The MagPi

Issue 118 | June 2022 | magpi.cc

The official Raspberry Pi magazine

RASPBERRY PI PHOTOGRAPHY

SETUP • EDITING • PROJECTS

TIME-LAPSE & BATCH CONVERT **TUTORIALS**

BUILD YOUR OWN **VECTREX MINI**

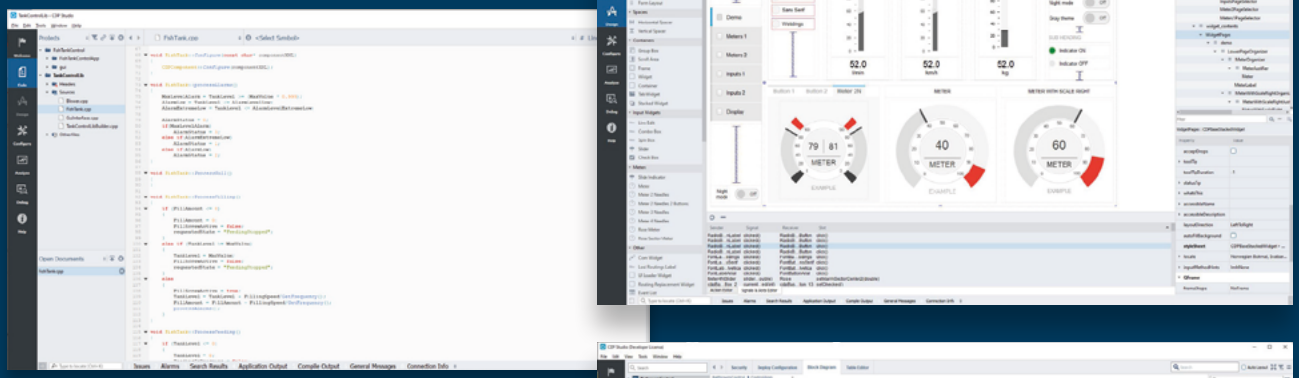
CRAFT WEARABLE KIT

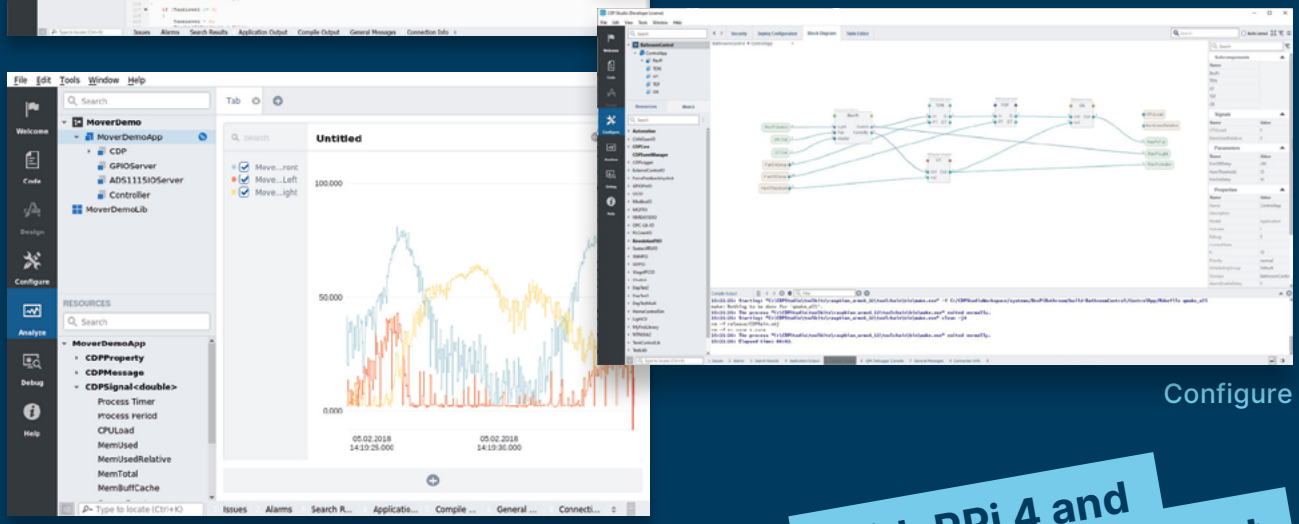
10 **FACE TRACKING** PROJECTS

WIN! CUTIEPI COMPUTE MODULE 4 TABLET

Design

Code





Configure

Analyze

With RPi 4 and Bullseye support

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@cdptech.com
CDP Technologies AS // Hundsværgata 8, 6008 Ålesund, Norway

CDPStudio

WELCOME
to The MagPi 118

Photography is a great way to make use of your Raspberry Pi single-board computer.

Raspberry Pi's camera modules provide our small computer with an eye to the world, enabling sight-based projects and digital camera creations. Using the newer High Quality Camera enables you to snap images every bit as impressive as with an expensive camera.

But, of course, you can do so much more with a Raspberry Pi because it's a fully programmable computer. Raspberry Pi can perform smart vision sensing tricks – capturing whatever you ask it to. And you can automate shots, taking one every few minutes to create stunning time-lapse shots. You can even use the images to create your own amazing works of art.

This month we have a superb photography feature (page 32) that collects everything you need to know about Raspberry Pi cameras. Plus, Nik Rawlinson has written an excellent time-lapse tutorial (page 40), and Sean McManus provides us with the first part of ArtEvolver (page 44) – a fantastic primer on batch-converting images with Raspberry Pi.

Some of my favourite Raspberry Pi projects involve cameras. And this issue is packed with some of the greatest image-based Raspberry Pi projects around.

Lucy Hattersley Editor

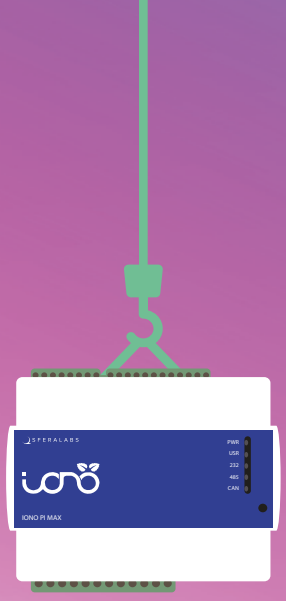
EDITOR
Lucy Hattersley

Lucy is editor of *The MagPi* magazine and keeps a beady eye on things. She is a big fan of art and photography computer projects.

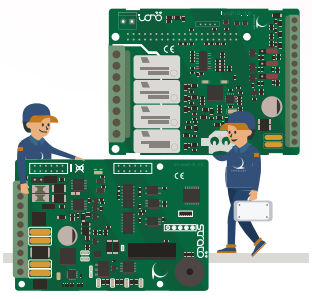
@LucyHattersley

SFERALABS

Raspberry Pi goes industrial



Long term availability, high reliability and support



Tailor-made embedded solutions

Our range of Raspberry Pi-based devices is wide.
Really wide.
But what if your project needs something more?

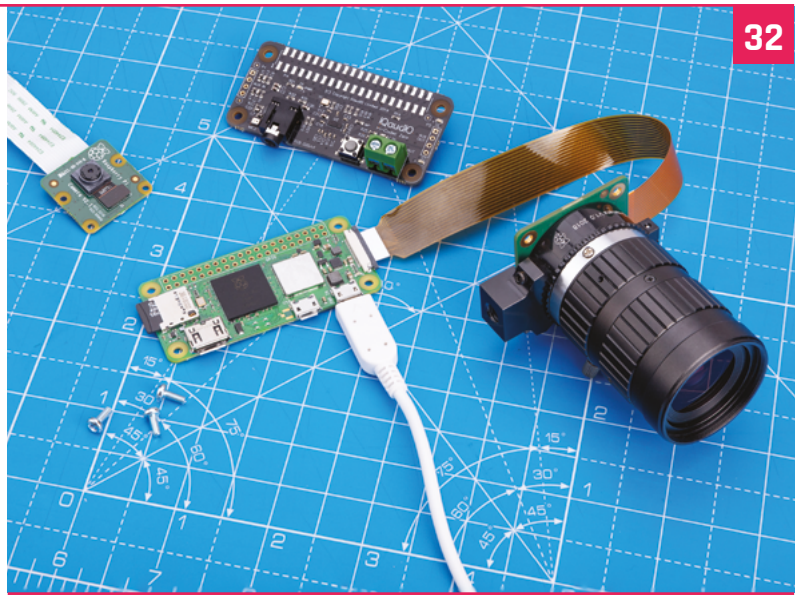
Visit our website www.sferalabs.cc

POWERED BY Raspberry Pi

Contents

Issue 118 > June 2022

Cover Feature



Regulars

Project Showcases

Szerafin MM5D Mushroom Farm

NMS Ceefax

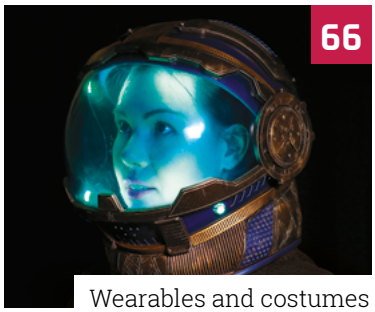
The MagPi magazine is published monthly by Raspberry Pi Ltd. Publishers Service Associates, 2406 Reach Road, Williamsport, PA, 17701, is the mailing agent for copies distributed in the US and Canada. Application to mail at Periodicals prices is pending at Williamsport, PA. POSTMASTER: Send address changes to The MagPi magazine c/o Publishers Service Associates, 2406 Reach Road, Williamsport, PA, 17701.

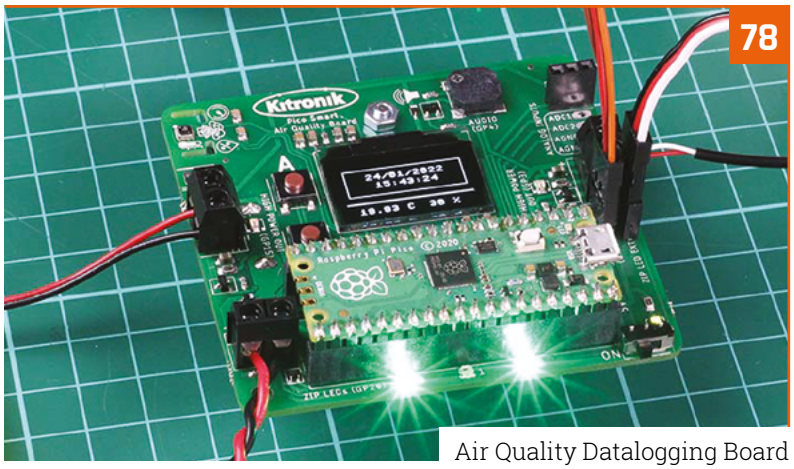
Tutorials

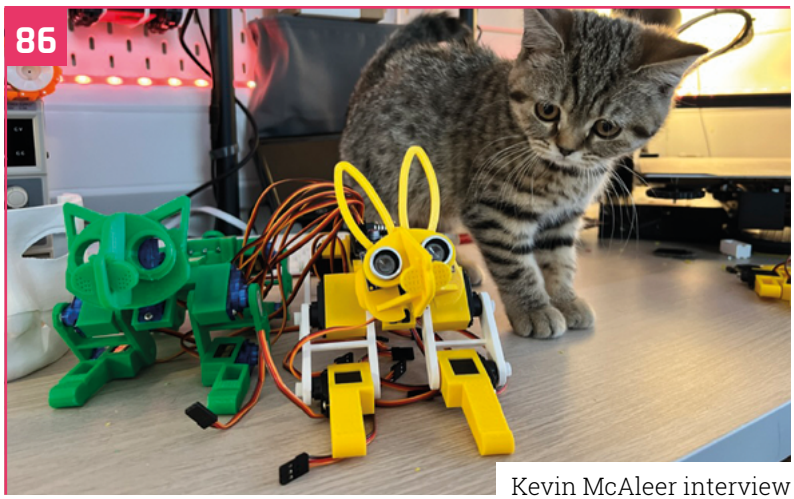
The Big Feature

Reviews

Community







WIN A CUTIEPI

WIZnet

WizFi360 Design Contest

at maker.wiznet.io

WizFi360 with

ARMKEIL

Raspberry Pi RP2040

ARDUINO

Bring your **creativity** and win an **Apple iPad Pro**

WizFi360

- Official Wi-Fi Shield on ARM Open-CMSIS-Pack and Keil Studio Cloud
- Easy-to-connect Wi-Fi to Pico RP2040
- Support Azure, AWS SDK Example

Free Sample

More details at maker.wiznet.io

WizFi360

or

WizFi360-EVB-Mini

or

WizFi360-EVB-Pico

Contest begins	Apply & get a free sample	Submissions close	Winners announced by
June 20	July 31	September 30	October 17

30 Winners will receive **Apple iPad Pro($1,099)** 12.9-inch 128GB Wi-Fi

Portsdown 4 Digital TV Transmitter

Fancy running your own digital TV station? Dave Crump did and he used a Raspberry Pi to do it. **PJ Evans** takes to the airwaves

MAKER

Dave Crump

Dave is a retired RAF pilot who qualified for an amateur radio licence at 14 and has been transmitting television since he was 16.

magpi.cc/portsdowndatv

Dave Crump is no couch potato; in fact, he much prefers it on the other side of the TV set. Since his early years, he has had a passion for amateur TV. Starting with analogue home-built equipment, his projects have raised him up to be a key player in the British Amateur Television Club. His latest project, Portsdown 4, brings the new world of digital television transmission to a wider audience than ever before.

"I was inspired by the desire to reproduce a capability that a few years ago would have occupied half a room and cost hundreds of thousands of pounds, and replace it with something cheap and portable that could be used by myself and my fellow amateur TV enthusiasts,'' Dave tells us. Fellow enthusiasts had been discouraged by the seeming complexity of DTV (digital TV) broadcasting. It was assumed to be out of the reach of the home enthusiast, but the advent of Raspberry Pi changed all that. "Raspberry Pi brought two key elements to the project at the beginning. The first was hardware H264 encoding. Radio amateurs are limited in the amount of bandwidth and power that they can use for communication. The second was easy image capture using the camera, which works seamlessly with the H264 encoder."

Amateur TV enthusiasts can use satellite dishes to transmit digital TV worldwide

Proof of concept

The surge in popularity of software-defined radios (SDRs) meant that a relatively cheap piece of hardware could handle the reception and transmission of DTV signals using the DTB standard used by services such as Freesat. Dave started to piece together a proof of concept connecting a Raspberry Pi 3B, SDR, camera, and audio via a £5 USB dongle to create a rudimentary DTV transceiver. It was named Portsdown, in tribute to a late president of BATC in whose Portsdown home the idea took shape.

> "A capability that a few years ago would have occupied half a room and cost hundreds of thousands of pounds"

Now Dave has completed Portsdown 4, using the more powerful Raspberry Pi 4 Model B. In addition,

Enclosed here is a Raspberry Pi Camera, which provides the video feed

An official Raspberry Pi 7" touchscreen provides control over operations

A robust case containing the Raspberry Pi ensures the unit is suitable for outdoor transmission

Quick FACTS

- Dave Crump is chair of the British Amateur Television Club (BATC)
- The concept came from a French radio amateur – Evariste Coujard
- The project uses the LimeSDR USB to transceive
- The project is open-source and anyone can build one
- The Es'hail 2 satellite can be used by amateurs

▲ For more adventurous transmissions, the Portsdown 4 can be taken anywhere

▶ Add a dish and you can receive transmissions from amateur satellites

Warning!
Frequency restrictions

Use of this project requires a valid UK licence and must not be operated in restricted frequencies. Different restrictions apply in different parts of the world. Do your research if attempting to recreate this project.

rsgb.org

BATC Portsdown 4 DATV Transceiver Main Menu

TX		RX	M2	M3
Modulation	Encoder	Output to	Format	Source
S2QPSK	H264	Pluto	16:9	TestCard
Freq	Sym Rate	FEC	Band/Tvtr	Pluto Pwr
437 MHz	333	2/3	70_cm	0
EasyCap	Caption	Audio	Atten	Att Level
Comp Vid	On	Auto	NONE	-10.00
Preset 1	Preset 2	Preset 3	Preset 4	Store
146.5_333	437_Lime	1255_HD	437-Pluto	Preset

Dave's touchscreen user interface is perfect for the job. It is high contrast with large buttons that are clearly labelled

a Raspberry Pi 7-inch touchscreen and strong case have made a complete, travel-ready unit suitable for outdoor transmission. The power and frequencies that can be used are heavily regulated by Ofcom and you'll need a licence to operate a

> You'll need a licence to operate a Portsdown 4

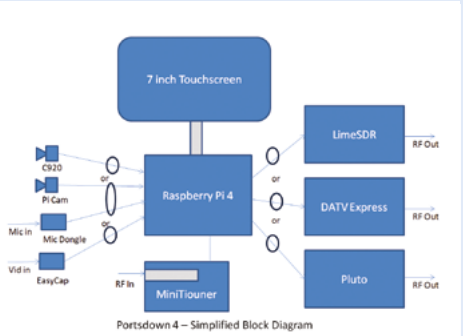
Portsdown 4, but such licences have been granted to those as young as ten years old. Amazingly, you can add a powerful enough antenna to relay your DTV signal to a satellite that has transponders available to amateur radio enthusiasts.

Compatibility

Dave's biggest challenge was compatibility. The build required specific components and peripherals. When he published the build and the software, he soon found that even very similar models of webcams and SDRs could cause problems. This led to the birth of the project wiki that provides a detailed bill of materials to anyone who wants to try their hand at building their own. It's estimated that over 500 hobbyists have built their own Portsdown transmitter.

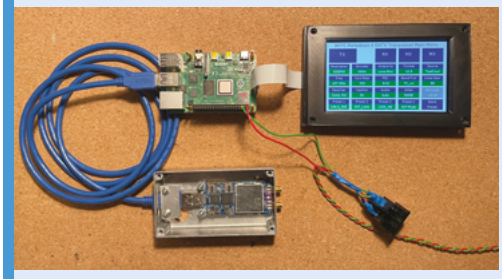
We asked Dave, what was next for Portsdown? "Development of the Portsdown system as a piece of radio test equipment is proving to be very popular. Current capabilities include a radio frequency signal generator, a simple radio spectrum analyser, and a receiver noise-figure meter. Developments are underway to provide additional test-bench capabilities using the existing hardware."

Evolution of Portsdown 4

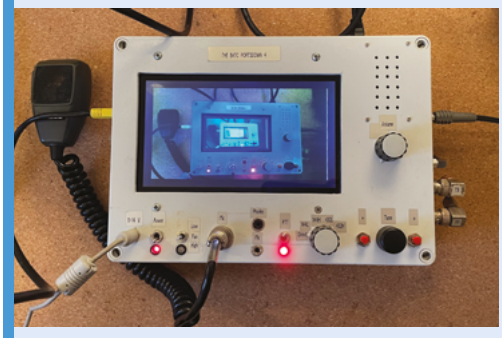


Portsdown 4 – Simplified Block Diagram

01 Dave started with a block diagram that showed how the key parts would link together. The key section is communication with a software-defined radio (SDR) that would handle transmission. A tuner unit allows incoming signals.



02 The next step was a proof of concept. Here we can see the three key parts: Raspberry Pi, SDR connected by USB, and touchscreen. Dave created the custom touchscreen software to control the system.



03 Dave wanted to be able to transmit from anywhere and, in particular, to relay signals from a satellite. This required a rugged enclosure with suitable connectors. This also shows the ability to receive digital video.

Air raid siren monitor

Dmytro Panin's project has the potential to save lives as war in Ukraine continues to devastate the country, as **David Crookes** explains

MAKER

Dmytro Panin

Dmytro Panin is a programmer based in Ukraine, and he wrote his first line of code aged eight. He works for a large provider of nearshore software engineering services.

magpi.cc/airraidmonitorgit

Last year, *The MagPi* featured two projects by Dmytro Panin: his Air Quality Traffic Light in issue 108 and his Solar System Display in issue 110. The first project sought to address concerns that his home city was one of Europe's most polluted, while the second allowed him to gaze at the skies. When we caught up with him recently, however, he was worried about his city and staring upwards for a very different reason.

Dmytro is a Ukrainian from Kyiv, and he and his family are among millions of people displaced and suffering since Russia invaded their country. "I was in Kyiv when the first wave of explosions surged across Ukraine and we left the city with nothing but our backpacks," he says. Yet among the items he managed to grab before evacuating were a Raspberry Pi Zero computer and an e-ink screen – an odd decision, he admits, but one that has proven to be very useful.

Getting alerts

By using those two components, Dmytro has created an air raid siren monitor that shows which parts of the country are being shelled. He says he never imagined that he would ever make such a device, but the war has shifted the paradigm of what is a must, a necessity, or useful. "I know where all the load-bearing walls in my apartment are," he laments, saying he now looks at old concepts in completely new ways.

Although there are lots of air raid sirens across Ukraine, there are still many places in a city where they cannot be heard

"When an air raid or shelling starts, we usually hear sirens going off, signalling that citizens should go to a bomb shelter or take cover," he explains, of his motivation to create the device. "I noticed that my family would try and get additional information on the probability of air raids before going out by browsing through media and other channels. I thought it would help to have a device that's always on that could show this information about active air raid sirens across Ukraine."

To do this, he turned to the popular instant messaging platform Telegram. "It has features resembling a social media platform," Dmytro says. "One of these is 'Channels' – pretty much a one-to-many information distribution platform. You subscribe to a channel you're interested in and get messages with links, photos and videos from people

> "We left the city with nothing but our backpacks"

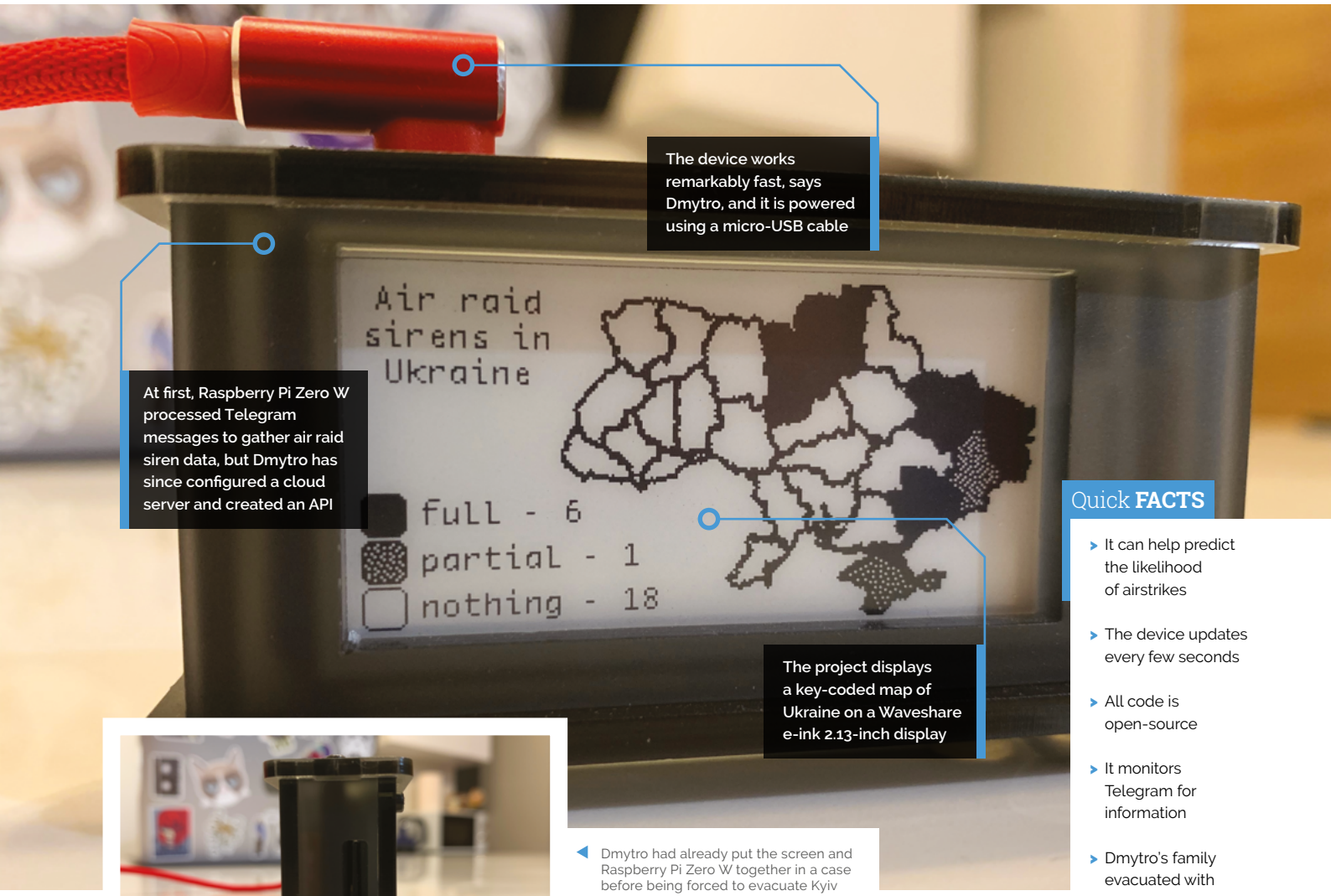
who run it. Telegram became a way for officials to notify where air raid sirens start and stop."

