic STORAGE DEVICE Media' with its storage capacity). Click Write to download Raspberry Pi OS and copy the software to the microSD card.





RASPBERRY PI OS BULLSEYE FAQ

▲ Read digital PDFs of your favourite magazines by downloading issues in the Bookshelf app

Find out what's changed in the new version of Raspberry Pi OS

With the basics of the desktop GUI and core applications covered, let's take a look at some of the new features in Raspberry Pi OS Bullseye and the differences from previous versions.

How do I get Bullseye?

Get Raspberry Pi OS Bullseye by installing a fresh installation of the latest version of Raspberry Pi OS from Raspberry Pi Imager (magpi.cc/imager).

Can I upgrade from Buster to Bullseye?

You can't upgrade from Debian Buster to Debian Bullseye using `sudo apt full-upgrade` - this only takes you to the latest version of the current (i.e. 'Buster') operating system. You have to install Bullseye onto a fresh microSD card. If you have data on your Buster installation you want to keep, we recommend copying it to a separate drive and then back to the fresh Bullseye installation.

What if I want to get Buster?

Buster is now known as Raspberry Pi OS (Legacy) and is available from Raspberry Pi's Software

page (magpi.cc/buster). It can also be found in Raspberry Pi Imager. Read more about Raspberry Pi OS (Legacy) in Gordon Hollingworth's blog post (magpi.cc/legacy).

How different does Bullseye look?

If you've used Raspberry Pi OS before, you'll immediately notice that the default wallpaper is different in Bullseye. If you like, you can change it in Preferences > Appearance Settings.

Another change you may notice is that desktop windows have a shadow effect on their borders, and animate as they open and close. Widgets and their tabs and buttons also look a little different.

In File Manager, the view modes have been simplified, with icon options to switch between icons or list mode. For more advanced options, you can use the View menu to zoom the icon size in/out and select thumbnail icon mode.

What's new in Bullseye?

One of the numerous under-the-hood changes in Raspberry Pi OS Bullseye is that the KMS (kernel mode setting) video driver is now

used by default, whereas previously it was an experimental option.

While you may not notice much of a change in how the video display works or performs in general, one major advantage of using KMS is that it's the standard open-source video driver used in Linux. This means that any application written using the standard Linux display APIs should run on Raspberry Pi without the need for modification.

In addition, now that all the display drivers for Raspberry Pi OS are part of the Linux kernel, this should make it easier for manufacturers of custom displays to add support for Raspberry Pi.

The upgrade to Bullseye brings with it an update of Chromium to version 92, which has also been optimised to use Raspberry Pi's hardware to accelerate video playback.

Bullseye takes its name from a character in Disney's *Toy Story* movie franchise

Anything else new?

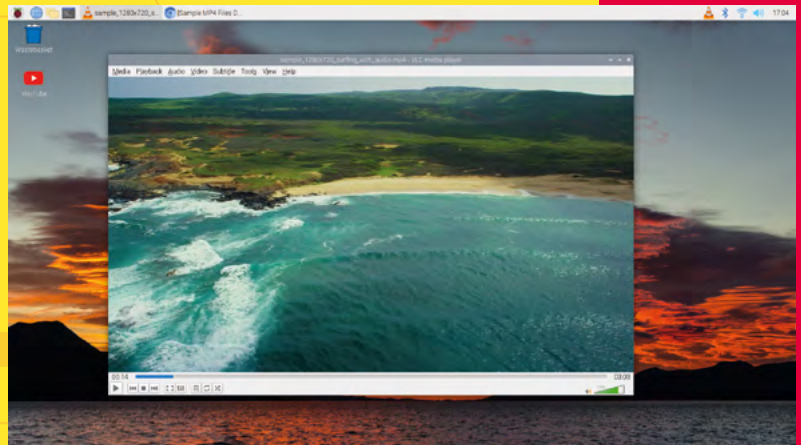
The Bookshelf application now includes free downloadable PDFs of our sister publication Custom PC, the UK's best-selling magazine for PC hardware, overclocking, gaming, and modding. Bookshelf also contains issues of *The MagPi*, HackSpace, and Wireframe magazines, along with our official Raspberry Pi books.

Will my Raspberry Pi run faster?

There's a speed boost bonus for owners of recent Raspberry Pi 4 and Raspberry Pi 400 models: the default turbo-mode clock has increased from 1.5GHz to 1.8GHz. For more details, see Eben Upton's blog post: magpi.cc/bullseyeboost. You can manually overclock many models of Raspberry Pi (magpi.cc/overclock).

Why doesn't my Raspberry Pi have the new interface?

The new look is down to the use of a new window manager called Mutter (magpi.cc/mutter). The switch to Mutter is due to the upgrade from GTK+2 to GTK+3 for user interface components. Mutter replaces the Openbox window manager used in previous versions of the OS. Openbox is still employed if your Raspberry Pi model has less than



▲ Playing a video in the VLC application; it runs very smoothly in Bullseye

2GB of RAM, because Mutter is quite demanding in terms of memory usage.

How do I manage notifications?

Raspberry Pi OS Bullseye now has a common notification manager which tells you about things like USB devices being inserted/removed and low power supply voltage, and memory allocation.

Notifications appear in grey windows under the right end of the taskbar, with the most recent messages shown at the top. Each notification will time out and automatically disappear after 15 seconds, or when you click on it.

The notification timeout setting can be altered in the Appearance tab of the Panel Preferences dialog, accessed by right-clicking the taskbar. If you set the timeout to zero, notifications will stay on screen until clicked. You can also turn notifications off completely (not advisable).

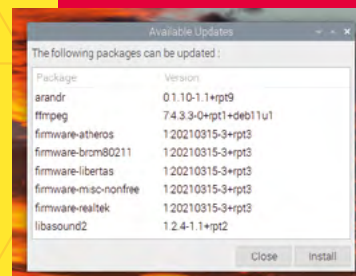
How should I keep Bullseye up-to-date?

Another major user-friendliness improvement in Bullseye is the introduction of an updater plug-in for the taskbar. Whenever you reboot Raspberry Pi, or every 24 hours if it's left powered up, it will check for any software updates and show an update icon in the taskbar with a notification telling you that updates are available.

Click the update notification to open a window showing you the packages that can be updated. Just click the Install button to start installing them; you can then carry on with whatever you were doing while they install in the background.

Why is it 'Bullseye'?

Every major version of the Debian Linux distro, upon which Raspberry Pi OS is based, takes its name from a character in Disney's *Toy Story* movie franchise. Previous versions include Wheezy, Jessie, Stretch, and Buster. If you've not seen the movies, Bullseye is Woody and Jessie's horse, one of the new characters introduced in *Toy Story 2*.



▲ Clicking the updater icon in the taskbar brings up a list of available software updates

BULLSEYE & RASPBERRY PI CAMERA SYSTEM

Get to grips with using Raspberry Pi cameras with Bullseye! By **David Plowman**

When we released our first Raspberry Pi OS image based on Debian Bullseye, we pointed to a change that is hugely important to people who have written code to use cameras with Raspberry Pi: the driver that Raspberry Pi uses to access Camera Modules has been replaced with **libcamera**.

These very significant changes mean less closed-source code, and they make it easier for people outside of Raspberry Pi to develop new camera hardware and software; but they also mean that new Raspberry Pi OS releases will no longer support the familiar raspicam apps and picamera Python library.

In the place of this older camera system is the new and almost entirely open-source camera stack based on standard Linux frameworks such as V4L2 (Video for Linux) and libcamera. Our kernel drivers have been moving in this direction for some time too, and have just recently taken further large strides towards the preferred new Media Controller architecture.

But the principal difference that users will notice is that OS releases from Bullseye onwards will no longer support the older camera system and applications, and Raspberry Pi's libcamera-apps will be built and pre-installed instead. Before we go further, note that Raspberry Pi OS Buster is still available to download if you're not ready to use Bullseye. If you are using camera applications with your Raspberry Pi, we recommend you take some time to weigh up whether to move to Bullseye at this point. A blog post on the Raspberry Pi website (magpi.cc/bullseyecamera) considers why, and why not, you might want to do so.

Raspberry Pi OS (Legacy)

A version of Raspberry Pi OS based on the older 'Buster' system is available. Read about it on this Raspberry Pi post. magpi.cc/legacy

David Plowman
is an Engineer at
Raspberry Pi

What are libcamera-apps?

Libcamera-apps are designed to copy most of the functionality that users will know from raspistill, raspivid, and raspivid. There are some unavoidable differences, which are examined in greater detail on Raspberry Pi's documentation pages (magpi.cc/libcamdiff). The new applications include:

libcamera-hello – a simple 'hello world' application that starts a camera preview stream and displays it on the screen.

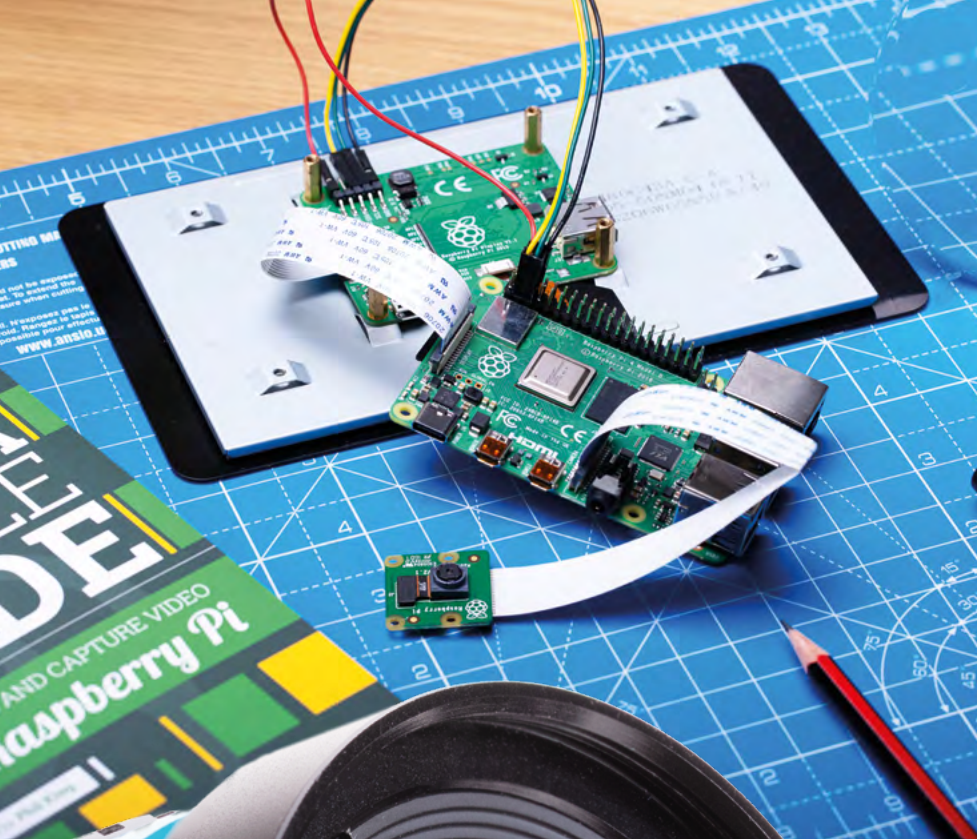
libcamera-jpeg – a simple application to run a preview window and then capture high-resolution still images.

libcamera-still – a more complex still image capture application that emulates more of the features of raspistill.

libcamera-vid – a video capture application.

libcamera-raw – a basic application for capturing raw (unprocessed Bayer) frames directly from the sensor.

libcamera-detect – this application is not built by default, but users can build it if they have TensorFlow Lite installed on their Raspberry Pi. It captures JPEG images when certain objects are detected.



Bullseye blog

For detailed information on Bullseye, straight from the horse's mouth. Read Simon Long's Bullseye blog post. magpi.cc/bullseye

“ A new camera stack based on standard Linux frameworks ”

Legacy Camera Support

Legacy Camera Support is a new option in Raspberry Pi OS (Bullseye). It replaces the “Enable/disable connection to the Raspberry Pi Camera” found in Raspberry Pi OS (Buster) and previous Raspberry Pi operating systems. (You no longer need to enable camera support in Raspberry Pi OS (Bullseye) as the libcamera API is enabled by default).

Legacy Camera Support is included in the latest update to Raspberry Pi OS. Open a terminal window and enter:

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

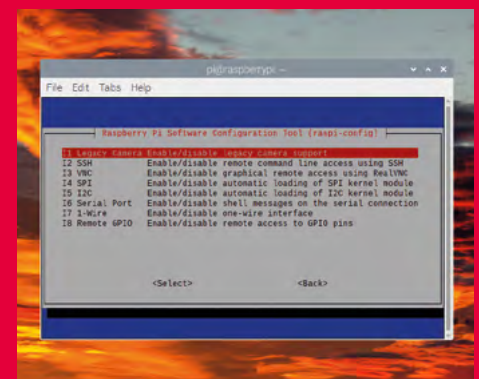
You will need to reboot Raspberry Pi OS for the changes to take effect (Menu > Shutdown and click Reboot).

Use raspi-config to enable Legacy Camera Mode. In a terminal window enter:

```
sudo raspi-config
```

Chose '3 Interface Options' and '1 Legacy Camera' and select 'Yes' in the 'Would you like to enable legacy camera support?' window. You will see a message saying 'Legacy camera support is enabled' along with 'Please note that this functionality is deprecated and will not be supported for future development'. Legacy Camera Mode will enable you to use Picamera-based projects with Raspberry Pi OS (Bullseye), but you should move your project to libcamera for future compatibility.

See the Raspberry Pi Documentation for more information on using the raspi-config tool (magpi.cc/raspiconfig).



▲ Enable Legacy Camera Mode in raspi-config

CONNECTING ELECTRONICS

Make use of Raspberry Pi's GPIO header to connect and control circuits. By **Phil King**

While Raspberry Pi can be used as an effective desktop PC, a major bonus is that it can easily be connected to electronic circuits and add-on boards (often called HATs) via its 40-pin GPIO header. This is found on every Raspberry Pi model, including the tiny Zero 2 W and also the Pi 400. See the GPIO diagram below for details of pin functions.

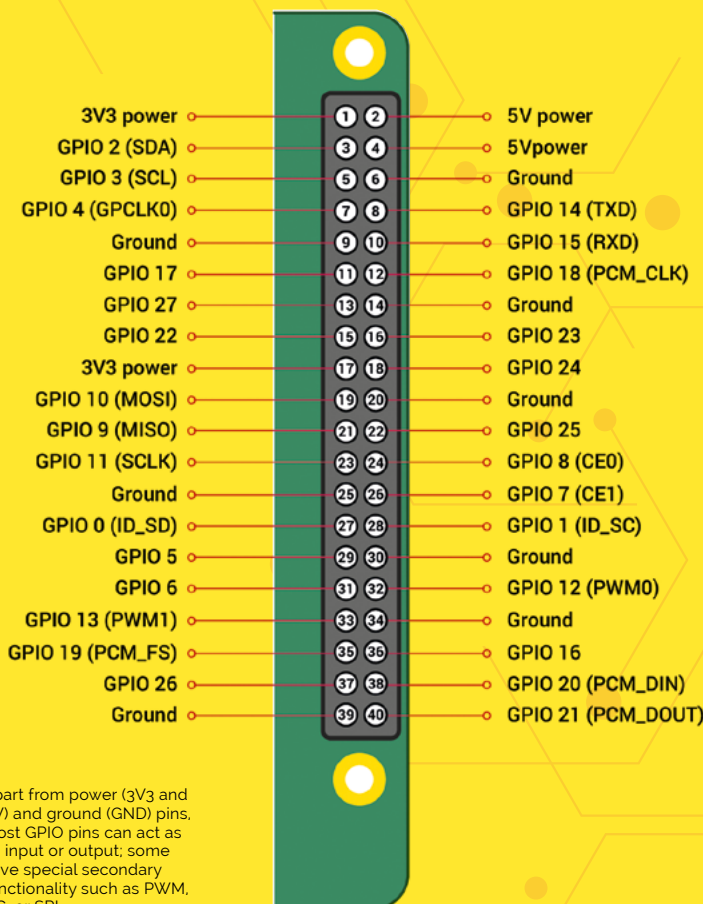
There's a wide variety of HATs (Hardware Attached on Top) and other expansion boards available for Raspberry Pi. On most Raspberry Pi models, the HAT just fits on top of the GPIO pins and is positioned over the computer's PCB. Since Raspberry Pi 400's GPIO header is located at the rear of the keyboard unit and is slightly inset, however, you may well need a ribbon cable or breakout adapter to connect a HAT.

Alternatively, connect and control your own electronic circuits to the GPIO pins using jumper wires. Standard electronic components such as LEDs, push-buttons, and sensors can be used, often via a breadboard (magpi.cc/breadboard).

JAM HAT

For this introductory electronics example we'll make things simpler by using a JAM HAT. Available from The Pi Hut (magpi.cc/jamhat), this add-on board features built-in LEDs, push-buttons, and a piezo buzzer. Just slot it onto your Raspberry Pi's GPIO header (or connect to Pi 400 via a ribbon cable or breakout adapter).

“ Set up the LEDs as traffic lights and use a push-button for a pedestrian crossing ”



▶ Apart from power (3V3 and 5V) and ground (GND) pins, most GPIO pins can act as an input or output; some have special secondary functionality such as PWM, I2C, or SPI

You are now ready to start programming your first electronics project in the popular Python language. In Raspberry Pi OS, open the Thonny IDE (Menu > Programming > Thonny IDE). The easiest way to read and control the GPIO pins in Python is to use the GPIO Zero library. This even has a special class for the JAM HAT – at the start of our code, we import this with: `from gpiozero import JamHat`.

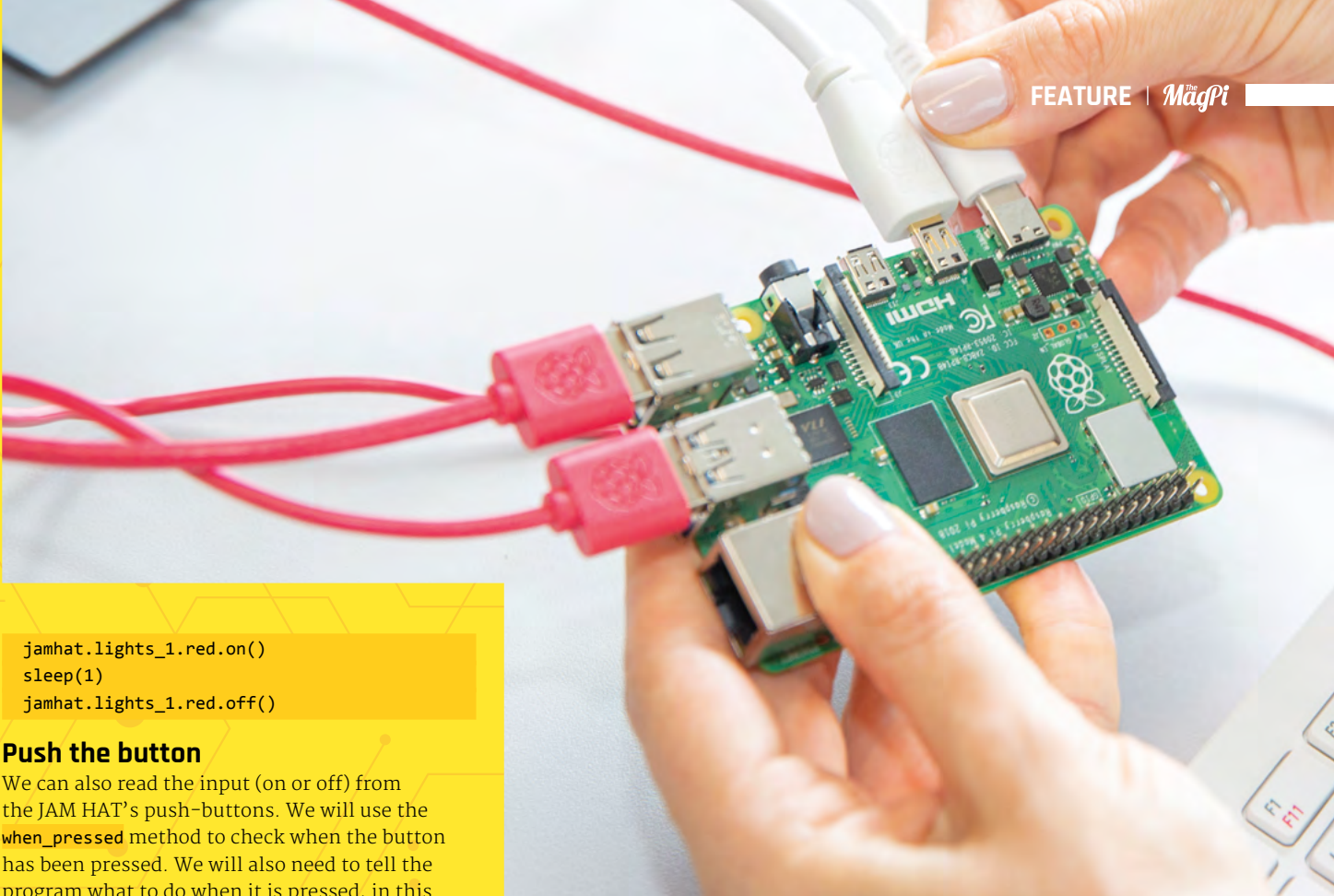
So that we can incorporate a delay in our program, we also import the `sleep` method from the time library: `from time import sleep`.

We then assign a variable, `jamhat`, to the `JamHat` class, with: `jamhat = JamHat()`.

Now, to blink the two rows (`lights_1` and `lights_2`) of LEDs in turn, we can use GPIO Zero's `blink` method:

```
jamhat.lights_1.blink()
sleep(0.5)
jamhat.lights_2.blink()
```

See the `jam-LEDs.py` listing for the full code. Run it in Thonny to see the effect. We can specify which colour LED in each row to blink. To turn on the red LED in row 1 for one second enter:




```
jamhat.lights_1.red.on()
sleep(1)
jamhat.lights_1.red.off()
```

Push the button

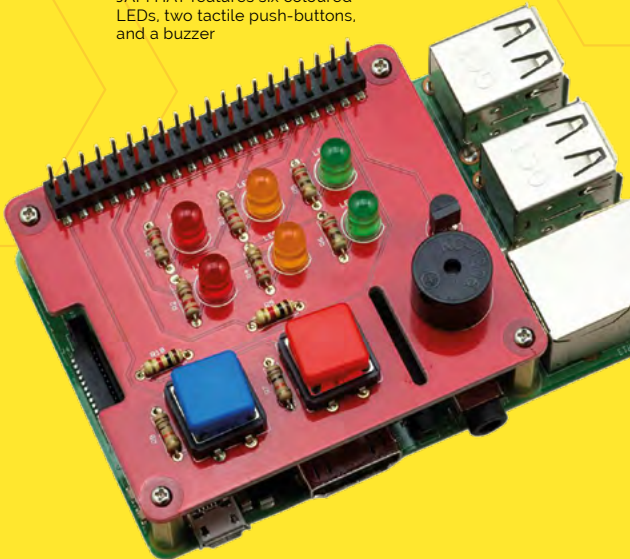
We can also read the input (on or off) from the JAM HAT's push-buttons. We will use the `when_pressed` method to check when the button has been pressed. We will also need to tell the program what to do when it is pressed, in this case turning on all the JAM HAT's LEDs and its buzzer using the `jamhat.on` command:

```
jamhat.button_1.when_pressed() = jamhat.on()
```

See the **jam-button.py** listing for the full code. Run it and every time you press button 1, the LEDs will light up and the buzzer will make a noise. Press button 2 to turn them off.

There's a lot you can do with the JAM HAT, including setting up the LEDs as traffic lights and using a push-button for a pedestrian crossing. For more details, see the JAM HAT getting-started guide (magpi.cc/jamhatguide) and further code on GitHub (magpi.cc/jamhatgit). 