Making predictions

When Dmytro began his project, the e-ink screen was already connected to Raspberry Pi Zero. The idea was to code a program that would monitor and parse messages in Telegram and create a visual snapshot of the current situation across Ukraine.

"Not only does it show regions where the air raid sirens are active, it helps to predict a potential air raid by looking at the spread and progression of the attack – chances are that if multiple regions are hit, it will spread," Dmytro says.

It wasn't all plain sailing, though. "The problem was that, even though I had a preconfigured microSD card installed in Raspberry Pi Zero, I needed to change the configuration in the 'boot' drive and a card reader wasn't something I considered essential when packing my bag! I ended up putting the microSD card into a family



The device works remarkably fast, says Dmytro, and it is powered using a micro-USB cable

At first, Raspberry Pi Zero W processed Telegram messages to gather air raid siren data, but Dmytro has since configured a cloud server and created an API

The project displays a key-coded map of Ukraine on a Waveshare e-ink 2.13-inch display

Quick FACTS

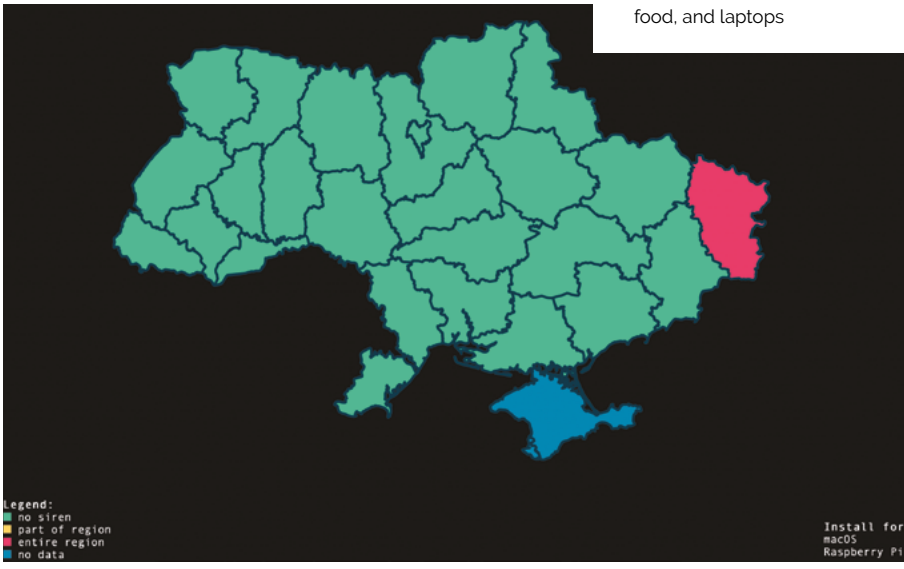
- It can help predict the likelihood of airstrikes
- The device updates every few seconds
- All code is open-source
- It monitors Telegram for information
- Dmytro's family evacuated with paperwork, clothes, food, and laptops

Dmytro had already put the screen and Raspberry Pi Zero W together in a case before being forced to evacuate Kyiv

member's phone and using a text editor to configure the cmdline.txt and config.txt to SSH into Raspberry Pi Zero through the USB connection."

The project has also led to the creation of a mobile-first online map at **sirens.in.ua**. "It's hard to get Raspberry Pi in Ukraine if you don't already have one," Dmytro reveals.

So far, the device has proven useful. "Last weekend, we were getting ready to go to the shop but, based on the data from the device, we postponed this and the air raid siren went off ten minutes later," he says. Since posting about the monitor online, lots of other Ukrainians have also created their own version. "I have open-sourced the code and the instructions. I hope it keeps people safe during these unthinkable times."



There's a macOS widget too, with Linux and Windows versions in the works

Synch.Live

Combining art and brain science with Raspberry Pi? It's not as strange as you think. **Rob Zwetsloot** ponders it

MAKER

Hillary Leone

A New York-based artist working at the intersection of art, science, technology, and social impact.

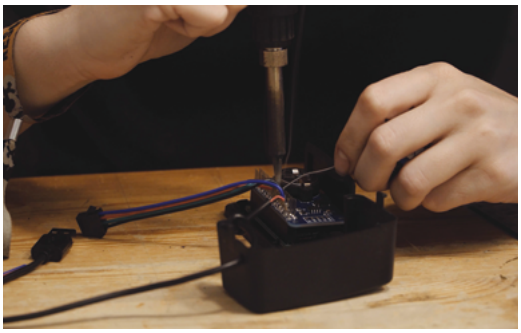
synch.live

Creating art with Raspberry Pi is one of our favourite things to see, which is why we have been excited about this project during an email exchange with its creator, Hillary Leone.

"Synch.Live is a joyful, participatory community art experience and cutting-edge brain science experiment, designed to spark emergent behaviour and fortify human connection," Hillary tells us. "Here's how it works. A group of players wear bowler hats equipped with randomly flashing LEDs. Their challenge is to synchronise the LEDs on all of the hats without talking or touching. A tracking system mounted above tracks collective movement and calculates the level of group co-ordination in real time. If the group can figure out how to move together in a coherent, emergent fashion, the LEDs will synchronise."

Hillary explains that emergence can mean many things: "capturing the creation of galaxies, intelligent behaviour in ant colonies, economics, ecosystems, and conscious brains." The use of it in this context is brand new.

▼ The tracking system uses an HQ Camera and hangs from the ceiling



▲ Construction of the hat system

Idea emerging

How does one go about creating such a system? "Functionally, the system needed to extract position and movement information, feed it into the emergence algorithm, and then create a visual feedback loop to signal how close the participants were to the goal of the game," Hillary explains. "The system needed to run the emergence criterion in real-time and provide feedback to the player hats. The hats needed to be able to 'blink' in perfect sync, like fireflies do in nature. They needed to be safe and comfortable; flexible to allow

> "The system needed to run the emergence criterion in real-time"

for complex patterns and artistic expression; and precise and reliable in order for the results of the scientific part of Synch.Live to be valid."

Building the system wasn't so easy, and the team also wanted to make the code easy to use as well. "We wanted to build a system that was easy for other artists, scientists, and enthusiasts to replicate and play with," Hillary says. "Scalability was also an important requirement: while we are currently running small pilots with only a few players (ten), we envision a future for Synch.Live where dozens of players meet up to create large mesmerising displays of emergent co-ordination.

The lights are used by the people in the experiment to know if they're co-operating

The whole thing is powered by a single mobile battery

All the components are stored in and around the hat

"With those requirements in mind, we designed the software to be highly modular and extensible. Precise clock synchronisation and scalable deployment were tough nuts to crack; we solved them with RTC modules and Ansible. We have been lucky to have a network of very supportive friends and collaborators, including professional software engineers and computer vision researchers, who have helped us make this project a reality. The project has been developed fully open-source, and our in-house software engineer Madalina has thoroughly documented the build process in her blog (**mis.pm/projects**)."

Scientific fun

If you're like us, this all sounds very fascinating, and also a fun experience to participate in.

"People are incredibly excited about the work and its potential applications," Hillary mentions. "As one attendee wrote after one of our presentations: 'The vision is powerful, the implications and potential are clearly articulated and beautifully expressed, and the need for this work is deep and significant.'"

▲ Here's how the experiment looks in action

The project is currently undergoing further refinements as pilots are run to test the system, and Synch.Live will show up on 18-19 June at Imperial College London.

"Once we have the data to demonstrate collective emergence, we want to scale-up the group size, design the wearable, develop new scenarios and rules sets, and explore applications such as conflict resolution and team building." Hillary finishes, "Ultimately, we want to make the experience of Synch.Live available to communities around the world, and make a film that celebrates our collective humanity."

Quick FACTS

- Synch.Live began as a thought experiment about future language
- Birds flocking together, without a leader or plan, is emergence
- The tracking system uses a Raspberry Pi 4 and camera...
- ...while the hats use Raspberry Pi Zero W and an RTC add-on
- Conversations about the project started in 2020

Szerafin MM5D Mushroom Farm

Hobbyist farmers Judit and Zsolt used engineering and tech know-how to expand their mushroom farming business. **Rosie Hattersley** hears how

MAKER

Zsolt Pozsár

Judit and Zsolt Pozsár's remotely managed mushroom farm is an excellent first Raspberry Pi project for the engineer and his wife.

magpi.cc/szerafin

With two decades of engineering experience gained during military service, Hungarian Zsolt Pozsár resumed his vocational qualifications and began a career maintaining the bottling lines for a multinational company. Meanwhile, his wife Judit's small farming concern seemed as though it could benefit from Zsolt's knowledge of how technology could improve their crops. The couple have been growing mushrooms for the past five years. Zsolt's MM5D Raspberry Pi-based monitoring system – a four-channel programmable control and remote monitoring system – has kept an eye on the mushrooms' development for the past three years, first in a cellar underneath the house, and more recently in dedicated mushroom fruiting chambers.

▶ The mushroom tents have recently been joined by similarly controlled outdoor vegetable crops

"The aim is to ensure the automatic operation of mushroom growing sites, so the characteristics of the growing environment can be monitored and modified remotely," says Zsolt. "Devices must operate continuously and reliably in a humid environment without user intervention." Remote management without smart controller devices such as relays and timers is impossible. "These operations should be user-friendly, with no programming knowledge required."

Zsolt wanted a small, low-power, Linux-based single-board computer. He chose Raspberry Pi for his mushroom farm on the recommendation of a technical vocational high school teacher who'd used them and said he wouldn't be disappointed. Zsolt now uses Raspberry Pi 3B+ in all his projects, all of which he self-designs, making use of standard items such as display, relay boards, and third-party libraries. The hardware cost no more than 40,000 Hungarian forints (approximately $110, or £90). "The hardest part of building a device is making the box and making the front panel. It required a closed box. I bought this, but it was hard to find a company that deals with Plexiglas cutting and printing," Zsolt explains.

Mushroom for growth

The mushroom monitoring system consists of a TTL-level input in which temperature and humidity are measured. The remotely accessible

▲ Mushrooms benefit from their optimal growing conditions

Quick **FACTS**

- Zsolt enjoys P languages: Pascal, Perl, and Python
- And has a weakness for rescuing abandoned cats
- His army years were spent looking after air defence radars
- He still can't resist repairing old electron tube equipment
- But he's currently tinkering with irrigation controls

Warning!
Mains Electricity

Do not connect 230V contacts to the relays on the printed circuit board, as their high coil voltage and current will also cause the contact to spark and interfere with the display's serial data traffic. Use low voltage and current fast relays and move them away from the control unit.

magpi.cc/electricalsafety

▲ Zsolt's controller allows the user to see at a glance the status of the crop and the current environment settings

system shows the status of the lights, fans, humidifier, and watering system – including the pressure, and whether or not the tent door is open. "Measurement, timing, and electrical equipment control is done by a Python-language program that runs as a service in the background," Zsolt tells us. The program requires an internet connection and access to data from **OpenWeatherMap.org**.

> Growing mushrooms this way has proved successful enough for Judit and Zsolt to operate a local delivery service

The MM5D setup (one of several he's developed and implemented at the family farm), uses Raspberry Pi 3B+ and has been in continuous use since 2019. "The devices were built one after the other, so I was able to use the experience gained in building for the next device." This iterative process is reflected in Zsolt's meticulous GitHub where he makes the installer software for his MM5D plant-monitoring devices available online (**magpi.cc/mm5dgit**). He also maintains his own Debian repository for Raspberry Pi OS, Debian, and Ubuntu Linux (**magpi.cc/pozsi**).

Grow more greens

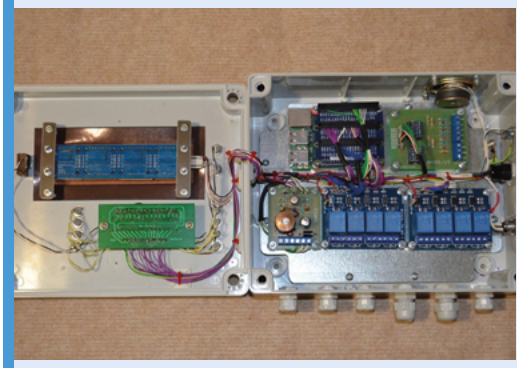
Growing mushrooms this way has proved successful enough for Judit and Zsolt to operate a local delivery service for customers in the vicinity

of their farm in Tiszaföldvár, close to Budapest. Their mushrooms, as well as oyster mushroom compost useful for growing other crops, are sold in environmentally friendly packaging.

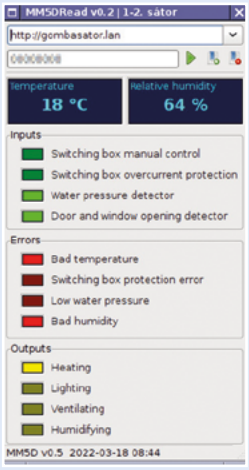
They've also expanded with additional mushroom fruiting chambers, as well as diversifying into other crops. The addition of these vegetable plots has also given Zsolt the excuse to come up with another monitoring project adding automated irrigation to the whole site, as well as the mushroom tents. "The water and electrical system, electrical cabinet, and pump shaft will be ready by the summer. Tomatoes, eggplants, and pumpkins will grow and the environment will be beautiful!"

▼ Judit and Zsolt's son shows off the latest mushroom crop grown in their cellar, produced with the help of the original 2018 prototype controller

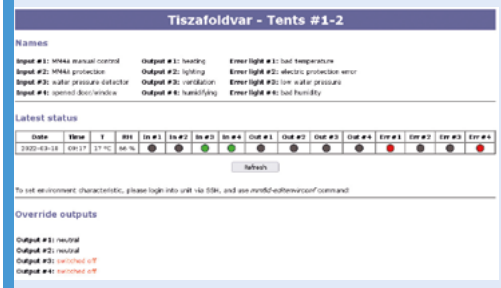
Grow your own



01 The MM5D controller unit houses Raspberry Pi and sensors and controllers to adjust the lights, temperature, and humidity sensors. It's linked to **OpenWeatherMap.org**.



02 Powered by Raspberry Pi 3B+, the MM5D has four inputs, four controllers, and four status relay contact outputs. Parameters are customisable. Details of the Python-based hardware controller are at: **magpi.cc/mm5d**.



03 Environmental settings can be manually overridden, but the system is largely designed to operate automatically.

Vectrex Mini

A bespoke version of a rare video games console is within your reach... if you're willing to make one. **Nicola King** learns how to 'mini-fy' an inimitable console of yesteryear

MAKER

Brendan Meharry

Brendan runs a YouTube channel about retro gaming called Retro Game On. As a keen maker too, he tries to mix the two hobbies.

retrogameon.com

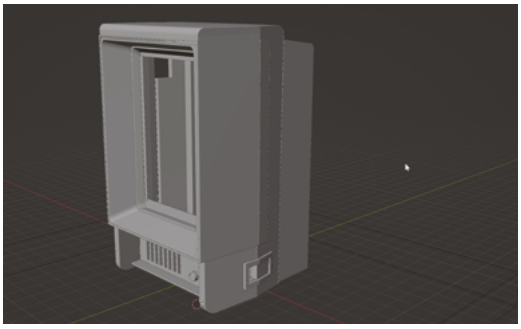
Remember the Vectrex video games system? Maybe not. Launched in 1982, it soon disappeared. Still, it was one of the most curious consoles ever made, with an integrated vector CRT monochrome monitor – colour could be added with the addition of an acrylic screen overlay that came with each cartridge-based game. Only 28 titles were officially released, including the built-in Mine Storm, although some homebrew games were later created by fans.

Anyone now wanting to get their hands on a Vectrex will find themselves hugely challenged, because this unique games system is extremely rare. With that in mind, Aussie retro gamer Brendan Meharry decided to fashion his own version of this classic console, investing some six months of his spare time into the project in the process.

Honey, I shrunk the console

Having experienced the console's delights, playing on an original Vectrex in a local gaming museum, Brendan was motivated to make one. "They were released in Australia, but they're quite rare," he tells us. "I was inspired by official mini consoles like the SNES Mini or PlayStation Classic. I thought it would be fun to 'mini-fy' something more obscure, especially since the original included a built-in monitor." So, with his 3D printer ready to roll, he set to work.

The Vectrex Mini didn't prove too expensive to create, totalling around $70 all-told, not including the spare Raspberry Pi that he already had in his stash. For the audio element of the build, he used a super-small PAM8302 amplifier by Adafruit. Brendan employed a Waveshare 2.5-inch LCD screen for the project, but would he entertain the idea of replacing that with a mini upcycled CRT display? "Maybe! To be honest, the high voltage of CRTs terrifies me (even the small ones), but I'd love to see if someone else could pull it off," he enthuses.



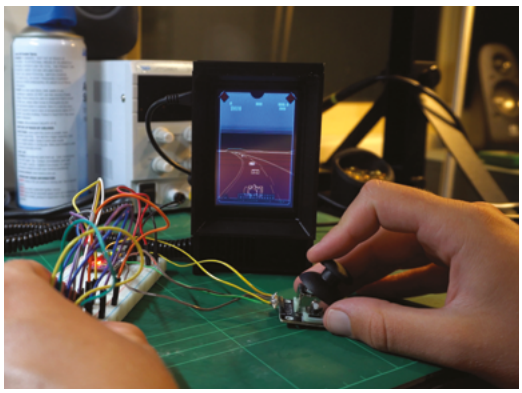
▲ Brendan designed the Vectrex Mini case in Blender and 3D-printed it

The Waveshare screen plugs into the GPIO pins of a Raspberry Pi 2 Model B which runs a Vectrex emulator using the latest build of RetroPie. "It took me a long time to figure out how to run the emulation in portrait mode with no weird scaling or stretching of the image," says Brendan of one of several challenges he faced along the way.

An achievable goal

Brendan says that building the separate controller, which is based on an Arduino Pro Micro with a KY-023 joystick module, proved to be the easiest part of the build. "If you've watched the [YouTube] video, it's plain to see that it was my first time designing something like that. But on the upside, the later designing/building of the controller went much smoother. It's obvious I learned from my mistakes," he affirms.

Despite the learning curve, Brendan is confident that such a project would be within the grasp of many hobbyists. "The 3D modelling is likely

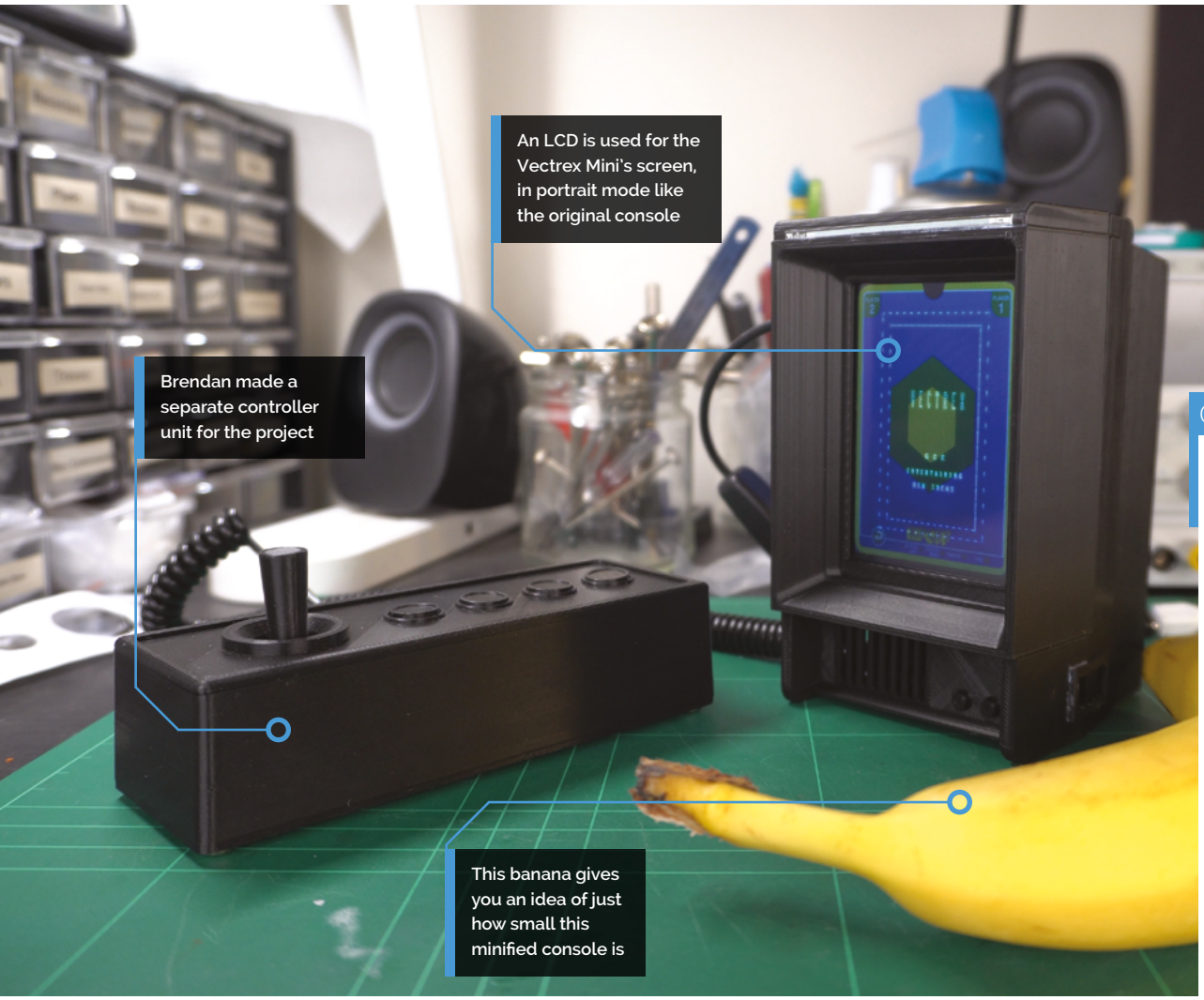


▶ Testing out the electronics for the controller unit

Alert!
Copyright

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

magpi.cc/legalroms



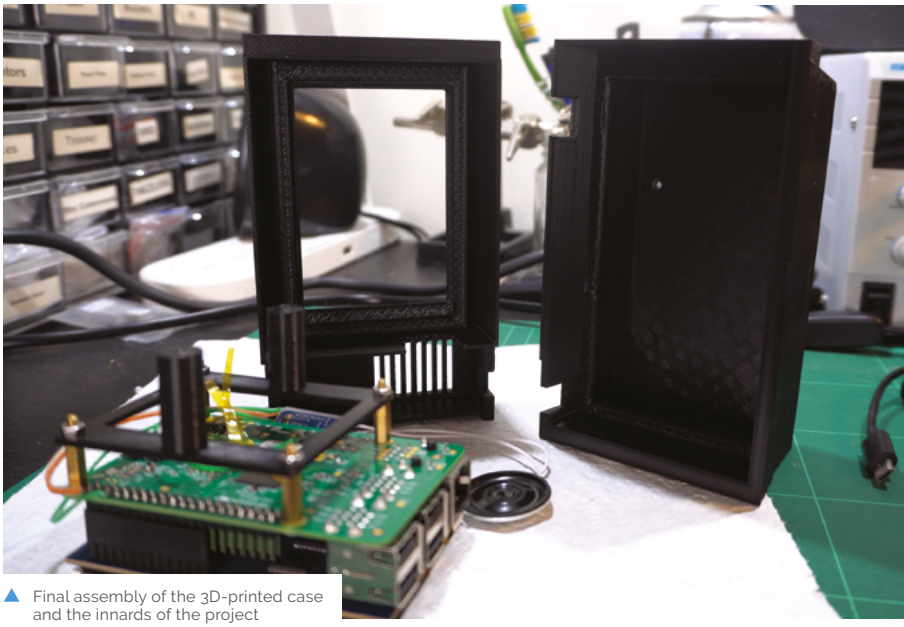
Quick **FACTS**

- It's powered by a standard, official 5.1V 2.5A micro-USB supply
- 3D printing time totalled around 30 hours
- The 3D models are on Thingiverse: **magpi.cc/vectrexmini3d**
- Brendan has also built a wooden arcade stick running RetroPie…
- View the video for it here: **magpi.cc/woodarcadestick**

> I thought it would be fun to 'mini-fy' something more obscure

the most complex portion of the project, but the wiring, software etc. is quite basic. Overall, I'd say it's a reasonably easy project for an experienced maker."

Brendan has a detailed and informative YouTube video (**magpi.cc/vectrexmini**) taking you through his build, which is well worth a watch if you're interested in fabricating your own version, and he's been bowled over by the response to his project. "As a newbie, I've been blown away by the encouraging comments and constructive criticism," he recounts. But, more than that, he's also gained a huge amount from the experience, and surely that's what we all want to acquire from every project we make.



▲ Final assembly of the 3D-printed case and the innards of the project

NMS Ceefax

The BBC closed its Ceefax service in 2012 but Nathan Dane is keeping memories of the iconic blocky service alive, as **David Crookes** explains

MAKER

Nathan Dane

Hailing from Northern Ireland, Nathan spends most of his time working on electronic projects and coding.

magpi.cc/nmsceefax

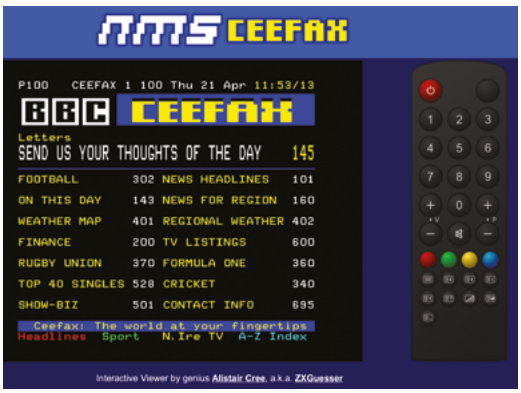
Before the internet, there was teletext – a brightly coloured, blocky-looking information service built into a huge number of television sets from the mid-1970s onwards. It allowed millions of people to read news and sport headlines, check the weather, find travel information, view TV and radio listings, and even play a quiz or two.

But while it still exists across the world, many popular services have long been axed – including BBC Ceefax which ended in 2012. Rather than allow its memory to wither, however, enthusiasts are determined to keep this charming service alive, among them Nathan Dane who had recreated his own version of Ceefax on a Raspberry Pi connected to a VBIT-Pi board.

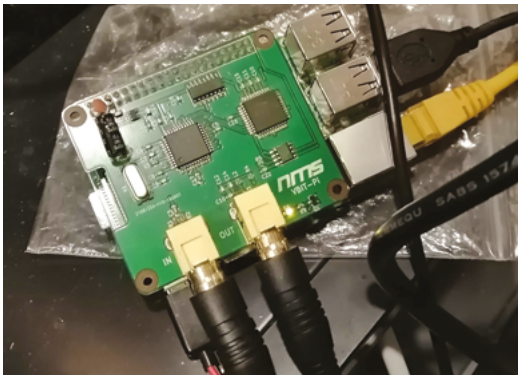
Top stories

Nathan's original aim was to produce a teletext service as a personal project. "I only really wanted a service that would be useful to me and my family, so I coded things like the national and local news headlines for my dad to read, and wrote 'school news' and other information for my sister and I," he explains.

To do this, he made use of Peter Kwan's VBIT-Pi project and Alistair Buxton's Raspi-teletext software. "The VBIT-Pi adds the teletext signal to any PAL composite video signal so you can play a TV channel through it and add teletext," he says. "Raspi-teletext adds the teletext signal to Raspberry Pi's composite video output, and anyone can use it to generate teletext without additional hardware."



You can view the pages on Nathan's website using an interactive viewer created by Alistair Cree



Nathan has designed his own VBIT-Pi 3 board which adds a teletext stream to a PAL video signal

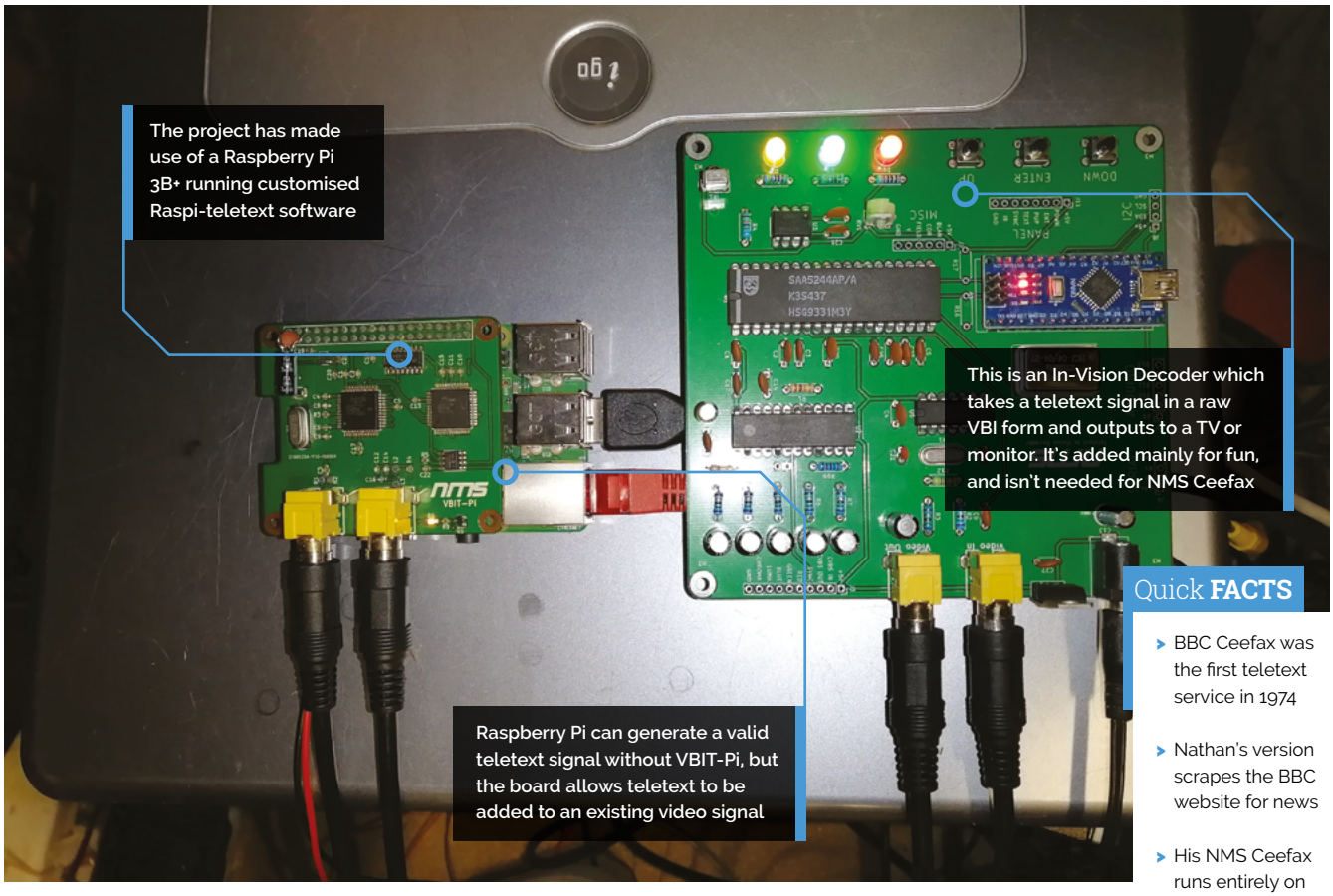
At first, he would manually type stories from BBC News – which he found too time-consuming, despite the 40-column text screen limitations. It was then that he started scraping the BBC website for news and sport content, going as far as adding the ever-popular football league tables.

"The BBC offers RSS feeds – XML documents automatically filled with links to the latest headlines," Nathan says. "My code parses these links and extracts the text from each page. One issue, of course, is that some stories don't fit well into the teletext format, such as reports made up mostly of photos or videos, so I had to build in checks to remove those."

Rain or shine

Nathan also tweaked the Ceefax page design so that it would allow headlines longer than 35 characters and he made his NMS Ceefax, as he calls it, draw data from other sources, too. TV listings, for instance, come directly off-air from the Freeview Electronic Programme Guide. Weather, meanwhile, is taken from the Met Office's API – a decision that overcame a major challenge.

"For those who don't remember, the Ceefax weather map was essentially a blocky outline of



Quick FACTS

- BBC Ceefax was the first teletext service in 1974
- Nathan's version scrapes the BBC website for news
- His NMS Ceefax runs entirely on Raspberry Pi
- He's built it up over six years
- It can be accessed via Nathan's website too

the UK with areas shaded in different colours," Nathan explains. "Coloured text around the edges told of the weather to be expected in the corresponding area, but this was challenging to recreate because it relies on a lot of 'fuzzy logic'."

> Raspi-teletext adds the teletext signal to Raspberry Pi's composite video output

At first, he hard-coded the four UK countries in different colours and included the data beside each of them. By taking weather from the Met Office, however, Nathan could download weather data for the UK's eleven regions. "Comparison code then grouped all the areas with similar conditions under one colour – it's not perfect, but it's close enough to do the job," Nathan reveals.

Having kept the project for personal use for a long time, he's now made his Ceefax publicly available online, where it's gone down well with visitors. "I was initially drawn to teletext because it was an old broadcast technology that I could recreate, but I very quickly came to love the simplicity of the format," he says. "People have said hard limits induce creativity and this couldn't be more true with teletext."

▲ This fuzzy image shows how the teletext signal looks in the Vertical Blanking Interval (VBI). "This was made visible by adjusting the V-Hold on a CRT to 'roll' the VBI down into view," Nathan says

AI Weather Station

Forecasting air quality with this Raspberry Pi weather station and machine learning, **Rob Zwetsloot** takes a whiff

MAKER

Kutluhan Akar

A self-taught full-stack developer, maker, and electronics enthusiast, he concentrates on proof-of-concept environmental and ecological projects.

Air quality is a big issue in many parts of the world – especially in cities and other built-up areas. While we've seen a couple projects like this before which just show current air quality readings, this project does something a bit more special.

"I focused on creating an AI-driven, budget-friendly device to collect local weather data with ozone concentration to train a TinyML (TensorFlow Lite) neural network model," Kutluhan Akar, the project's maker tells us. "[It runs] the trained model to forecast air quality levels so as to get prescient warnings regarding pulmonary risk factors to avert potential hazardous respiratory diseases."

It's built like a regular home weather station, with sensors that can track local conditions. Combining those with the ozone concentration and online data in the training model adds that extra sauce to the project.

Arduino and Raspberry Pi

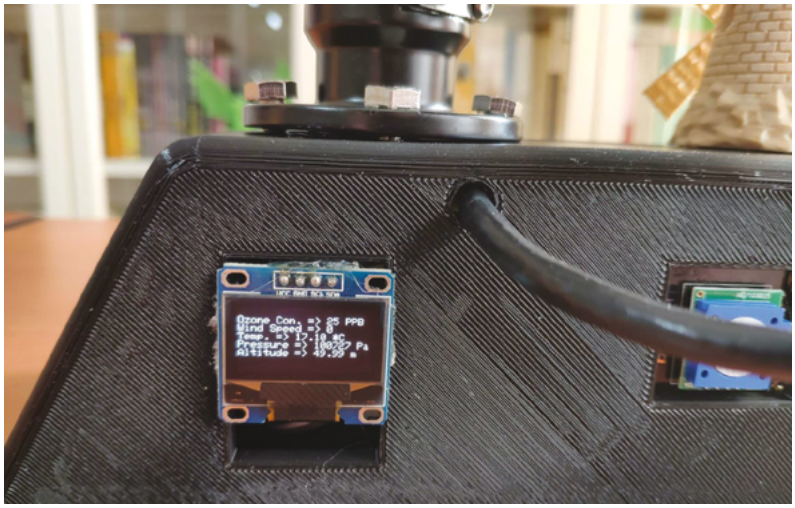
Data is fed to Raspberry Pi from an Arduino Nano 33 BLE for analysis."I decided to utilise an Arduino Nano 33 BLE in this project since it can easily collect local weather data with ozone concentration and run my neural network model outdoors after being trained," Kutluhan explains. "To collect the required data to train my model, I connected an I2C ozone sensor, an anemometer, and a BMP180 precision sensor to the Nano 33 BLE. Then, I added an SSD1306 OLED display to monitor the collected data in the field.

▲ The completed kit is very sturdy and has some weather resistance

> "The predictions are at roughly 92% accuracy for the three quality classes"

"Since I collected local weather data with ozone concentration on my balcony, I was able to transmit the collected data from the Nano 33 BLE to a Raspberry Pi 4 in my house over BLE instead of sending data packets to a web server as usual. In that regard, I was able to transfer data packets via the Nano 33 BLE without requiring any additional procedures."

After running the data through TensorFlow, he assigned the results to three categories: Good, Moderate, and Unhealthy. With the training done, he transferred the model back to the Nano 33 BLE as a C array so it could use the model on the system for predictions.



▼ Multiple screens show different data

Quick FACTS

- If you wish to replicate this, Kutluhan says a Raspberry Pi 3B+ will work too
- Displays around the weather station display predictions
- The C array is a .h file that the Nano 33 uses
- Raspberry Pi was able to read the wireless data from the Nano 33...
- ... which were stored in CSV files for analysis

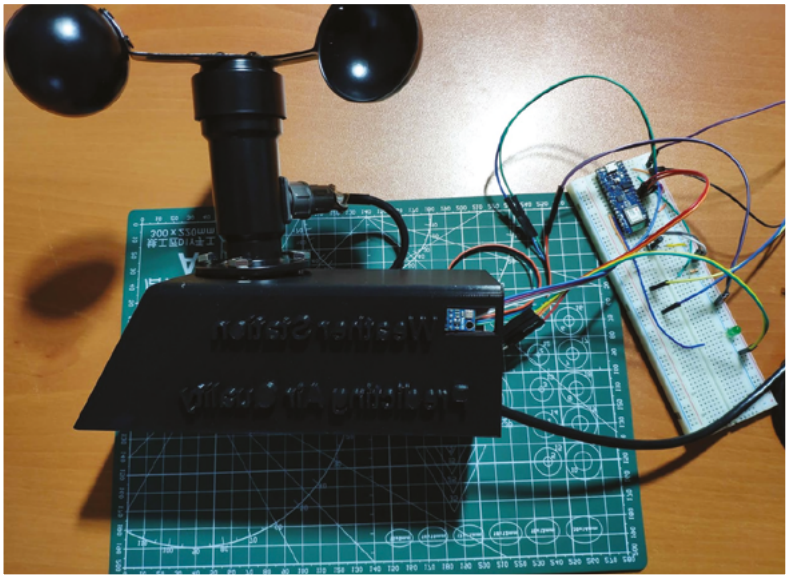
"Lastly, to make the weather station as sturdy and robust as possible while enduring harsh weather conditions, I designed a windmill-themed case (3D-printable)," Kutluhan finishes.

Good quality

According to Kutluhan, the predictions are at roughly 92% accuracy for the three quality classes.

"After publishing my project, I received encouraging comments to keep improving this weather station as an SDG (Sustainable Development Goals) project," he reveals. "[I also got] questions regarding how to make the station compatible with WiFi or GPRS by utilising different development boards."

Kutluhan is planning to add LoRaWAN and GPS capabilities to the weather station in the future, for collecting data in forests and industrial areas. We look forward to seeing how it goes!



▲ Testing the weather station to make sure it all works

Fibre Optic Matrix Display

This unique Raspberry Pi Pico-powered display produces some amazing arty effects. **Phil King** is mesmerised

MAKER

ElliotMade

A technical product manager for an online retail company, Elliot spends his spare time on electronics, machining, gardening, and just making things in general.

elliotmade.com

Warning!
CNC Cutter

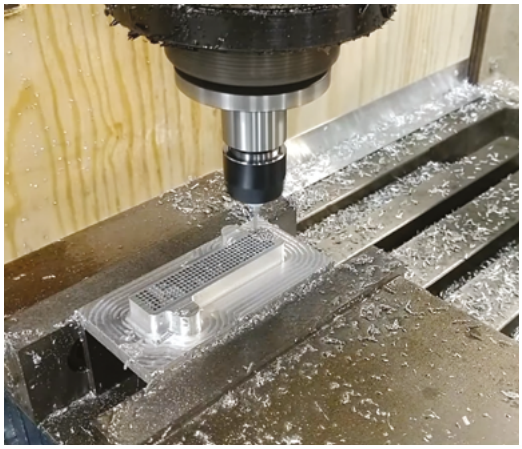
Be careful when using CNC cutters in your projects. Wear ear protection and safety glasses and stand clear of the machinery as it works. Understand the basic safety functions of your machine.

magpi.cc/cncsafety

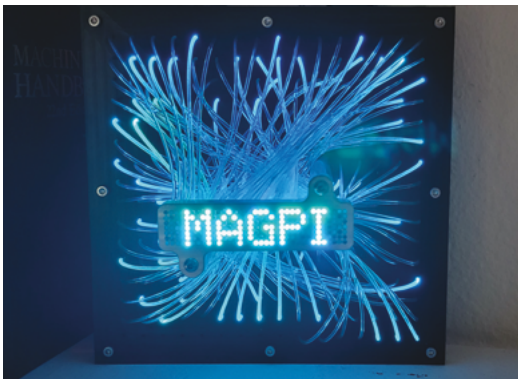
We've seen a wide variety of displays used in projects, but none quite like this. With a standard 16×16 NeoPixel LED matrix board connected by optic fibres to a grid of tiny holes in a smaller 35×7 front plate, it results in some cool colour effects from the light leaking from the sides of the fibres. To get the full impact, check out the video (**magpi.cc/fibreopticyt**) on ElliotMade's YouTube channel.

Elliot tells us the concept is a natural extension of the word clocks he's made previously. Having been introduced to Raspberry Pi Pico on a course at **teachmepcb.com**, he opted to use it for this project: "It fit the requirements for this display, plus all of the other good reasons: it's cheap, easy to get, runs CircuitPython, really easy to use, and has pretty great coverage in the community online."

After spending a year or two thinking about the idea – in which time he learned to use the SolidWorks CAD software, which came in useful – Elliot set about building the innovative display. "I spent a week of evenings and one weekend to knock out the mechanical construction and to get some basic code running."



▶ Elliot used a CNC machine to mill two of the panels and drill all the tiny holes, but they could be 3D-printed instead



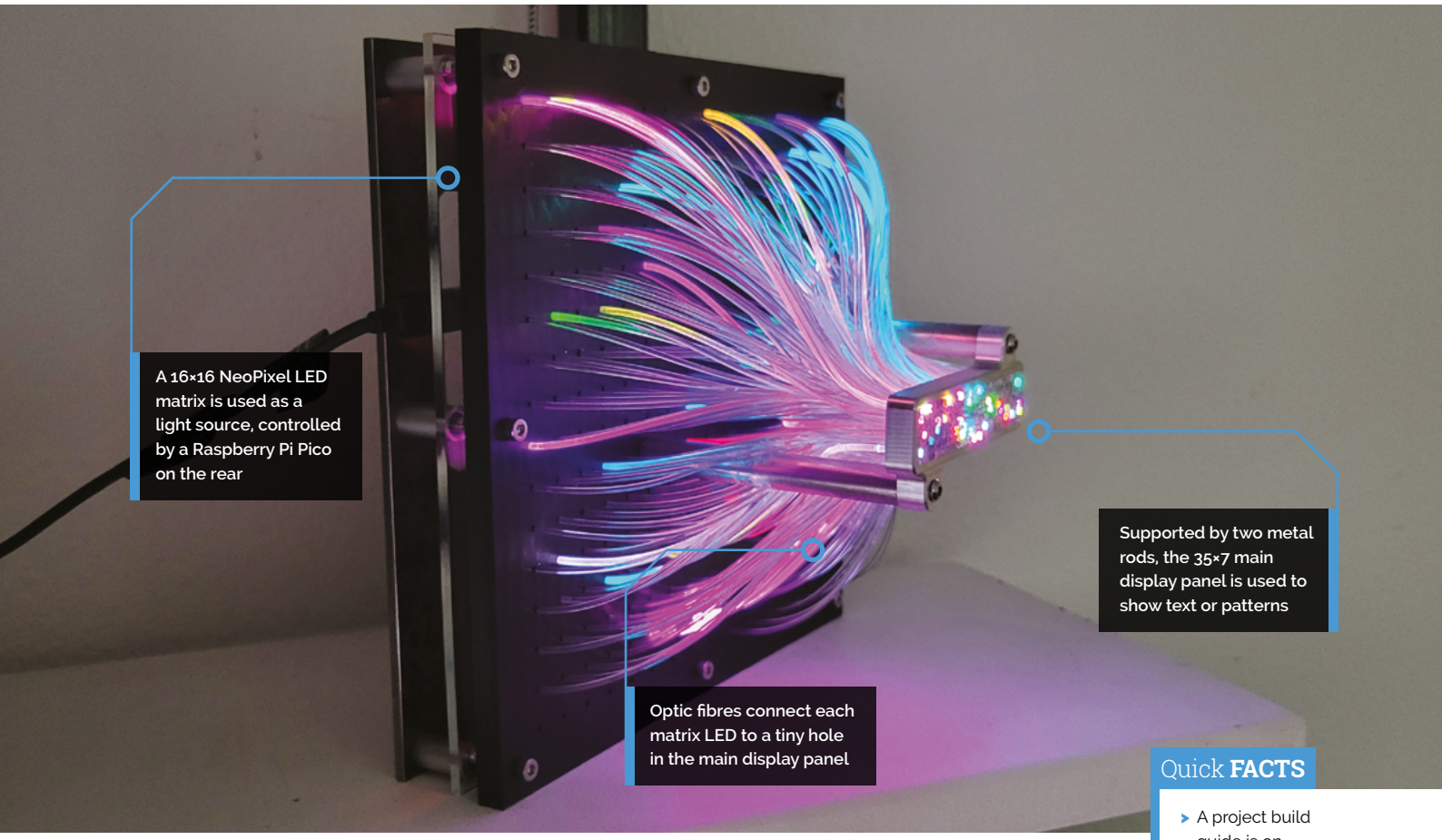
▲ Light leaking from the sides of the optic fibres results in spectacular patterns and effects

Sandwiched plates

The build comprises a number of plates, designed in SolidWorks, sandwiched together. At the very front, held by two metal rods, the main display grid has 35×7 tiny holes connected with optic fibres via a front plate to the LED matrix board behind, whose rear is supported by a middle plate. An optional back plate is used to mount the electronics, including Pico.

One of the main challenges was plugging all of the optic fibres in place: "They were not completely uniform in thickness," says Elliot, "so some fit easily, some were loose, and some didn't fit at all – by the end of it, my fingertips were pretty sore from threading them all in."

Always up for a fun challenge, he opted to make the plates on a CNC machine, milling their shape and drilling every small hole, which wasn't without its difficulties. "I used the same kind of solid carbide drill bit that is used in PCB manufacturing, which in hindsight was probably not a great choice for aluminium – they are very easy to break, particularly if the flutes fill up with chips in a hole. My CNC mill tops out at 3800 rpm as well, which isn't the best for small-diameter tools like this."