▼ Making electronics easier, the JAM HAT features six coloured LEDs, two tactile push-buttons, and a buzzer



jam-LEDs.py

► Language: **Python 3**

```
001. from gpiozero import JamHat
002. from time import sleep
003.
004. jamhat = JamHat()
005.
006. jamhat.lights_1.blink()
007. sleep(0.5)
008. jamhat.lights_2.blink()
```

jam-buttons.py

► Language: **Python 3**

```
001. from gpiozero import JamHat
002.
003. jamhat = JamHat()
004.
005. jamhat.button_1.when_pressed =
    jamhat.on
006. jamhat.button_2.when_pressed =
    jamhat.off
```




Richard Hayler

MAKER

Richard is an engineer who's easily distracted by comics, music, rugby, and LEGO. His quest to combine them into a single activity is ongoing.

raspberrypi.org



Marc Scott

MAKER

Marc is a former computer science teacher, and manages informal learning content for the Raspberry Pi Foundation.

raspberrypi.org

Make a Build HAT game controller

Build a Pong paddle-style controller using Build HAT and bricks

In this project, you will use the Raspberry Pi Build HAT, a LEGO® Technic™ motor encoder and wheel, and the Python Turtle library to make a simple game controller that you can use to play Pong.

Pong is one of the earliest arcade video games, originally released in 1972 by Atari. It is a table tennis game featuring simple two-dimensional graphics. Players control paddles on each side of the screen, which they use to hit a ball back and forth.

You will learn how to read the degrees of rotation from LEGO Technic motors, how to draw and move Turtle graphics using LEGO Technic motors, and how to detect collisions between graphics using x and y coordinates.

01 Set up Build HAT

Before you begin, you'll need to have set up your Raspberry Pi computer and attached a Build HAT. If you have a Maker Plate™, mount your

Raspberry Pi to it using M2 bolts and nuts, making sure your Raspberry Pi is mounted on the side without the 'edge'.

Mounting Raspberry Pi this way round enables easy access to the ports as well as the SD card slot. The Maker Plate will allow you to connect Raspberry Pi to the main structure of your dashboard more easily (this is an optional extra).

Line up the Build HAT with your Raspberry Pi, ensuring you can see the 'This way up' label. Make sure all the GPIO pins are covered by the HAT, and press down firmly.

You should now power your Raspberry Pi using the 7.5V barrel jack on the Build HAT, which will allow you to use the motors.

You will also need to install the `buildhat` Python library. Open a Terminal window on your Raspberry Pi and enter:

```
sudo pip3 install buildhat
```

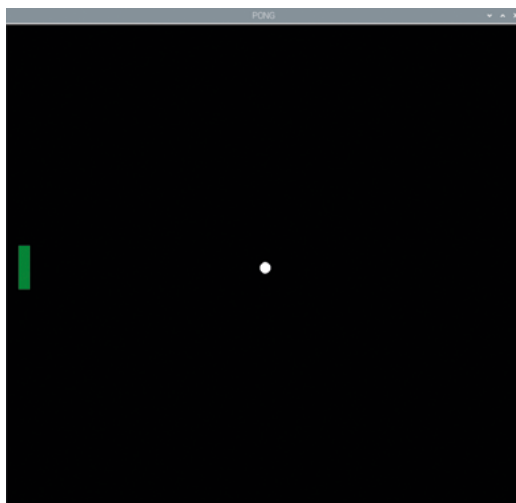
Press **ENTER** and wait for the 'installation completed' message.

02 Motor encoders

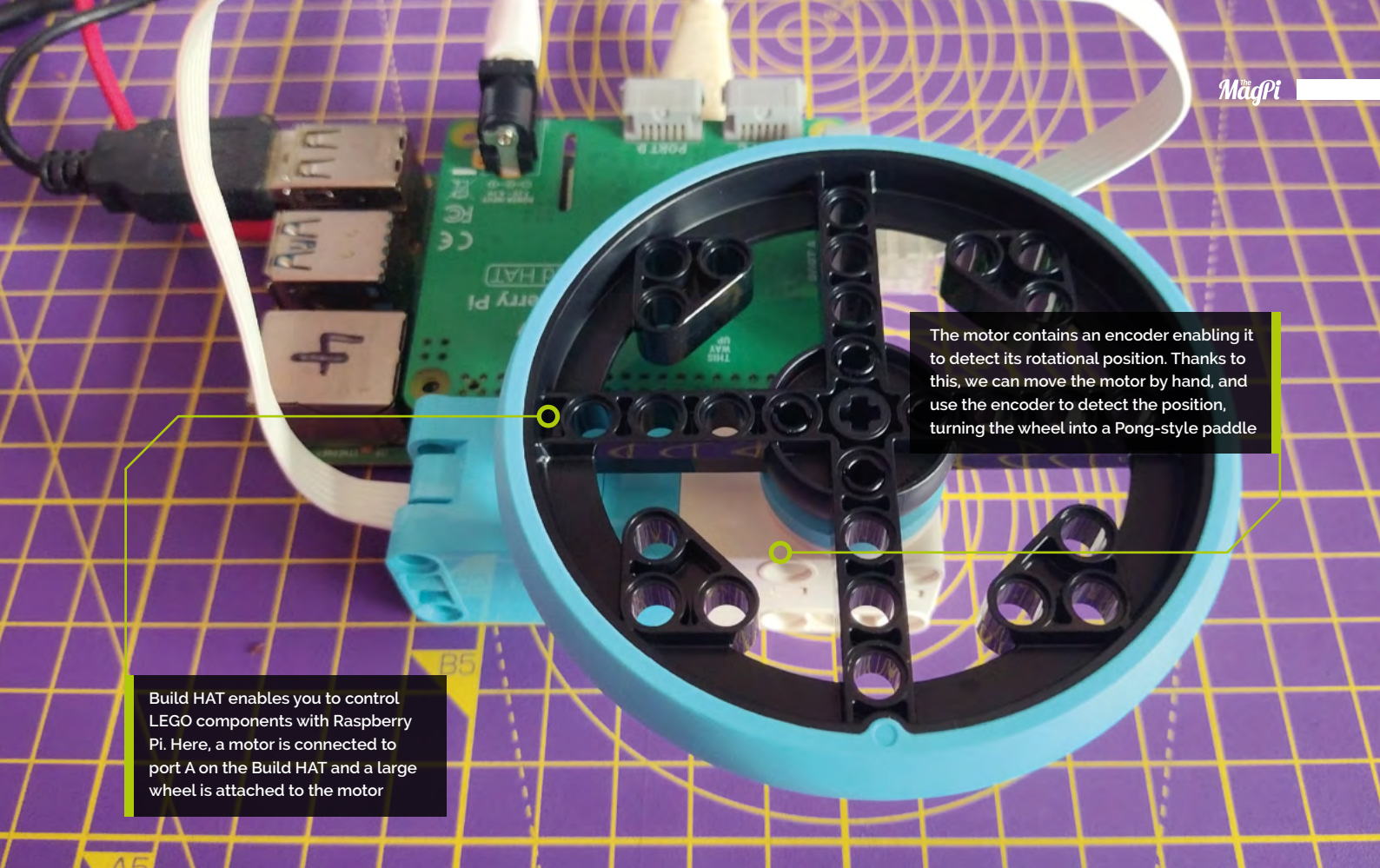
Motor encoders can not only rotate: they can also accurately detect how many degrees they have been rotated.

The LEGO Spike™ motors all have encoders. If you look at the rotating disc part of the motor, you will see a mark shaped like a lollipop that can be lined up with the 0 mark on the white body of the motor itself. This is the encoder set to zero degrees, and any angular movement of the motor shaft can be measured relative to this point.

A motor encoder, also called a rotary or shaft encoder, is an electromechanical device that allows you to record the angular position or



► Here is our early one-player Pong game, with a white ball in the centre of a black window, with a single green paddle on the far left



Build HAT enables you to control LEGO components with Raspberry Pi. Here, a motor is connected to port A on the Build HAT and a large wheel is attached to the motor

The motor contains an encoder enabling it to detect its rotational position. Thanks to this, we can move the motor by hand, and use the encoder to detect the position, turning the wheel into a Pong-style paddle

motion of the axle. It normally does this by converting the angular position to an analogue or digital output.

If a motor has an encoder, that means you can very accurately set the position of the axle. It also allows you to use the motor as an input device so that if something changes the position of the axle, this can be registered and used to trigger other actions in a computer program.

Open Thonny from the Programming menu, click on the Shell area at the bottom, and enter:

```
from buildhat import Motor
motor_left = Motor('A')
```

Depending on how well you positioned the motor at the start, you should get a value close to '0'. Move the motor and type the second line again, and see how the value changes.

03 React to motor encoder movement

To use the LEGO Technic™ motors as a controller for a game, you'll need to be able to constantly read their absolute positions.

In the main Thonny window above the Shell, you can use the commands you already know to find the absolute position of the motor. Then, in a **while True:** loop, you can print the value of the position.

Enter this code, save the file, and press Run.

```
from buildhat import Motor
motor_left = Motor('A')

while True:
    print(motor_left.get_aposition())
```

You should see that your program continually prints the position of the motor. If you rotate the motor, these values should change.

There is a better way of doing this, though. You only need to read the motor position if it is moved.

Delete the **while True:** loop from your program and create this simple function that prints the absolute position of the motor. You will also need to add another import line to use the **pause()** function.

```
from buildhat import Motor
from signal import pause
motor_left = Motor('A')

def moved_left(motor_speed, motor_pos,
               motor_a_pos):
    print(motor_a_pos)
```

Now set this function to run when the motor's encoder is moved:

```
from buildhat import Motor
from signal import pause
```

You'll Need

- Raspberry Pi computer
- Raspberry Pi Build HAT
magpi.cc/buildhat
- Assortment of LEGO, including wheels (we used a selection from the LEGO Education SPIKE Prime kit)
magpi.cc/spikeprime
- Small breadboard, buzzer, and jumper leads (optional)
- .5V power supply with a barrel jack (optional)

Top Tip



Get started

See *The MagPi* issue 112 for a quick guide to setting up Build HAT. magpi.cc/112

```
motor_left = Motor('A')

def moved_left(motor_speed, motor_pos,
motor_apos):
    print(motor_apos)

motor_left.when_rotated = moved_left
pause()
```

Run your code and you should see the values printed out in the Shell change when the motor is moved.

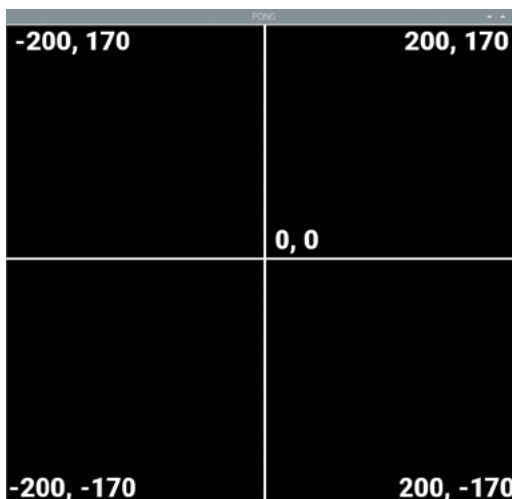
04 Make your Pong screen

Open a new file in Thonny and add the following code to import the Turtle, time, and Build HAT libraries, and then set up a screen. Run the file and you should see a black window with the title 'PONG' open.

```
from turtle import Screen, Turtle
from time import sleep
from buildhat import Motor

game_area = Screen() #Create a screen
game_area.title("PONG") #Give the screen a
title
game_area.bgcolor('black') #Set the
background colour
game_area.tracer(0) #Give smoother
animations
```

The Turtle library also has a useful way of setting the co-ordinates for a screen area. Add this line to your program:



► **Figure 1** The co-ordinates for the game area in our version of Pong

```
game_area.tracer(0)
game_area.setworldcoordinates(-200, -170,
200, 170)
```

This creates a rectangular window 400 pixels wide and 340 high, with 0 being in the centre (see **Figure 1**).

Now, you need to update your game area, to see the paddle and ball. Add a game loop to the bottom of your code, and call the `update()` method.

```
while True:
    game_area.update()
```

Run your code and you should see a black window appear. Next, you can make a ball by using a Turtle that is set to be a white circle. The ball should start in the middle of the screen, and shouldn't draw a line when it moves. Above your `while True:` loop, add the following code:

```
ball = Turtle()
ball.color('white')
ball.shape('circle')
ball.penup()
ball.setpos(0,0)

while True:
```

Run your code again. You should see a white ball appear in your window. Next, you can set up a paddle in the same way. It will be a green rectangle, and positioned on the far left of the screen.

```
paddle_left = Turtle()
paddle_left.color('green')
paddle_left.shape('square')
paddle_left.shapesize(4, 1, 1)
paddle_left.penup()
paddle_left.setpos(-190, 0)
```

Run your code and you should see your ball and paddle.

05 Move the ball

The ball is going to bounce around the screen, so two variables are needed to keep track of its speed in both the 'x' and 'y' directions. These numbers can be larger to make the game harder, or smaller to make the game easier.

```
ball.speed_x = 1
```

```
ball.speed_y = 1
```

You can check where a Turtle is by using `turtle.xcor()` and `turtle.ycor()` to find the 'x' and 'y' co-ordinates, respectively. So to make the ball move, you can combine the position and speed. Add the lines below to your program:

```
while True:
    game_area.update()
    ball.setx(ball.xcor() + ball.speed_x)
    ball.sety(ball.ycor() + ball.speed_y)
```

Run the program and see what happens! The ball should move diagonally upwards towards the top right corner of the game area... and then keep on going! If you want your game to be fast and challenging, you can increase the `speed_x` and `speed_y` values to make the ball move more quickly.

The ball should bounce off the top wall rather than disappear off the screen. To do this, the speed can be reversed, making the ball travel in the opposite direction, if its 'y' position is greater than 160.

Add the following code into your game loop and run it.

```
while True:
    game_area.update()
    ball.setx(ball.xcor() + ball.speed_x)
    ball.sety(ball.ycor() + ball.speed_y)
    if ball.ycor() > 160:
        ball.speed_y *= -1
```

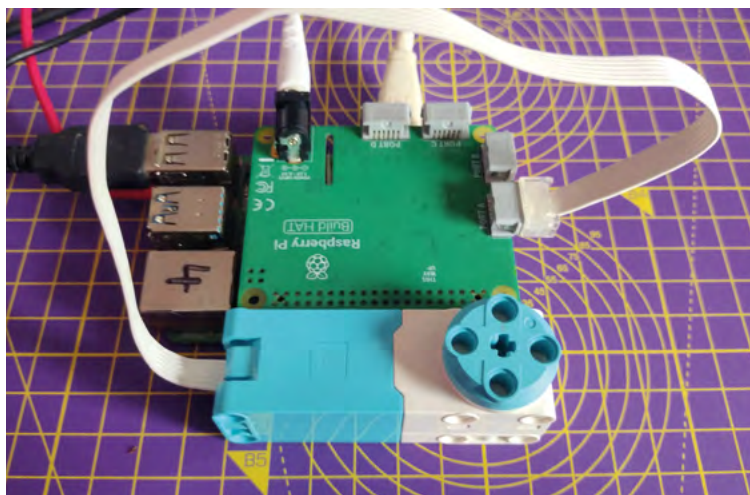
Run your code again, and the ball should bounce off the top of the screen, but disappear off the right of the screen.

In the same way that the code checks the upper 'y' position of the ball, to make it bounce, it can check the right 'x' position and the lower 'y' position, in your game loop.

Add these checks on the ball's position.

```
if ball.ycor() > 160:
    ball.speed_y *= -1
if ball.xcor() > 195:
    ball.speed_x *= -1
if ball.ycor() < -160:
    ball.speed_y *= -1
```

The ball should now bounce around the screen, and fly off the left edge. Next, you will control your paddle to reflect the ball back from the left edge.



06 Control the paddle

The LEGO Spike motor is going to be used to control the position of the paddle, but you don't want to be able to make full turns.

A simple way to limit the motion of the wheel is to add a LEGO element to prevent the wheel turning through a complete rotation.

Line up the encoder marks on your motor using the wheel, like before. Insert a peg or axle as close to level with the markers as possible.

▲ Connect a motor to port A on the Build HAT

“ The LEGO Spike motor is going to be used to control the position of the paddle ”

Add a line to create the `motor_left` object after the import line.

```
from buildhat import Motor

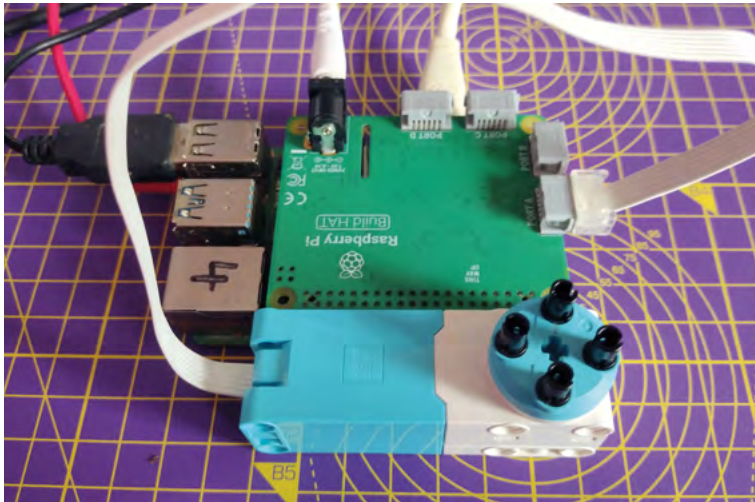
motor_left = Motor('A')
```

Now a new variable is needed to keep track of the location of the paddle. This will be called `pos_left` and set to 0.

```
ball.speed_x = 0.4
ball.speed_y = 0.4

pos_left = 0
```

Create a function for the paddle that will run when the motor encoder moves. Note that it uses a global variable so that it can change the value of the `pos_left` variable.



▲ Use pegs to attach the wheel to the motor

```
def moved_left(motor_speed, motor_rpos,
motor_apos):
    global pos_left
    pos_left = motor_apos
```

Now add a single line that will use that function each time the motor is moved. It can be just before your `while True:` loop.

```
motor_left.when_rotated = moved_left
```

Then, add a line to the `while True:` loop to update the paddle object on the screen to the new position.

```
if ball.ycor() < -160:
    ball.speed_y *= -1
    paddle_left.sety(pos_left)
```

Run your code and then turn the wheel on your motor encoder. You should see your paddle moving up and down the screen.

In case there are errors, your code should currently look like **pong.py**.

07 Paddle collisions

The game is nearly complete – but first you need to add some extra collision detection that covers the ball hitting the paddle.

Within the `while True:` loop, check if the ball's 'x' position is within the horizontal area covered by the paddle. Also use an `and` to check the ball's 'y' position is in the vertical line in which the paddle moves.

```
paddle_left.sety(pos_left)
if (ball.xcor() < -180 and ball.xcor() >
-190) and (ball.ycor() < paddle_left.ycor()
+ 20 and ball.ycor() > paddle_left.ycor() -
```

```
20):
    ball.setx(-180)
    ball.speed_x *= -1
```

Try the program out. You should be able to bounce the ball off your paddle and play a solo game of 'squash'!

Now you have a way of preventing the ball from disappearing off-screen, it's time to think about what happens if you fail to make a save. For now, let's just reset the ball back to the start.

Add this code within the `while True:` loop:

```
ball.speed_x *= -1
if ball.xcor() < -195: #Left
    ball.hideturtle()
    ball.goto(0,0)
    ball.showturtle()
```

Once you're happy with the various settings, it's time to add in the second paddle. Using what you've created for the left-hand paddle as a starting point, add a second paddle on the right-hand side of the game area.

“ Once you're happy with the various settings, it's time to add in the second paddle ”

First of all, connect a second LEGO Technic motor to the Build HAT (port B) and set it up in the program.

```
motor_left = Motor('A')
motor_right = Motor('B')
```

You can copy and paste your code for setting up your left paddle, and change the name and values for your right paddle. Create your right paddle.

```
paddle_right = Turtle()
paddle_right.color('green')
paddle_right.shape("square")
paddle_right.shapesize(4,1,1)
paddle_right.penup()
paddle_right.setpos(-190,0)

paddle_right = Turtle()
paddle_right.color('blue')
```

```
paddle_right.shape("square")
paddle_right.shapesize(4,1,1)
paddle_right.penup()
paddle_right.setpos(190,0)
```

Add a variable for the right paddle position, a function for the paddle, and the line to call the function when the right motor is moved.

```
pos_left = 0
pos_right = 0

def moved_left(motor_speed, motor_rpos,
motor_apos):
    global pos_left
    pos_left = motor_apos

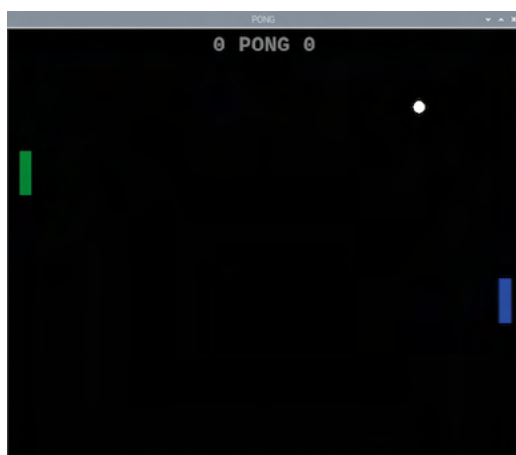
def moved_right(motor_speed, motor_rpos,
motor_apos):
    global pos_right
    pos_right = motor_apos

motor_left.when_rotated = moved_left
motor_right.when_rotated = moved_right
```

Add a line to update the paddle on screen to the `while True:` loop:

```
paddle_left.sety(pos_left)
paddle_right.sety(pos_right)
```

Currently, the ball will bounce off the right-hand wall. Modify the lines of your program that make that happen so that the ball is instead reset to the centre. Now add a similar condition for the right paddle as you did with the left, to handle collisions.



▲ A view of the game window with the score displayed at the top

pong.py

> Language: **Python 3**

**DOWNLOAD
THE FULL CODE:**



magpi.cc/pongpy

```
001. from turtle import *
002. from time import sleep
003. from buildhat import Motor
004.
005. motor_left = Motor('A')
006.
007. game_area = Screen()
008. game_area.title('PONG')
009. game_area.bgcolor('black')
010. game_area.tracer(0)
011. game_area.setworldcoordinates(-200,-170,200,170)
012.
013. ball = Turtle()
014. ball.color('white')
015. ball.shape('circle')
016. ball.penup()
017. ball.setpos(0,0)
018.
019. paddle_left = Turtle()
020. paddle_left.color('green')
021. paddle_left.shape("square")
022. paddle_left.shapesize(4,1,1)
023. paddle_left.penup()
024. paddle_left.setpos(-190,0)
025.
026. ball.speed_x = 0.4
027. ball.speed_y = 0.4
028.
029. pos_left = 0
030.
031. def moved_left(motor_speed, motor_rpos, motor_apos):
032.     global pos_left
033.     pos_left = motor_apos
034.
035. motor_left.when_rotated = moved_left
036.
037. while True:
038.     game_area.update()
039.     ball.setx(ball.xcor() + ball.speed_x)
040.     ball.sety(ball.ycor() + ball.speed_y)
041.     if ball.ycor() > 160:
042.         ball.speed_y *= -1
043.     if ball.xcor() > 195:
044.         ball.speed_x *= -1
045.     if ball.ycor() < -160:
046.         ball.speed_y *= -1
047.     paddle_left.sety(pos_left)
```


Top Tip

Turtle Race

Turtle is a drawing and animation library and you can learn more about it with the excellent Turtle Race projects. magpi.cc/turtlerace

```

if (ball.xcor() < -180 and ball.xcor()
> -190) and (ball.ycor() < paddle_left.ycor()
+ 20 and ball.ycor() > paddle_left.ycor() -
20):
    ball.setx(-180)
    ball.speed_x *= -1
if (ball.xcor() > 180 and ball.xcor() <
190) and (ball.ycor() < paddle_right.ycor()
+ 20 and ball.ycor() > paddle_right.ycor() -
20):
    ball.setx(180)
    ball.speed_x *= -1
    
```

You should now be able to enjoy a basic two-player game of Pong. Your code should currently look like `two_player_basic.py`.

08 Improve your project

There are a few additional features you can add to finish off your game. Keep track of the score by using two variables (one for each player) and update them whenever a round is lost.

First of all, declare the new variables towards the top of the program and set the score to zero.

```

score_r = 0
score_l = 0
    
```

Whenever a ball is missed, increment the appropriate score variable by one. There are two conditional tests you'll need to modify.

```

if ball.xcor() > 195: #Right
    ball.hideturtle()
    
```

▼ The rotating disc of the motor has a mark shaped like a lollipop, this can be lined up with the 0 mark on the white body of the motor. This is the encoder set to zero degrees



```

ball.goto(0,0)
ball.showturtle()
score_r+=1
if ball.xcor() < -195: #Left
    ball.hideturtle()
    ball.goto(0,0)
    ball.showturtle()
    score_l+=1
    
```

Now you need to display the score in the game area. You can use a fourth Turtle to do this. Add the following to your program after the creation of the paddle and ball Turtles, but before the `while True:` loop.

```

writer = Turtle()
writer.hideturtle()
writer.color('grey')
writer.penup()
style = ("Courier",30,'bold')
writer.setposition(0,150)
writer.write(f'{score_l} PONG {score_r}',
font=style, align='center')
    
```

You can look at the documentation for the Turtle library to see what other options there are for how the text is displayed.

“ There are a few additional features you can add to finish off your game ”

If you run your program now, the score and Pong legend should appear, but the scores themselves won't get updated.

Find the two conditionals for each of the scoring situations – when the ball is missed by a paddle and disappears to the left or right – and update the score by rewriting the new value.

```

writer.clear()
writer.write(f'{score_l} PONG
{score_r}', font=style, align='center')
    
```

09 Adding a buzzer

To include some simple sound effects, connect a buzzer to the GPIO pins on Raspberry Pi.

Instead of using a breadboard, you could use jumper leads with female sockets at both ends

two_player_basic.py

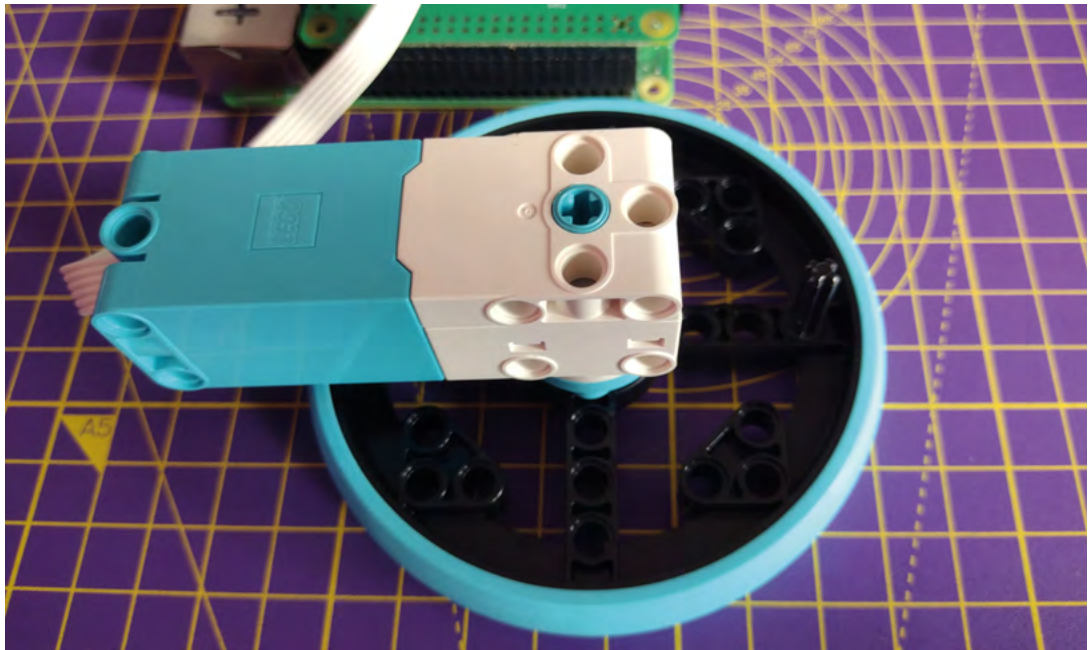
**DOWNLOAD
THE FULL CODE:**Language: **Python 3**magpi.cc/twoplayerbasicpy

```

001. from turtle import *
002. from time import sleep
003. from buildhat import Motor
004.
005. motor_left = Motor('A')
006. motor_right = Motor('B')
007.
008. game_area = Screen()
009. game_area.title('PONG')
010. game_area.bgcolor('black')
011. game_area.tracer(0)
012. game_area.setworldcoordinates(-200,-170,200,170)
013.
014. ball = Turtle()
015. ball.color('white')
016. ball.shape('circle')
017. ball.penup()
018. ball.setpos(0,0)
019.
020. paddle_left = Turtle()
021. paddle_left.color('green')
022. paddle_left.shape("square")
023. paddle_left.shapesize(4,1,1)
024. paddle_left.penup()
025. paddle_left.setpos(-190,0)
026.
027. paddle_right = Turtle()
028. paddle_right.color('blue')
029. paddle_right.shape("square")
030. paddle_right.shapesize(4,1,1)
031. paddle_right.penup()
032. paddle_right.setpos(190,0)
033.
034. ball.speed_x = 0.4
035. ball.speed_y = 0.4
036.
037. pos_left = 0
038. pos_right = 0
039.
040. def moved_left(motor_speed, motor_rpos,
motor_apos):
041.     global pos_left
042.     pos_left = motor_apos
043.
044. def moved_right(motor_speed, motor_rpos,
motor_apos):
045.     global pos_right
046.     pos_right = motor_apos
047.
048. motor_left.when_rotated = moved_left
049. motor_right.when_rotated = moved_right
050.
051. while True:
052.     game_area.update()
053.     ball.setx(ball.xcor() + ball.speed_x)
054.     ball.sety(ball.ycor() + ball.speed_y)
055.     if ball.ycor() > 160:
056.         ball.speed_y *= -1
057.     if ball.xcor() > 195:
058.         ball.hideturtle()
059.         ball.goto(0,0)
060.         ball.showturtle()
061.     if ball.ycor() < -160:
062.         ball.speed_y *= -1
063.     paddle_left.sety(pos_left)
064.     paddle_right.sety(pos_right)
065.     if (ball.xcor() < -180 and ball.xcor() >
-190) and (ball.ycor() < paddle_left.ycor() + 20
and ball.ycor() > paddle_left.ycor() - 20):
066.         ball.setx(-180)
067.         ball.speed_x *= -1
068.     if (ball.xcor() > 180 and ball.xcor() < 190)
and (ball.ycor() < paddle_right.ycor() + 20 and
ball.ycor() > paddle_right.ycor() - 20):
069.         ball.setx(180)
070.         ball.speed_x *= -1
071.     if ball.xcor() < -195: # left
072.         ball.hideturtle()
073.         ball.goto(0,0)
074.         ball.showturtle()

```


► Attach a peg to the rear of the wheel to prevent the wheel from spinning indefinitely



and poke the legs of the buzzer into the socket. Then use some LEGO elements to mount the buzzer so that it doesn't flop around and become disconnected during frantic gaming sessions.

Now add the gpiozero library to the list of imports at the start of your program:

```
from gpiozero import Buzzer
```

Then, make the buzzer available for the program to use, by setting which pin you have connected the positive (+) leg to. here, we used GPIO17.

```
buzz = Buzzer(17)
```

If you didn't use GPIO17, change the value to reflect the pin your buzzer is connected to.

Now, whenever the paddle and ball make contact, you want the game to play a short tone.

Add this line to each action part of the collision detection if conditionals for the ball and paddle:

```
buzz.beep(0.1,0.1,background=True)
```

Then add a line to play a longer tone whenever the player misses the ball.

```
buzz.beep(0.5,0.5,background=True)
```

You can read more about the options available with buzzers in the GPIO Zero documentation (magpi.cc/buzzer).

“ Add even more randomness to the speed and trajectory of the ball, and make the ball move faster ”

10 Improve the Pong game

Here are some ideas to improve your game of Pong. You can add even more randomness to the speed and trajectory of the ball, and make the ball move faster as the game progresses.

Right now the game carries on forever – consider having a target score that a player must achieve in order to win and then start a new set of rounds; change the scoring method to count how many times the players return the ball to one another, and reset when someone misses. Or, introduce some haptic feedback, so that the motors turn a small amount when a point is lost.

At the moment it doesn't matter what part of the paddle connects with the ball; it will always bounce off at the same angle as it hit. Modify the collision code so that the angle becomes more obtuse if the ball makes contact close to the end of the paddle.

Create more games that use the LEGO Technic motors as controllers.

How about a game in the style of Angry Birds, where two controllers are used to set the launch trajectory and the amount of force applied to the catapult? 🦘

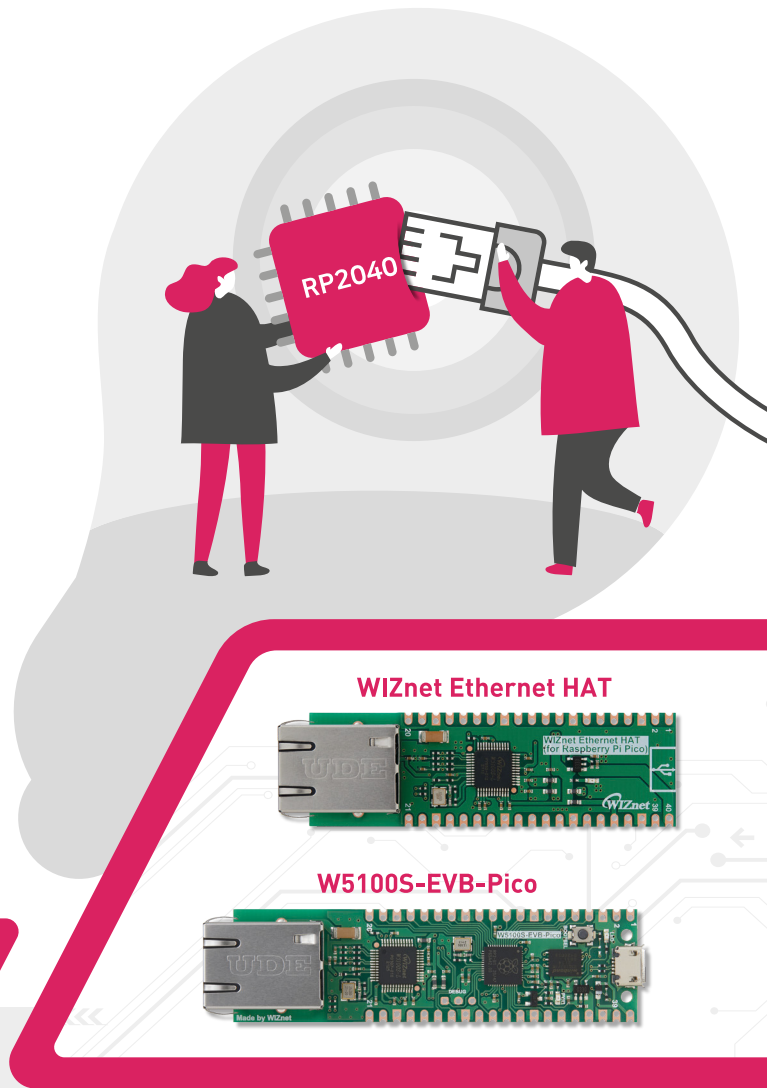
WIZnet Ethernet HAT Contest 2022

for **Raspberry Pi Pico**
and **RP2040** projects

What can you achieve if
RP2040 had access to Ethernet?

Hardware Giveaway!

Submit your idea and get one for **FREE**



Contest
begins

December 15, 2021



Idea submissions &
applications for
Hardware close

January 31, 2022



Project submissions
close

Mar 31, 2022



Winners
announced

April 25, 2022

Sum of prize for H/W & S/W winners will be **\$30,000**



\$ 5,000

1st PRIZE

X2



\$ 3,000

2nd PRIZE

X2



\$ 1,000

3rd PRIZE

X2



\$ 500

**Honorable
Mention**

X24



Explore the sensory world:

Build a simple weather station



Phil King

Long-time contributor to *The MagPi*, Phil is a freelance writer and editor with a focus on technology.

@philkingeditor

You'll Need

- ▶ DHT11 sensor
magpi.cc/dht11
- ▶ UV sensor
magpi.cc/uvsensor
- ▶ MCP3008 ADC
magpi.cc/mcp3008
- ▶ Jumper wires

Measure temperature, humidity, and UV light with this basic weather station

Previously in this series, we have built a couple of alarms: one for fire and gas safety, the other for detecting intruders.

This time we're doing something a little different by using a couple of sensors to measure certain weather conditions: temperate, relative humidity, and ultraviolet light. One of the sensors gives an analogue output, so we'll learn how to convert that to a digital reading using an ADC.

01 Connect DHT11 sensor

The DHT11 sensor measures temperature and relative humidity. We're using one from the Waveshare Sensors Pack, available from The Pi Hut (magpi.cc/wavesensors), and also sold separately. You can buy a similar DHT11 sensor elsewhere, sometimes without the PCB, or you could upgrade

to a DHT22 or use a different sensor such as a BME280 (which also reads barometric pressure).

The DHT11 combines a digital thermistor with a capacitive humidity sensor and outputs digital readings, so it's easy to use. With the power turned off, connect the DHT11 sensor to Raspberry Pi as in **Figure 1**. We're powering it from Raspberry Pi's 3V3 pin, grounding it with a GND pin (both via the breadboard side rails), and the digital output (marked DOUT on the sensor) is going to GPIO14.

02 Install DHT11 library

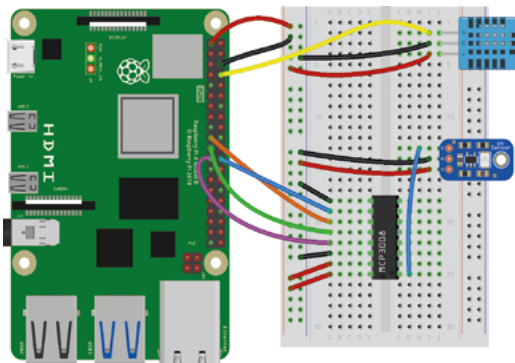
With the DHT11 sensor wired up, turn on Raspberry Pi. You should see the sensor's red power LED (on the right) light up.

Since the DHT11 outputs its data in binary form to the GPIO14 pin, we'll need a way of converting that into decimal numbers. The easiest way is to use a ready-made Python library. We're using szazo's DHT11 library, which can be installed with the following Terminal command:

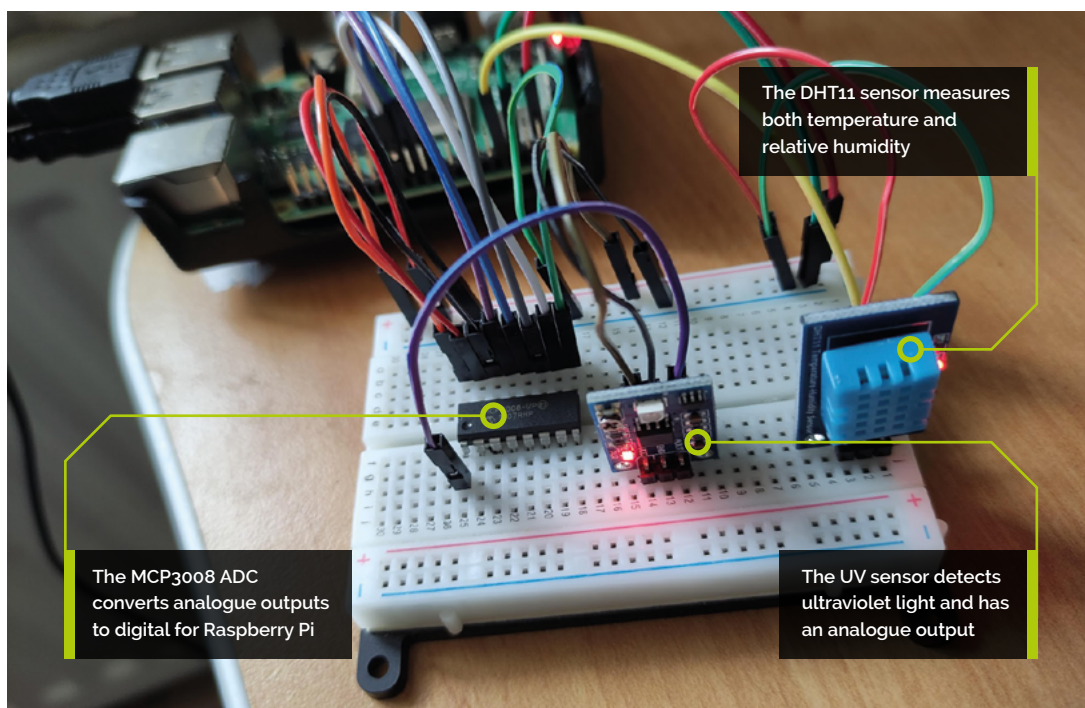
```
pip3 install dht11
```

03 Temperature and humidity test

To begin, we'll create a simple Python program (`dht11_test.py`), to read the sensor's temperature and humidity outputs. From the desktop menu, go to Programming > Thonny IDE.



▶ **Figure 1** The wiring diagram for the weather station, including the DHT11 sensor, UV sensor, and ADC



Top Tip

Off board

We've placed our DHT11 sensor on the breadboard, but you could wire it directly to Raspberry Pi's GPIO pins.

As our DHT11 library uses the RPi.GPIO Python library, we import that as 'GPIO' at the top of the code. We then initialise the GPIO and do a standard clean-up to reset all pins to inputs.

Within an infinite `while True:` loop, we read an instance of the GPIO 14 pin value. If the result is valid (i.e. not an error), we print the temperature and humidity values to the Shell. In our print statement, the `%-3.1f` format parameter sets each output to a minimum three digits including one decimal place. We add the `end = "\r\n"` parameter so the message is always printed on the same line.

Run the `dht11_test.py` code and check that you are getting realistic readings. Try breathing on the sensor to increase both temperature and humidity.

04 Add an ADC

The Waveshare UV sensor (GUVA-S12SD) we're using outputs an analogue signal, so we'll need to convert that to a digital value using an ADC (analogue-to-digital converter). We're using the MCP3008 ADC, which has eight input channels.

Since this chip uses the SPI interface, we'll need to enable SPI in the Raspberry Pi Configuration tool. It's also best to enable full SPI support in Python 3. To do so, open a Terminal window and enter:

```
sudo apt-get install python3-spidev
```

With the power to Raspberry Pi turned off, it's time to wire up the ADC. Place the MCP3008 in the

dht11_test.py

> Language: **Python 3**

```
001. import RPi.GPIO as GPIO
002. import dht11
003.
004. # initialise GPIO
005. GPIO.setwarnings(False)
006. GPIO.setmode(GPIO.BCM)
007. GPIO.cleanup()
008.
009. while True:
010.     instance = dht11.DHT11(pin = 14)
011.     result = instance.read()
012.     if result.is_valid():
013.         print("Temperature: %-3.1f C" % result.temperature,
              "Humidity: %-3.1f %" % result.humidity, end = "\r\n")
```

middle of the breadboard, straddling its central groove. Make sure it's the correct way round, as shown in **Figure 1**, with the top of writing on top of the ADC nearest Raspberry Pi.

Now connect the jumper wires as in **Figure 1**. Two of them go to the '+' breadboard power rail, connected to a 3V3 pin; two others are connected to a GND pin via the '-' rail. The four middle legs of the ADC are connected to the SPI interface on Raspberry Pi: GPIO pins 8 (CEO), 10 (MOSI), 9 (MISO), and 11 (SCLK).

uv_test.py

> Language: Python 3

```
001. from gpiozero import MCP3008
002.
003. uv = MCP3008(0)
004.
005. while True:
006.     print("UV: %-.3f V" % (3.3 * uv.value), end = "\n")
```

uv_index.py

> Language: Python 3

```
001. from gpiozero import MCP3008
002.
003. uv = MCP3008(0)
004.
005. def uv_range():
006.     global uv_mv
007.     global uv_index
008.     uv_mv = int(3300 * uv.value)
009.     if uv_mv in range(0,227):
010.         uv_index = 0
011.     elif uv_mv in range(227,318):
012.         uv_index = 1
013.     elif uv_mv in range(318,408):
014.         uv_index = 2
015.     elif uv_mv in range(408,503):
016.         uv_index = 3
017.     elif uv_mv in range(503,606):
018.         uv_index = 4
019.     elif uv_mv in range(606,696):
020.         uv_index = 5
021.     elif uv_mv in range(696,795):
022.         uv_index = 6
023.     elif uv_mv in range(795,881):
024.         uv_index = 7
025.     elif uv_mv in range(881,976):
026.         uv_index = 8
027.     elif uv_mv in range(976,1079):
028.         uv_index = 9
029.     elif uv_mv in range(1079,1170):
030.         uv_index = 10
031.     elif uv_mv >= 1170:
032.         uv_index = 11
033.
034. while True:
035.     uv_range()
036.     print("UV index: ",uv_index, end = "\n")
```

05 Connect the UV sensor

With the ADC wired up to Raspberry Pi, we can now add our UV sensor to the setup. As in **Figure 1**, we connect its VCC pin to 3.3V via the breadboard power rail, and its GND pin to GND on Raspberry Pi via the breadboard ground rail.

Finally, we connect the sensor's AOUT (analogue out) pin to the MCP3008's channel 0 pin, as shown in **Figure 1**. You could wire it to any of the eight channels on that side, but we're using this one in our code.

06 UV test

Let's create a simple program to test the sensor. In the **uv_test.py** listing, this time we're using the GPIO Zero library as it has a handy MCP3008 class, which we import at the top. We assign the **uv** variable to the MCP3008's channel 0 to read the connected sensor's analogue output.

In a **while True:** loop, we multiply the digitally converted sensor output (which ranges from 0 to 1) by the 3.3 maximum voltage to get an accurate voltage reading. If you're indoors, it should be very low. To test it, you could move the sensor outside, or just hold it through an open window. Alternatively, you could try shining an ultraviolet light (such as from a counterfeit currency detector) onto it. You should see the output value change.

07 UV index level

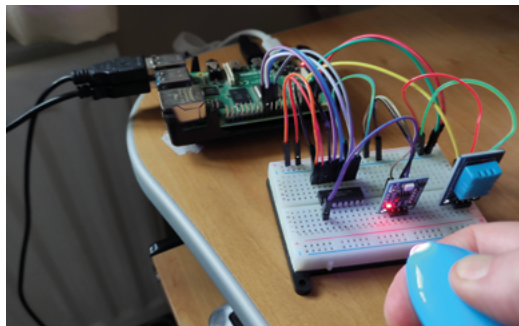
We're getting an accurate voltage reading from our UV sensor, but what does it mean? We need to convert it to the international standard UV index for it to be useful.

In the **uv_index.py** listing, we create a **uv_range** function with a series of **if** and **elif** statements to check which range of values the reading (converted into microvolts in the **uv_mv** variable) falls into and then set the UV index level (**uv_index**) accordingly.

Note: We've estimated the ranges for this based on a table of typical voltage values for each index that we found online for the sensor, but it may vary depending on temperature.

08 Bringing it together

Finally, let's combine the code with our earlier DHT11 program to read both sensors and print the appropriate weather information, in the



▲ Testing the UV sensor with ultraviolet light. It needs to be of the correct wavelength to be sensed

weather.py listing. We can use both the `RPi.GPIO` and `GPIO Zero` libraries (the latter is based on the former), but we need to initialise the `GPIO` pins before setting any `GPIO Zero` values.

We include our `uv_range` function to determine the UV index and add a call to it in the `while True:` loop. We also add its output to the `print` function.

09 Take it outside


If you want to locate your weather-sensing project outside, you'll need a weatherproof case for the electronics. Alternatively, you could even house it in a couple of connected drainpipe bends, as we used for an air sensor project in *The MagPi* issue 92 (magpi.cc/92) – in which case, it's best to use a Raspberry Pi Zero W to save space.

You'll need to power the project via an extended USB cable. Assuming you want to read the outputs from it remotely, via SSH or possibly a web dashboard, you'll also need to connect Raspberry Pi to your router – if doing so wirelessly, depending on the project's distance from the router, you may need a WiFi range extender.

Taking it further

We have made a very basic weather station. To improve it, you could add a barometric pressure sensor such as the `BMP280` – or switch out the `DHT11` for a `BME280`, which measures temperature, humidity, and pressure, but you'll need a different Python library for that.

If locating it outdoors, you could also add special weather sensors such as an anemometer (wind speed), rain gauge, and wind vane, as featured in the SparkFun `SEN-15901` Weather Meter Kit (magpi.cc/sfweatherkit).