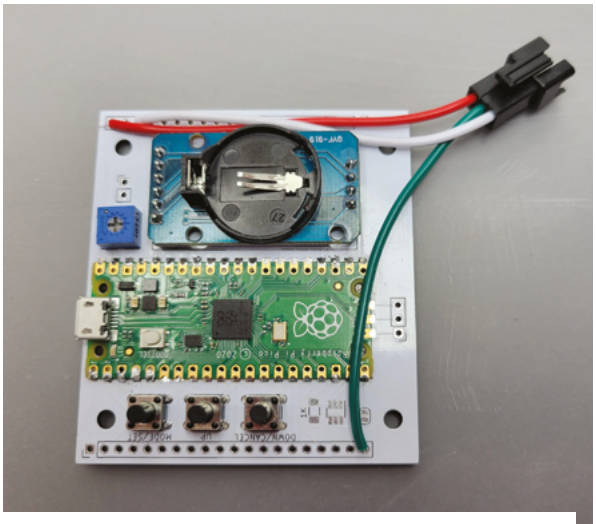
However, he says that anyone wanting to replicate the project could 3D-print the plates instead. "In fact, another user on Instructables took my files and printed his own version and it looks like it turned out great! This design is also quite easy to make with a laser cutter."

> One of the main challenges was plugging all of the optic fibres in place

Coding the matrix

With the hardware assembled, Elliot set about creating the Pico firmware to run it in CircuitPython, which he found challenging. "I code at a hobbyist level, so it takes a lot of effort to get what I want to actually happen. CircuitPython was a big help: it makes it super-fast to test changes by simply saving a file on the device and letting it reboot."

An array in the code is used to map 245 of the LEDs on the matrix to the corresponding positions in the 35×7 front display panel, as connected by the optic fibres. This makes it easy to adapt to show any text or pattern. So far, Elliot has used the display as a digital clock (with an RTC) and for scrolling text/graphics and animated effects, but "it can do just about anything you want, only limited by your imagination."



▲ A Raspberry Pi Pico controls it all, shown here mounted on an optional PCB with buttons and a real-time clock module

Quick FACTS

- A project build guide is on Instructables: **magpi.cc/fibreopticdisplay**
- The code and 3D model files can be found on GitHub: **magpi.cc/fibreopticgh**
- An EVA foam gasket is used to prevent light leaking to neighbouring matrix LEDs
- An external 5V power supply is required to achieve full NeoPixel brightness – and it's very bright!
- Elliot's previous projects include a calendar using a Raspberry Pi 3 and DAKboard

CASE STUDY

Crux Labs digital telephony

Raspberry Pi was this company's clear choice to enable seamless home working. By **Rosie Hattersley**

Crux Labs' CEO and founder Rohit Gupta noticed that, while home communications technology had become well-integrated, many small businesses were still using analogue telephony solutions, with no means of recording missed calls and following up on the potential business opportunities they represent.

Many firms with under 100 employees lack a specialist IT expert to maintain a classic PBX telephone system and integrated communications hub and, in any case, the cost of installation is prohibitive for many businesses at this scale. Gupta founded Singapore-based Crux Labs to offer an accessible digital telephony solution to SMEs, from sole traders up to around 250 staff.

THE CHALLENGE

Crux Labs had to ensure that neither cost nor complexity would be a barrier for small companies setting up their first digital telephony solution. Its customers needed an easy-to-install product that would require minimal support, while reliably delivering business-critical functionality.

THE SOLUTION

Crux LX, Crux Labs' first SME product, is a digital telephony solution aimed at businesses from sole traders up to around 250 staff, with a Raspberry Pi 3 at its heart. It uses SIP telephony and a wide area network to create a low-cost, scalable, device-agnostic and location-agnostic digital telephone network. Features such as call re-routing allow the integration of telephony across multiple office locations, as well as enabling employees to pick up calls from their home offices as though they were in the business base.

Customer demand led Crux to develop its second digital telephony product. Crux VX, a "call centre in a box", takes all the functionality of the Crux LX and adds CRM and call management features.

WHY RASPBERRY PI?

Crux Labs did their research: having trialled no fewer than eight other options, CEO Rohit Gupta says Raspberry Pi was the clear choice. Had Crux chosen a standard business telephony option, he believes it would have cost as much as £300,000 just to acquire a prototype. The powerful and low-cost Raspberry Pi 3 was ideal for large-scale rollout.

With Raspberry Pi, the stable hardware is suitable for the approximately 800 software packages that Crux LX contains, from minor handling features to UI overlays that ensure a consumer-friendly interface. The result is a high-quality product that can be set up in around 30 minutes by a non-specialist.

For Crux's multinational clients with offices in both Malaysia and Singapore, this sophisticated Raspberry Pi-based system offers substantial savings on international calls, at a cost to the client of less than a sixth of the price of a comparable enterprise system.

> "The powerful and low-cost Raspberry Pi 3 was ideal for large-scale rollout"

THE RESULTS

When Covid-19 lockdowns began to impact severely on businesses across the world in 2020, Crux Labs customers who were already using Crux LX reported that they were able to switch to remote working comfortably, with staff continuing to work from home with no discernible negative effect on the business. Crux LX reports over 10,000 users across 11 countries. The MD of Crux VX client Far East Flora, Singapore's leading online florist, noted that choosing the Raspberry Pi-powered system was "one of the wisest technology decisions [we] made."

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**

RASPBERRY PI PHOTOGRAPHY

The Raspberry Pi Camera Module offers a world of creative and fun digital photo opportunities. By **Rosie Hattersley**

Less than two years after Raspberry Pi revolutionised the world of personal computing, the tiny but powerful Camera Module introduced a totally different approach to digital photography.

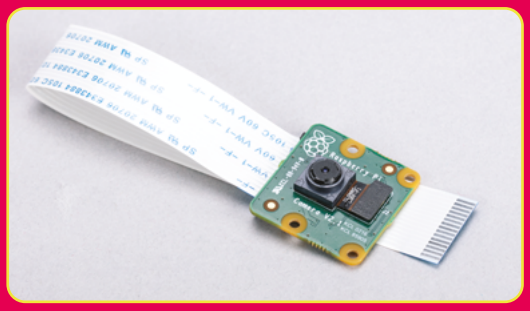
The three Raspberry Pi Camera options open up dozens of fantastic projects to try that involve little more than adding a tiny-but-mighty lens and a smattering of code. Try point-and-shoot photography via a designed-it-yourself camera, capture superb sports day action shots and all-day time-lapses; explore artificial intelligence using image recognition; set up smart surveillance in case of unwelcome guests; use wildlife cams to enjoy the birds and the bees; a delivery cam just in case the postman doesn't ring twice; encase a camera in a waterproof housing to explore the world aquatic; or bolt to a telescope to peer at celestial realms unknown.

RASPBERRY PI CAMERA OPTIONS

There are three main camera options available for Raspberry Pi owners

CAMERA MODULE 2

The original model, launched in 2013, offered 5MP (megapixel) image capture whereas its successor, the cleverly named Camera Module 2 (**magpi.cc/cameramodule**), features an 8MP Sony sensor. It can capture both stills and videos.

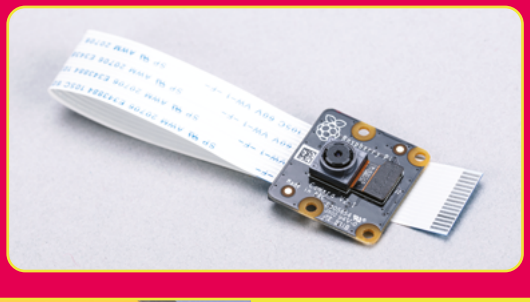


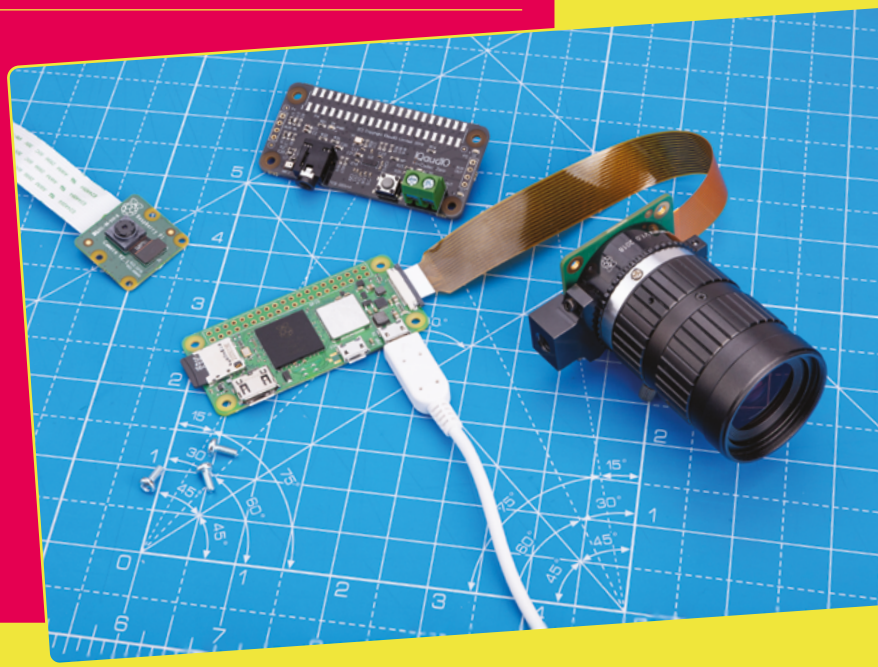
HIGH QUALITY CAMERA

There's also a 12.3-megapixel High Quality Camera, which is designed to work with interchangeable lenses, notably Micro Four Thirds lenses, using either a C or CS mount. The Pi Hut has a good selection of available lenses (**magpi.cc/cameralenses**).

RASPBERRY PI NOIR

The Raspberry Pi NoIR Camera Module offers low-light and night-time photography thanks to its lack of infrared filter (**magpi.cc/noir**), hence its name, but can also be used for daytime shots if you're after some unusual effects.





CONNECTING AND USING THE CAMERA

Find out how to connect your High Quality Camera or Camera Module, enable it, and take your first shots

We are going to show you how to connect the High Quality Camera or Camera Module to your Raspberry Pi using the supplied ribbon cable.

Attaching a Camera Module to Raspberry Pi is easy, using the CSI (camera serial interface) found on most Raspberry Pi boards.

You'll need a compatible ribbon cable. The larger Model A and Model B Raspberry Pi boards use a standard cable (**magpi.cc/cameracable**), while the smaller Zero boards use a Zero Camera Cable (**magpi.cc/zerocameracable**).

With a camera attached to Raspberry Pi, you can then start to capture images and video using terminal commands and Python code.

We will then enable it in Raspberry Pi OS, before entering some commands in a Terminal window to start shooting photos and video. Let's get started...

You'll Need

- Raspberry Pi Zero, Model A, or Model B **magpi.cc/products**
- Camera Module **magpi.cc/camera**
- Camera cable **magpi.cc/cameracable**

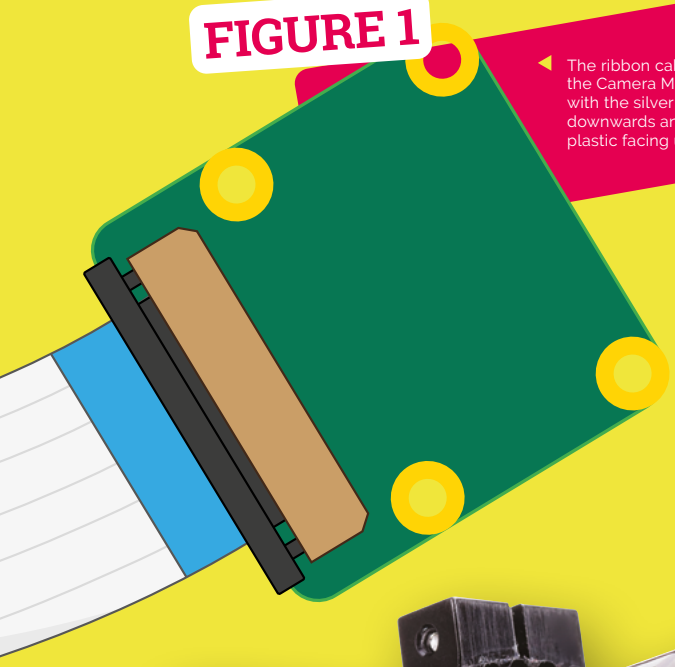
RASPBERRY PI CAMERA GUIDE

Inspired to try more Raspberry Pi photography? Download for free *The Official Raspberry Pi Camera Guide* and try stop-motion videography, learn how to capture high-speed action shots, do Raspberry Pi flash photography, and more. We'll also teach you to build a photo booth (Minecraft optional), how to shoot underwater footage, and how to take photos remotely.

magpi.cc/cameraguide

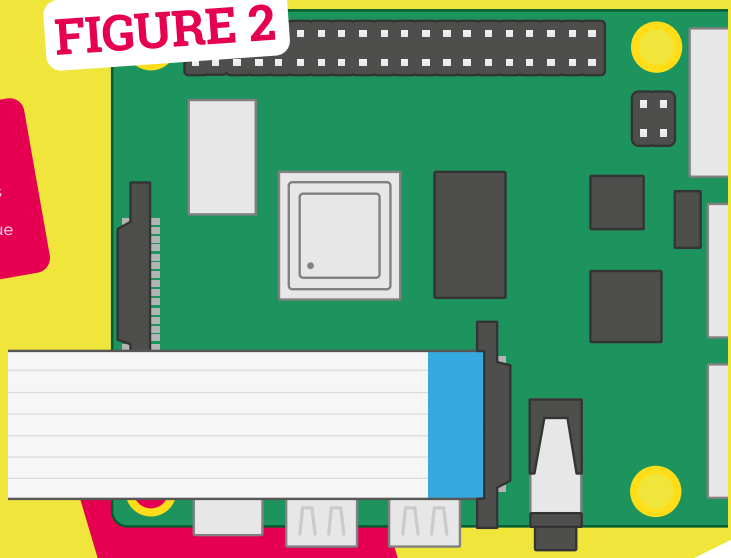
GET THE CABLES
THE RIGHT WAY AROUND

FIGURE 1



The ribbon cable on the Camera Module fits with the silver edges downwards and the blue plastic facing upwards

FIGURE 2



Hold Raspberry Pi with the HDMI ports facing down and slide the ribbon cable in so the silver edges are to your left and the blue plastic to your right

The High Quality Camera with a stock 6 mm lens enables you to capture seriously impressive shots

Top Tip

Camera documentation

Take a look at the High Quality Camera documentation to set up your camera. **magpi.cc/hqcamgettingstarted**

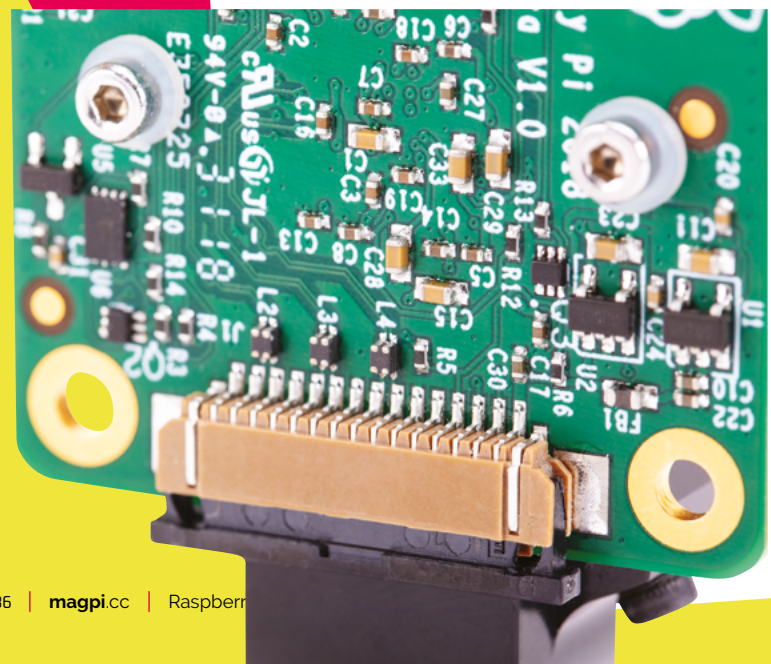
If you have a Camera Module V2, or Camera NoIR, you are ready to connect it directly to Raspberry Pi.

The High Quality Camera will need a lens mounting on top of the sensor on the camera board before you can take photographs. A low-cost 6 mm lens is available for the High Quality Camera (**magpi.cc/6mmlens**). This lens is suitable for basic photography. It can also be used for macro photography because it can focus objects at very short distances. It does not require the adaptor ring and connects directly to the HQ Camera.

Start by screwing your lens onto the High Quality Camera and adjusting the focus rings. Follow the High Quality Camera documentation to set up the lens (**magpi.cc/hqcamgettingstarted**).

01 Connect ribbon cable to camera

On the HQ Camera or Camera Module board, you'll find a flat plastic connector. Carefully pull the sticking-out edges until the connector pulls part-way out. Slide the ribbon cable, with the silver edges downwards and the blue plastic facing upwards, under the flap you just pulled out, then push the flap gently back into place with a click (**Figure 1**, previous page); it doesn't matter which end of the cable you use. If the cable is installed properly, it will be straight and won't come out if you give it a gentle tug; if not, pull the flap out and try again.



Gently pull the plastic tabs out on the CSI connector to open it. Slide the camera cable in and gently push the plastic tabs back to lock it into place

02 Connect cable to Raspberry Pi

Find the CSI port on Raspberry Pi and pull the plastic flap gently upwards. With Raspberry Pi positioned so the HDMI port is facing you, slide the ribbon cable in so the silver edges are to your left and the blue plastic to your right (**Figure 2** – previous page), then gently push the flap back into place. If the cable is installed properly, it'll be straight and won't come out if you give it a gentle tug; if not, pull the flap out and try again.

If using a Raspberry Pi Zero, its camera port is found on the edge of the board.

03 Enable the camera

Connect the power supply back to Raspberry Pi and load Raspberry Pi OS. Before you can use the raspistill tool in the next step, you will need to enable Legacy Mode. Open a Terminal window and open raspi-config:

```
sudo raspi-config
```

Use the arrow and **ENTER** keys to choose three Interface Options and I1 Legacy Camera. Select Yes to the 'Would you like to enable legacy camera support?' Raspi-config will let you know it has been enabled. Choose Finish and Yes to the 'Would you like to reboot now?' message.

04 Test the camera

To confirm that your camera is correctly installed, you can use the raspistill tool. This, along with raspivid for videos, is designed to capture images from the camera using Raspberry Pi's command-line interface (CLI). To take a test shot, type the following into the Terminal:

```
raspistill -o test.jpg
```

As soon as you hit **ENTER**, you'll see a large picture of what the camera sees appear on-screen. This is called the live preview and, unless you tell raspistill otherwise, it will last for five seconds. After those five seconds are up, the camera will capture a single still picture and save it in your home folder under the name **test.jpg**. If you want to capture another, type the same command again – but make sure to change the output file name, after the `-o`, or you'll save over the top of your first picture.

05 More advanced commands

The raspistill command has a list of options so long that it borders on the intimidating. Have no fear, though: you won't need to learn them all, but there are a few that might be useful to you, such as:

```
raspistill -t 15000 -o newpic.jpg
```

The `-t` option changes the delay before the picture is taken, from the default five seconds to whatever time you give it in milliseconds – in this case, you have a full 15 seconds to get your shot arranged perfectly after you press **ENTER**.

06 Rotate the image

If the live preview was upside-down, you'll need to tell raspistill that the camera is rotated. The Camera Module is designed to have the ribbon cable coming out of the bottom edge; if it's coming out of the sides, or the top, as with some third-party camera mount accessories, you can rotate the image by 90, 180, or 270 degrees using the `-rot` switch. For a camera mounted with the cable coming out of the top, use the following command:

```
raspistill -rot 180 -o test.jpg
```

You can discover more commands that work with raspistill in the Raspberry Pi Documentation (**magpi.cc/raspistilldoc**). This includes an example of a short script that takes a photograph and adds the date and time.

Top Tip

Raspberry Pi 400?

Sadly, you cannot connect Raspberry Pi Cameras to Raspberry Pi 400 computers, although you can use a stock USB camera instead (**magpi.cc/usbcamera**).

RASPISTILL VS LIBCAMERA

It is a somewhat confusing time for Raspberry Pi camera coding as the raspistill functionality we are using here is being slowly replaced by the newer libcamera option.

Sadly, you can only use one or the other, and most Raspberry Pi projects are using raspistill while makers get up and running with libcamera.

To use raspistill, you must enable Raspberry Pi Legacy mode (as shown in Step 3) or install a special custom version of Raspberry Pi OS (Legacy).

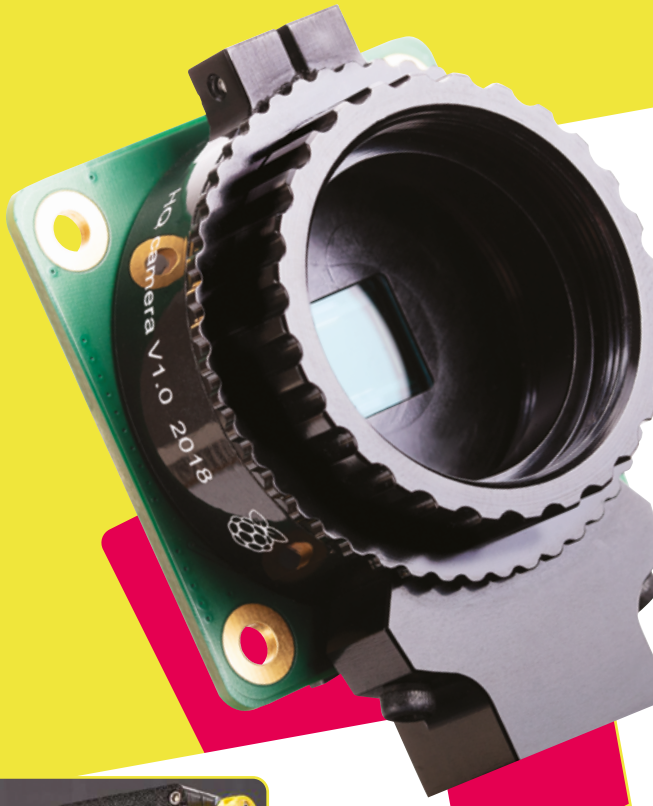
Unfortunately, this prevents you from using the new libcamera applications. Worry not, though: it's easy to turn off Legacy Support by running raspi-config again and choosing No to the 'Would you like to enable legacy camera support?' (Don't forget to reboot.)

Now try out these commands in Terminal:

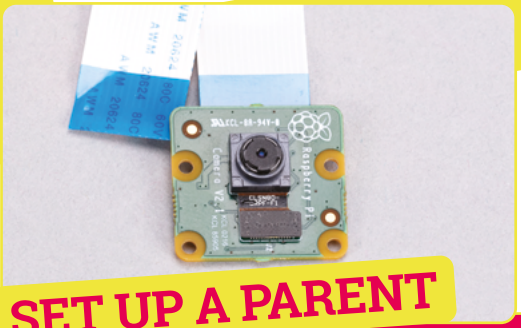
- **libcamera-hello** A simple 'hello world' application which starts a camera preview stream and displays it on the screen.
- **libcamera-jpeg** A simple application to run a preview window and then capture high-resolution still images.
- **libcamera-still** A more complex still image capture application which 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.