Next time we'll use moisture and liquid level sensors to monitor a plant. See you then. 

weather.py

> Language: Python 3

DOWNLOAD
THE FULL CODE:



magpi.cc/github

```
001. import RPi.GPIO as GPIO
002. import dht11
003. from gpiozero import MCP3008
004.
005. # initialise GPIO
006. GPIO.setwarnings(False)
007. GPIO.setmode(GPIO.BCM)
008. GPIO.cleanup()
009.
010. uv = MCP3008(0)
011.
012. def uv_range():
013.     global uv_mv
014.     global uv_index
015.     uv_mv = int(3300 * uv.value)
016.     if uv_mv in range(0,227):
017.         uv_index = 0
018.     elif uv_mv in range(227,318):
019.         uv_index = 1
020.     elif uv_mv in range(318,408):
021.         uv_index = 2
022.     elif uv_mv in range(408,503):
023.         uv_index = 3
024.     elif uv_mv in range(503,606):
025.         uv_index = 4
026.     elif uv_mv in range(606,696):
027.         uv_index = 5
028.     elif uv_mv in range(696,795):
029.         uv_index = 6
030.     elif uv_mv in range(795,881):
031.         uv_index = 7
032.     elif uv_mv in range(881,976):
033.         uv_index = 8
034.     elif uv_mv in range(976,1079):
035.         uv_index = 9
036.     elif uv_mv in range(1079,1170):
037.         uv_index = 10
038.     elif uv_mv >= 1170:
039.         uv_index = 11
040.
041. while True:
042.     uv_range()
043.     instance = dht11.DHT11(pin = 14)
044.     result = instance.read()
045.     if result.is_valid():
046.         print("Temperature: %-3.1f C" % result.temperature,
"Humidity: %-3.1f %" % result.humidity, "UV index: ",
uv_index, end = "\r")
```


Composite video with Raspberry Pi Zero 2 W



PJ Evans

PJ is a writer, software engineer, and tinkerer. His ambient music collection has been described as "inspiring" (PJ Evans) and "just a load of random noise" (Mrs Evans, music teacher).

twitter.com/mrpjevans

Raspberry Pi Zero 2 W continues support for composite video out, making it a great candidate for retro projects. You'll have to get the soldering iron out, though

Every full-size model of Raspberry Pi has incorporated some form of composite video output, from the RCA connector of the original to the integrated 3.5 mm video/audio jack of current models. The Zero range also supports composite out, but requires a little work on the owner's part. The latest Raspberry Pi Zero 2 W has access and you can get a signal. Composite is a great medium for retro projects, helping to recreate the authentic look of retro games. It's also handy for connectivity where HDMI isn't an option. Here we'll show you how to get that slightly fuzzy video feed.

02 Find the pads

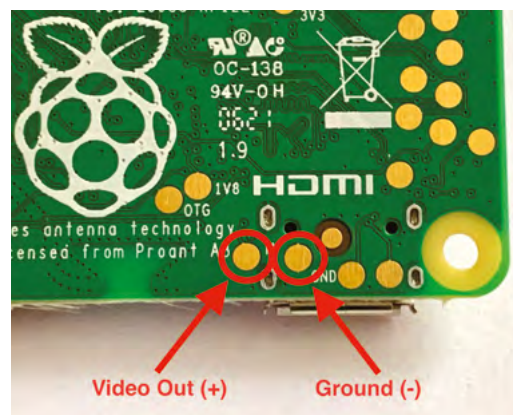
On the original Raspberry Pi Zero, composite video access was via two through-hole soldering points; on Zero 2 W, access to the composite signal has been replaced with PCB test pads on the reverse of the board. These small, flat discs are normally intended for using logic probes to test parts of a PCB. However, they have been repurposed here for their space-saving properties. On the back of the PCB, look near the HDMI connector for a pad marked TV and one near it marked GND. We're going to solder on to these.

You'll Need

- ▶ Raspberry Pi Zero 2 W magpi.cc/zero2w
- ▶ RCA male plug terminal block magpi.cc/rca
- ▶ 30AWG 0.05 mm silicon wrapped wire (or similar) magpi.cc/30awg
- ▶ TV or monitor with RCA composite video input (PAL or NTSC)

01 Choose and prepare your operating system

We're sure you'll be interested in taking the all-new Bullseye release of Raspberry Pi OS with Desktop for a spin. However, have a think about the needs of your project. If you're interested in recreating the intended aesthetic of 1980s video games, check out RetroPie (retropie.org.uk) or, if you're driving an old TV showing vintage programmes, maybe Raspberry Pi OS Lite will save you precious resources as it can play video using OMXPlayer from the command line. We're going to go for the Desktop version. As ever, prepare your microSD card, and make sure everything is up-to-date before continuing using Software Update.



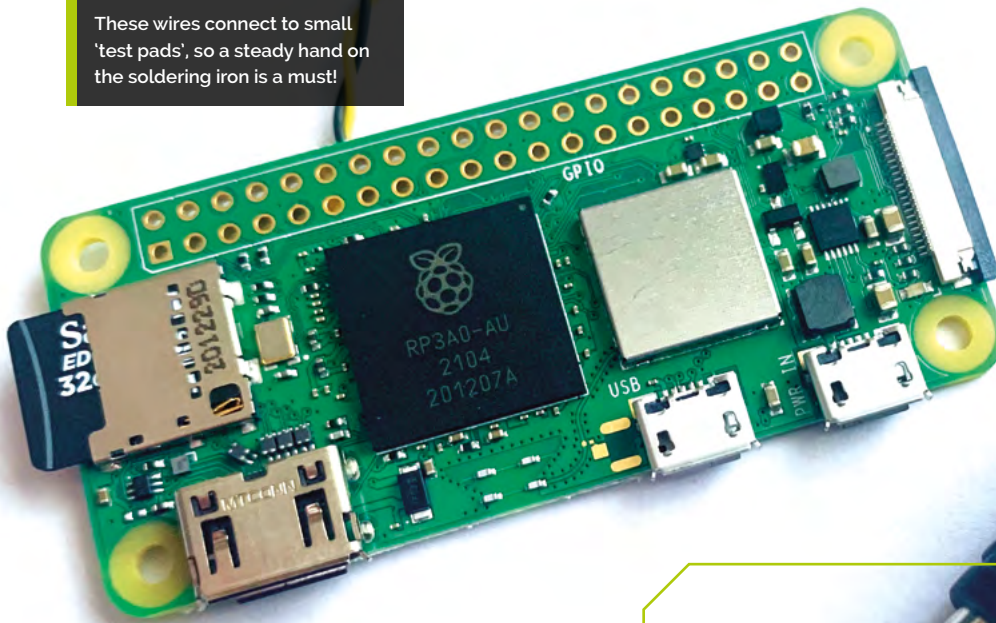
▲ These PCB test pads allow us to tap the composite video feed. Make sure you get the right ones!

**Alert!**

Although not part of this project, opening up a CRT television is very dangerous, risking electric shock. Be careful if you consider putting a Zero 2 W inside a CRT.

magpi.cc/crt

These wires connect to small 'test pads', so a steady hand on the soldering iron is a must!

**03 Measure up**

Composite video is not what is termed a 'balanced' signal. It's analogue in origin, not digital. What this boils down to is 'the shorter the better', as a composite video signal will degrade over distance. It's also sensitive to external electrical interference. As we do not want to be resoldering on to the small pads in the future, be mindful of how much wiring you need (although always allow yourself a little bit more for mistakes). As we are using very thin wire in this project, you may also want to consider an additional sleeve or heat-shrink covering to protect the signal further.

04 Prepare your wiring

As the PCB pads are so small, we've opted for a 30 AWG silicone-sleeved wire. This has two important properties. Firstly, the wire is very small, helping us to hit our target. Secondly, the silicone allows the wire to flex freely, so it is much less likely to snap off the pad through wear and tear, unlike nylon-covered wires. That said, if you don't have this type of wire, that's OK, but be careful with soldering and handling. Having worked out your optimum length, cut two equal length pieces of wire and strip the sleeve on all ends, as short as you dare on the PCB pad side, and enough on the other to fit the RCA screw terminals without poking out.

We've chosen this screw-block RCA connector for simplicity, but you can solder one in place if you prefer

05 Get soldering

Take your time with this next step. Start by coating the ends of the two wires with a small amount of solder ('tinning' or 'silvering'). Watch for any protrusions that could cause a short circuit and if the ends are too long, trim them with snips. Next, solder one wire to the TV pad and another to the GND pad (it's common to use a black sleeved wire for GND). We recommend carefully mounting the Raspberry Pi Zero 2 in a PCB holder or 'helping hands', and holding the wire in place with some masking tape.

06 Tidy up and wire up

As gently as possible, secure the wire close to the solder points so that any inadvertent pull on the wire will not cause it to snap. You could use a piece of sticky tape or a dab of hot glue. Now, if you wish, you can carefully twist the two wires together

Top Tip **What about audio?**

The Raspberry Pi Zero 2 W does not have audio out (except for HDMI), so if you need analogue audio too, try an inexpensive USB sound adapter.

**Warning!**
Hot solder

Soldering irons get very hot, and stay hot for a long time after they're unplugged. Read up on soldering before connecting the wires.

magpi.cc/soldering

09 Troubleshooting

No video? Here are some things to try. First, ensure that you have soldered to the correct pads. The positive pad is marked TV with the label to the right of the pad. The pad to the left of GND is your ground wire. It's very cramped on the PCB and this can be confusing. Also double-check you've wired GND to the negative (outer part) of the RCA connector. Make sure your TV is on the correct input. If the video quality is unacceptable, shorten the wiring or move your Raspberry Pi Zero 2 W closer to the TV.

10 Try out a retro game!

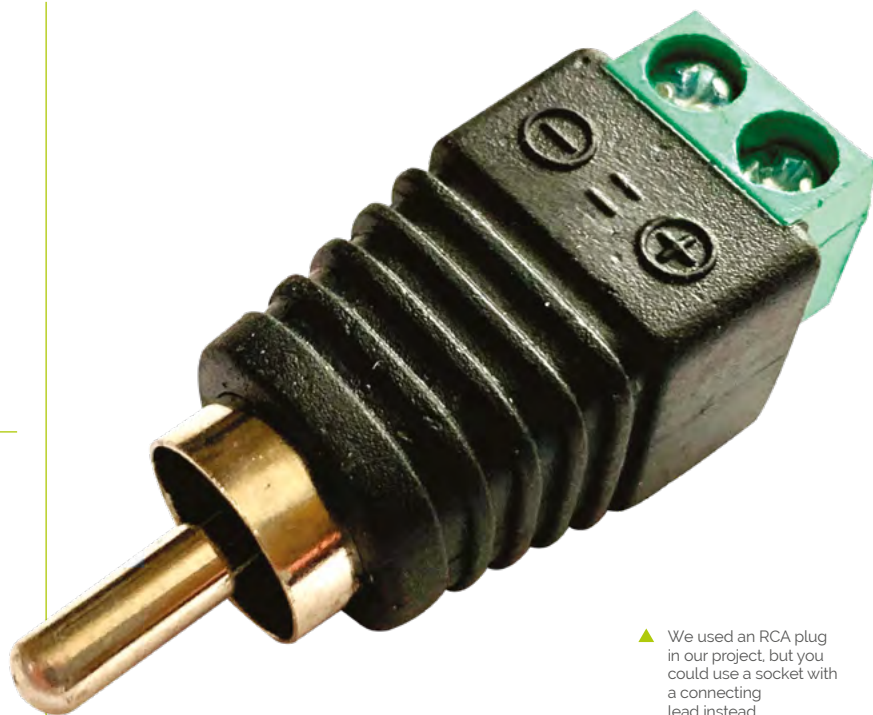
We've now covered the how of a composite video output, so let's look at the why. There is lots we can do with composite, especially in terms of easy compatibility, and one of the most fun is retro gaming. Although we can run old games using HDMI, this level of sharpness was never intended by the 1980s graphics designers, who used all kinds of tricks to use the 'fuzziness' of a composite signal to their advantage. The result of using composite video is a much more faithful representation of the artists' work. Check out RetroPie and Lakka, both great gaming platforms.

11 On old TVs

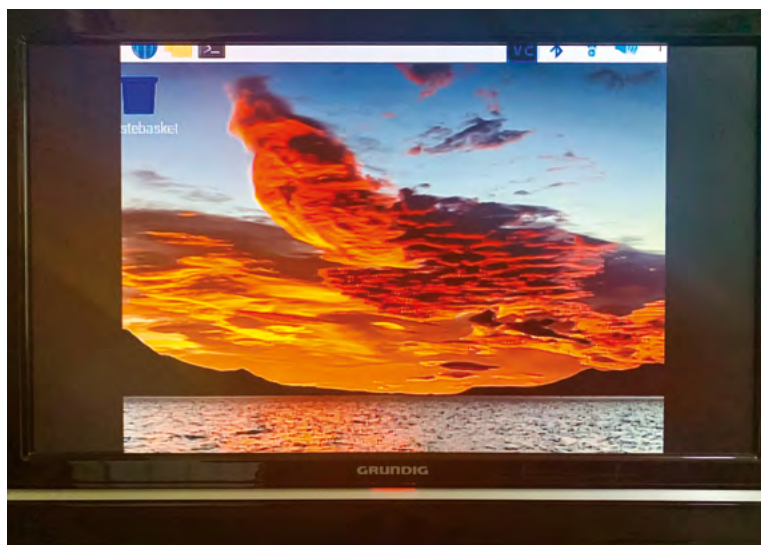
For a true retro experience, try out our Raspberry Pi Zero 2 W modification using an original CRT TV. Most will accept the composite video input using an RCA jack like ours, but some will have a SCART connector. Adapters to make these work are cheap and readily available. Although new CRT-based TVs are almost impossible to find these days, there are plenty of second-hand sets available. Just always be extra-careful with second-hand electronics. If you have a plan to place the Zero 2 W inside the TV, please read up on the dangers of opening up a CRT set – or, better still, don't do it at all.

12 Over to you

Raspberry Pi Zero 2 W is an astonishingly powerful device for its price and size. At speeds that rival even Raspberry Pi 3 and its siblings,



▲ We used an RCA plug in our project, but you could use a socket with a connecting lead instead



▲ Here's the result on a LCD monitor. This screen overscans the image, so we've lost a bit of the desktop. Most will allow you to adjust this

to be able to secrete one of these devices away and produce old-school video output is fuel for the imagination. Rather than add to landfill with that old TV set, why not create a random vintage TV show player using YouTube? Or, upcycle a broken games console for retro gaming how it should be. You could even add a Raspberry Pi Camera and add the setup to a low-cost CCTV system. As ever, it's over to you. **W**

Make a binary clock with Raspberry Pi Zero 2 W

Part art installation, part mathematical curiosity, this clock uses blocks of light to display the time as ones and zeroes



Nik Rawlinson

MAKER

Esperanto-speaking, pencil-wielding, single-board computing fan who likes hyphens and remembers what that icon on the save button depicts.

nik.co.uk

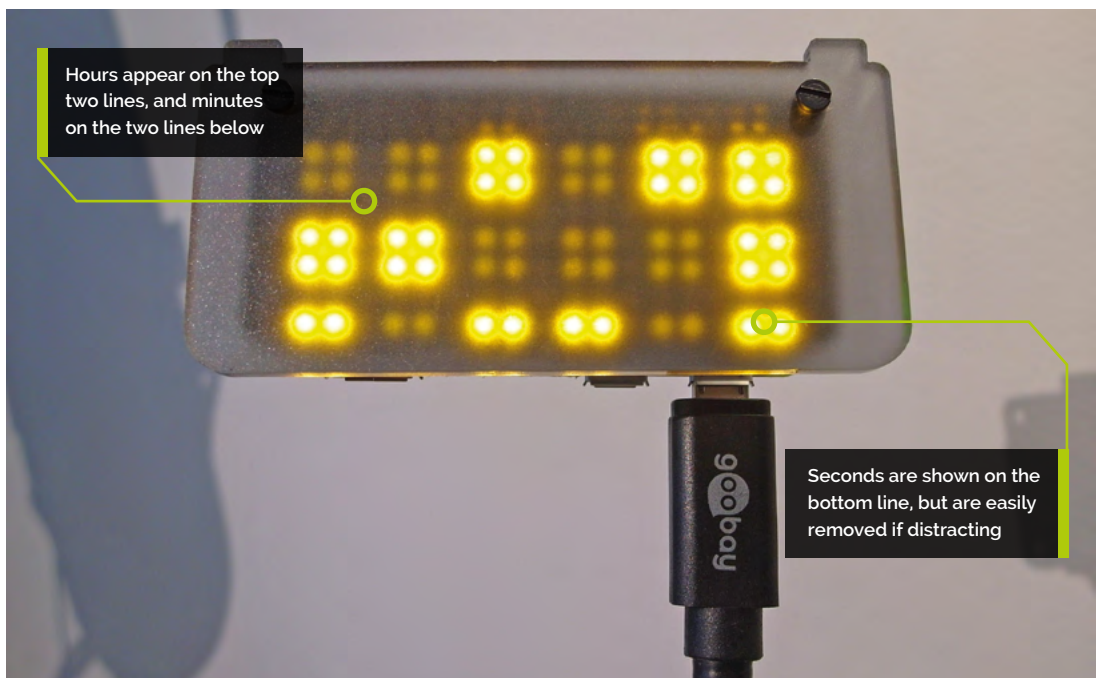
You'll Need

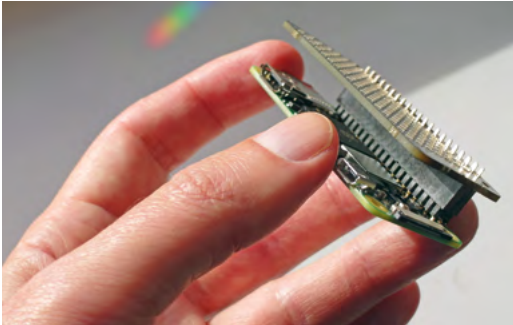
- ▶ Scroll pHAT HD magpi.cc/scrollphatd
- ▶ pHAT Diffuser magpi.cc/phatdiffuser
- ▶ Male 40-pin HAT Header or Hammer Header magpi.cc/header magpi.cc/hammerheader

Every 1960s sci-fi movie computer was awash with blinking lights, yet here we are – living in the future – and they're nowhere to be seen. Unless you have a binary clock. Rather than using hands or digits, our Raspberry Pi Zero 2 W-based timepiece opts for LEDs, illuminating them to indicate the ones and zeroes (pun coincidental) with which it tells the time. It takes a bit of practice to get used to reading the time this way, but you'll soon become proficient at adding the value of each column. In the meantime, just enjoy the hypnotic display.

01 Hack the hardware

Raspberry Pi Zero 2 W lacks the physical interface required to communicate with the Scroll pHAT HD panel, so your first job is to solder a 40-pin male GPIO header. The Scroll pHAT HD is supplied with a female header, which you'll also need to solder to the pHAT. Once they've both attached to their respective boards, press them together to make the connection. We've used Pimoroni's GPIO Hammer Header kit. The Male + Female + Installation Jig option (£7/\$10, magpi.cc/hammerheader) contains all you need,





▲ If you're using a Raspberry Pi without integral GPIO pins, you'll need to add your own

aside from the hammer, and is easier and quicker than soldering.

02 Attach the diffuser

Because the 'digits' on our clock will be rendered in binary, they'll be composed of zeroes and ones. To indicate this, we'll dimly illuminate a block of LEDs for each zero, and brightly illuminate the blocks that represent a one. To maximise the contrast, we'll set the bright LEDs to shine at 100%, but the dim ones at just 20%, and use a diffuser to soften them. This not only gives the project an attractive bokeh feel, but makes the brightest LEDs less harsh. Use the screws supplied with the diffuser to attach it to the Scroll pHAT HD, with two nuts on each screw between the diffuser and the panel to maximise the spread.

03 Enable the interface

Every Raspberry Pi, including Raspberry Pi Zero 2 W, sports the I2C multi-device bus, which is used to communicate with attached peripherals like the LED panel that we're using to display the time on our clock. Go to Menu > Preferences > Raspberry Pi Configuration, choose the Interfaces tab, and set I2C to Enabled. Set SSH to Enabled and you'll be able to access the clock remotely from another computer with SSH with:

```
ssh pi@<IP>
```

...replacing <IP> with the IP address of your Raspberry Pi (hover the mouse over your network icon in the menu bar to reveal it).

04 Install the libraries

Although Raspberry Pi OS has Python built in, it doesn't include the libraries required to address the Scroll pHAT HD. You'll need to install

these at the command prompt. In a Terminal window on Raspberry Pi, or remotely if connected via SSH, type:

```
curl https://get.pimoroni.com/scrollphatd
| bash
```

Press **ENTER**. Follow the prompts to install the libraries and included examples, which are worth examining in a code editor like Thonny, to get a better idea of the pHAT's full potential.

05 Write the code

Either use Thonny to type in the code from the **binary.py** listing here or download it from GitHub. Save it as **binary.py** in your user folder. Before we run it for the first time, we'll explain how it works. The first line:

```
#!/bin/python3
```

...tells the operating system where to find the Python interpreter, and the three lines that follow, each of which is preceded by 'import', call the libraries that the code uses to calculate and render the time. These are immediately followed by a line that informs the user that the binary clock is running, and how they can quit it. This will only appear if the routine is called at the command prompt. We'll be invoking it using cron so, in most cases, this won't be seen.



Warning!
High temperatures

If you solder the GPIO header, take care with the high-temperature soldering iron and flux.

magpi.cc/soldering

Top Tip

Seconds best

If you find the updating seconds distracting, delete lines 34 to 40 and adjust the hour and minute block heights to fill the gap.



▲ We've added a diffusion layer to soften the glow of the brightest LEDs and help them work as blocks, rather than the individual lights seen here



▲ To read the time, add up the brighter columns, which are worth 32, 16, 8, 4, 2, and 1. Here, the clock shows 11:28am (and 25 seconds)

06 Start a loop

We now begin a never-ending loop with:

```
while True:
```

Everything below this is indented because it's part of the loop. Further indents within it signify embedded loops, each of which ends when the code moves back to the left. The first thing we do inside our loop is to get the current time, with:

```
now = datetime.datetime.now()
```

This stores the year, month, date, hour, minute, second, and millisecond in the `now` variable, which is far more detailed than we need – and in the wrong format. So, we'll extract and convert the components we need.

07 Extract the binary variables

The next three lines define three variables for hour (`bhour`), minute (`bmin`) and second (`bsec`). We've preceded each with a 'b' to signify to ourselves that they're binary. Examining the `bhour` line in more detail, we're first extracting the hour from the time stamp stored in the `now` variable with `now.hour`, then converting it to binary by making it an input for the binary conversion function, like this:

```
bin(now.hour)
```

Unfortunately, this doesn't quite give us the answer we're after as the function adds two extra characters to the front of the result '0b' to signify that the string is binary, rather than decimal.

08 Strip out the surplus characters

To remove the '0b' characters we add `[2:]` to the `bin(now.hour)` function, which instructs the interpreter to slice out the first two characters. Finally, because we need our time read-out to be right-aligned, so that it always counts from the '1' column on our binary display, we add `.zfill(6)`, which pads out the result by adding sufficient zeroes to the start to make it exactly six digits long. The result, if it was 6pm (1800 hours), would therefore be `010010`. And for 6am (0600 hours) it would be `000110`. We repeat this for the minute (`bmin`) and second (`bsec`) readings. 59 minutes or seconds would become `11101`.

“ The first thing we do inside our loop is to get the current time ”

09 Understand the layout

The Scroll pHAT HD has 17 columns and seven rows of LEDs. We'll use these to create six 2x2 blocks of LEDs to represent the hours, six of the same, below, to represent minutes, and six 2x1 blocks for seconds on the bottom row. The rows are numbered from zero to six, and the columns from zero to 16, which we need to bear in mind when telling the panel where to position the upper left corner of each block. The line of code on which we specify this will also define the block's width, height, and brightness.

10 Build your blocks

We now define a new variable called `pos`, to represent the column position as we arrange the illuminated blocks, then create new loops for the hours, minutes and numbers. Each loop breaks the six binary digits of each number down into their individual ones and zeroes, so will cycle through six times apiece. In each cycle, they examine the relevant digit and, if it's a zero, set a brightness variable to 20% (0.2) or, if it's a one, to 100% (1.0). It then creates a block in memory using the `scrollphatd.fill` command, which combines the brightness, value of `pos` (column), the row, and the width and height of the block (2x2). The `pos` variable is then incremented by three, and the process repeats for the next digit in the binary value.

Top Tip

Using this in a bedroom?

Reduce the brightness values on lines 17, 25 and 33. A setting of 0.0 is off, and 1.0 is full brightness. Choose something between.