Take a look a the libcamera documentation for more in-depth information about the new functionality: **magpi.cc/libcamera**

INCREDIBLE CAMERA PROJECTS



All the incredible things you can make with Raspberry Pi and a Camera Module



SET UP A PARENT DETECTOR

This one is great for kids, and is a superb introduction to using the Camera Module. Suspicious that someone's been in your room and having a good old nose? So unfair, right? Teens and house-sharers keen to put a hunch to the test can set up a spy cam to alert them whenever someone dares enter their domain. The easy-to-follow tutorial explains how to set up the Raspberry Pi Camera Module and PIR motion sensor to trigger video recording should an unexpected visitor be detected, plus how to adjust the sensitivity so you don't get jumpy every time the door rattles. There's even a stealth mode so the telltale red recording light is shielded should your housemates have their own suspicions. Helpfully, you get an alert when an intruder is detected and can then view the footage to check it wasn't just a cheeky puss paying you a visit before you launch in and raise merry hell with the oldies about an invasion of privacy.

magpi.cc/parentdetector

TOY CAMERA

The Raspberry Pi High Quality Camera can be used with the CS mount and lenses, but it's not the only option. As maker Volzo explains, S-mount lenses (ideal for machine vision and infra-red photography) can also be used, which is what he chose for this Raspberry Pi Zero build designed to add zany features and introduce unpredictable effects to digital photography – something more generally known as Lomography. Depending on your level of make-ability, you can either use an existing old camera body or another vessel to house your camera, or fashion one yourself, for which Volzo provides demos and links to ample alternative design ideas. His 3D-printed and CNC-cut version is held together with magnets as well as screws, while the viewfinder is from a pair of night-vision goggles. Exhorting others to rediscover the quirkiness of analogue photography, this project shows how much fun it can be to experiment when the basic Raspberry Pi camera and other parts are cheap enough to allow you the freedom to do so.

magpi.cc/digitaltoycamera

INSTANT PHOTO PRINTER

Adafruit's Philip Burgess offers some great project ideas including this super means of printing out the results of your Raspberry Pi photography on demand, making use of the sort of $45 thermal printer more commonly found at a grocery till. The thermal photo printer works with any standard size Raspberry Pi and the retro photo results are just as good whether taken on a standard Camera Module 2 or the High Quality Camera Module. You'll need an SD card for the Python code and Raspberry Pi OS, a large push-button, and a means of connecting this and the printer to Raspberry Pi, plus four AA NiMH batteries. A case for the setup could be as simple as a cardboard box or something fancier you design yourself or 3D-print from Thingiverse.

magpi.cc/instantcamera

FACE RECOGNITION SMART LOCK

A great way to control access to a building is using the Raspberry Pi High Quality Camera and a smartphone to ascertain who's calling. When a familiar face pops up on-screen, you can then grant that person access. This works really well – as long as you're around to check your phone when a visitor notification pops up. For scenarios in which you aren't around, you could train Raspberry Pi 3 or 4 to recognise friends' faces and allow them entry. In Seeed Studio's walkthrough, you take photos of people you want to let in, as per this amusing face-recognition setup along with Grove's Relay To LTE HAT, a wireless antenna, and of course Raspberry Pi with the Camera Module attached. A text message is sent to the owner stating who was let in whenever someone is recognised and their door unlocked.

magpi.cc/facerecogsmartlock

SMART BIRD BOX

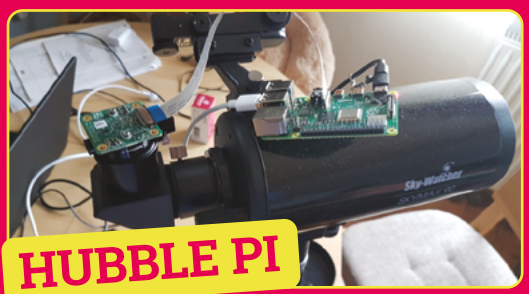
Blue skies, flowers blooming, and birds tweeting are good for the soul. Getting a close-up view of the magic of nature without disrupting said critters from going about their daily business is thoroughly worthwhile. This version will stream bird box footage for remote viewing.

Make sure the nest box you're going to use is not currently inhabited (or create/acquire a new one) before you place your High Quality Camera – or Raspberry Pi NoIR Camera Module if you're keen to observe nesting birds at night too – inside.

You won't be able to adjust things once the bird box is in use (as disturbing birds is likely to cause them to desert the nest), so make sure you test everything to confirm it works before installing it in place.

magpi.cc/infraredbirdbox

> "Observers of the night sky with even a basic telescope can use it in tandem with Raspberry Pi and the High Quality Camera"



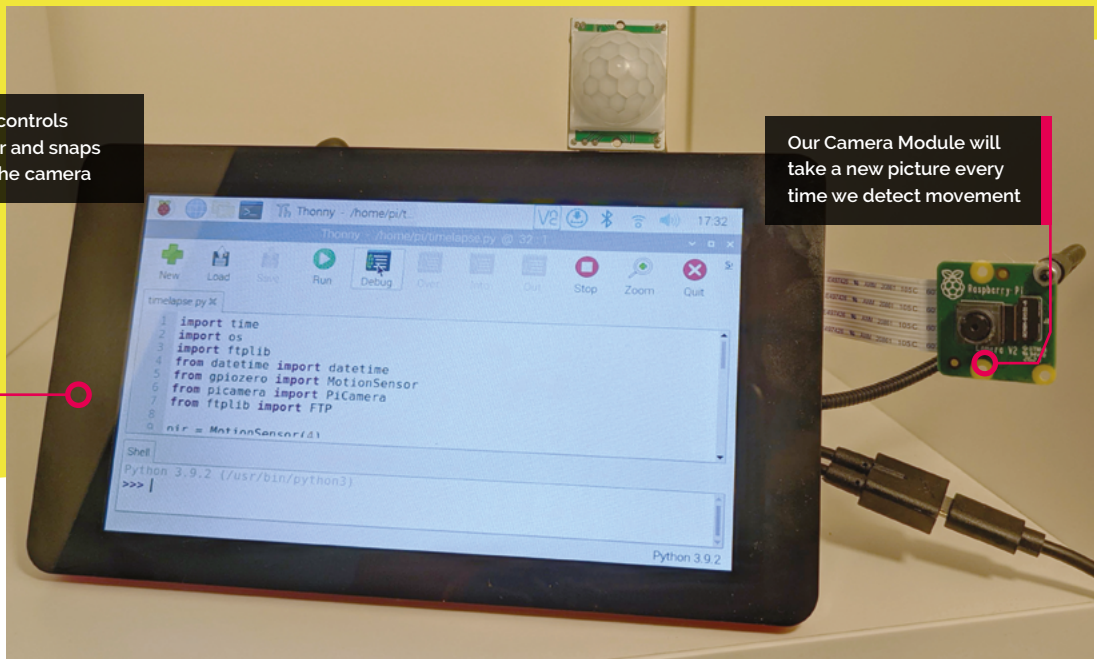
HUBBLE PI

Observers of the night sky with even a basic telescope can use it in tandem with Raspberry Pi and the High Quality Camera plus a C-mount-to-telescope adapter to capture incredible astronomical sights. Hubble Pi pairs Raspberry Pi 4 with free astronomy software KStars which displays a live map of the night sky on the telescope's display. Maker Santiago exploited the large lens of the HQ Camera and wrote Python code he calls AstroCam to control its shutter speed, the ISO, and exposure time. Bonuses include being able to automatically take multiple RAW photos and using either remote desktop or a touchscreen to trigger a shot.

magpi.cc/hubblepi

TURN YOUR PHOTOS INTO TIME-LAPSE VIDEOS

Can we get to the good bit? Shooting time-lapse videos makes drawn-out processes more engaging



MAKER

Nik Rawlinson

Coffee-drinking, typewriter lover with a penchant for pencils and paint. Frequently found staring at the sea from the back of a camper-van.

nikrawlinson.com

Good things are usually worth the wait – unless they come at the end of a long and largely repetitive video. In this workshop, we'll show you how pairing Raspberry Pi with a Camera Module 2 and a PIR motion detector lets you shoot stills only when there's something worth watching. Discover how to convert images into a time-lapse video packed with action. We've been using this technique to show how time-consuming artistic processes result in a finished image, but you could also use it to see what your pet does when you've left it home alone.

01 Enable legacy camera support

The functions we need to use in our Python code will run natively on older versions of Raspberry Pi OS, but require legacy support if you're running Raspberry Pi OS Bullseye or later. To enable support in these versions, open a new Terminal window by pressing **CTRL+ALT+T** (or connect to your Raspberry Pi remotely using SSH) and type:

```
sudo raspi-config
```

Select option **3 – Interface Options** and **1 – Legacy camera**, then confirm that you want to enable legacy camera support. Quit raspi-config and allow your Raspberry Pi to reboot and implement the changes.

02 Correct your screen

We're setting up our time-lapse camera using a Raspberry Pi 3 B+ connected to a Raspberry Pi Touch Display (**magpi.cc/touch**). Enabling legacy camera support caused our display to rotate through 180 degrees (only locally; the orientation was still correct when connecting via VNC), and removed the Displays option from Raspberry Pi OS's graphical interface. If you experience the same, open a Terminal window and type:

```
sudo nano /boot/config.txt
```

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

```
lcd_rotate=2
```

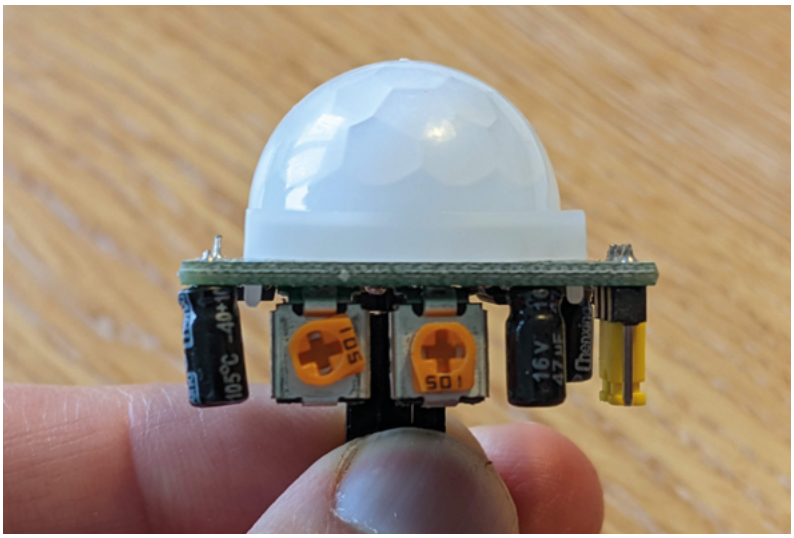
Press **CTRL+X** to quit and press **Y** when asked if you want to save, then reboot by typing:

```
sudo reboot
```

> "Enabling legacy camera support caused our display to rotate through 180 degrees"

03 Connect your camera

We're using the regular Camera Module 2 here because we only want to shoot time-lapses in regular lighting. However, this tutorial works equally well with the Camera Module 2 NoIR, which can shoot low-light and night-time stills and video (you will need to provide an infra-red light source for night-time use). Connect the camera to Raspberry Pi's CSI (Camera Serial Interface) using the bundled ribbon cable. Don't confuse it with the display connector: on Raspberry Pi Model B computers, the camera connector is the one between the HDMI and headphone ports.



▲ The screw on the right controls the sensitivity of the PIR motion detector

04 Connect your PIR sensor

The passive infra-red (PIR) sensor lets you take pictures only when there's some kind of movement. Place your Raspberry Pi on a flat surface so the USB ports are to the right and the GPIO pins are running along the top edge, and hold your PIR sensor so that its pins are also running along the top edge and pointing towards you. Use jumper wires to connect the PIR's left pin and right pins to GPIO pins 6 and 2 respectively, and the PIR's middle pin to GPIO pin 7 (**magpi.cc/pinout**).

05 Position and test your camera

At this point, you want to get your Raspberry Pi and camera in position. We've mounted ours on a goose-neck stand, which plugs into an unused USB port, so we can point it directly at the scene we want to capture. Take a test photo by returning to the Terminal prompt and typing:

```
raspistill -o test.jpg
```

Open the resulting **test.jpg**, which will be saved in your home folder (**/home/pi/test.jpg**) to check the orientation, so you can apply correction later, if necessary.

06 Set up your FTP server

To avoid filling up your Raspberry Pi's microSD card, our code uploads each captured image to a server and then deletes the local copy. It doesn't matter whether you're running your own server on-site or using a remote commercial offering: what does matter is that you have FTP access. Log on to your host's control panel and create a folder for your

Top Tip

Zeroes ain't heroes

If any of your saved images are 0kB in size, delete them: FFmpeg may quit early if it encounters them.

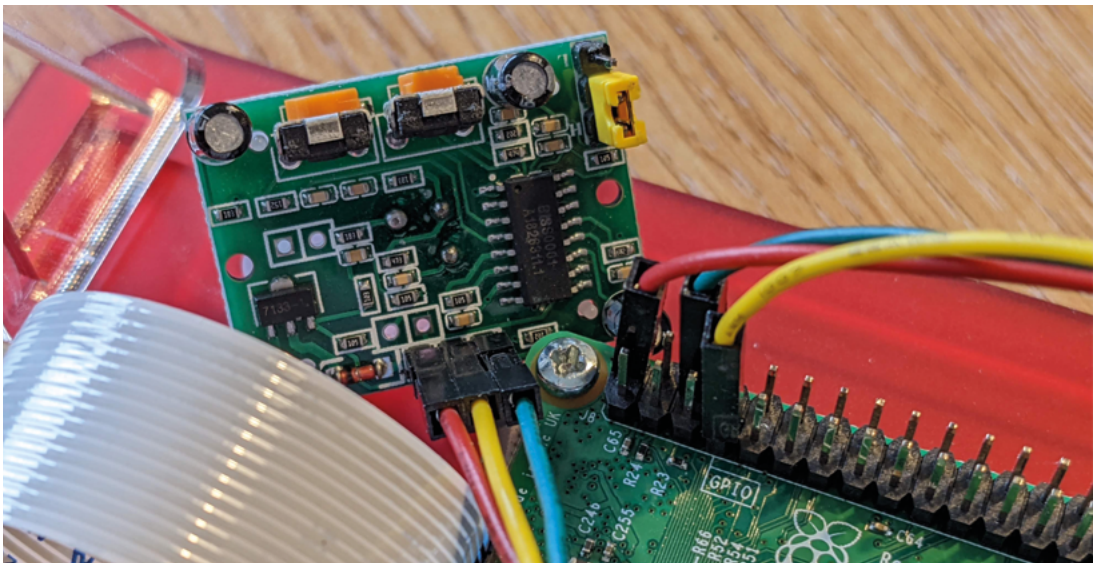
You'll Need

- Camera Module 2 **magpi.cc/cameramodule2**
- PIR motion sensor **magpi.cc/pir**
- Three female-female jumper wires **magpi.cc/jumperjerky**

Top Tip

Running too slow?

FFmpeg defaults to a framerate of 25 fps. To swap it for 60 fps, add '-framerate 60' before '-pattern_type' in Step 12.



▲ You'll need three jumper leads to connect the PIR motion detector to the GPIO pins on your Raspberry Pi

uploaded images, as well as a user account that logs straight into that folder. The process for doing this will vary between providers. Make a note of your username, password, and server address.

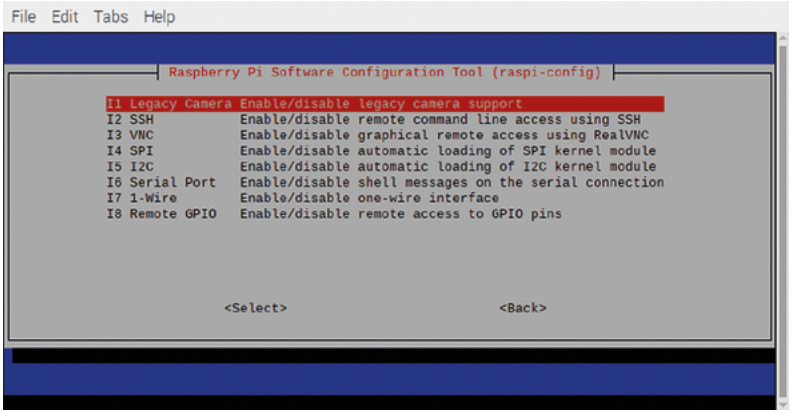
07 Supply your login details

Download the code from GitHub (**magpi.cc/timelapsepy**) and save it in your home folder (**/home/pi/**) as **timelapse.py**. Open it in Thonny Python IDE (in the Programming menu of a standard install of Raspberry Pi OS) or an editor of your choice. Edit lines 21 and 22, inserting your server address where indicated on line 21, and your username and password where indicated on line 22.

08 Correct the camera... if necessary

If the test shot you took in Step 5 showed that the camera was rotated, uncomment line 17 by removing the # at the start, and adjust the number at the end of the line to correct the framing. Our camera is positioned with Raspberry Pi's logo printed on the baseboard in the top left corner, so we need to rotate the image by 270 degrees to correct its orientation. However, you may instead need to make an adjustment of 90 or 180 degrees, depending on your setup.

▼ If you're running Raspberry Pi OS Bullseye or later, use raspi-config to enable legacy camera support



09 Automate your capture

Save your edited code, then return to the Terminal and type:

```
crontab -e
```

If you've never edited crontab before, you'll be asked which tool you want to use. Select 1 for Nano. When Nano opens, key down to the bottom of the file, create a new line and type:

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

This tells Raspberry Pi OS to run your code as soon as it starts, with the & at the end telling it to run the process in the background. Press **CTRL+X** to quit Nano, confirm that you want to save your edit, and reboot your Raspberry Pi.

10 Testing, testing, 123

Your Raspberry Pi should begin capturing images and uploading them to your server when the sensor detects motion. If it doesn't, either reposition the sensor (not the camera) or adjust its sensitivity. To set the sensitivity to maximum, hold

the sensor so the dome is pointing up and the pins are away from you, then rotate the orange screw on the right (which controls sensitivity) as far as you can to the right, to maximise the range across which it detects motion. If you get false positives, dial back a little. Finding the optimal position can require some trial and error.

11 Download your stills

Download the stills from the server to a folder called **timelapse** inside your home folder. Open Terminal and type:

```
cd timelapse
```

Press **RETURN**, then type:

```
lv -v | cat -n | while read n f; do mv -n
"$f" "$n.jpg"; done
```

Press **RETURN**. This renames your files sequentially. You now need to add padding so the file names are all the same length.

Type the following:

```
sudo apt install -y rename
```

Press **RETURN**, then type the following and press **RETURN**:

```
rename 's/\d+/sprintf("%05d",$&)/e' *.jpg
```

12 Create your time-lapse

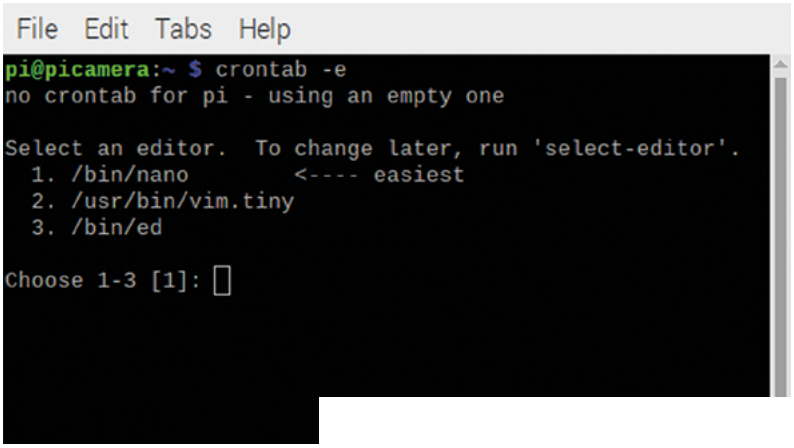
Install FFmpeg by typing:

```
sudo apt install -y ffmpeg
```

Press **RETURN**. When installation completes, compile your images into a video by typing:

```
ffmpeg -pattern_type glob -i "*.jpg"
-s:v 640:480 -c:v libx264 -pix_fmt yuv420p
timelapse.mp4
```

If you want to change the output resolution, adjust `640:480` to your preferred dimensions. The time taken to complete the process depends on both the resolution and the number of images you've saved.



timelapse.py

DOWNLOAD THE FULL CODE: magpi.cc/timelapsepy

> Language: **Python**

Credit: magpi.cc/repaddzeros and magpi.cc/renamingfilesseq

```
001. import time
002. import os
003. import ftplib
004. from datetime import datetime
005. from gpiozero import MotionSensor
006. from picamera import PiCamera
007. from ftplib import FTP
008.
009. pir = MotionSensor(4)
010.
011. def thegrab():
012.     thetime = datetime.now()
013.     detectiontime = thetime.strftime("%y-%m-%d-%H-%M-%S")
014.     extension = ".jpg"
015.     filename = detectiontime + extension
016.     camera = PiCamera()
017.     #camera.rotation = 270
018.     time.sleep(2)
019.     camera.capture(filename)
020.     camera.close()
021.     ftp = FTP('[server address]')
022.     ftp.login('[username]','[password]')
023.     ftp.cwd('')
024.     ftp.storbinary("STOR " + filename, open(filename, 'rb'))
025.     ftp.quit()
026.     os.remove(filename)
027.
028. while True:
029.     pir.wait_for_motion()
030.     thegrab()
031.     time.sleep(30)
```

Part 01

ArtEvolver: batch-convert images with ImageMagick

Prepare to build an ever-evolving artwork using Python code that merges images endlessly. Start by discovering how to bulk process images with ImageMagick

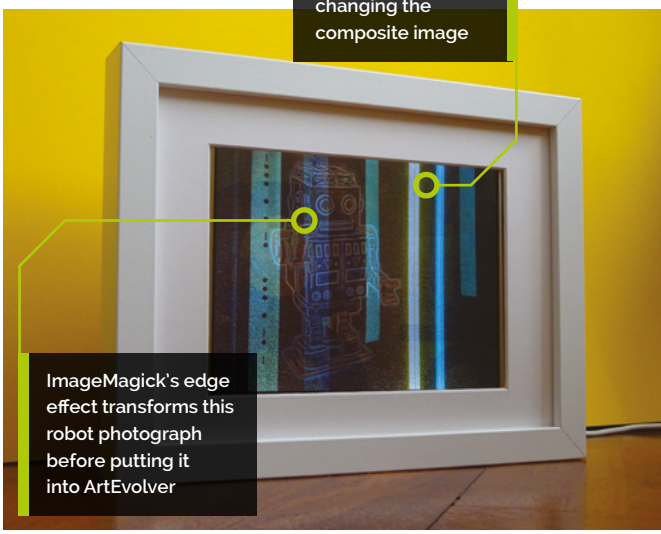
WRITER

Sean McManus

Author of *Mission Python*, *Scratch Programming in Easy Steps*, and *Raspberry Pi For Dummies* (with Mike Cook). Get free chapters at Sean's website.

sean.co.uk

Brian Eno's *77 Million Paintings* is a video installation that merges slides together to create endless variations of abstract art. I created my own version, called ArtEvolver, which runs on a Raspberry Pi and uses a Pimoroni 8-inch LCD screen. To work well, it needs a large library of images. This issue, you'll see how to curate them and prepare them using ImageMagick's (**imagemagick.org**) powerful batch processing. You'll learn how a single command can resize, crop, or transform hundreds of pictures. There's a short Bash script to automate image rotation, and we'll take a tour through some of ImageMagick's special effects.



ArtEvolver blends the robot with other pictures, constantly changing the composite image

ImageMagick's edge effect transforms this robot photograph before putting it into ArtEvolver

You'll Need

- Raspberry Pi
- Raspberry Pi OS
- Some images
- ImageMagick **imagemagick.org**

01 Collect your images

In Add/Remove Software (Raspberry Pi menu > Preferences), search for 'ImageMagick' and select the 'image manipulation programs – binaries' option. Click Apply to install the software. Alternatively, open a terminal window and enter:

```
sudo apt update
sudo apt install imagemagick -y
```

In this tutorial, you will make ArtEvolver unique by curating the images that you feed into it. Scour your personal photo archive and pick images that resonate. You can bulk these up with free images from sites such as **unsplash.com**, **pixabay.com**, and **pexels.com**. Your images will be layered in unpredictable ways, so search for colours, textures, and shapes that could be part of an abstract artwork. Textures like paper, stone, and paint make the art feel more organic; illustrations often work well. We've collected about 1000 images for this version, but you only really need a hundred or so. Put all your images (and nothing else) in a folder, and keep a separate, safe copy as a backup.

You can use ImageMagick from the desktop, but the command line enables powerful batch processing capabilities

02 Experiment with the desktop app

ImageMagick installs into a desktop menu's **Graphics** folder. Start ImageMagick and click the splash screen. The splash screen is actually an image you can edit, but you'll get better results by loading a photo using the File menu. Try the various options in the Effects and F/X menus. These include emboss, sharpen, blur, sepia tone, and oil paint. The Enhance menu has options for changing the colour and tone of your image. You'll find the option to resize your image in the View menu. The Image Edit menu enables you to draw. You choose an element such as a filled circle, a fill colour, and 'stipple.' The stipple is a pattern for the fill, such as brickwork, waves, or fish scales. Drag on the canvas to draw your shape, but be warned that it can be slow on high-resolution images.

03 First steps with the terminal

Go into your images folder on the desktop and press **F4** to open a terminal window in that directory. There are two main ImageMagick commands: `convert` and `mogrify`. Convert is good for changing individual images or experimenting with effects. For example, you can mirror an image vertically with the `-flip` operation like this:

```
convert image_file.jpg -flip new_image_file.jpg
```

Use `-flop` to mirror it horizontally. With mogrify, you can process many images at the same time, which is a huge plus over the desktop app. You use wild cards, where * represents every file, and *.jpg would be every file ending with .jpg. Here's an example:

```
mogrify -flip *
```

Beware: mogrify overwrites your image files.

04 Make all your images landscape format

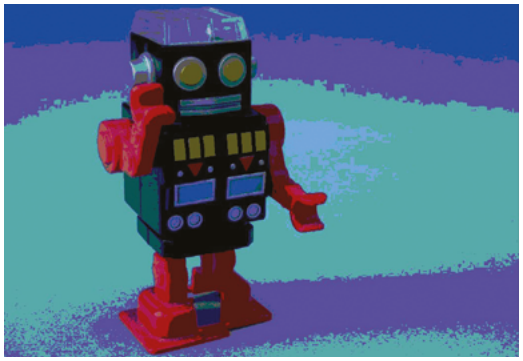
We are assuming you want to display your artwork on a landscape format screen (wider than it is tall). It's OK to have some portrait shape images in ArtEvolver, but it works best if most images fill the screen. With abstract images, it doesn't matter if you rotate them. The **landscapify.sh** listing shows a Bash script that rotates all the portrait images in a folder by 90 degrees. You can download it at **magpi.cc/artevolver** or create it using Text editor. Save it in the same folder as your images, with the name **landscapify.sh**. Open the folder in the terminal and enter:

```
chmod +x landscapify.sh
```

This will make the script executable. Then run it with `./landscapify.sh`. The script uses ImageMagick's `identify` command to get the dimensions of each image, and its `convert` command to rotate any pictures where the height is greater than (`-gt`) the width. Unchanged versions of the rotated images are saved in the new **original_images** subfolder.

05 Combining images

For some images, you could instead make a landscape file by joining two or more portrait images, side by side. The `montage` command enables you to join images together. You list the images you want to combine and specify the layout with the `-tile` parameter. We are using `2x1` to place the images side by side. You could create a 4×2 grid of images as well, and give the command eight files to combine. To remove any gaps between the images, you use the `-geometry` parameter with two zero values. Try this (replacing the 'image_file.jpg' names with your image files):



The posterize effect (here used with a value of 4) gives your image the style of a vintage PC palette

Top Tip

View progress in the desktop

Use ImageMagick's `display image_file.jpg` command to see an image. It's easier to use the Image Viewer in the desktop to quickly review image batches.

▲ The implode effect, here used with a value of 0.5, distorts the image

```
montage image_file1.jpg image_file2.jpg
-tile 2x1 -geometry +0+0 new_file.jpg
```

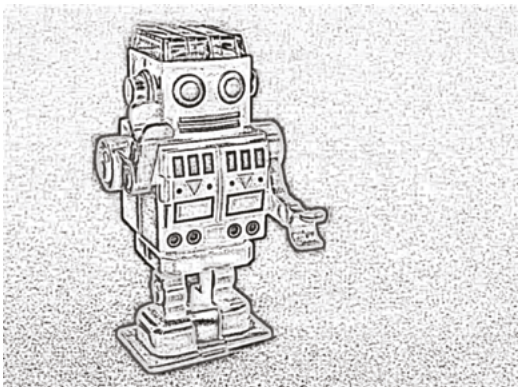
This will create a **new_file.jpg** file from the two images you supply.

06 Resize your images

The Pimoroni display we are using has a resolution of 1024×768 pixels, but camera images are typically much larger. To resize all the images in one go, use ImageMagick's `mogrify` command, like this:

```
mogrify -resize 1024x768^ *
```

The `^` symbol after the resolution sets the images to fill the screen, with some spilling over. If you leave the `^` off, the resized images will fit the screen. In that case, you see the entire image, but the empty spaces at the top and bottom won't work well in our final project. Beware: this command might take a while, and it will overwrite your original images.



▶ Charcoal images like this can work well when blended with coloured textures by ArtEvolver. The line thickness here is 5

Top Tip

Experiment with small batches

Image transformations can be slow, so experiment with test batches before running a command on a big batch.

07 Crop your images

For typical landscape camera images, any overmatter will be on the width of the image. Let's use mogrify to crop it off. The `-gravity` parameter specifies which part of the image you want to keep, using compass points. To keep the left and trim the right, use `west`, for example. Set the gravity to `center` to trim both sides equally. The best-looking crop depends on the image. I manually sorted my images into three folders for cropping left, right, and centre, and then ran a version of this command in each folder.

```
mogrify -gravity center -crop 1024x768+0+0 *
```

If you have images with unusual dimensions, you may need to crop `north` or `south` instead. (You can also use a north crop to extract the top of a portrait-shaped image before you resize it.)

08 Convert to greyscale

Some photos will blend better with other images if you convert them to greyscale. I use `convert` for changes like this so I can keep and compare the results of different effects. You can convert an image using:

```
convert -colorspace Gray image_file.jpg
new_image_file.jpg
```

You can convert to sepia (a browned-out photo style) using `-sepia-tone`, where a higher number makes the image darker:

```
convert -sepia-tone 75% image_file.jpg new_
image_file.jpg
```

09 Adjust the colours

Swapping colours often makes striking images. Use `-negate` to switch black and white, and swap complementary colours (e.g. blue and yellow). Try this:

```
convert -negate image_file.jpg new_image_
file.jpg
```

You can also try negating only the red, green, or blue channel:

```
convert -channel blue -negate image_file.jpg
new_image_file.jpg
```

Posterize reduces the number of colours in the image. Use a value of 2 for an 8-colour palette, 3 for 27 colours, and 4 for 64:

```
convert -posterize 2 image_file.jpg new_
image_file.jpg
```

10 Add visual effects

There are a number of special effects you can apply to images, including `-emboss`, `-charcoal`, `-edge`, `-paint`, and `-spread`. Experiment with them to transform photographs creatively. Spread gives you a frosted glass effect. The value for charcoal is a line thickness. Start here:

```
convert -emboss 2 image_file.jpg new_image_
file.jpg
```

11 Distort images

The `-wave` effect adds ripples to your image. You give it the height of the wave (amplitude) and the distance between two waves (wavelength), like this:

```
convert -wave 5x20 image_file.jpg new_image_
file.jpg
```

You can also use the `implode` effect to collapse an image, like this:

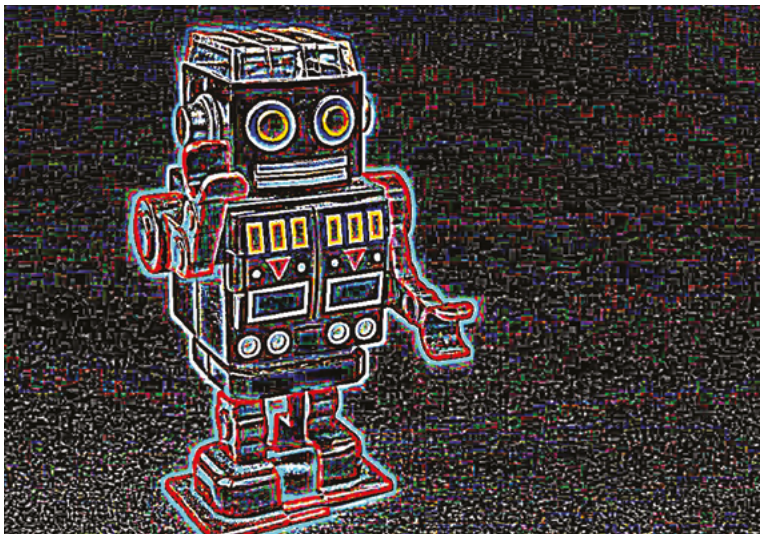
```
convert -implode 1 image_file.jpg new_image_
file.jpg
```

Use a negative number for the implode value to explode the image instead.

12 Combine effects

You can combine multiple transformations in one command. They're carried out in the order you list them. Here's an example that resizes, mirrors, and negates an image in a single command:

```
convert -resize 1024x768^ -flop -negate
image_file.jpg new_image_file.jpg
```



▲ The edge effect creates striking results like this

Using ImageMagick enables you to batch-convert images into a range of different styles. In the next ArtEvolver tutorial, we will collate these into a physical project that uses these transformations.

landscapify.sh

DOWNLOAD THE FULL CODE:
magpi.cc/artevolver

Language: **Bash**

```
001. #!/bin/bash
002. # Rotates portrait images (only) in the current folder
003. # From ArtEvolver Tutorial in The MagPi - by Sean McManus
004.
005. mkdir original_images
006.
007. # Remove any extensions in the list below that you're not
     using to avoid error messages
008. for image_file in *.jpg *.JPG *.png *.PNG;
009. do
010. # Make sure there is no space around the = below
011.     width=$(identify -format "%w" $image_file)
012.     height=$(identify -format "%h" $image_file)
013.     if test $height -gt $width
014.     then
015.         echo "$image_file is portrait shape [$width x
     $height]. Rotating..."
016.         new_name="rotated-${image_file}"
017.         convert -rotate 90 "$image_file" "$new_name"
018.         mv $image_file original_images
019.     else
020.         echo "$image_file is landscape already [$width x
     $height]."
021.     fi
022. done
```

Make a SpecDeck: Digital tape loader for the ZX Spectrum

Fed up with 'R Tape Loading Error' errors on your classic Speccy? Load ZX Spectrum software with ease using Raspberry Pi and Pirate Audio

WRITER

PJ Evans

PJ has owned a ZX Spectrum since 1982. He's pictured here with Richard Altwasser and Dr Steven Vickers, the hardware and software designers of the ZX Spectrum.

twitter.com/mrpjevans

You'll Need

- Pirate Audio Headphone Amp HAT **magpi.cc/pirateaudiohead**
- Headphone amplifier **magpi.cc/compactstereoamp**
- 2 × 2.5″ audio cables
- 3D-printed case (optional) **magpi.cc/pirateaudiocase**

Forty years ago, Sir Clive Sinclair brought home computing to the masses with his affordable ZX Spectrum. It was soon the centrepiece of living rooms across the UK as young gamers and coders battled with parents for control of the television. As wonderful a machine as it still is, the tape loading system for games and other software was fraught with errors. Wonky cassettes and unreliable playback hardware caused no end of frustration. Now we can use a Raspberry Pi Zero to emulate a tape onto an original ZX Spectrum with a reliability you could only have dreamt of in 1982.

01 Raspberry Pi, Assemble!

Our first challenge with this project is volume. ZX Spectrums like it loud and the audio output from Model A and B Raspberry Pi computers just isn't enough for these old computers to hear. So we start by adding a Pimoroni Pirate Audio HAT, specifically the Headphone Amp version, which will get us some of the way to an acceptable output level. It also means we can use the more portable Raspberry Pi Zero W for this project. Start by carefully adding the HAT to Raspberry Pi Zero (it'll need GPIO headers, so a WH variant is perfect). If you're not planning on using a case, we recommend adding some pillars for stability.

02 Prepare your SD card

It's software time now. We don't need a full-blown OS, so Raspberry Pi OS Lite is perfect for our needs. If you're building this project headless (with no keyboard or monitor attached), then we recommend the new advanced features in Raspberry Pi Imager (**magpi.cc/imager**). Run Imager and select Raspberry Pi OS (Other) then Raspberry Pi OS Lite as the image and your SD card as the storage. Now click the cog to set the hostname, enable SSH, create an account, and set wireless LAN credentials. Now you can write your image and, on boot, your Raspberry Pi will connect to the network and be ready to go.

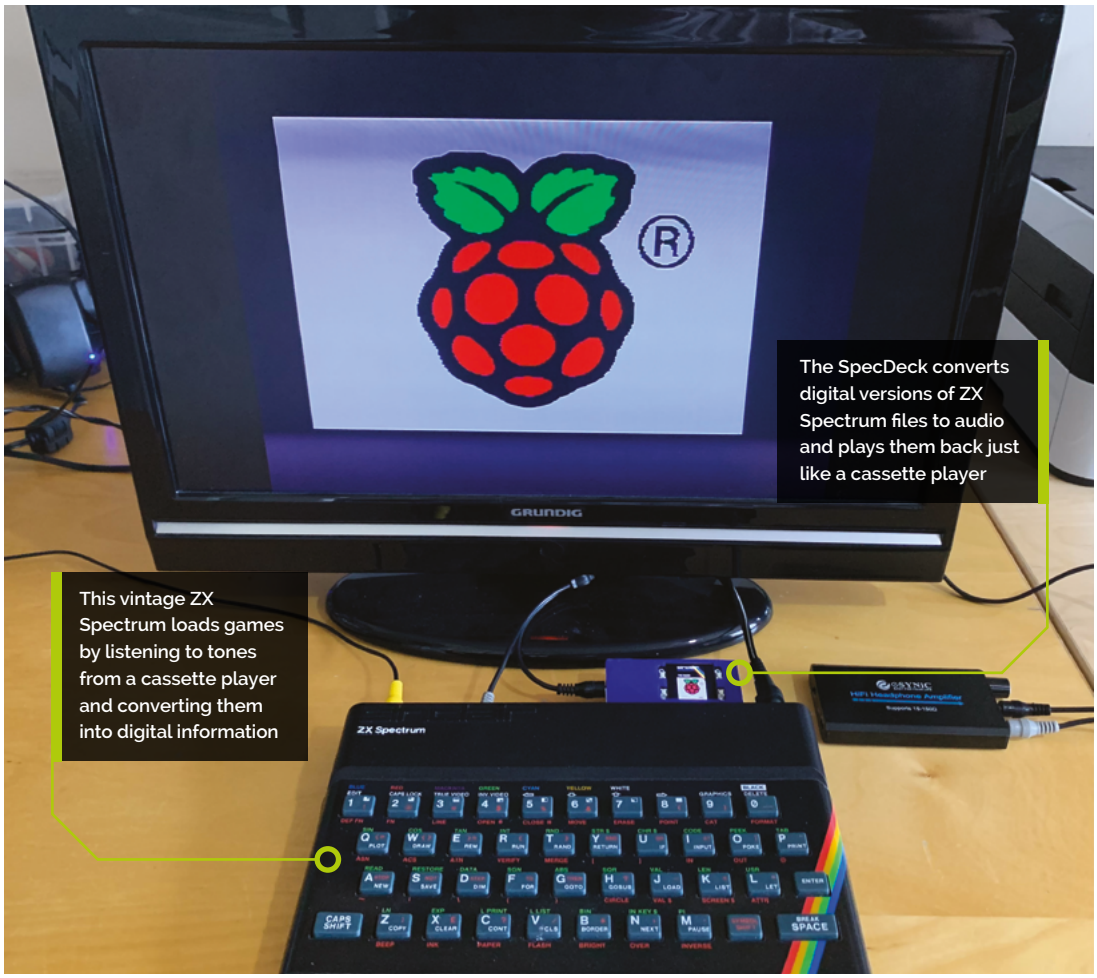
03 Enable and test Pirate Audio

By now you should be able to boot your Raspberry Pi and gain access via SSH. Don't worry if you've connected a monitor and keyboard: the instructions are exactly the same. Now we need to enable Pirate Audio. Log in and enter the following command:

```
sudo nano /boot/config.txt
```

Then, at the bottom of the file, add these lines:

```
dtoverlay=hifiberry-dac
gpio=25=op,dh
```




```
dtparam=audio=off
```

Save (**CTRL**+**X**) then reboot (`sudo reboot`). After reboot, connect the audio out to headphones (careful of the volume!) and run this command:

```
speaker-test -c 2
```

Do you hear static-like white noise? If so, you're good to go. Press **CTRL**+**C** to stop the racket.

04 Will it work? It depends on dependencies

The lines we added to **config.txt** in the previous step did two things: enable audio output to the DAC (digital-to-analogue converter) on the HAT and disable any other audio output. However, to get control over the screen, we have to install libraries that will help us write and display data. These are known as dependencies. Start by making sure you have updated everything already installed to the latest version:

```
sudo apt -y update && sudo apt -y upgrade
```

Once complete, you can install all the libraries you need:

```
sudo apt install git libsdl2-mixer-2.0-0
python3-rpi.gpio python3-spidev python3-pip
python3-pil python3-numpy libatlas-base-dev
libportaudio2
```

Finally, run `sudo raspi-config` and under Interfacing Options, enable SPI and I2C.

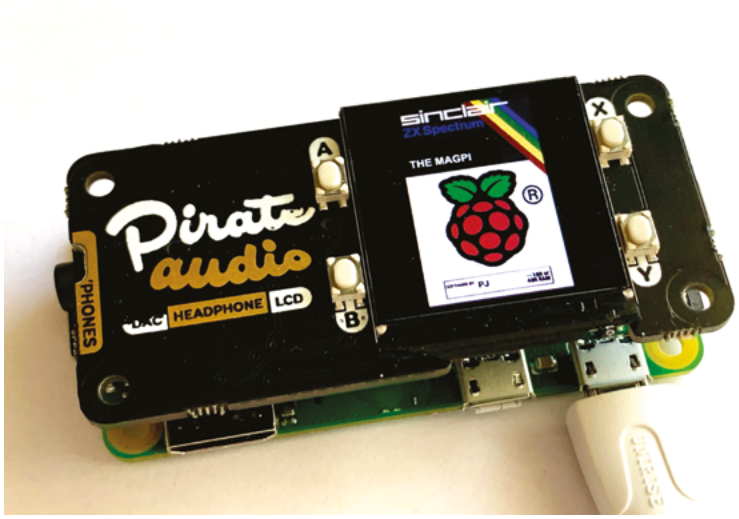
05 Turn it up to 11

What you do next depends on what kind of hardware you are intending to use with this project. If it is an emulated ZX Spectrum such as FUSE or Spectaculator, or a modern recreation such as Harlequin, you can skip this. However, if you want to load to an original ZX Spectrum, including later variants such as the 128+, you'll need further amplification. What will do the trick is a standard

Top Tip

Keep it down

Please be careful when testing playback, ZX Spectrums need it loud, so please don't listen directly to playback on headphones! If you need to test hold them away from you, you'll be able to hear!



▲ The Pirate Audio HAT is a DAC (digital-to-analogue converter) combined with a small headphone amp

battery-powered headphone amp. You need to connect its input to the Pirate Audio HAT and the output to the ZX Spectrum. It will provide the final boost needed for reliable loading.

06 Soundcheck

We've now got enough of the project working to test loading on a real ZX Spectrum (or whatever you are using) but we need something to load. We provided a test program. To download it, enter this at the command line:

```
wget https://github.com/mrpjevans/specdeck/raw/main/raspberrypi.wav
```

Once downloaded, hook up the output from your amplifier to the EAR socket on the ZX Spectrum, turn it up to max, run **LOAD ""** on the ZX Spectrum and enter:

```
aplay raspberrypi.wav
```

If all is well, the program will start to load!

specdeck.service

DOWNLOAD THE FULL CODE:
magpi.cc/specdeckservicepy

› Language: **Service**

```
001. [Unit]
002. Description=specdeck
003.
004. [Service]
005. ExecStart=/usr/bin/python3 /home/pi/specdeck/specdeck.py
006. Restart=on-failure
007.
008. [Install]
009. WantedBy=multi-user.target
```

07 Playing with Python

For our project, we want to be able to select from a number of games, and also handle conversion from the compact TZX format directly to WAV. We're going to create a Python app that will do this for us, as well as allow playback control and display cover art on the Pirate Audio's tiny display. To demonstrate how you can control playback of audio in Python, take a look at the **specdeck_1.py** code listing. We need the amazing Pygame module and a few other libraries to help us first:

```
sudo pip3 install pygame keyboard st7789 tzxtools
```

Now, if you enter the code into a file called **playback.py** in the same directory as the WAV file, you can start playback with:

```
sudo python3 playback.py
```

08 Get the full code

The final project code is a bit long for these pages, so we've provided an easy way to download everything and get running. From your home directory, run the following:

```
git clone https://github.com/mrpjevans/specdeck.git
```

This will download everything you need. The main file is called **specdeck.py**, but we've also included several iterations of the project (**specdeck_1.py** to **specdeck_6.py**) to show how the program evolved from the original code listing to the finished version. This method of coding is much more satisfying than trying to build everything all at once. You'll find everything in the **specdeck** directory.

09 Add some files

In the **specdeck** directory, you will find three further directories; **tzx** takes the standard ZX Spectrum TZX tape image format. When played for the first time it will be converted into a WAV file and placed in the **wav** directory, so we don't have to convert it again. If a JPG file is found with the same name in the **image** directory, it is loaded and displayed on the screen (it will be automatically

This inexpensive headphone amplifier can be combined with the Pirate Audio HAT to produce a sound loud enough for a ZX Spectrum to load reliably

resized if needed). You can add as many files as you like here from legitimate sites such as World of Spectrum (**worldofspectrum.org**). We've provided our test program so you can try things out immediately.

10 Add some files

From the command line, run this:

```
cd ~/specdeck
sudo python3 specdeck.py
```

After a few seconds, 'SpecDeck!' will be displayed on the screen and then Raspberry Pi's logo. This means all is working. Pressing button A on the HAT will start conversion of the TZX to WAV. This takes a little time on the Raspberry Pi Zero, but the resulting WAV will be kept so it will be instant next time. You can pause playback by pressing button A again and rewind by pressing B. On the right-hand side, X and Y will scroll through all the TZXs available. Press **CTRL+C** to stop the program.

11 Spectrum service

We don't want to have to log in via SSH every time we want to use our SpecDeck, so let's start everything up on boot by installing the program as a system-level service.

First, create a new file:

```
sudo nano /usr/lib/systemd/specdeck.service
```

Now add the text in the **specdeck.service** listing and use **CTRL+X** to save and exit. Now we have our service file, we can enable it to run on boot:

```
sudo systemctl enable /usr/lib/systemd/
specdeck.service
```

specdeck_1.py

DOWNLOAD THE FULL CODE:
magpi.cc/specdeck1py

Language: **Python 3**

```
001. from time import sleep
002. import pygame.mixer
003. pygame.mixer.init()
004. pygame.mixer.music.set_volume(1)
005. pygame.mixer.music.load("./raspberrypi.wav")
006. pygame.mixer.music.play()
007. while True:
008.     sleep(0.1)
```

Here we can see a healthy loading tone: the equal bars in the border show that we have the levels just right

Finally, test it with a reboot:

```
sudo reboot
```

On startup, you should see the 'SpecDeck!' announcement and then our test file. You can shut down your Raspberry Pi safely by pressing and holding button B for five seconds.

12 Add some files

Our SpecDeck is a little exposed and there's lots of delicate stuff that needs protecting. Fortunately, there are a lot of cases that people have designed for the Pirate Audio HAT that can be 3D printed. Our favourite, by Yasuhiro Wabiko, can be downloaded here: **magpi.cc/pirateaudiocase**. This can add that all-important protection and make the SpecDeck truly portable. Other improvements could be to add a battery to make it truly portable, or add support for other computers such as the ZX81 or Commodore 64. As ever, it's over to you.