▲ If you only want to display hours and minutes, you could increase the height of those blocks to three rows, as we've done here

11 Output the time

By the time the routine has cycled through the three loops for hour, minute, and second, 18 individual blocks of light will have been defined but, so far, they only exist in memory. To send them to the pHAT, we use the command:

```
scrollphatd.show()
```

This tells Python to output the buffer to the Scroll PHAT HD. Finally, we pause for a second with:

```
time.sleep(1)
```

And then the overall loop begins again, to update the time so it's correct for the next second.


12 Launch at boot

You can run the routine from the command prompt by navigating to your user folder and typing `python3 binary.py`. However, we want this routine to launch at boot so that as soon as the computer is plugged in, it starts working as a clock. Type:

```
crontab -e
```

If you've never edited crontab before, you'll be asked which editor you want to use. Select Nano (option 1). Key down to a blank line at the bottom of the file and type:

```
@reboot python3 /home/pi/binary.py &
```

Press **CTRL+X**, press **Y** to save, then reboot. 

binary.py

DOWNLOAD
THE FULL CODE:



magpi.cc/binclock

> Language: Python

```
001. #!/bin/python3
002.
003. import datetime
004. import time
005. import scrollphatd
006.
007. print('Binary clock is running. Press ctrl-c to stop.')
008.
009. while True:
010.     now = datetime.datetime.now()
011.     bhour = bin(now.hour)[2:].zfill(6)
012.     bmin = bin(now.minute)[2:].zfill(6)
013.     bsec = bin(now.second)[2:].zfill(6)
014.     pos = 0
015.
016.     for x in bhour:
017.         if x == '0':
018.             bright = 0.2
019.         else:
020.             bright = 1.0
021.         scrollphatd.fill(bright,x=pos,y=0,width=2,height=2)
022.         pos = pos + 3
023.     pos = 0
024.
025.     for x in bmin:
026.         if x == '0':
027.             bright = 0.2
028.         else:
029.             bright = 1.0
030.         scrollphatd.fill(bright,x=pos,y=3,width=2,height=2)
031.         pos = pos + 3
032.     pos = 0
033.
034.     for x in bsec:
035.         if x == '0':
036.             bright = 0.2
037.         else:
038.             bright = 1.0
039.         scrollphatd.fill(bright,x=pos,y=6,width=2,height=1)
040.         pos = pos + 3
041.
042.     scrollphatd.show()
043.     time.sleep(1)
```




Figure 1 ♦ The white buttons are used to play notes, the coloured buttons are used to move between tracks. The cheese was delicious!

Make a Pico musical cheese box

Create syncopated rhythms and computer-generated music using a Pico



Rob Miles

🐦 @robmiles

Rob Miles has been playing with hardware and software since almost before there was hardware and software. You can find out more about his so-called life at robmiles.com.

Make a portable MIDI controller that you can use to create complex rhythms and musical sequences and use as a tiny keyboard. In a cheese box. The article 'Make Music with Pico', in HackSpace

magazine issue 46, showed how easy it is to use a Raspberry Pi Pico device to send MIDI notes to a synthesizer. We are going to build on that to create a MIDI instrument, the 'Pico musical cheese box', as shown in **Figure 1**. The musical cheese box uses a ring of twelve pixels to display notes, and twelve buttons for user input. You can use the cheese box with a synthesizer program that connects to MIDI, and there is also a specially written Pure Data patch for it.

INSIDE THE BOX

Figure 2 shows the contents of the box. There are quite a lot of wires, but the task of assembly wasn't too onerous. The box contains a twelve NeoPixel ring and twelve buttons which are connected to a Raspberry Pi Pico. Each button has its own dedicated input pin on the Pico.

Figure 3 shows the circuit for the box. The button signals are pulled low when a button is pressed. To simplify the wiring, the ground signal is connected in a ring around the switches. The pixel ring display only needs a single data connection, along with power and ground connections.

The prototype used wire wrap wire for the connections. One end was wrapped round pins on

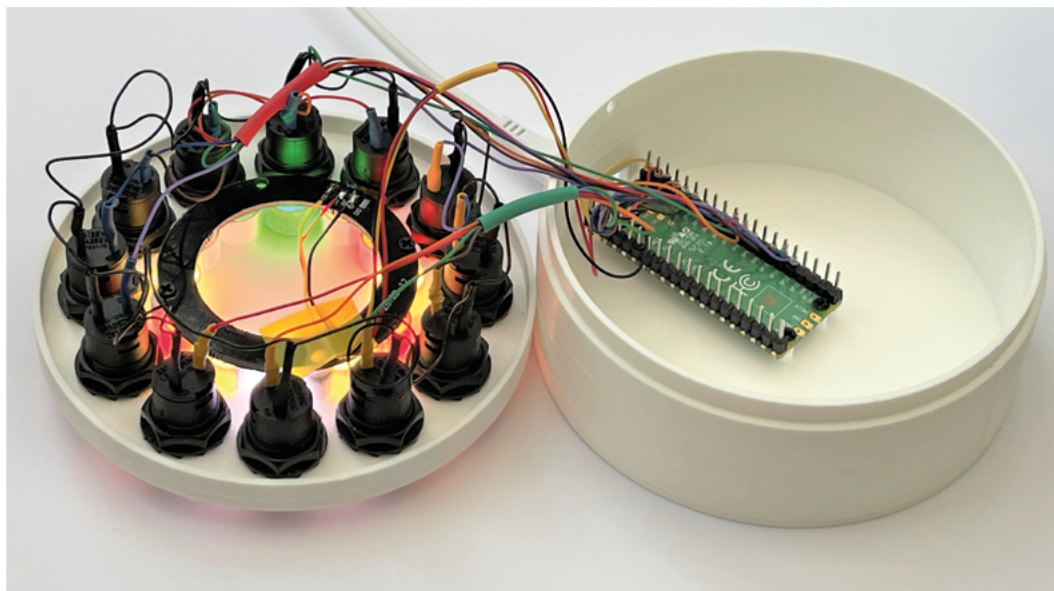


Figure 2 ♦ The cables have been organised in two sets of six coloured wires (a red set and a green set), plus three (the yellow set) for the pixel. And there are still plenty of spare pins for more buttons

the Pico device, the other end was hooked inside the hole in the pin on the switch, and then soldered in place as shown in **Figure 4**. All the switch connections were encased in heat-shrink tubing.

HARDWARE MAPPING

The buttons are used by the CircuitPython program running inside the Pico that controls the device. The program contains **Note** and **Select** classes to represent the two types of button on the device. Note and Select objects are created with hardware pins and pixel number settings when the box starts.

```
notes = (
    Note(pin=board.GP6, pixel=11, num=0),
    Note(pin=board.GP5, pixel=10, num=1),
    Note(pin=board.GP4, pixel=9, num=2),
    Note(pin=board.GP3, pixel=8, num=3),
    Note(pin=board.GP2, pixel=7, num=4),
    Note(pin=board.GP1, pixel=6, num=5),
    Note(pin=board.GP12, pixel=5, num=6),
    Note(pin=board.GP11, pixel=4, num=7)
)

selects = (
    Select(pin=board.GP7, pixel=0, col=Col.RED),
    Select(pin=board.GP8, pixel=1, col=Col.GREEN),
    Select(pin=board.GP9, pixel=2, col=Col.BLUE),
    Select(pin=board.GP10, pixel=3, col=Col.YELLOW)
)
```

The code above is the Python that creates the collections of Note and Select objects that provide the user interface for the box. The note values are assigned a pin, a pixel number, and a note number

//

You can use the
cheese box with a
synthesizer program that
connects to MIDI

//

which will be used with they are played. The select values are assigned a pin, a pixel number, and the colour of the button which will be used to select the appropriate voice. This colour is then mapped to an instance of the appropriate **Voice** class. The mappings opposite were used for the first cheese box. You can change them to match the hardware that you build. Look for the statements in the program (they are near the bottom), and change the GP and pixel values to match your hardware.

BOX SOFTWARE

You can download the software for the box from the GitHub site for this project: [hsmag.cc/CheeseBox](https://github.com/hsmag/cheesebox). The software is written in CircuitPython, which can be installed on your Raspberry Pi Pico using the image from here: circuitpython.org/downloads. The software uses the Adafruit `adafruit_bus_device` and `adafruit_midi` libraries, which need to be copied into the **lib** folder on the CircuitPython drive of your Pico. You can find these here: circuitpython.org/libraries.

You can use Thonny (thonny.org) or Mu (hsmag.cc/MuEditor) on your computer to open the program and download it into your Pico. →

YOU'LL NEED

- ♦ **A Raspberry Pi Pico**
- ♦ **A 12-pixel WS2812 pixel ring** (search for 'WS2812 ring 12')
- ♦ **12 × coloured push-buttons**
Ideally eight white buttons for the notes and a red, green, blue, and yellow button for the controls. The buttons should be non-locking, push to make
- ♦ **A micro USB cable to link the Pico to the host**
- ♦ **Connecting wire**
The author used coloured wire wrap (search for '30 AWG wire wrap') which needed a wire wrap tool (search for 'wire wrap tool'). He also used a fair amount of heat-shrink tubing
- ♦ **A Raspberry Pi or machine running Windows, macOS, or Linux running Pure Data to make the sounds**
- ♦ **A box** There is a 3D-printable design, or you can just drill some holes for the lights and buttons in a cheese box and stick the pixel ring and Pico inside it
- ♦ **Screws** You'll need some screws size M2.4 mm in length to fix the pixel ring and Pico to the case (search for 'laptop screws')

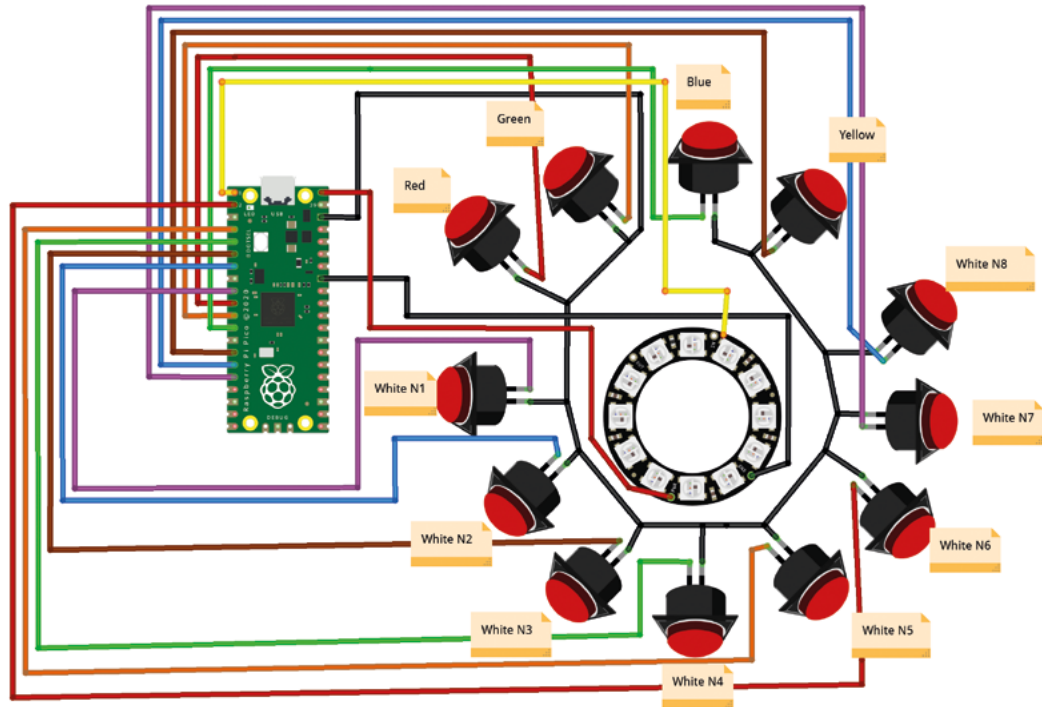


Figure 3 ♦ This diagram took almost as long to draw as it took to build the hardware

Figure 4 ♦ Wire wrap cable is single core, which makes it very easy to work with. It is also very flexible, so it is less likely to strain the connections at each end



Different elements in the application are managed by software objects called classes

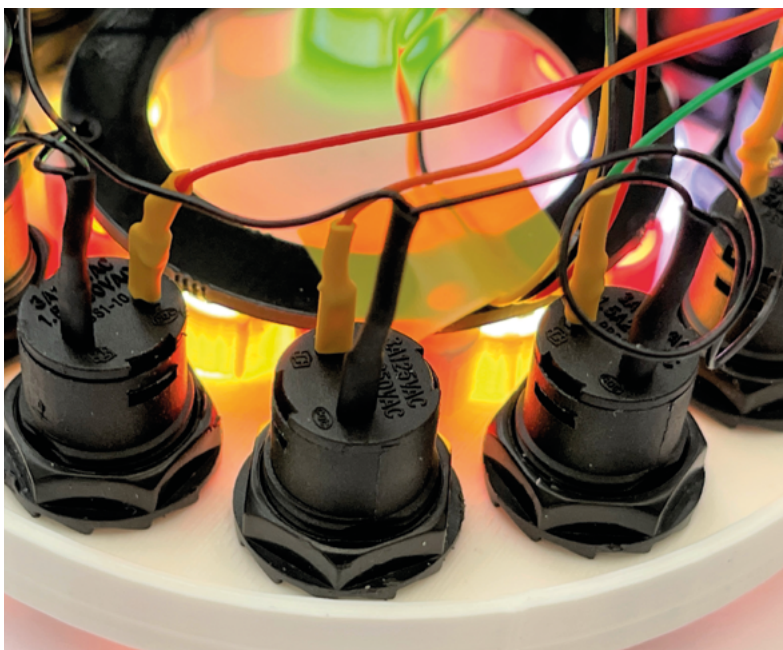


A full description of how the program works would take more space than we have for this article, but we can take a look at how it is structured.

Figure 5 shows a simplified class diagram for the software. Different elements in the application are managed by software objects called classes. The **Button** classes (Note and Select) manage the information for the buttons. The **Voice** classes (Keyboard and Rhythm) manage the information for the voice tracks that are played. The **Setting** class holds setting information for a collection of setting values. The **Step** class is based on the **Setting** class and holds information for a step in a track. The **MidiBox** class contains lists of buttons and voices, and repeatedly updates them to keep the box running and provide the illusion of several things happening at the same time.

MAKING TRACKS

The software manages four performance tracks: one keyboard track and three rhythm tracks. A track is selected by pressing one of the coloured buttons. When a track is selected, the pixel next to its selection button is lit in the colour of the button. The red keyboard track lets you play notes. When this track is selected (indicated by a red pixel next to the red button), the box starts a MIDI note playing when



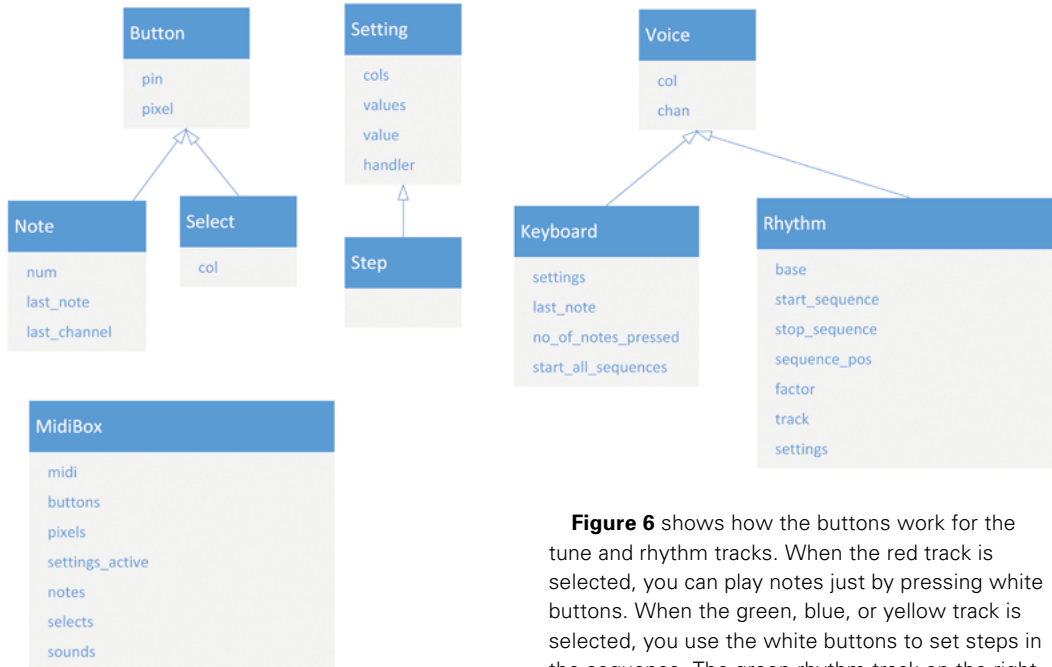


Figure 5 ♦ A child class (one with an arrow pointing to the class above it) inherits all of the properties of the parent above it. The items inside each class are the data values stored within it

Figure 6 ▣ The two diagrams show the switches around the outside and the pixels in the middle

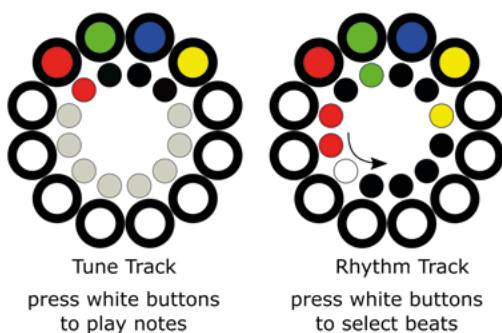
a white note button is pressed and stops the note when the button is released.

When a rhythm track is selected (by pressing a green, blue, or yellow button), the box will display the steps in the track and a moving white cursor that shows the position in the track. The user can select beats to be played at points in the track sequence by pressing the corresponding white button. All the sequencers run in parallel and work through a sequence of up to eight steps. Each step can produce a drum sound or a musical note. The speed of each sequencer can be changed, as can the length of sequence for that track. By making tracks of different lengths running at different speeds, you can create surprisingly complex rhythms as tracks go in and out of time with each other.

Figure 6 shows how the buttons work for the tune and rhythm tracks. When the red track is selected, you can play notes just by pressing white buttons. When the green, blue, or yellow track is selected, you use the white buttons to set steps in the sequence. The green rhythm track on the right of **Figure 6** starts with two beats from the kick drum (the red pixels) and ends with a ride cymbal (the yellow pixel).

The rhythm track functions in the same way for all three tracks. You can step through the options for each step by repeatedly pressing the white button to change the colour of the pixel for that step. A red step means 'kick drum'. When the sequencer reaches that step, it will send a MIDI note to play that sound.

Figure 7 shows the mapping of pixel colours to drum sounds. The number after each sound name is the MIDI note that will be sent to trigger that sound. These notes correspond to the General MIDI standard for musical instruments (hsmag.cc/MIDI) so the cheese box will work with most synthesizers. →



A NOTE ABOUT SCALES

The white buttons on the box don't play successive MIDI note values. Instead, they play the notes from a particular musical scale. This is so that the box sounds musical when you play it. Different scales use different arrangements of gaps between the successive notes. The most common scale is the 'major' scale, but there are others. Some have fantastic names like 'Phrygian' and 'Mixolydian', as you can see in the Scale setting in **Figure 8**. You can experiment with different scales to find the one that sounds best for a particular tune.



HackSpace

This tutorial is from HackSpace magazine. Each issue includes a huge variety of maker projects inside and outside of the sphere of Raspberry Pi, and also has amazing tutorials. Find out more at hsmag.cc.

Drums

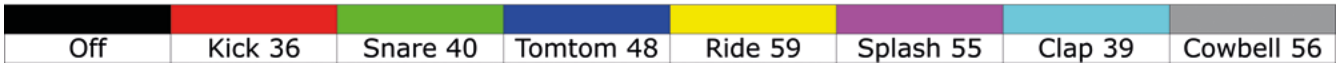


Figure 7 ♦ Each time you press a step button, the selected beat for that step moves to the next colour. If it reaches the end, it wraps around to off (black)

Figure 8 ♦ The pixel colours in the centre of the settings show the default setting options

SETTING UP

You adjust the settings for the box by holding down the button for the selected track, and then setting the colour of pixels that are mapped onto setting values. If you press the red button when the red track is selected, the tune settings are displayed.

Figure 8 shows the tune settings. There are seven different setting items. Each of them is assigned a pixel and the value of the setting selected by stepping through the colours for that setting. There are also seven settings for the rhythm track...

Figure 9 shows the rhythm settings for the green track. You can set the speed of the track in beats per minute. You can also set the pattern length and the 'speed factor'. The speed factor is used to slow down a rhythm track. A factor of 1 means normal speed. A factor of 3 a third speed, and

so on. You can also change the mode of a rhythm track to play notes rather than drums. This lets the track play short musical sequences.

MAKING SOUNDS WITH PURE DATA

You can use the cheese box with any synthesizer, but you can also use it with a specially written Pure Data patch. Pure Data is a programming language for creating music applications. You can find out more about it in the article in issue 47 of HackSpace magazine. Pure Data programs are expressed as a network of connected objects. The connections between the objects can be either individual values or audio streams.

Figure 10 shows the Pure Data patch that generates sound output for the cheese box. It works as a 'switching station' which acts on incoming MIDI messages. The 'notein' object at the top left of the patch accepts incoming MIDI messages and sends out the note number, volume, and channel (these are the three outputs along the bottom of the notein object). The patch uses spigot objects to control the flow of the data around the patch. Values will pass through the spigot if the right-hand input to the spigot is true.

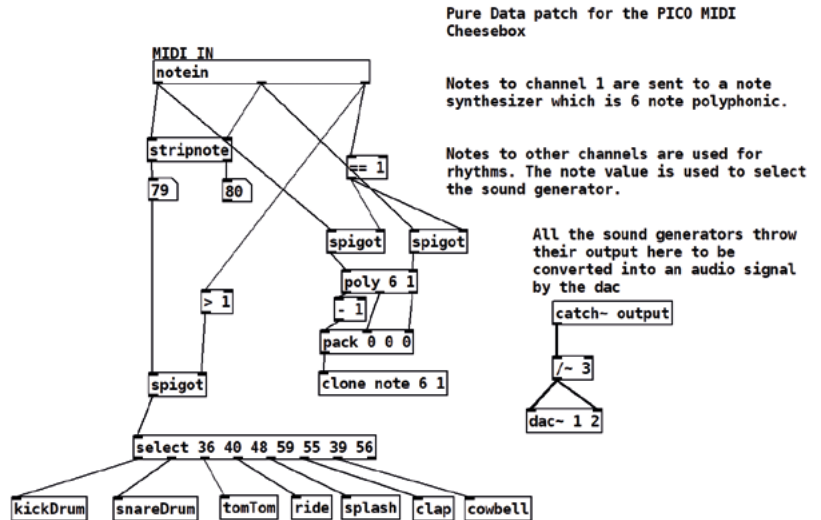
Any messages sent to MIDI channel number 1 are passed into the polyphonic object (poly) and

// You can use the cheese box with any synthesizer, but you can also use it with a specially written Pure Data patch //

Tune Settings



then on to a set of six note synthesizers. Messages sent to any other MIDI channel are stripped of their note off elements (these have no meaning for drums) using the stripnote object and then the note number is used to select the sound generator to be used. You will notice that the note numbers in the select object are the same as those in **Figure 7**. You can find this patch, along with installation instructions, at the GitHub site for this project: hsmag.cc/CheeseBox. At the moment, the note generators only output simple sine waves, but in the next issue of HackSpace magazine, we will be discovering how to use Pure Data to produce more interesting sounds and create a crackers controller to go with the cheese.



EUCLIDEAN FUN

By changing the speeds of the tracks and playing notes and drums at different points, you can generate interesting sounds that change over time. These are called Euclidean rhythms. You can find out more here: hsmag.cc/Euclidean. Don't be afraid to experiment with the sounds. If you change the rhythm mode of a track to 'Euclidian', the cheese box calculates a note frequency by adding together note values from other Euclidian tracks. By careful choice of track speeds, you can create extended sequences of changing notes. Once you have something interesting, try changing the scale and listen to what happens. You can pick different octaves for the note playback so you can generate bass sounds too. Remember that there is no such thing as bad music, just stuff that only you like. ▣

Rhythm Settings

Figure 10 ♦ Pure Data programs can run on a huge range of computers, from Raspberry Pi Zero upwards

Figure 9 ♦ By default, each rhythm track sends on a different MIDI channel so that the rhythms can be used to trigger different voices. However, you can select the MIDI channel for each track so that you can make the rhythm tracks play on the same channel as the keyboard



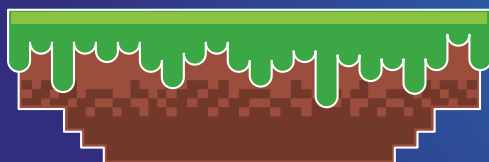
RASPBERRY PI

GAMING

No-emulation gaming direct on Raspberry Pi. By **Rob Zwetsloot**

We love RetroPie. It's a fantastic OS for Raspberry Pi that makes it really, painfully easy to get retro gaming working on a Raspberry Pi. However, did you know that some games run on Raspberry Pi OS without any form of emulation? And we don't just mean Doom.

Raspberry Pi is a computer after all, which means you can easily make games for it – we've done it several times ourselves with Python and Scratch. There are many games that have been released on or ported to Raspberry Pi, and we think it's about time we show you some.



Steam Link

Stream games without the hassle

The quickest way to play modern games on Raspberry Pi is to stream them from another PC using Steam Link – here's how to set it up.

Install on Raspberry Pi OS

Raspberry Pi OS has Steam Link in the repository, so you can easily install it! You can find it in the Software Installer, or simply install it from the Terminal using:

```
sudo apt update
sudo apt install steamlink
```

You'll find it in the main menu under Games – just make sure Steam is running on your main PC, plug in a controller, and you're good to go.

📌 Raspberry Pi OS has Steam Link in the repository, so you can easily install it! **📌**

Install on RetroPie

If you have a dedicated retro gaming Raspberry Pi setup, you can still make use of Steam Link. Go to RetroPie Setup, scroll down to Manage Packages, and then go to 'exp' for the Experimental packages. There you'll find a steamlink package. Install it, and then find it in the Ports section of the main interface.



Xbox Cloud Gaming

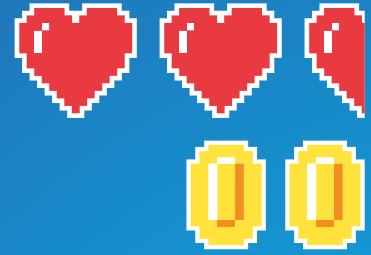
Want to try out the xcloud on Raspberry Pi? Our friends at Pi My Life Up have a guide: magpi.cc/xcloud

Steam Link tips!

For the best experience, make sure at least one of the systems in this setup is connected via an Ethernet cable. Preferably your PC running Steam and your Raspberry Pi will both be wired.



Originals on Raspberry Pi



Linux games running natively on a Raspberry Pi

Micropolis

> magpi.cc/micropolis

Created by the SimHacker team, this free and open-source game is based on the original SimCity. For those unaware of it, it's a simulation game which allows you to plan out and build a city while also facing any consequences your actions might result in. And, sometimes, giant monsters.

How to install:

```
sudo apt install micropolis
```



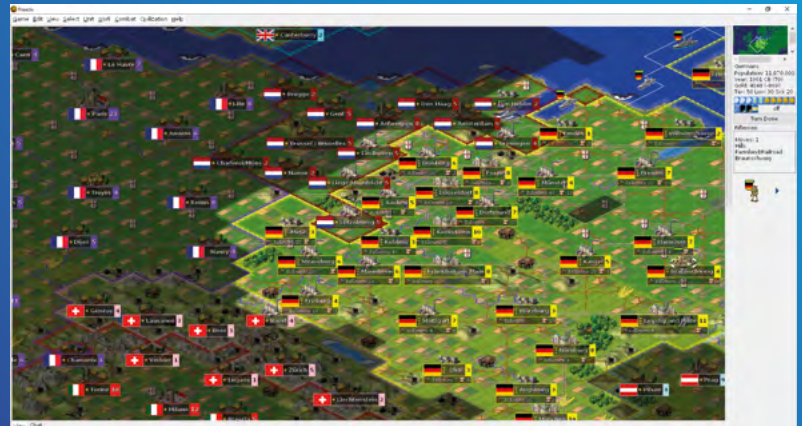
Freeciv

> freeciv.org

As you may have gathered from the name, Freeciv is a clone of the Civilization games, albeit the earlier ones. This version supports up to 126 people in multiplayer, and has a load of mods and scenarios for you to play around with.

How to install:

```
sudo apt install freeciv-client-sdl
```



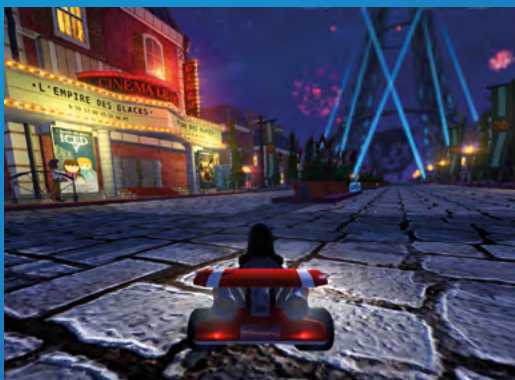
SuperTuxKart

> supertuxkart.net

This mascot kart racer includes mascots from all your favourite libre operating systems and software, including Tux, the mascot of Linux himself. This is a fun racer full of power-ups, much like your Mario Karts, Sonic Karts, or Diddy Karts, and it even has a story!

How to install:

Download from the main website or follow the instructions here: magpi.cc/tuxkartinstall



Hurrican

> magpi.cc/hurrican

This game describes itself as 'THE ultimate Turrican Mega-Mix', taking ideas from several instalments of the original franchise while at the same time expanding on its universe. It's a side-scrolling run-and-gun game with a killer soundtrack.

How to install:

Follow the instructions here: magpi.cc/hurricaninstall



Pingus

> magpi.cc/pingus

A Lemmings-style puzzle game with little penguins instead. Basically, you have to guide the penguins through a level using a variety of commands you can give them, like dig, redirect, build a bridge, etc. Unfortunately, the penguins do not have any regard for their own well-being.

How to install:

```
sudo apt install pingus
```





Games on Itch.io

Indie games galore for Raspberry Pi

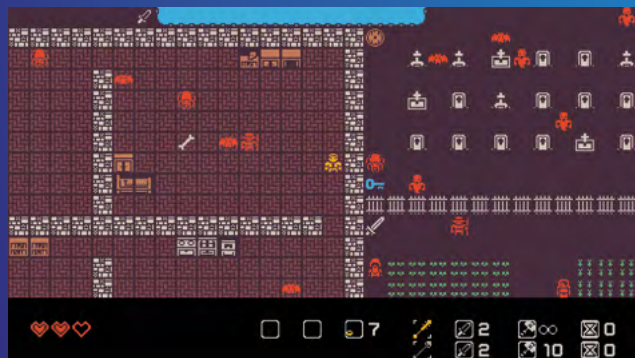
Ben Was Assimilated

> magpi.cc/benassimilated

With a very cool, low-colour aesthetic, Ben Was Assimilated is a hack 'n' slash game where things only move as you do, resulting in careful deliberation and interesting tactics. It also has Roguelike elements, including a permadeath option for those wanting a real challenge.

How to install:

Get it from the link.



Art Treachery

> magpi.cc/arttreachery

This game was made for the British Library Labs Crowdsourcing Game Jam. You play an art thief who has to steal specific objects from a museum. It's simple and short, but a pretty fun concept.

How to install:

Get it from the link.



You play an art thief who has to steal specific objects from a museum



Zombusters: Raspberry Pi Edition

> magpi.cc/zombusters

A isometric, twin-stick shooter with Roguelike elements where you have to bust zombies with guns. Lots of guns. You can also play with up to four players on one Raspberry Pi for some excellent combo zombie take-downs.

How to install:

Get it from the link.



SPACETRADER

SpaceTrader

> magpi.cc/spacetrader

This game runs in the Terminal for a different kind of playing experience. You're a trader in Earth's space port, and need to make your way around a changing economy and pirate bands to make as much money as you can.

How to install:

Get it from the link.

gray water

> magpi.cc/graywater

Another Terminal game, this game takes a darker turn into horror with this simple description: 'You enter an apartment building. You do not know your neighbours.' It's not very long, but it is very effective, bringing new ideas to the classic command line-based game.

How to install:

Get it from the link.

gray water

Classic games

These games from yesteryear have been made available to everyone, in most cases by releasing the source code. You can run them straight off your Raspberry Pi thanks to the community making them ARM-friendly.

As the space marine Doom Guy, you have to fight off the forces of hell

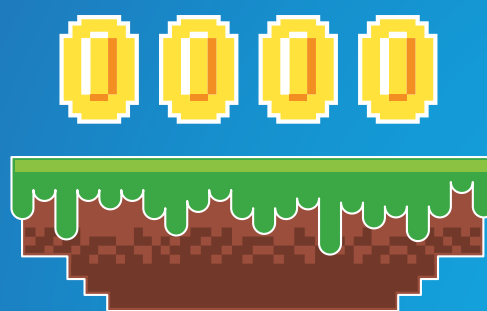
Doom (1993)

magpi.cc/doom

A game so good, it defined a genre for years. Seriously – FPS games used to be called Doom-clones. As the space marine Doom Guy, you have to fight off the forces of hell with a large variety of guns and a rocking soundtrack. The source code was released by id Software in 1997, and became open-source in 1999, which is why there’s been a long-running joke about trying to get every piece of tech to run it.

How to install:

```
sudo apt install chocolate-doom
```



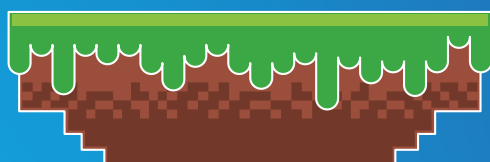
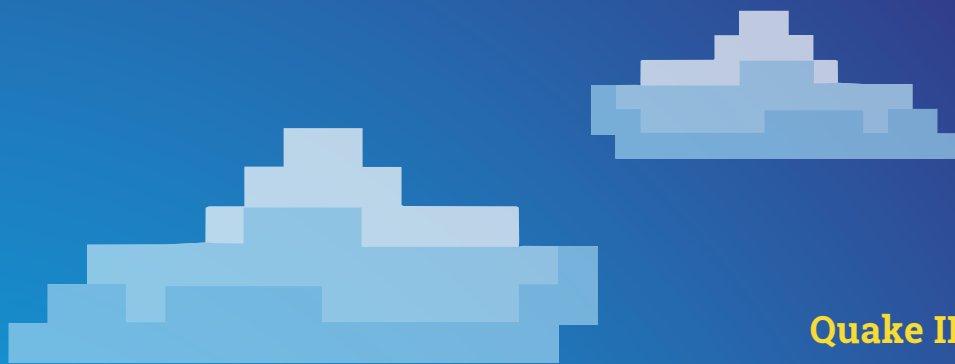
Beneath a Steel Sky (1994)

magpi.cc/steelsky

A classic point-and-click adventure game, Beneath a Steel Sky has been available as freeware since 2003. It’s a cyberpunk story set in a dystopian Australian future and... well it’s a story game, so we won’t spoil too much. A sequel for it came out just last year, so it’s a great time to familiarise yourself with it.

How to install:

```
sudo apt-get install beneath-a-steel-sky
```

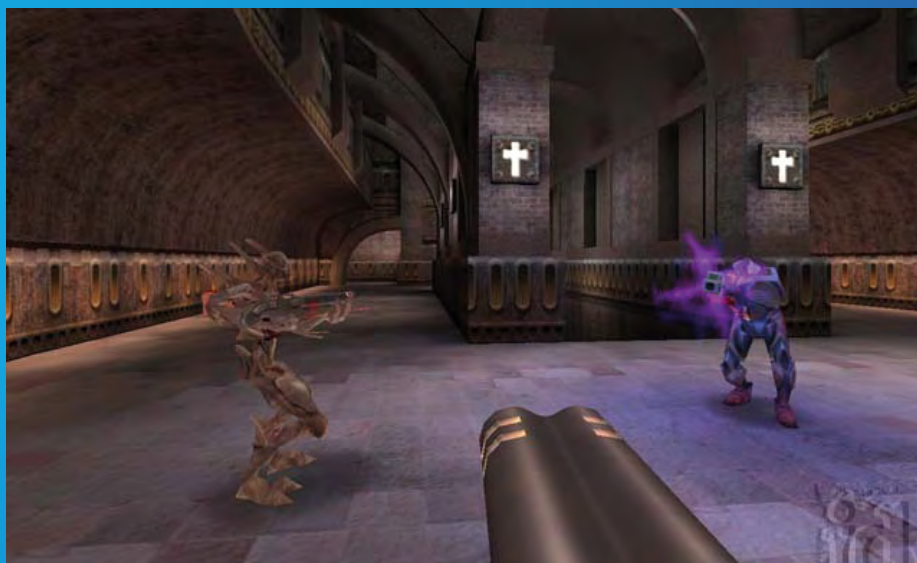
Quake III Arena (1999)

> magpi.cc/quake3arena

Another id Software classic, and the granddaddy of multiplayer twitch shooters, Q3A was the bane of school IT administrators in the early 2000s because of how easily available it was. Only the demo data is free, but if you have a copy of the game, you can add the files yourself to get the full game.

How to install:

```
sudo apt install quake3
game-data-packager quake3 -i --gain-root-
command sudo
```



Minecraft (2011)

> magpi.cc/mcpi

Minecraft Raspberry Pi Edition is a free game on Raspberry Pi. However, it has not been officially updated in many years. You can still hack it with Python to do fun things, and we've covered that in the mag before. Minecraft Raspberry Pi Edition Reborn is an effort to mod it to be much closer to the Minecraft you can play elsewhere – it's still in development, though.

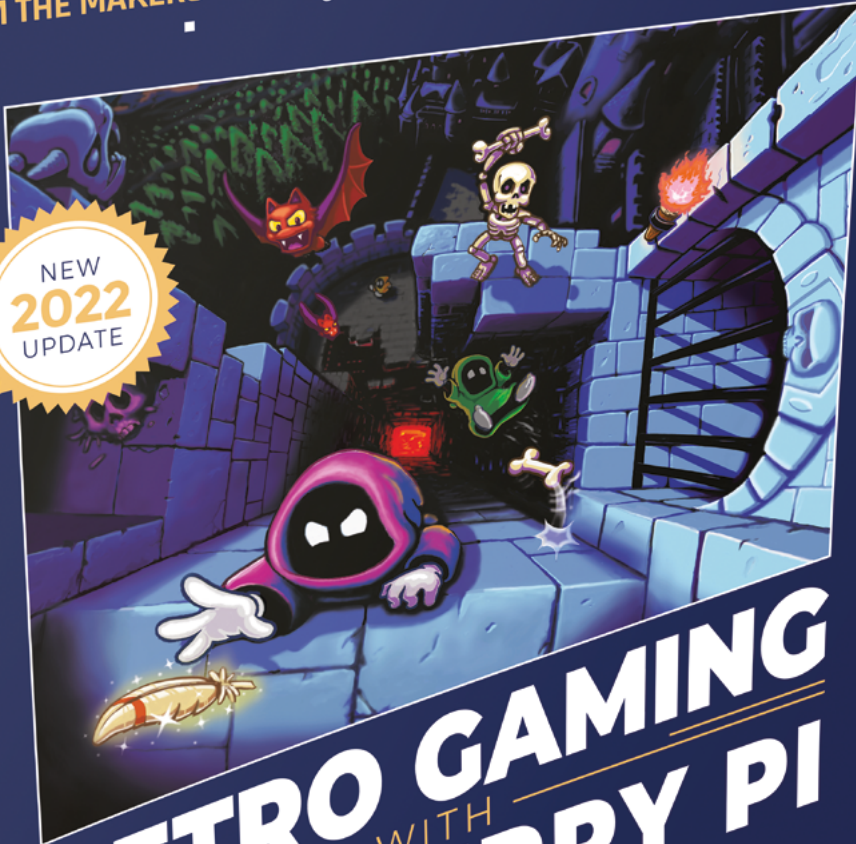
How to install:

Follow instructions on the website linked above. 



FROM THE MAKERS OF *The MagPi* THE OFFICIAL RASPBERRY PI MAGAZINE

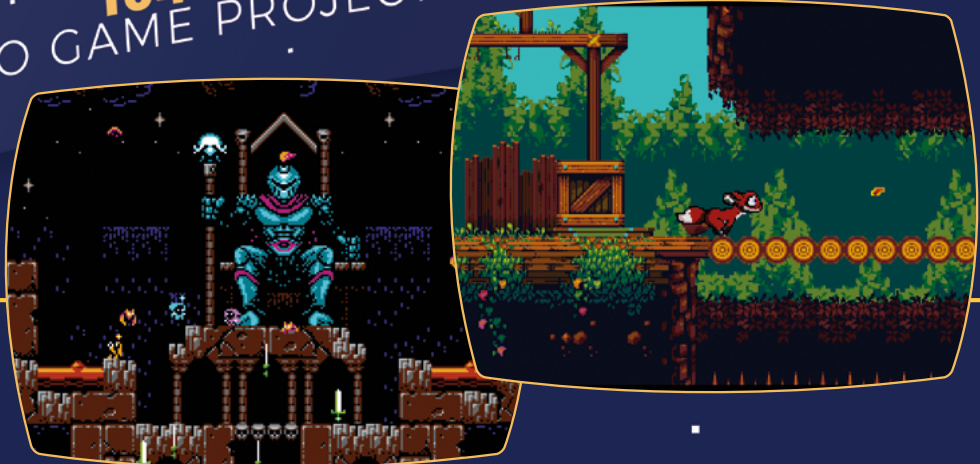
NEW
2022
UPDATE



**PLAY
& CODE
GAMES!**

RETRO GAMING WITH RASPBERRY PI 2ND EDITION

164 PAGES OF
VIDEO GAME PROJECTS



RETRO GAMING

WITH

RASPBERRY PI

2ND EDITION

Retro Gaming with Raspberry Pi shows you how to set up a Raspberry Pi to play classic games. Build your own games console or full-size arcade cabinet, install emulation software, and download classic arcade games with our step-by-step guides. Want to make games? Learn how to code your own with Python and Pygame Zero.

- ***Set up Raspberry Pi for retro gaming***
- *Emulate classic computers and consoles*
- ***Learn to code your own retro-style games***
- *Build a console, handheld, and full-size arcade machine*



BUY ONLINE: magpi.cc/retrogaming

Argon IR Remote

▶ Argon40 ▶ magpi.cc/argonir ▶ £9 / \$10

Looking for a sleek minimalist remote for your Raspberry Pi media centre?
PJ Evans sits back on his sofa and gets clicking

SPECS

DIMENSIONS:

150 mm × 35 mm
× 10 mm

POWER:

2 × AAA batteries

COMMUNICATION:

Infrared

HARDWARE

COMPATIBILITY:

Argon ONE &
EON cases

SOFTWARE

COMPATIBILITY:

LibreELEC or
Raspberry Pi OS

Using a Raspberry Pi 4 as a media centre makes for a brilliant way to access your favourite sounds and movies. Add Argon40's sleek ONE case and it looks as good, if not better, as any other set-top box. There's even mature software in the form of the LibreELEC operating system and Kodi media centre app. All that's missing is an easy way to control the action. The eagle-eyed may have noticed the ONE case's circuitry includes an IR sensor, and now Argon40 has made the companion Argon IR infrared remote control available.

Thankfully, Argon40 has been watching other tech design teams rather than those of most modern TVs. What you get for your very

reasonable £9 is a minimalist, slim remote with just enough to make controlling Kodi a breeze. Basic directional control, selection, volume, and standard navigation work alongside a power button that works out-of-the box with any Raspberry Pi 4, regardless of operating system. It's light too, at just 65g with two AAA batteries.

More than media

If media centres are not your thing, or you have other epic plans for a remote control, Argon40 provides software that allows you to assign any key press to a button using the standard LIRC library with ease, making the addition of IR remote control to your project easier than ever before. In our tests, this worked flawlessly. However, we did have some issues configuring Kodi due to some missing steps in the supplied instructions. Once resolved, the remote performed admirably from across the room. **M**

“ A minimalist, slim remote with just enough to make controlling Kodi a breeze ”



▲ The lack of extraneous buttons and a small form factor make the Argon IR comfortable to hold and use

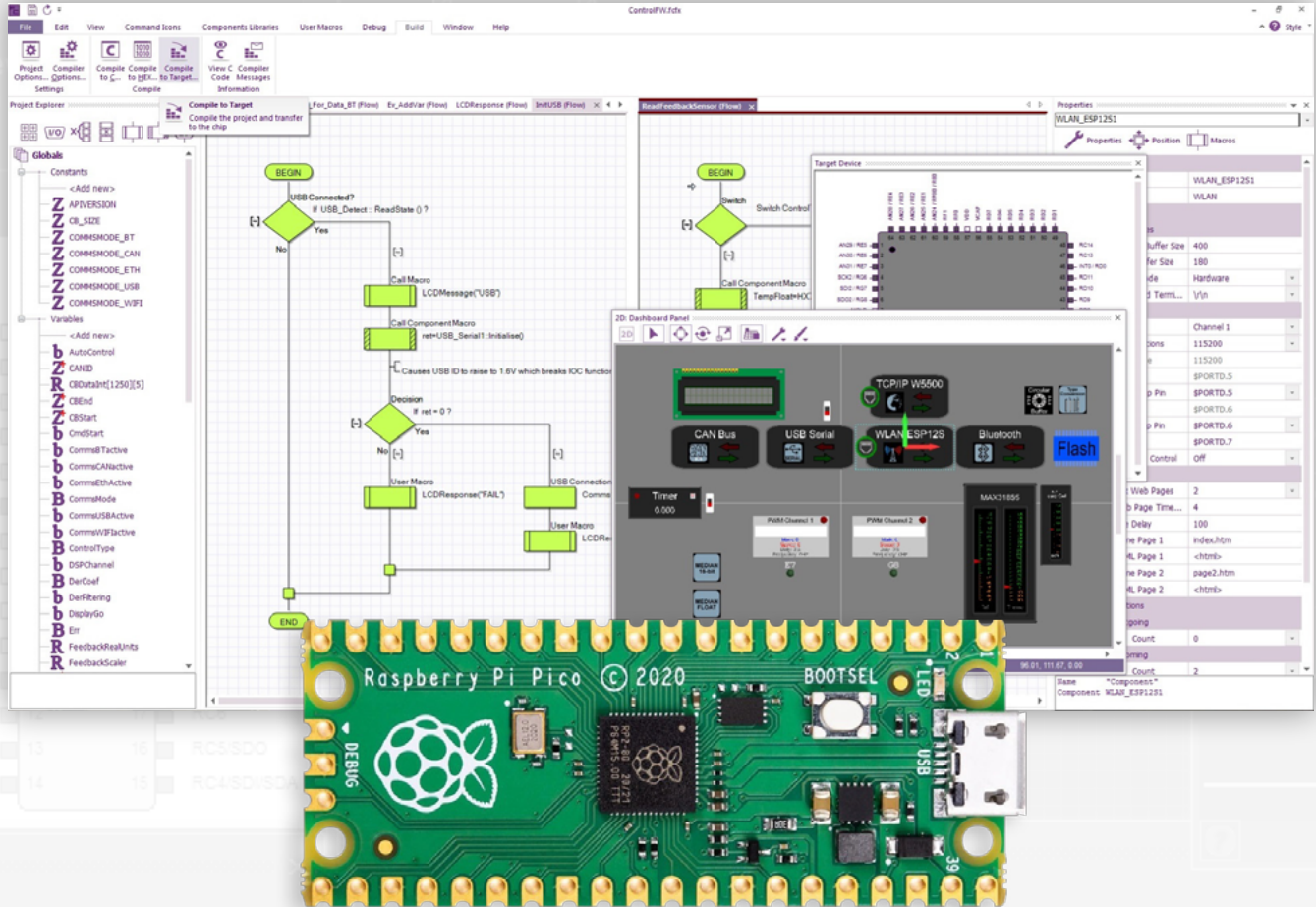


▲ You'll need an Argon ONE case (or the forthcoming EON) to use the Argon IR

Verdict

Although the setup instructions could be improved, Argon40 has delivered a sleek, simple, and reasonably priced IR remote, both for Raspberry Pi media centres and homebrew projects.