Top Tip

Legal ROMs

Looking for content for your SpecDeck? The ZX Spectrum homebrew scene is as busy as ever. See **magpi.cc/legalroms.**

Part 03

Learn ARM assembly: Raspberry Pi Pico

Learn how to code a small assembly language program for Raspberry Pi Pico

MAKER

Stephen Smith

Stephen is a retired software developer who has written three books on ARM Assembly Language Programming. He is a member of Sunshine Coast Search and Rescue and enjoys mountain biking, hiking, and running. He is also a member of the Sunshine Coast Writers and Editors Society (**scwes.ca**).

magpi.cc/stephensmith

You'll Need

- Raspberry Pi
- Raspberry Pi OS 32-bit
- Raspberry Pi Pico
- Raspberry Pi Pico SDK
- Serial and debug connectors

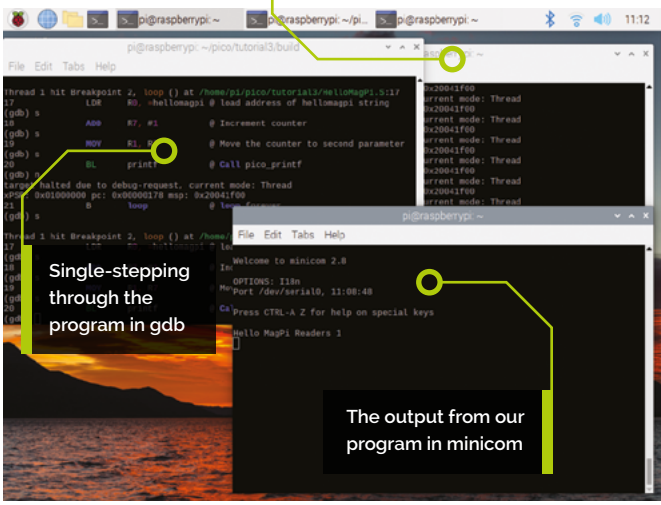
Raspberry Pi Pico is a low-cost microcontroller board built around Raspberry Pi's custom RP2040 system-on-a-chip (SoC) containing dual-core 32-bit ARM processor cores. The RP2040 uses ARM's microcontroller M-series instruction set, sometimes referred to as the 'thumb' instruction set, which is based on the 32-bit instruction set, except most instructions use 16 bits of memory. This makes programs compact and allows you to do quite a lot in the RP2040's 264kB of memory.

There is no operating system on Pico, and just one program running, which is your program. However, there is an SDK that you link into your program to provide useful function libraries, including those to initialise the processor. The SDK is open-source, and you can browse the processor initialisation assembly language code in **src/rp2_common/pico_standard_link/crt0.S**.

01 Create the program

Start by setting up Raspberry Pi Pico and install Pico's SDK on your Raspberry Pi. Connect your Pico to Raspberry Pi via the Serial Wire Debug (SWD) pins using jumper leads. Follow Chapter 5 in the Getting Started with Pico documentation if you need a refresher (**magpi.cc/getstartedpico**).

In Raspberry Pi OS, create a folder named **tutorial3** in the **pico** folder that was created in your home folder by Raspberry Pi Pico SDK's setup script. The source code for this tutorial is in **HelloMagPi.S** and **CMakeLists.txt**. Copy the file **pico_sdk_import.cmake** from the SDK's external folder to the **tutorial3** folder. In this folder, create a new folder called **build**. The **tutorial3** folder should now look like:




```
pi@raspberrypi:~/pico/tutorial3 $ ls -l
total 16
drwxr-xr-x 6 pi pi 4096 Apr  9 10:39 build
-rw-r--r-- 1 pi pi  412 Apr  9 10:37
CMakeLists.txt
```

```
 -rw-r--r-- 1 pi pi  665 Apr  9 10:37
HelloMagPi.S
 -rw-r--r-- 1 pi pi 2763 Apr  9 10:37 pico_
sdk_import.cmake
```

02 Build the program

If you have built a C program using Pico's SDK, then these steps are identical. The difference is that the **CMakeLists.txt** file lists an assembly language source file rather than a C source file. Open a Terminal window and `cd` to the **tutorial3** folder.

```
cd pico/tutorial3/build
```

Run `cmake` with the option to perform a debug build, so we can step through the program in the gdb debugger in a later step. Placing this command in a script file in the **$HOME/bin** folder with a short filename can be a real time-saver.

```
cmake -DCMAKE_BUILD_TYPE=Debug ..
```

The `cmake` command doesn't build the program, instead it creates a makefile used to compile by running `make`:

```
make
```

The `make` command takes time to execute and produces a lot of output. Even though our program is one small assembly language file, `make` compiles the entire Pico SDK and links it into our program.

Without an operating system running on Pico, just our program, everything needs to be in the one executable.

The **build** folder should now look like the example below:

```
 pi@raspberrypi:~/pico/tutorial3/build $ ls
-l
 total 1080
 -rw-r--r-- 1 pi pi  18809 Apr 11 09:21
CMakeCache.txt
 drwxr-xr-x 5 pi pi   4096 Apr 11 09:22
CMakeFiles
 -rw-r--r-- 1 pi pi   1670 Apr 11 09:21
cmake_install.cmake
 drwxr-xr-x 6 pi pi   4096 Apr 11 09:22
elf2uf2
 drwxr-xr-x 3 pi pi   4096 Apr 11 09:21
generated
 -rwxr-xr-x 1 pi pi  20224 Apr 11 09:22
HelloMagPi.bin
 -rw-r--r-- 1 pi pi 324975 Apr 11 09:22
HelloMagPi.dis
 -rwxr-xr-x 1 pi pi 307468 Apr 11 09:22
HelloMagPi.elf
 -rw-r--r-- 1 pi pi 226499 Apr 11 09:22
HelloMagPi.elf.map
 -rw-r--r-- 1 pi pi  56944 Apr 11 09:22
HelloMagPi.hex
 -rw-r--r-- 1 pi pi  40448 Apr 11 09:22
HelloMagPi.uf2
 -rw-r--r-- 1 pi pi  74243 Apr 11 09:21
Makefile
 drwxr-xr-x 6 pi pi   4096 Apr 11 09:21 pico-
sdk
```

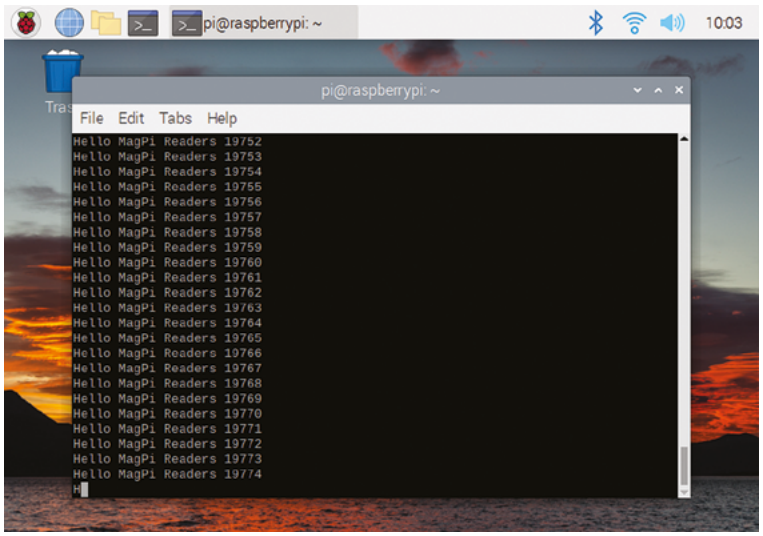
> Everything needs to be in the one executable

03 Run the program

Power on Raspberry Pi Pico by plugging the USB cable into Raspberry Pi, while holding down the BOOTSEL button. When the file explorer window appears, open it, and copy the file **HelloMagPi.uf2** to Pico; it then reboots and the

Stephen's stuff

He's written three books on Assembly Language Programming. The most recent one is *RP2040 Assembly Language Programming for the Raspberry Pi Pico*, which is the place to go for a deeper understanding of the topics touched on in this tutorial. The first one is *Raspberry Pi Assembly Language Programming* for 32-bit ARM code, and the second is *Programming with 64-bit ARM Assembly Language*.



▲ **Figure 1** Running our program and observing the output in minicom

program runs. The program prints out 'Hello MagPi Readers' and a counter. Open a serial port program to see this output.

```
minicom -b 115200 -o -D /dev/serial0
```

The `minicom` command displays data being sent from Pico to Raspberry Pi (see **Figure 1**).

> "The minicom command displays data being sent from Pico to Raspberry Pi"

04 Debug the program

Let's look at how the program works using the gdb debugger. Gdb will run on Raspberry Pi and remotely control Raspberry Pi Pico via the debug pins. To communicate with Raspberry Pi Pico's debug hardware, use the Open On-Chip Debugger (OpenOCD). First, configure gdb to talk to OpenOCD, by creating a file **.gdbinit** in the home folder. Create **.gdbinit** and place the following line in it:

```
target remote localhost:3333
```

Next, open a Terminal window to run OpenOCD. Enter the following command in the Terminal window:

```
openocd -f interface/raspberrypi-swd.cfg -f
target/rp2040.cfg
```

Now, let's debug our program, with the following command:

```
gdb-multiarch HelloMagPi.elf
```

To load the program, enter the `load` command:

```
(gdb) load
```

05 Step through the program

The program starts by executing Raspberry Pi Pico SDK's startup code. To skip this part and step through the program in our code, we will set a breakpoint at our main routine.

```
(gdb) b main
Breakpoint 1 at 0x1000035c: file /home/pi/
pico/tutorial3/HelloMagPi.S, line 14.
Note: automatically using hardware
breakpoints for read-only addresses.
```

Now, when we execute the program using the `continue` command, execution will initialise the RP2040 and Pico's SDK before halting at the start of the main routine.

```
(gdb) c
Continuing.
target halted due to debug-request, current
mode: Thread
xPSR: 0x01000000 pc: 0x00000178 msp:
0x20041f00
```

```
Thread 1 hit Breakpoint 1, main () at /home/
pi/pico/tutorial3/HelloMagPi.S:14
14 MOV R7, #0 @ initialize counter to 0
```

We can now execute a `step` command, the MOV R7, #0 instruction will execute, and we can use `info registers` to see that `R7` has been set to 0. `R7` holds an index that we increment in our loop.

```
(gdb) s
15 BL stdio_init_all @ initialize uart or usb
```

```
(gdb) i r r7
r7             0x0                 0
```

The next statement calls the SDK routine `stdio_init_all`, which initialises Pico's SDK to either send text output to the serial port or the USB port. This is configured in the **CMakeLists.txt** file by the following two lines:

```
pico_enable_stdio_uart(HelloMagPi 1)
pico_enable_stdio_usb(HelloMagPi 0)
```

Here, we configure output to the serial port. If we want to see our output file debugging, we must use this option. When gdb stops Pico's processor, it stops everything since our program is the only thing running. This stops and disconnects the USB connection, preventing it from working. The serial port doesn't require continuous attention, so it doesn't mind.

This routine contains a lot of code, and it will take quite a long time to step through it all. So, we will set a breakpoint at the next statement which happens to have a label, namely `loop`. Then we will continue, so we stop at the `loop` label after `stdio_init_all`.

```
(gdb) b loop
Breakpoint 2 at 0x10000362: file /home/pi/
pico/tutorial3/HelloMagPi.S, line 17.
(gdb) c
Continuing.
target halted due to debug-request, current
mode: Thread
xPSR: 0x01000000 pc: 0x0000012a msp:
0x20041f00
```

```
Thread 1 hit Breakpoint 2, loop () at /home/
pi/pico/tutorial3/HelloMagPi.S:17
17 LDR R0, =hellomagpi @ load address of
hellomagpi string
```

The next instruction is `LDR R0, =hellomagpi` which loads the address of the `hellomagpi` string into the register `R0`. This will be the first argument to `printf` which we call in a few steps. We step through this instruction.

```
(gdb) s
```

CMakeLists.txt

DOWNLOAD THE FULL CODE:
magpi.cc/learnassembly3

Language: **CMake**

```
001. cmake_minimum_required(VERSION 3.13)
002.
003. include(pico_sdk_import.cmake)
004. project(HelloMagPi C CXX ASM)
005.
006. set(CMAKE_C_STANDARD 11)
007. set(CMAKE_CXX_STANDARD 17)
008.
009. pico_sdk_init()
010.
011. include_directories(${CMAKE_SOURCE_DIR})
012.
013. add_executable(HelloMagPi
014.   HelloMagPi.S
015. )
016.
017. pico_enable_stdio_uart(HelloMagPi 1)
018. pico_enable_stdio_usb(HelloMagPi 0)
019.
020. pico_add_extra_outputs(HelloMagPi)
021.
022. target_link_libraries(HelloMagPi pico_stdlib)
```

```
18 ADD R7, #1 @ Increment counter
```

The `ADD R7, #1` instruction increments our counter that is printed out. Again, we step through this instruction.

```
(gdb) s
19 MOV R1, R7 @ Move the counter to second
parameter
```

Perform `MOV R1, R7`, which moves the counter from register 7 to register 1. This is since `R1` holds the second parameter to `printf`. We can't use `R1` as our counter, since `printf` will not preserve the value of this register. Step through this instruction.

```
(gdb) s
20 BL printf @ Call pico_printf
```

The `BL printf` instruction will call Pico SDK's `printf` function. This isn't the `printf` function from the C runtime, but part of the SDK. It is a good implementation of `printf` and helpful to use from any programming system using the SDK. If we step, it will take a lot of step calls to get through this function, so we use the next command to execute this instruction without debugging into it.

```
  (gdb) n
  target halted due to debug-request, current
mode: Thread
  xPSR: 0x01000000 pc: 0x0000012a msp:
0x20041f00
  21 B loop @ loop forever
```

HelloMagPi.S

> Language: **Assembly Language**

```
@
@ Assembler program print out "Hello MagPi Readers"
@ using the Pico SDK.
@
@ R0 - first parameter to printf
@ R1 - second parameter to printf
@ R7 - index counter
@

.thumb_func                  @ Necessary because sdk uses
BLX
.global main         @ Provide program starting address
to linker

main:
        MOV     R7, #0          @ initialize counter to
0
        BL      stdio_init_all  @ initialize uart or
usb
loop:
        LDR     R0, =hellomagpi  @ load address of
hellomagpi string
        ADD     R7, #1          @ Increment counter
        MOV     R1, R7          @ Move the counter to
second parameter
        BL      printf          @ Call pico_printf
        B       loop            @ loop forever

.data
.align  4                       @ necessary alignment
hellomagpi: .asciz   "Hello MagPi Readers %d\n"
```

> "Raspberry Pi Pico's SDK has full support for assembly language programming"

If you have `minicom` running, you should see the string print. The `B loop` instruction branches back to the `loop` label. In most environments, executing an infinite loop is a bug. In the microcontroller world this is typical because you want your program to run forever – there is nothing else for the processor to do. Execute a `step` command to jump back to the `LDR` statement. We need to re-initialise `R0`, since `printf` will have overwritten the value we placed previously.

```
  (gdb) s
  Thread 1 hit Breakpoint 2, loop () at /home/
pi/pico/tutorial3/HelloMagPi.S:17
  17 LDR R0, =hellomagpi @ load address of
hellomagpi string
```

06 Modify the program

Congratulations! Welcome to the world of writing assembly language for microcontrollers. Raspberry Pi Pico's SDK has full support for assembly language programming. If you browse the SDK source code, you will find quite a few functions written in assembly language. Try making some simple modifications to this program to ensure you understand it. Try using the `MUL` instruction instead of the `ADD` instruction, counting down with the `SUB` instruction, and adding additional elements to the print string.

HiPi.io

HIGHPI PRO

The new case from the HiPi.io team

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

Available at these great Pi stores:

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

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

Part 1

Write your own Game Boy game in C

Learn the basics of making a game on original handheld hardware with the Game Boy Development Kit

AUTHOR
EDWIN JONES

Edwin Jones is a senior software engineer at Mediatonic who's been working in the games industry for around 17 years. **edwinjones.me.uk**

Download the code from GitHub: **wfmag.cc/wfmag62**

How can you make your own game for a handheld console that came out in the 1980s? It's actually quite easy, thanks to a few factors. Because the Game Boy is so old, it's simple to program. And, thanks to the open-source community, its tools are better now than they've ever been. If you have a desktop computer, you can write code and run it on a Game Boy.

One of the simplest ways to get started is with the Game Boy Development Kit, or GBDK. We can use this to write C code and compile it into a Game Boy program, also known as a ROM. There are a few versions of software around, but we're using the modern version on GitHub: **wfmag.cc/gbdk2020**.

There's quite a bit of code required for this project – too much to print here in full, but you can download the files you'll need from our GitHub at **wfmag.cc/wfmag62**. We'll be going over all the code you need step by step, so don't worry if you don't understand it all at first. If you're a C/C++ wiz, you'll probably find this easier than previous projects, but the Game Boy still has some quirks you may be unfamiliar with. If you want to jump ahead, I can recommend this primer from freeCodeCamp: **wfmag.cc/c-begin**.

Our first step is to configure our tools to allow us to write some code, compile it, and deploy it. First, you'll need a Game Boy or another compatible console such as the Game Boy Advance or DS. You'll also need a flash cart to copy ROMs onto, such as an EZ Flash: **wfmag.cc/ezflash**. If you don't have access to a Game Boy, you can just run your code on an emulator. I'm using BGB (**wfmag.cc/BGB**), which has excellent tools for visualising what our code's doing as it runs.

Once you have these tools, you'll need to install GBDK. Follow the instructions on GitHub for your preferred operating system to get it running. Follow the README to get started (**wfmag.cc/GBDK**). Download the latest release and unzip it to the directory of your choice. Once you've got that set up, we need to write some code. Open your editor of choice (my favourite is VS code: **wfmag.cc/vscode**) and create a file called **main.c**. Add the following text to this file:

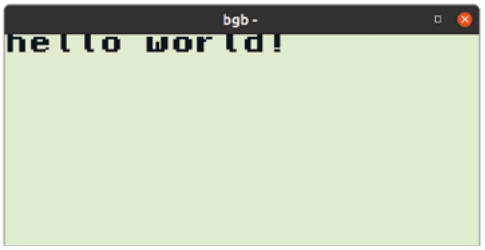


Figure 1: This is what your Hello World should look like. Notice how a Game Boy-compatible font has automatically loaded.

```
#include <gb/gb.h>
#include <stdio.h>

void main()
{
        printf("hello world!");
}
```

INCLUDES

You've probably twigged that our 'main' function is the code that our program executes at run time. Those lines above that which start with a '#' are includes. These tell the compiler to add some code in place of these lines when compiled. We're using this to import the GBDK code we'll need to make our code talk to the Game Boy hardware directly.

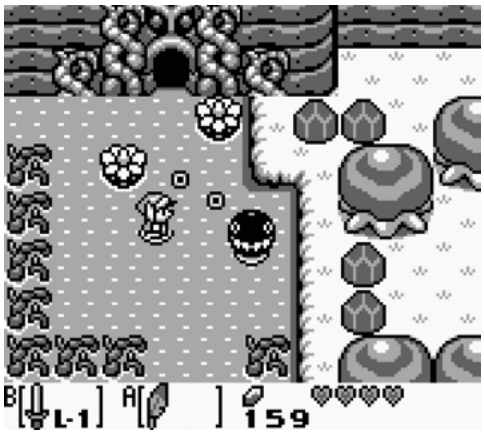


Figure 2: The player's health, money, and equipped items are drawn to the window layer.

Now run the following commands in a terminal from the directory you saved this file in:

```
"{path where you installed the GBDK}/bin/lcc"
-c -o ./main.o ./main.c
"{path where you installed the GBDK}/bin/lcc"
-o ./main.gb ./main.o
```

If you open that folder you should see a new file in there called **main.gb**. Open this in your Game Boy emulator and you should see the output in **Figure 1**.

Congratulations! You've written and deployed your first Game Boy program. Feel free to move this onto a flash cart and run it on your own hardware.

"You've written and deployed your first Game Boy program"

HELLO WORLD

Now our tools work, let's look at the Game Boy's specs in more detail so we can understand how it works before we make a real game:

- A 160×144 pixel four-colour LCD screen (256×256 addressable, can draw off-screen)
- A four-way direction pad and four buttons (start, select, A, B) for a total of eight inputs
- An 8-bit Sharp LR35902 CPU
- 8kB of working RAM
- 8kB of graphics memory
- At least 32kB of ROM memory from the game carts themselves

These are the basics we need for this project. You may notice that the Game Boy can 'draw' to 256×256 pixels with four different colours, which would mean we'd need at least 16kB of memory to control the colours. The Game Boy only has half of that for graphics memory. How is this possible? Because the console doesn't control every pixel individually – it uses tiles to save on memory. These are split into two kinds – background tiles, and sprite tiles which are used for foreground objects like the player character and enemies. Sprites can be 8×8 or 8×16 pixels wide.

The Game Boy can draw a maximum of 40 sprites on screen at any one time with a limit of ten per line. We'll avoid this limitation by making sure to draw fewer than this number. The console can also draw onto two layers: the background layer and what's called a window layer. This is used for UI elements such as the player HUD or health bars. If you go back and play some of your favourite Game Boy titles, like *The Legend of Zelda: Link's Awakening* (**Figure 2**), you can see where the layer's been placed.

The Game Boy lets us scroll the background and move sprites freely, so we don't need to worry about clearing any previously used pixels. It even handles wrapping for us – if a background or sprite moves too far in one direction, it will appear on the other side of the screen.

Now you understand a bit more about how the Game Boy works, let's get started with our own game: a simple infinite runner called *Drop Bear*. A bear's falling from the sky, dropping coins; how many can he collect before the time runs out?

First, we'll need to create some sprites using a browser-based tool called the Game Boy Tile Data Generator (**wfmag.cc/gbtdg**). Open the folder the project has downloaded to, then open the **index.html** file in your browser, and you should see something like **Figure 3**.

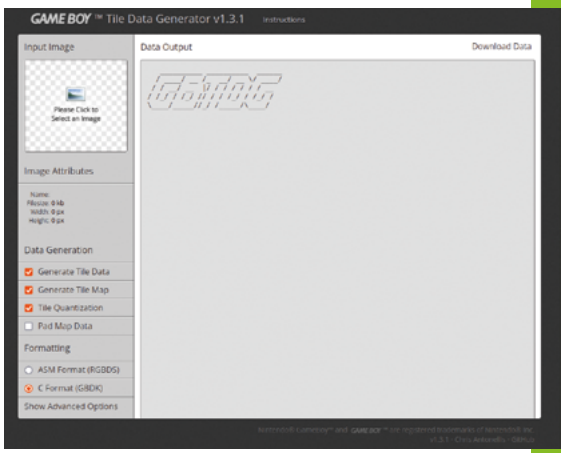


Figure 3: This is how the Tile Data Generator looks in a browser without any data uploaded.

ON THE TILES

Why so much memory per tile? Each 8×8 pixel tile is encoded into 16 bytes that contain which colour each pixel of the tile can be. That's 64 pixels that could be one of four different colours the Game Boy screen could draw – that's 128 bits. Divide that by 8 bits per byte and you get 16 bytes.

Wireframe

This tutorial first appeared in Wireframe, our sister magazine that lifts the lid on the world of video games. Every issue includes tutorials and in-depth interviews, along with news and reviews of the latest indie and triple-A games.

To find out more, visit their website at **wfmag.cc**.

Check out their subscription offers at **wfmag.cc/subscribe**.