9/10



PROGRAM RP2040 WITH EASE USING

FLOWCODE

20% off your Flowcode purchase using code: **MAGPI20**

Use code at checkout: flowcode.co.uk/buy

DACBerry 400 S

► The Pi Hut ► magpi.cc/dacberry400 ► £17 / \$19

Add an audio output, and mic input, to your Raspberry Pi 400. By **Phil King**

SPECS

AUDIO FEATURES:

Up to 96kHz/32-bit, 102 dB SNR DAC, 92 dB SNR ADC, integrated DSP, 3D effects and de-emphasis

INPUT/ OUTPUT:

Gold-plated 3.5 mm jack for mic or stereo out

GPIO:

Pass-through; GPIO pins used: 2, 3, 18–21, 26

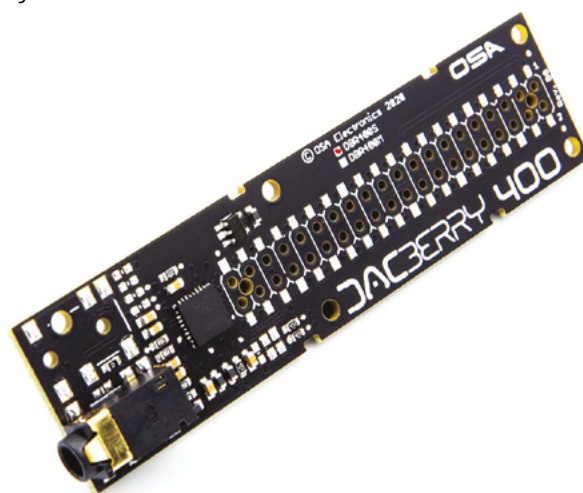
While Raspberry Pi 400 has an array of ports at the rear, one notable omission is a dedicated socket for audio output. The DACBerry 400 S is designed to address this while also offering a mic input via the same gold-plated 3.5 mm socket.

The DACBerry 400 S fits onto the GPIO header at the rear of Raspberry Pi 400. While it's fairly obvious which way up the board should go, care needs to be taken to line up its header with the GPIO pins correctly – with a small gap showing at the Pin 1 end.

Get the alignment wrong and not only will you get no sound, but you may damage the board. This is something manufacturer OSA Electronics will be addressing in the second batch of boards, by introducing a small bump in the middle, to line up with a notch in the GPIO header surround.

Let there be sound

A one-line Terminal command installs the software needed to drive the DACBerry 400 S. At the time of writing, this required the earlier Buster version of Raspberry Pi OS (magpi.cc/buster), but it should



▲ The DACBerry 400 S has a gold-plated audio connector and pass-through for the GPIO pins

“ Care needs to be taken to line up its header with the GPIO pins correctly ”

be available in Bullseye soon. After rebooting, the audio will be routed through the DACBerry – with Raspberry Pi 400's audio chip disabled, so it's not easy to switch back if needed. It also works with OSMC and LibreELEC, among other OSes.

The line-out audio signal is strong enough for decent volume on connected headphones, or you can use powered speakers or an amp. The sound quality is very good with a nice warm tone and strong bass, possibly due to DSP's integrated low-pass filter. The audio can be adjusted using the numerous settings in alsamixer, but OSA plans to introduce a GUI-based tool for this.

A bonus is the ability to connect a microphone for recording, using the same 3.5 mm socket – OSA is also planning another DACBerry 400 model with a separate mic socket. [M](#)

Verdict

If you need a separate audio output (and/or mic input) for your Raspberry Pi 400, this provides one at little cost.

8/10



▲ Make sure to connect the board to the GPIO header the right way around and align the pins correctly

HackSpace

TECHNOLOGY IN YOUR HANDS

THE **MAGAZINE**
FOR THE **MODERN MAKER**



SUBSCRIBE AND
SAVE UP TO
35%
on the cover price



ISSUE #50

OUT NOW

hsmag.cc



10 Amazing: Starter kits

These Raspberry Pi kits will help you take the first steps on your making journey

Did you get a Raspberry Pi for Christmas? Maybe thinking of getting one with any holiday money you received? Here are some great kits to help you get properly started with a Raspberry Pi or Raspberry Pi Pico. [M](#)



▲ Picade

Arcade kit

This mini arcade cabinet is a wonderful kit that allows you to create a high-quality retro games console, with a nice stick and buttons which can be upgraded in the future if you wish.

magpi.cc/picade | £225 / \$249



▲ Raspberry Pi 400 Personal Computer Kit

All-in-one Raspberry Pi

A box full of just about everything you need to create a PC – a Raspberry Pi 400, mouse, power supply, microSD card, and a nice *Beginners Guide* book. Just add a monitor or TV.

magpi.cc/pi400info | £94 / \$125



▼ CamJam EduKit 3 – Robotics

Beginner's robot

A classic kit for beginners that lets you easily build and program a robot of your very own, and quite cheaply as well! You can even use the box as the chassis.

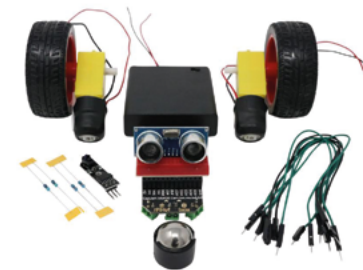
magpi.cc/edukit3 | £18 / \$24

▲ Grow Kit

Automated herbs

Growing your own herbs is a great way to not have to constantly buy fresh herbs for cooking, and the Grow Kit makes it very easy to keep them watered so they'll grow properly.

magpi.cc/growkit | £30 / \$40

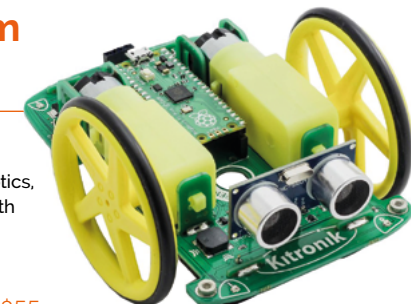


▼ Autonomous Robotics Platform for Pico

Pico robot

We're entering a new world of Pico robotics, and this Kitronik kit will get you going with it. Pico offers different advantages over Raspberry Pi for robots, so give it a look.

magpi.cc/robotplatform | £41 / \$55



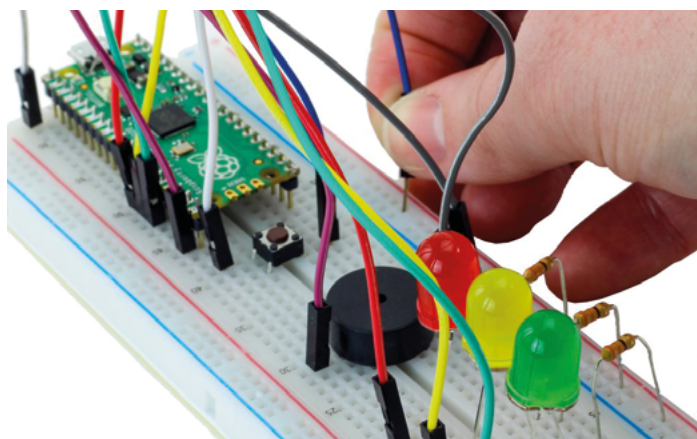


▲ Pico Explorer Base

Pico playground

This board allows you to really explore what can be done with a Pico, with plenty of stuff for beginners and loads of things you can then build up from, thanks to its breakouts and screen.

magpi.cc/picoexplorer | £22 / \$29



▲ Discovery Kit for Raspberry Pi Pico

Electronics and microcontrollers

Loads of little electronics bits that you can use with a Raspberry Pi as well as Pico. It comes with a little guide with several projects too.

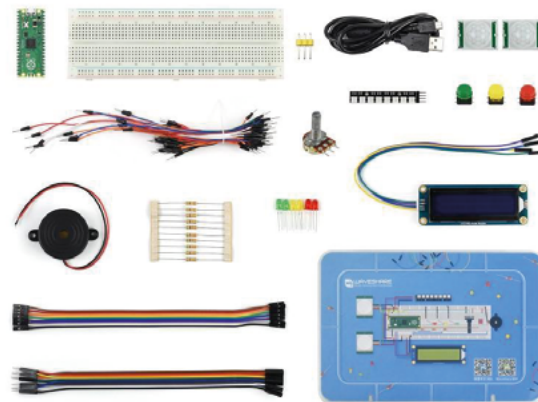
magpi.cc/discoverykit | £12 / \$16

▶ Monk Makes Project Box 1

Intro to electronics

With enough components to create ten projects, these Monk Makes boxes are a great way to get stuck in with a new Raspberry Pi.

magpi.cc/projectbox1 | £12 / \$15

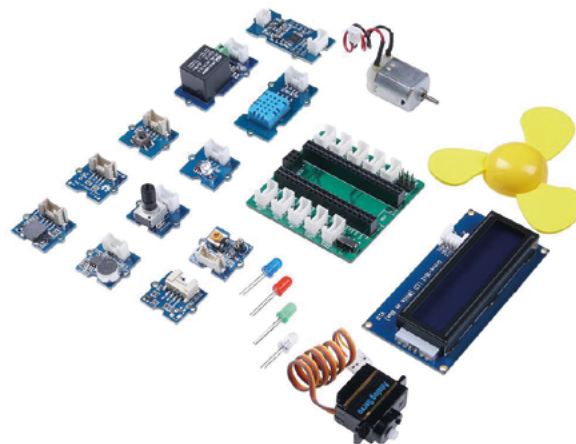


▲ Raspberry Pi Pico MicroPython Learning Kit

More electronics

This is basically a bigger version of the Discovery kit with many more components to play around with. It even comes with a Pico.

magpi.cc/learningkit | £32 / \$43



▲ Grove Starter Kit for Raspberry Pi Pico

Sensors and more

This kit is a great way to play around with sensors and motors and more without needing to worry too much about wires – just the programming.

magpi.cc/groestarterkit | £44 / \$59

Learn to code in Scratch

Bright colours and an array of projects to get stuck into make Scratch the ideal first step into coding. By **Rosie Hattersley**

Raspberry Pi Projects

CREATOR

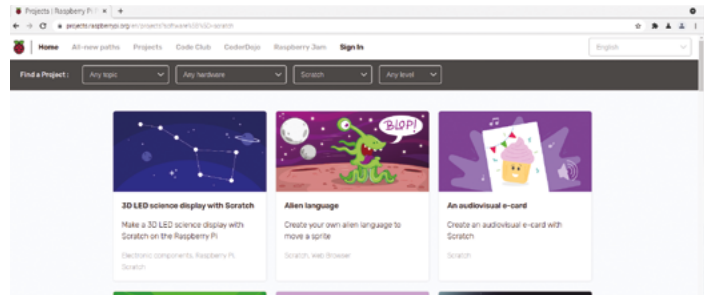
Raspberry Pi Foundation

Price:
Free


[magpi.cc/
scratchprojects](http://magpi.cc/scratchprojects)

Raspberry Pi provides a heap of projects and games to help you get to know Scratch, complementing those offered by MIT. These projects challenge you to beat the buzzer, press particular keys corresponding to colours in order to create sounds, or to log how many birds of a particular hue appear in your virtual wildlife camera.

The skills you learn can then be used for your own Raspberry Pi projects. Videos at the start of each project show you what

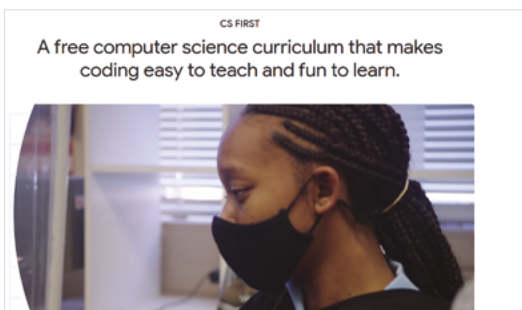


you'll create, with lists of materials needed and details of how it all fits into educational learning goals. Of course, you can just enjoy making things

with code! Web-based Trinkets enable you to practise and check your code works. Step-by-step instructions and diagrams are included throughout. 

Scratch courses

Teach yourself or others with a bit of online assistance



▲ CS First is a great place to learn Scratch skills

COURSERA

Coursera has a range of Scratch courses to help you learn the visual programming language. The "Introduction to basic game development course" is a great place to start, before moving on to a range of more advanced courses. [▶ magpi.cc/courserascratch](http://magpi.cc/courserascratch)

CS FIRST

Learn Scratch skills yourself, as well as getting ideas and project plans for use in the classroom or at a coding club you might be involved

with. Separate logins for individual students and educators mean that everyone can learn at their own pace, and backup resources are always on hand.

[▶ magpi.cc/csfirst](http://magpi.cc/csfirst)

UDEMY

Ideal for teachers or individuals, classes are aimed at adults keen to learn coding in order to teach it to others. The course focuses on problem-solving, computational thinking, and creativity.

[▶ magpi.cc/udemyscratch](http://magpi.cc/udemyscratch)

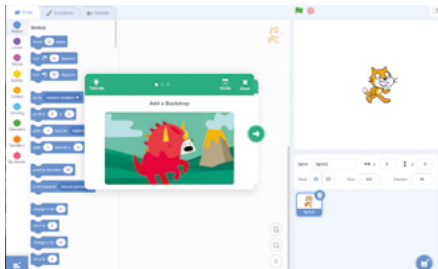
Scratch ideas

CREATOR

MIT


Price:
freescratch.mit.edu/ideas

An ideal starting point for newcomers to coding, MIT (Massachusetts Institute of Technology) is the home of Scratch. Scratch is a visual means of learning to code and provides an accessible route to learning using images and animations. Importantly, the visual approach mimics the types of instruction you will encounter when using text-based coding languages such as Python to program Raspberry Pi, Arduino, and other computers.



Start by watching the Getting Started video, which shows how instructional blocks are used to build up actions for each sprite or character, after which you can jump straight into using Scratch yourself.

The basic premise of Scratch is that coding can be done by ordering physical actions. This is as simple as dragging the blocks you want to use over to the workspace, and then clicking the green flag to play each action. Brief video overviews introduce topics such as Making Music, Imagine A World, and Video Sensing, providing a good steer before you give each one a try. Learner and educator info cards keep you on track.

Open Scratch 3 (under Programming in the accessories menu) and click on Tutorials, or visit Scratch online. 

CS50 Scratch


CREATOR

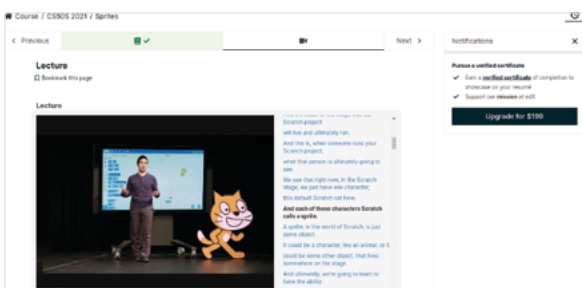
Harvard

Price:
freemagpi.cc/cs50scratch

CS50 is Harvard University's highly acclaimed introduction to computer science and programming. Most of the course materials cover text-

based languages, such as C, but this "Introduction to programming with Scratch" introduces computer concepts using visual programming.

It's a great mix, combining Harvard's know-how, excellent video seminars, and deep explanations with Scratch's easy-to-understand interface. The course covers key programming concepts such as functions, conditions, loops, variables, and abstraction. When you've finished you can move on to further CS50 courses. 



Learn along with others

Learn coding independently or as part of a peer group



SCRATCHED

Ideas for how to use Scratch across the curriculum, from music and maths to literacy and art. This site contains lots of ready-made quizzes and challenges, plus Scratch-based animations that précis important topics you want students to remember.

► magpi.cc/scratched

SCRATCH.IE

A useful set of Scratch-based activities separated into primary school-age and post-primary (age 12 upwards, reflecting the Irish education system). As with other sites, teacher and students have their own login and learners can progress independently, choosing which projects to complete when. We'll be heading straight to the retro gaming option!

► magpi.cc/scratchie

CODERDOJO

Part of the Raspberry Pi community, CoderDojos provide both online projects you can complete, and in-person groups where you can join other young coders working on Scratch, Python, and HTML challenges. Click the Resources link for ready-made projects to try.

► magpi.cc/coderdojo



Stewart Watkiss

The Penguin Tutor

- Name **Stewart Watkiss** | ➤ Occupation **Data Centre Manager**
- Community role **STEM Ambassador** | ➤ Website **penguintutor.com**

Stewart may be familiar to long-time readers of *The MagPi* – he’s written articles for it and had his projects featured in these very pages many a time. He’s also a well-known face in the community, having been around since the beginning.

“I became a member of the Raspberry Pi community from the launch of the first Raspberry Pi,” Stewart tells us. “I couldn’t wait to get my hands on one. Being a

member of the community has opened new doors for me; it’s given me an opportunity to make projects which were previously out of reach, to teach skills to others, and fulfil my ambition of writing a book.”

What is your history with Raspberry Pi?

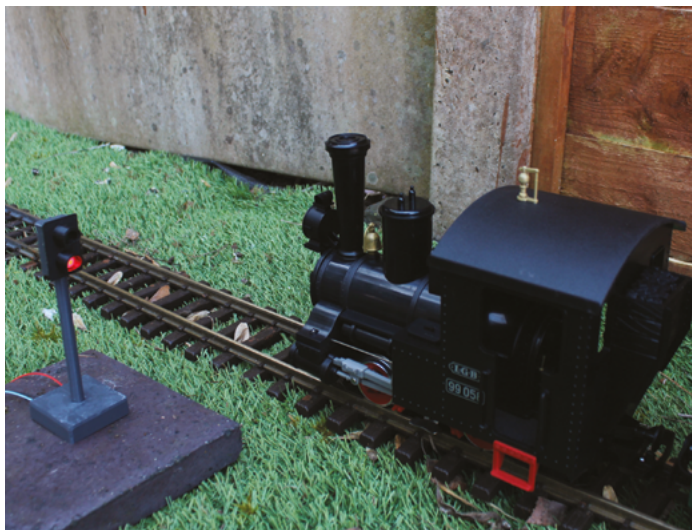
As a supporter of Linux, I first heard about Raspberry Pi a few days before the launch. I heard it was a new low-cost Linux

computer, and wanted to get hold of one. I was up early on the day of the launch, but didn’t manage to place an order until later that day when the retailers’ websites had recovered. It was then a few months’ wait before it was delivered. Once I had one, it became clear that it was much more than just a Linux computer, and the GPIO connectors meant that I could use it for so much more.

I created some initial projects and then got involved with more community events, giving talks and workshops at Raspberry Pi events, including some in Cambridge and those nearer home at Birmingham. I also signed up as a STEM ambassador, and I’ve run some Code Clubs in schools.

How long have you been making for?

I started getting interested in making during the early 1990s whilst studying electronics. After finishing university, I no longer had access to the equipment needed to go beyond basic projects, so I dropped the



▶ This model railway’s train signal is automated via Raspberry Pi



▲ The Bee Box was one of the first projects Stewart made with Raspberry Pi, working with his kids on it



▲ This 3D-printed house had lights controlled via a Raspberry Pi and touchscreen

hardware side and concentrated more on software projects.

I bought my first Arduino in 2008, which made electronics much easier and expanded the scope of what the home electronics enthusiast could achieve. I started to get back into electronics, working on some simple projects. It was really in 2012 when I first got my first Raspberry Pi when my making took off. I started with some simple projects and then it grew from there, learning new skills and progressing to more adventurous projects. Since then, I've now got a 3D printer as well, which has expanded the scope of what I'm able to do.

Could you tell us about writing for *The MagPi* in the early days?

I wrote a few articles for *The MagPi* when it was a purely volunteer magazine. I took some projects that I'd created already and created a write-up for the magazine. At the time, I was wanting to write a book and writing for *The MagPi* was a great way to get more experience. Despite being written by volunteers, it followed all the same process as a professional publication, including a technical review and proofreading. I was impressed by how professional

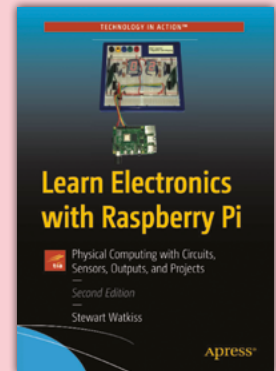
it was, and it was valuable experience for when I got my first book deal.

What is your favourite thing that you've made with a Raspberry Pi?

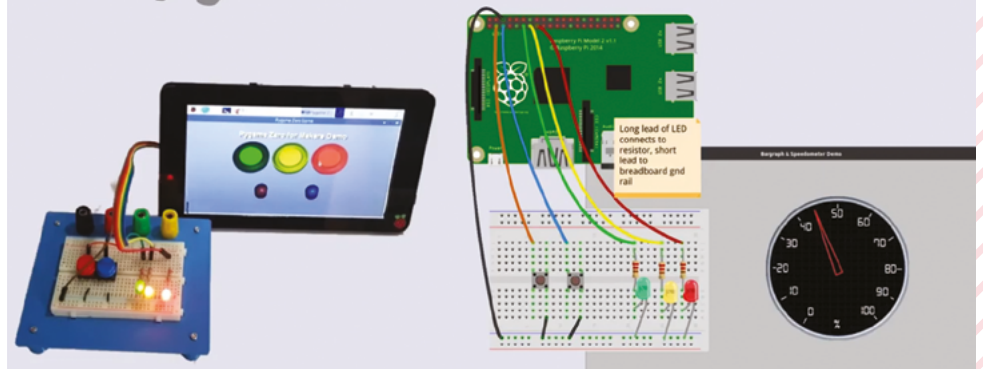
My favourite project is one of my first, known as the Bee Box. This was the first project that I did with my children, and I really enjoyed that we were able to work together on it. It was a minibeast project for school using some reed switches to determine the position of a model bee. It then showed a picture on the screen and included sound effects. 

Stewart's stuff

"I've written two books (or three if you count the second edition). The first is *Learn Electronics with Raspberry Pi*, which has even been translated to Chinese, and *Beginning Game Programming with Pygame Zero* which was published in 2020. More recently, I've been focused on creating content for my YouTube channel, penguintutor, and website."



Pygame Zero for Makers



MagPi Monday

Amazing projects direct from our Twitter!

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.

Here's a selection of some of the awesome things we got sent this month, and we got quite a lot! Remember to follow along at the hashtag #MagPiMonday! 📷

01. We love a bit of OctoPrint here, and this is a great use of Raspberry Pi Zero 2 W by Dr Footleg.
02. We'd have to do some maths to figure out just how much more powerful this Zero 2 W array would be – orders of magnitude at least.
03. This is an interesting console idea, which seems to have taken some inspiration from the Switch.
04. As long as there are ultrasonic sensors attached to Raspberry Pi, there will be theremins.
05. The recycling of this drift wood into a tide indicator is very cool.
06. Open-source farming robot is an excellent combo of words – we're keeping our eye on this!
07. The bare mental synth project keeps growing.
08. Lorraine always comes up with great and fun projects, and this is no different.
09. This cute plushy night light is something that should exist as a product, we think.
10. Getting readings to look good and be informative at a glance is an art in itself.




Dr Footleg - Roboteer
@drfootleg

01

Replying to @TheMagPi

I designed and 3D printed a case for my new Pi Zero 2 W and the Pi camera which mount on aluminium extrusion, then set it up running OctoPrint on my 3DPrintMill.
(Pi case was remixed from an original idea posted by Emily Murry on grabcad.com/library/raspbe...)
#MagPiMonday





Kevin McAleer
@kevsmac

02

Replying to @TheMagPi

#magpimonday no Christmas decorations yet - but i am building a Clustered-Pi based on the Cray-1 super computer from 1976. #cluster #raspberrypi #raspberrypi2 #ansible #opensource #stem #3dprint #3Dprinting





Stewart Watkins
@stewartwatkiss

03

Replying to @TheMagPi

I received a hardware preview of the PIP. #RaspberryPi based handheld games console for retro games and budding games developers.

A bit limited at the moment (waiting on software updates), but I've been taking a look at the hardware.

penguintutor.com/news/raspberry...

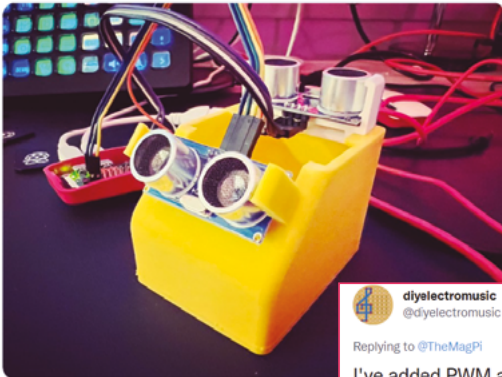
#MagPiMonday



Kevin McAleer @kevsmac 04

Replying to @TheMagPi

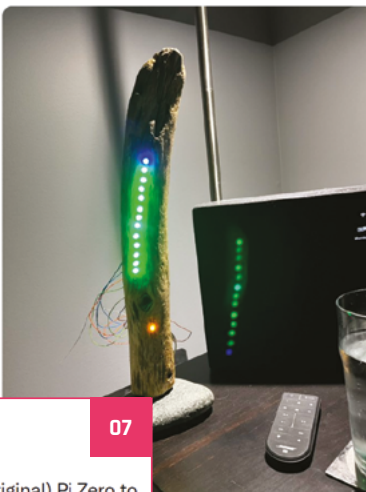
I've created a raspberry pi zero 2 based theremin using two range finders and a few lines of python (link in bio to the video) #MagPiMonday #raspberrypi #python #opensource #STEM



Roger Provost @13piRager 05

Replying to @TheMagPi

Working on my tide level and direction drift wood integration



diyelectromusic @diyelectromusic 07

Replying to @TheMagPi

I've added PWM audio output to my (original) Pi Zero to use with my "bare metal" synth.

#MagPiMonday

diyelectromusic.wordpress.com/2021/11/12/ras...

Taylor Alexander @TLAlexander 06

Replying to @TheMagPi

Yes! I am doing a major new rev of the motherboard for our raspberry pi powered open source farming robot! It will be open source (done in Kicad) once I am done with it.

Taylor Alexander @TLAlexander · Nov 12

Getting much closer now. Showing two views of the main board, then the serial board and the power distribution board. PDU switch is out of stock tho so I have to rework that to use P channel MOSFETs. Will do a full BOM stock check (and order!) before going to fab. [twitter.com/TLAlexander/st...](https://twitter.com/TLAlexander/status...)

Show this thread



Lorraine Underwood @LMcUnderwood 08

Replying to @TheMagPi

I'm creating an interactive light up window for my neighbours/passing strangers to play with! It's got a micro:bit on the outside controlling buttons. Inside is a raspberry pi connecting to and controlling the lights.

Lorraine Underwood @LMcUnderwood · Nov 14

My #micro:bit #raspberrypi interactive light up window is working! I'm squishing the wires between my hands as capacitive buttons, not clap sensors! Just needs bolting to the wall and to be made pretty

Show this thread



TGD @TGD_Consulting 10

Replying to @TheMagPi

I've been working on this Raspberry Pi Zero 2 W based #LoRaWAN CO2 sensor node for @thethingstwrk. The Python code still needs a little tweaking to make the information on the #SSD1306 OLED easier to read. #MagPiMonday #IoT #stem #opensource #CitizenScience

TGD @TGD_Consulting · Nov 20

#Reboi yell. Yee-haw Bullseye! PITS is ready for it & transmits all readings from @senerion SCD40 CO2 sensor as #opendata via @thethingstwrk to @opensensemap.

#ITN V3 #LoRaWAN-node based on @Rasoberry_Pi #PiZero2 with @Acacfruit #LoRa Radio Bonnet. #MINT #IoT #DIY #CO2 #SCD4x



Temperatur	
21.2 °C	vor 25 Minuten
Luftfeuchte	
65.5 %	vor 25 Minuten

Amanda Haughs @MsHaughs 09

Replying to @TheMagPi

Finished my Pico Plushy for baby boy! Fun project 😊

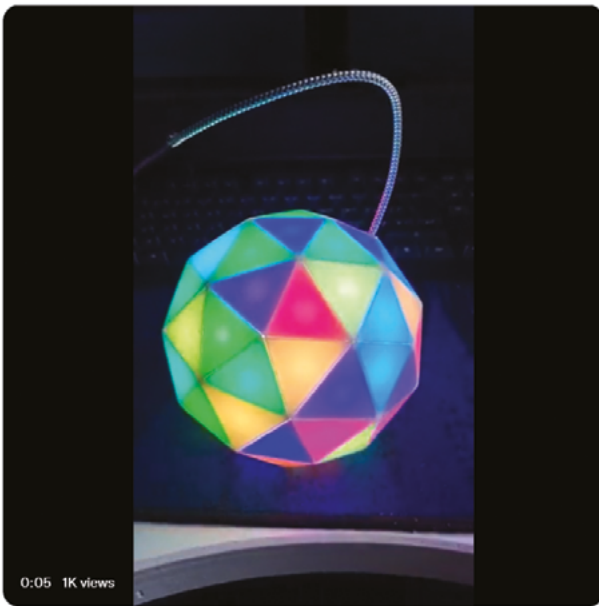




Andrew Porter
@defsdoor

Replying to @TheMagPi

Pico inside (tm)



0:05 1K views

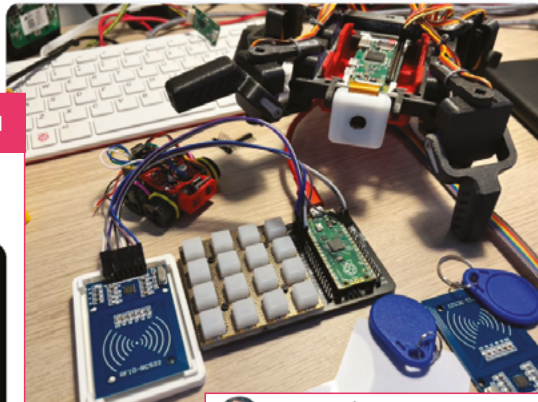
11



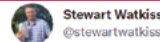
Kevin McAleer
@kevsmac

Replying to @TheMagPi

This weekend I've been working with the Raspberry Pi Pico and RFID tags and MicroPython - video here!
youtu.be/hV9GTqXLMpg #MagPiMonday #rasberrypico #pico #rasberrypi #robots #robotics #micropython #smallrobots #stem #opensource



12



Stewart Watkiss
@stewartwatkiss

Replying to @TheMagPi

Christmas lights are not up yet, here is last years. Controlled using Pi with Energenie remote control sockets.

Saves climbing under the tree to get to the sockets. Lights turn on and off automatically. Manual override using a Pi touch screen.



youtube.com

Christmas light controlled by Raspberry Pi. Home automati... This video explains about the Christmas lighting display I have at my house and how it's controlled using 3 Raspberry...

13



Mark Cantrill
@AstroDesignsLtd

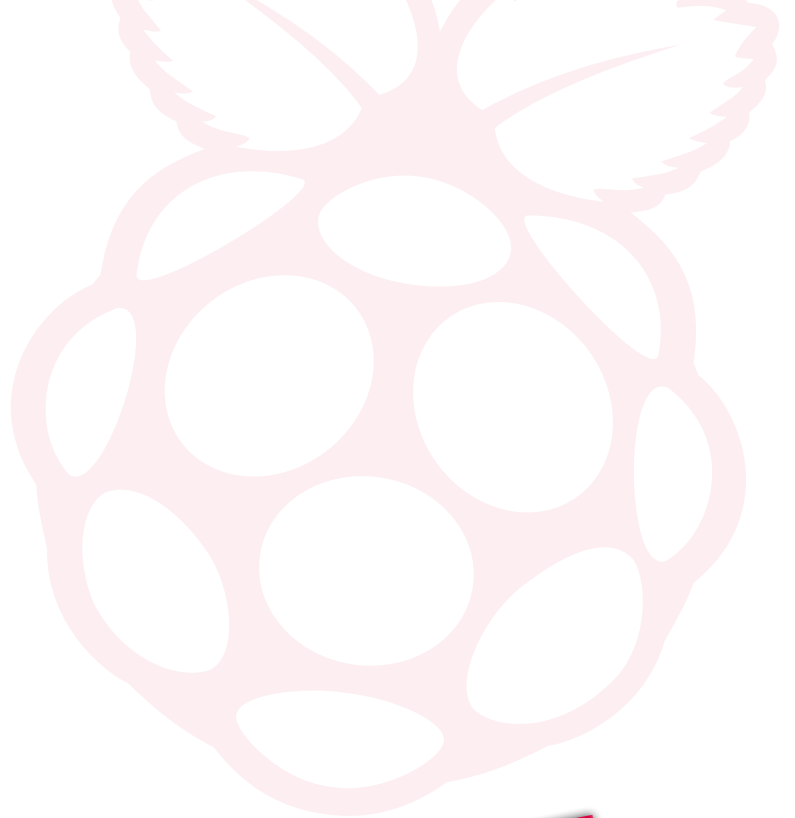
Replying to @TheMagPi

After a break from all things Pi for a while, I finally finished a build i've wanted to do for about 6 years - a #rasberrypi based central heating thermostat. Includes Timer, thermostat, monitor and data logger! Plus you could probably play games on it too!

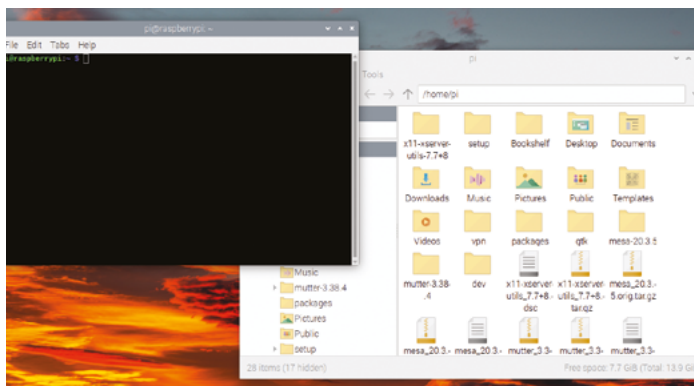
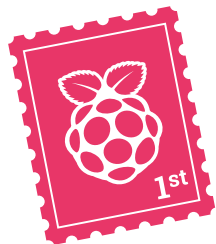


14

11. We're up pondering this orb.
12. RFID and robotics is a great combo, as it can help automate movement.
13. Send us your Christmas projects when you've finished them, we'd love to see!
14. This thermostat design is very nice, and we hope it helps keep Mark warm throughout the winter.



Your Letters



► Features Ed Rob's Dutch family enjoy seeing this version of *The MagPi*

► Bullseye is the recommended version of Raspberry Pi OS, but nothing is stopping you using an older version

The MagPi translated

I've been subscribed to the Dutch version of the magazine for several months and love getting it. I have a subscription query though, who should I contact? Are they the same people I should contact if I have a project to show?

Mo via Email

Each of the translated versions of *The MagPi* have specific contact forms on their website (magpi.nl, magpi.fr, magpi.de, etc.) and they are the best places to go about your subscription to them. As for submitting projects, you can definitely send them our way, and they will eventually make their way into the translated editions!

Buster or Bullseye?

I'm working on a new project, and I was wondering whether or not to use the new Bullseye version of the OS or stick with Buster for now as I know it?

Alex via Facebook

Generally, Bullseye is very similar to Buster, so you shouldn't have many issues having to learn anything new – mostly it's software updates and security fixes. Some projects may work a little smoother on Buster though, especially ones that involve the camera. For that, it might be better to use the new Legacy version of Raspberry Pi OS. You can find a copy in Raspberry Pi Imager and read all about it on Raspberry Pi's website (magpi.cc/legacy).



Popping up

I saw that there was a pop-up Raspberry Pi Store in London recently. Do you have any other plans for pop-ups, maybe in other countries?

Dana via Twitter

Unfortunately there are no plans, that we know of, for other pop-up versions of the Raspberry Pi Store. However, that's the nature of pop-ups really! If there is enough demand, I'm sure the team will keep note in case the opportunity arises. Never say never!



▲ The two-day London pop-up looked very swish

Contact us!

- ▶ Twitter [@TheMagPi](https://twitter.com/TheMagPi)
- ▶ Facebook [magpi.cc/facebook](https://www.facebook.com/magpi.cc/facebook)
- ▶ Email magpi@raspberrypi.com
- ▶ Online forums.raspberrypi.com

3 ISSUES FOR £5



☎ Subscribe by phone:
01293 312193

📧 Subscribe online:
magpi.cc/subscribe

✉ Email: magpi@subscriptionhelpline.co.uk

THE OFFICIAL Raspberry Pi Beginner's Guide

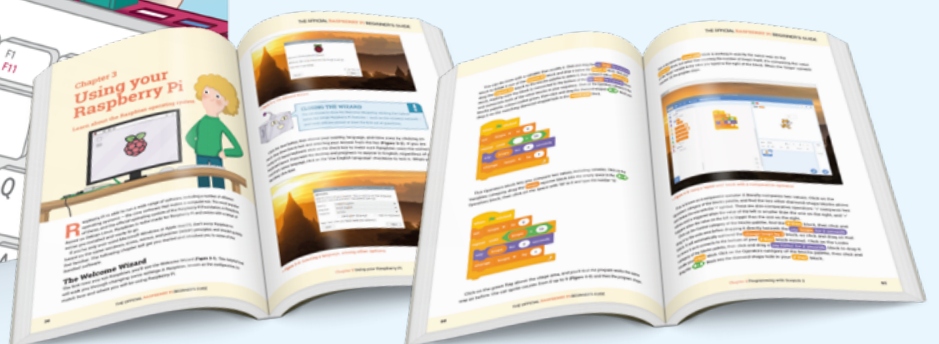
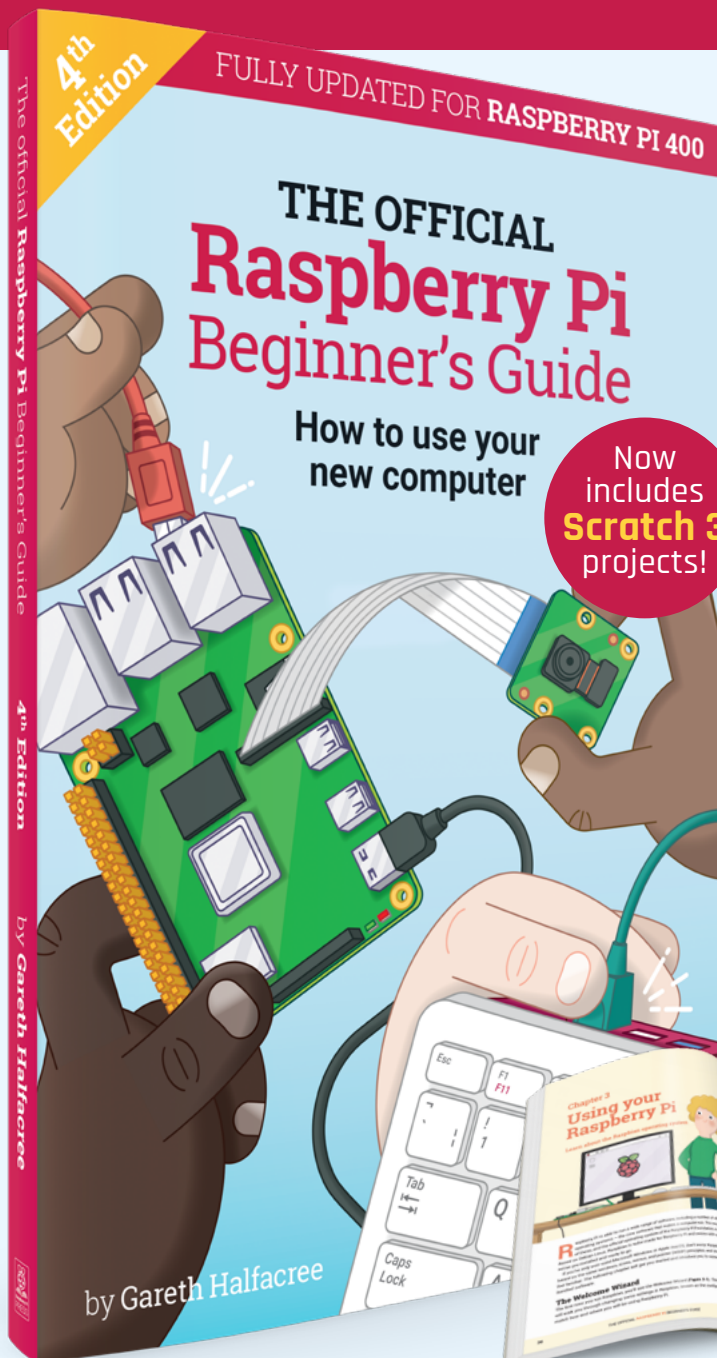
The only guide you
need to get started
with Raspberry Pi

Inside:

- Learn how to set up your Raspberry Pi, install an operating system, and start using it
- Follow step-by-step guides to code your own animations and games, using both the Scratch 3 and Python languages
- Create amazing projects by connecting electronic components to Raspberry Pi's GPIO pins

Plus much, much more!

£10 with **FREE**
worldwide delivery



Buy online: magpi.cc/BGbook

WIN ONE OF FIVE

PICOSYSTEM CONSOLES! IN ASSOCIATION WITH PIMORONI

The amazing PicoSystem is a great handheld system powered by RP2040 that lets you prototype and play games quickly. Last month we gave it 9/10, and now you have a chance to win one of five.



Head here to enter: magpi.cc/win | Learn more: magpi.cc/picosystem

Terms & Conditions

Competition opens on **15 December 2021** and closes on **27 January 2022**. Prize is offered to participants worldwide aged 13 or over, except employees of the Raspberry Pi Foundation, 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 The MagPi 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 or Facebook.

Wireframe

Join us as we lift the lid
on video games



Visit wfmag.cc to learn more

LEARN TO CODE WITH RASPBERRY PI

Discover Scratch and Python programming



THE MAGPI #114 ON SALE 27 JANUARY

Plus!

Know your maker tools

Build a LEGO® remote control car

Make a plant monitor

DON'T MISS OUT!
magpi.cc/subscribe

TWITTER @TheMagPi

FACEBOOK fb.com/MagPiMagazine

EMAIL magpi@raspberrypi.com

EDITORIAL

Editor

Lucy Hattersley
lucy@raspberrypi.com

Features Editor

Rob Zwetsloot
rob@raspberrypi.com

Sub Editor

Nicola King

ADVERTISING

Charlotte Milligan
charlotte.milligan@raspberrypi.com
+44 (0)7725 368887

DESIGN

criticalmedia.co.uk

Head of Design

Lee Allen

Designers

Olivia Mitchell, Sam Ribbits,
Ty Logan

Illustrator

Sam Alder

CONTRIBUTORS

David Crookes, PJ Evans,
Rosemary Hattersley, Richard
Hayler, Nicola King, Phil King,
Rob Miles, David Plowman,
Nik Rawlinson, Marc Scott

PUBLISHING

Publishing Director

Russell Barnes
russell@raspberrypi.com

Director of Communications

Liz Upton

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
magpi.cc/subscribe
magpi@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.

The MagPi magazine is published by Raspberry Pi (Trading) Ltd, Maurice Wilkes Building, St John's Innovation Park, Cowley Road, Cambridge, CB4 0DS. 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: 2051-9982.





Decade

2022 marks a decade of Raspberry Pi. How things have changed. By **Rob Zwetsloot**

When I first heard about Raspberry Pi, it was late 2011 and the technology landscape was quite different. The Motorola Atrix and Xoom were the premier Android phones, iPhone’s signal went wonky if your finger was in the wrong place, and the Xbox 360 and PS3 were about halfway through their quite extended life cycle. With months until Netflix streaming would hit the UK, I was reading about the upcoming Raspberry Pi and getting pretty interested. I had no idea what I could use it for, but I knew I could definitely use it for something, even if a Twitter acquaintance thought it would flop like the Ouya (they have since deleted the tweet, ending with ‘it doesn’t even have a battery’).

When Raspberry Pi came out, I had just started writing for a long defunct Linux magazine at a now dead publisher, and we were immediately enthralled. I spent years writing tutorials using a 256MB Raspberry Pi 1 Model B – turning Raspberry Pi into robots, media centres, file servers, routers, BOINC nodes, and much more. One day, I even managed to break the entire office LAN while playing around with network

settings, oops. It was a lot of fun though, and I learned loads doing it.

Late bloomer

I was one of the kids that grew up in the late nineties and early noughties who experienced the loss of proper computing lessons at schools, so to be able to learn to code and learn practical programmable electronics in my twenties was a huge thing for me. It’s one of the reasons I landed on *The MagPi* in 2015, and more importantly, the reason I have stayed for around 80 issues.

“ The passion of the community that first year was electric ”

Seeing the passion of the community that first year was electric. When we put out issue 40 with Raspberry Pi Zero on the cover, we thought it might be popular, but we didn’t realise just quite how popular. It took about a week for the constant stream of messages from people wanting to get a copy to die down, and it took years for us to

stop getting messages asking if we had any left at all.

Since then, I’ve been through many things with Raspberry Pi. The launch of Raspberry Pi 3, the AIY Vision Kit we did with Google, sending Raspberry Pi to space, new cameras, Raspberry Pi 4, Raspberry Pi 400, Raspberry Pi Pico, and recently Raspberry Pi Zero 2 W, just to name a few.

Now where?

As the second decade of Raspberry Pi looms ahead, my enthusiasm hasn’t died down a bit. I still get new ideas every month for something to do with Raspberry Pi, and I will never cease to be amazed at the creativity of the community and how much they’ve supported us over the years. I’ll be right here with you to see what 2022 brings to Raspberry Pi, and the ever-changing world it’s helping. 🍷

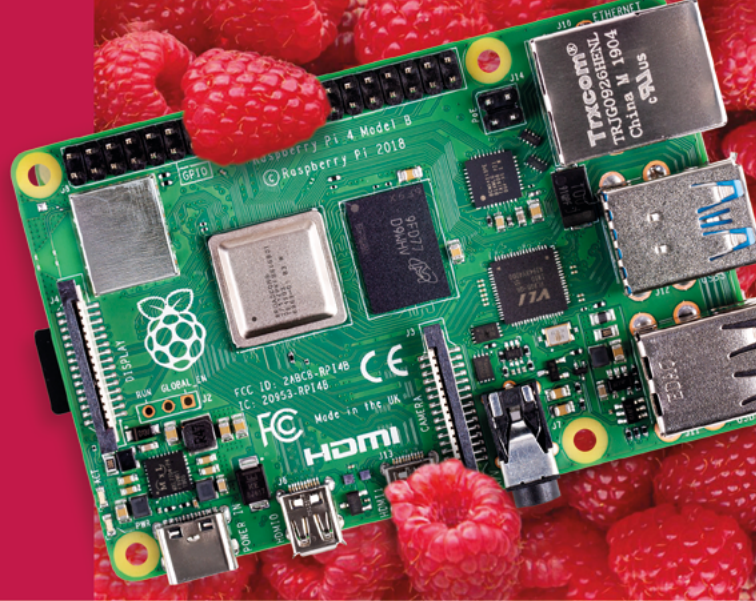
Rob Zwetsloot

AUTHOR

Rob is nearly 35 and has decided that makes him very old, despite being born the same year as *Star Trek: The Next Generation* started.

magpi.cc

American Raspberry Pi Shop



- Displays
- HATs
- Sensors
- Cases
- Arcade
- Swag
- Project Kits
- Cameras
- Power Options
- Add-on Boards
- Cables and Connectors
- GPIO and Prototyping

Partner and official reseller for top Pi brands:



and many
others!

Price, service, design,
and logistics support for
VOLUME PROJECTS

USA

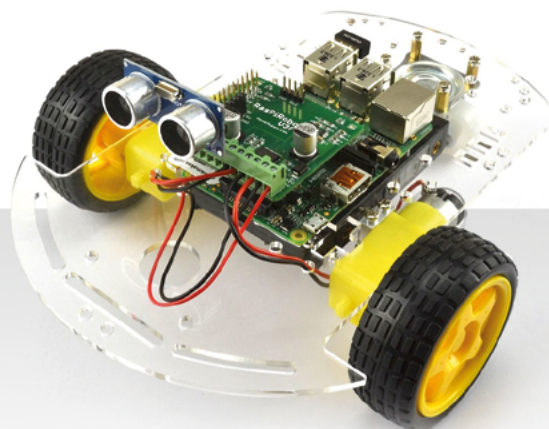


PiShop.us

Canada



PiShop.ca



Raspberry Pi

APPROVED RESELLER

HiPi.io

HIGHPI PRO

•———— The new case from the HiPi.io team ————•



- Rapid tool-free assembly and disassembly
- Large internal volume for HATs
- Compatible with Pi 2/3/4
- Multiple lid options
- Passive & Active Cooling options
- Secure microSD card cover
- VESA mount support
- Molding-configurable output ports customizable for volume orders
- Printed logo for your branding

Available at these great Pi stores:



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