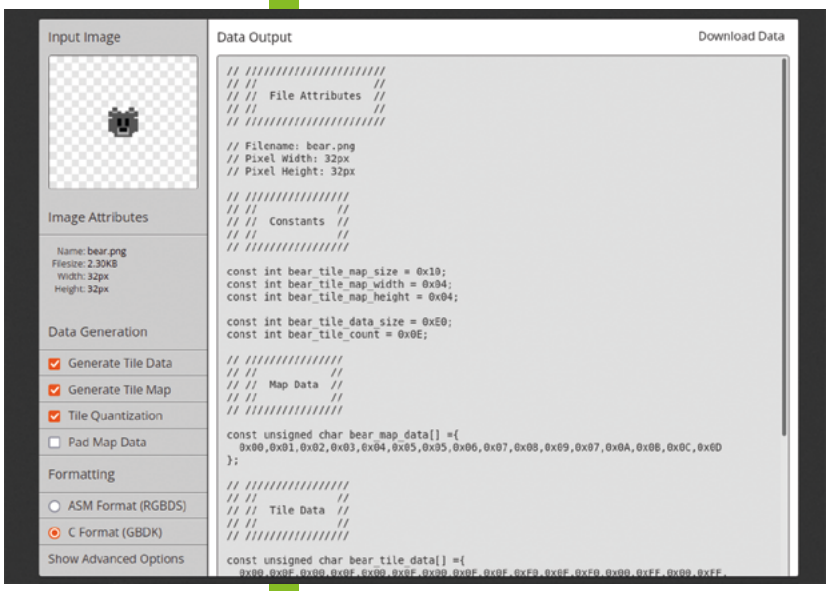


Figure 4: This is how the Tile Data Generator looks with a sprite loaded. Note how text has generated to the right of the loaded image.

If you look at the bottom left, you'll see two options under 'formatting' – it's important we use 'C format GBDK' for all our assets as that means the output will work with GBDK. I've already created all the assets our game needs, but let's walk through how to do this yourself so that you can make your own or edit them later. All you need to do is click under 'Input Image' on the top left and select an image from your local hard drive to load it into the editor. When we do this with a pixel art image of a bear's face I've prepared for the project, we get the output in **Figure 4**.

See the text on the right under 'Data Output'? That's the raw binary data the Game Boy needs to draw this image alongside a few constants, like how many tiles wide and high the sprite needs to be – as the PNG I've used is 32×32 pixels, we'll need four 8×8 tiles to represent the image on a Game Boy. You can see the height and width constants are the number 4. I'll include the relevant data in-line when we need it, as we go along. Feel free to import your own assets, though, as that's half the fun of making your own game.

First, let's create the title screen. For this, we'll need a logo asset and some on-screen text. Let's start with the easiest bit, printing the text. We can do this by modifying our 'Hello World' program – find the code you need in the file **tile-screen-code.c** on our GitHub: **wfmag.cc/wfmag62**.

If you compile and run this program the same way as before, you should see the output in **Figure 5**.

HOLY MACRO

A macro is a fancy word for something the compiler replaces at compile time. You can create your own macros to replace any text with any other bit of code. For instance, you could replace 'printf("hello")' with a macro by adding a line like '#define HELLO printf("hello")' to the top of your code file, then by using 'HELLO' in place of the formal function call.

LET'S MAKE A SPLASH!

There's our splash screen! That was a lot of code, though, so what did we just do? Well, everything outside the main function is just asset data for the logo, generated the same way as we've described before, but with one crucial difference – we've offset everything in the tilemap by the value 0×80. This is a bit of a hack, as the Game Boy has limited memory. We're already using a third of it for the font we need to print text with, so we push the logo data into a different part of the video memory so we don't override the font that's automatically loaded with the first `print` statement. You can view graphics RAM in BGB to see it for yourself by right-clicking then using the Other > Vram viewer menu option (**Figure 6**).

The first page is taken up by the Nintendo logo – this is a form of primitive DRM as the Game Boy won't boot if this logo isn't the first thing loaded into memory. We'll override it soon, though, as once you're past the logo screen, it's no longer needed. The *Drop Bear* logo is in one segment of memory and the font's in another, while tiles currently rendering are highlighted in green. We're only using a few characters of our font, but all of our logo files. Note that the composed logo reuses some of these tiles to save memory as we covered earlier.

Interestingly, the Game Boy's video memory layout starts at address 0×8000 and ends at 0×97FF. Addresses 0×8000 to 0×8FFF were designated for sprite data, as you can see with the first two blocks in the bgb vram viewer screenshot (**Figure 6**). Background memory runs from 0×8800 to 0×97FF. "Hold on, those addresses overlap!" I hear you cry, and you're correct. The middle page of graphics memory from 0×8800 to 0×8FFF can be used by either sprite or background tile data, but not both. This is why we needed to move the logo tilemap into a different part of memory, so we could load our logo into tile memory without overriding the font that lives in the highest part.

"The *Drop Bear* logo is in one segment of memory and the font's in another"

We add 0×80 to each tilemap value because the Game Boy treats the first numbered background tile (0×00) as living at address 0×9000, not 0×8000. Adding 0×80 pushes the tile indices into the correct portion or VRAM where we've loaded our logo. This is because 0×80 is 128 in decimal, which is exactly how many tiles

each third of the video memory can hold – 128 tiles * 3 blocks * 16 bits per tile = 384 available tiles to use.

Inside the main function, we invoke a few macros to clear the screen. We then call **printf** with a blank string to tell GBDK to load the default font into memory without drawing anything on screen. We move the background layer a little, then load the tile logo into memory by setting the tile data and tile layout map. We call another macro to force the background layer to render and show our logo. We then call another function to move the print cursor and print the words 'Press Start' onto the bottom of the screen.

Let's move on to input. Most games have multiple 'states'. Here, we'll keep it simple by only using three – the game splash screen, the game play screen, and a game over screen. We can get the Game Boy to wait for our input with another command. Add the following lines to the end of your main function:

```
waitpadup();
waitpad(J_START);
cls();
```

These functions wait for all buttons to be released, just in case the user has any depressed. When the Start button is pressed, the screen is cleared. Try it yourself in the BGB emulator: press the **RETURN** key, and the screen should clear. We can flesh out the gameplay by adding a background that loads when we move to the second stage. As our game involves a dropping bear, a simple trick is to load a background we can then scroll vertically to generate the feeling of movement. You can make a striped tile via the tools mentioned above, but for now, copy the code from the file **background-tile-code.c** on our GitHub and paste it above your main function. Then add the following lines to the end of your main function:

```
// Load tileset into background memory
set_bkg_data(128, BACKGROUND_TILE_SET_COUNT,
background_tile_set);

// Load tile map into memory.
set_bkg_tiles(0, 0, BACKGROUND_TILE_MAP_WIDTH,
BACKGROUND_TILE_MAP_HEIGHT, background_tile_
map);
```

We're loading this new tilemap to the background layer as we did with the logo – you'll see each tilemap entry is adjusted by 0×80, so we don't interact with the loaded font as before. Compile your program again and load it into BGB. You should now see **Figure 7** (overleaf).

Now, at the end of your main program, add the following code to the end of your main function:

```
// scroll the background forever
while(1)
{
    scroll_bkg(0, 3);
    // Wait until VBLANK to keep time
    wait_vbl_done();
}
```

Figure 5: Our title screen.

ABOUT VBLANK

VBLANK is the time taken from the end of drawing the last line on the screen to the start of drawing the next. By waiting until the VBLANK period, we know we're not looping faster than the screen's drawing. We can use this to make sure each loop executes no faster than a single frame, which takes roughly 16 milliseconds (1 second / 60 fps = 0.01666).

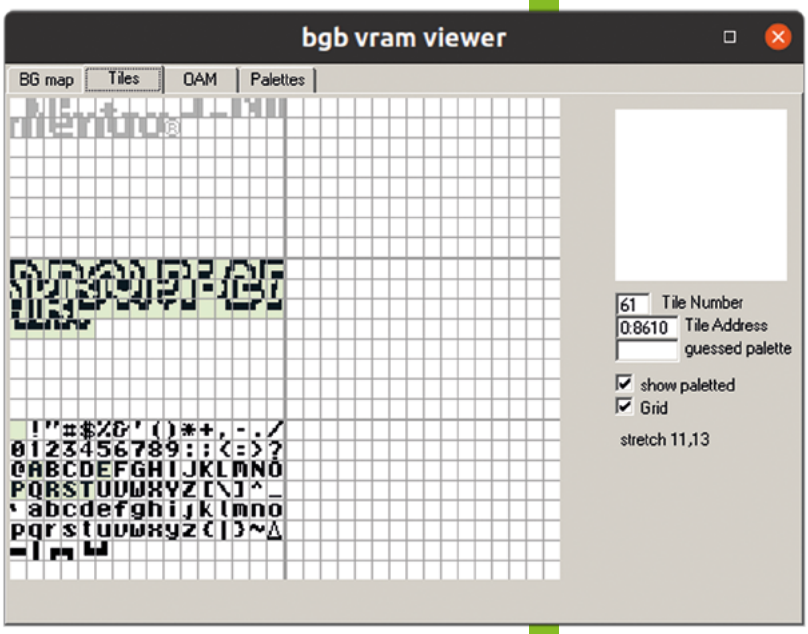


Figure 6: You can see the sprites that make up each letter in this view. This is the same font we used for our Hello World program.

SHORTHAND

You might have noticed we used a new type here called 'UINT16'. This is a 'typedef': a way C programs can use their own shorthand to refer to existing types. The GBDK defines these for us so we have a way to define unsigned integers by size instead of using byte for UINT8 etc.

If you compile and run your game again, then press Start, you'll see the background scrolls. We've used an **infinite** loop that scrolls the background up three pixels and then waits for a vertical blank. This ensures the code only runs once a frame. The Game Boy has a frame rate of ~60 fps, which means we're moving the background at roughly ~180 pixels a second.

FALLING DOWN

Next, let's add a little code to allow the main game to run for a while, then stop. We can then add some code that resets us back to the main splash screen, creating a basic game loop. We know that each refresh of our loop happens roughly 60 times a second, so if we want to create a quick ten-second cycle, we can start a counter at 600, decrement it each loop, and exit when it finishes. Add the following code to the above main function:

```
UINT16 game_time = 0;
```

This lets us store a value of 16 bits, more than enough for number 600. Next, add this at the very top of your main function:

```
start:
```

This is a label for a **goto** statement which we'll use in a minute. Next, update the **loop** statement like so:

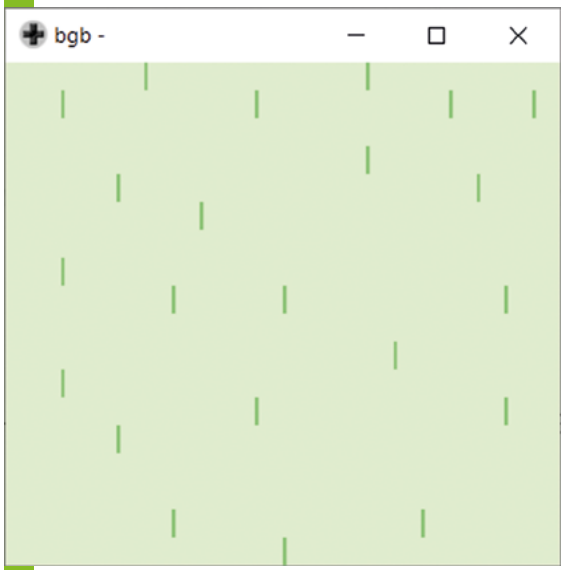


Figure 7: Notice how the tiles repeat over and over – this is how we can make a little bit of data useful over a large screen area.

```
// set game timer
        game_time = 600;
        while(game_time)
        {
        // scroll the background
        scroll_bkg(0, 3);

        //decrement the game timer
        game_time--;

        // Wait until VBLANK to avoid
corrupting memory and keep time
        wait_vbl_done();
        }

        //jump back to the start of the
program
        goto start;
```

Recompile and run the game again. The screen should fall for ten seconds then you should come back to the start screen.

Next up, let's add a player character with sprites. I've prepared the bear sprite data mentioned earlier, which you can find in the file **player-character-code.c**. Just copy this code and paste it outside of the main function (as you have previously) for now.

First, we need to define some variables to control the position of our player character. Add this code before your main function like so:

```
UINT8 player_x = 72;
UINT8 player_y = 32;
```

Now we can load the player character into memory with a similar set of commands as before. Add these lines to the bottom of your main function just before the **while** loop:

```
// Load the 'sprites' tiles into sprite
memory
set_sprite_data(0, BEAR_TILE_SET_COUNT, bear_
tile_set);

// Set the first movable sprite (0) to be the
first tile in the sprite memory (0)
for (UINT8 i = 0; i < BEAR_TILE_MAP_SIZE; i++)
{
    set_sprite_tile(i, bear_tile_map[i]);
}

SHOW_SPRITES;
```

Figure 8: Here's our bear sprite drawing over the background.

You may have noticed a difference in the name of the command from before – this is because we're setting sprite data, not background data. As we need more than one 8×8 tile to draw an image, we also need to set multiple sprite tiles at a time, hence the loop. We then invoke a macro to make sprites visible. Positioning the sprites is a little trickier – add a new block inside your **for** loop before the **wait_vbl_done()** call like this:

```
// move player
  for (UINT8 y = 0; y < BEAR_TILE_MAP_WIDTH;
y++)
  {
      UINT8 yOffset = y * 8;

      for (UINT8 x = 0; x < BEAR_TILE_MAP_
HEIGHT; x++)
      {
          UINT8 xOffset = x * 8;
          move_sprite(tileCounter++, player_x
+ xOffset, player_y + yOffset);
      }
  }
  // update input
  // LEFT
  if (joypad() & J_LEFT)
  {
      player_x -= 2;
  }
  // RIGHT
  if (joypad() & J_RIGHT)
  {
      player_x += 2;
  }
```

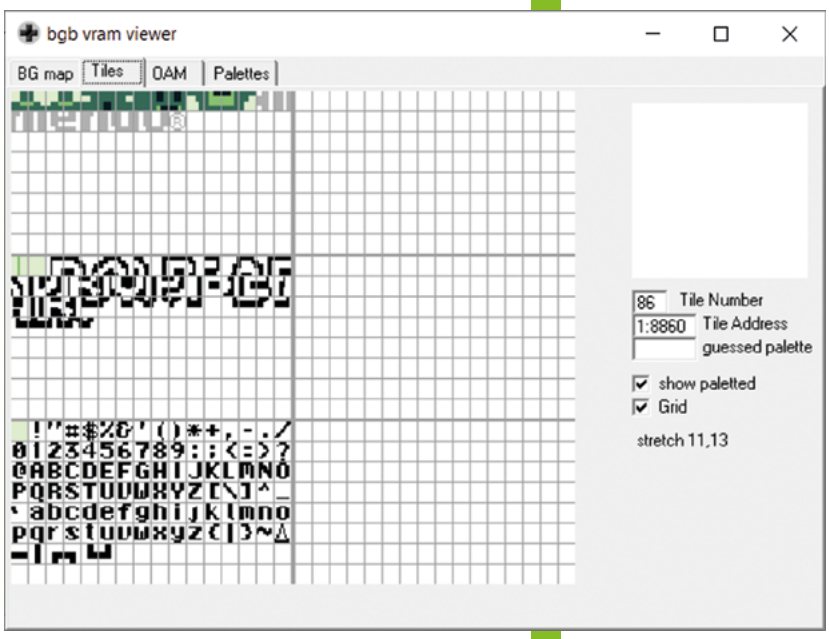


Figure 9: Here you can see the bear sprite in VRAM. Note how some tiles are repeated to save memory.

Here, we write some loops to go over every tile of our sprite and move them around based on the current player x and player y values we configure. We then wait for player input to see if the sprite's x position should move. If you're wondering about what '8' and '2' mean, the '8' is the size of a tile in pixels and the '2' how many pixels we want to move per update.

BEAR NECESSITIES

If you recompile your game and load it in an emulator, you should see **Figure 8** if you press Start on the splash screen, and if you press the left and right keys/buttons, the bear will move around the screen.

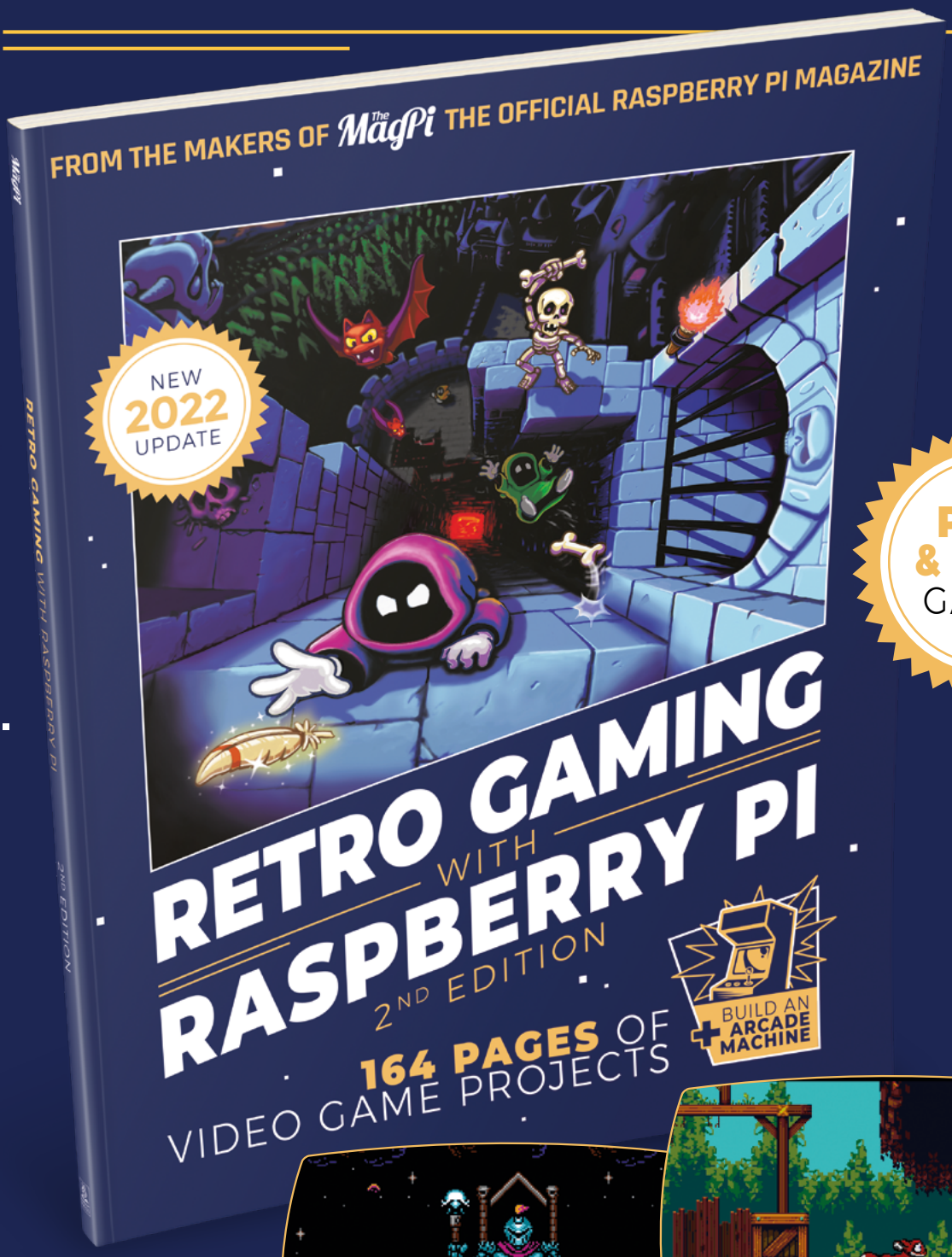
If you load up the VRAM debugger view again, you'll see that our bear sprite has overwritten some of the Nintendo logo in that memory area (**Figure 9**).

If you're wondering why some tiles are highlighted in this view, it's just BGB's way of telling you what's currently being drawn on screen. The other tiles represent that state of VRAM for things not currently being drawn. This can be handy for figuring out why things you load in aren't being drawn – usually because their positions haven't been correctly set.

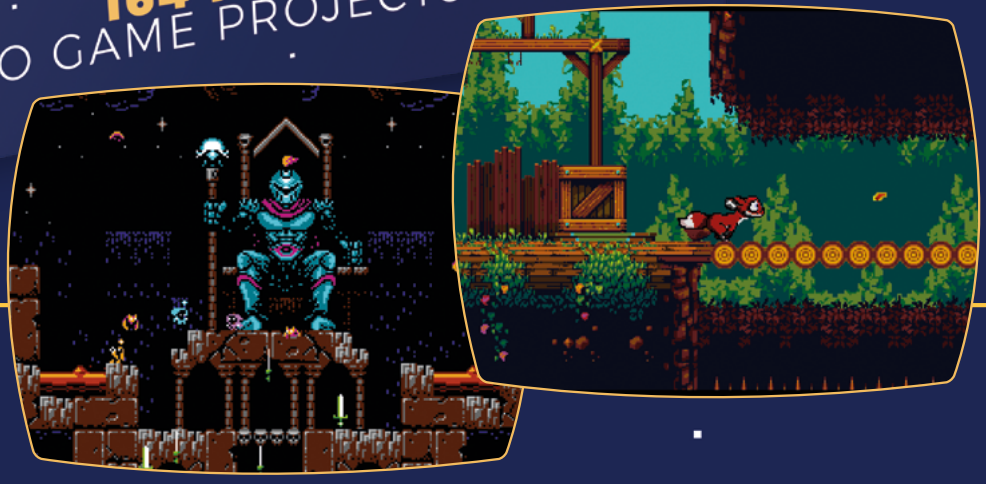
And that's it for now. In the second part of the guide, we'll add objectives, a timer, and end game states. See you next issue.

GOTO JAIL

GOTO is a simple C command that lets you move execution to any other part of the code defined by a label. It's fallen out of favour in modern times as it's considered risky, but it still has uses now and then as you can see in our code – as long as you keep an eye on exactly where your labels are!



PLAY & CODE GAMES!



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/store**

WEARABLES & COSTUMES

Make your clothes Raspberry Pi-powered masterpieces

with **Rob Zwetsloot**

Costumes and cosplay aren't just about sewing. Crafting armour from foam or thermoplastics, or turning some PVC piping into a spear is all part of it these days. So are electronic add-ons like screens, LEDs, or moving parts.

Raspberry Pi Zero and Raspberry Pi Pico are perfect devices to use with more interactive costumes. With their small size, low power requirements, and ease in which to program, they're a great choice for your Iron Man costume.

Time to make some plans and try to get your costume finished before the con.

Cosplay resources

Help with the costume part of your build

Cosplay photography

If you want to take decent selfies or other shots of your own cosplay, this guide is a great start to getting into that.

magpi.cc/costog

Coscraft

A store that stocks and sells many cosplay materials, from wigs to fabric, to even craft foam and thermoplastics.

coscraft.co.uk

KamuiCosplay

Tutorials, books, and patterns to create your own incredible costumes from Kamui.

kamuicosplay.com

Wearable Tech Projects

From Sophy Wong, and our colleagues over at HackSpace magazine, comes this great book full of excellent wearable projects and ideas beyond Raspberry Pi.

magpi.cc/wearablebook

Alysson Tabbitha

Alyson has incredible make-up tutorials, something she is an expert on, being able to transform herself into many different characters.

magpi.cc/tabbitha

LIGHTING UP

LEDs and displays

Want to really make your costume eye-catching? Install lights and displays on it so that you instantly light up the room.

Cosplay lights

Maker
Freya and Rob Zwetsloot

Web
magpi.cc/cosplayeyes

Project
Using NeoPixel LEDS, these lights were programmed to recreate colour patterns for the character Sans from Undertale. Using a button press cycled through different settings.

01 Build the circuit

For this project, a round NeoPixel board was used. NeoPixel lights require power, a data input, and ground. While there are not many lights in this setup, they usually require more power than a Raspberry Pi can deliver, so you'll need a battery pack as well. A button is also added, which acts as the trigger for the different light effects.

02 Coding

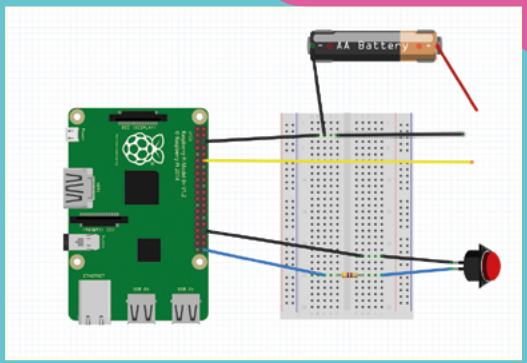
NeoPixels require special libraries to work with Python and MicroPython, including Adafruit_Blinka (**magpi.cc/blinka**) and the CircuitPyhon libraries, like so:

```
sudo pip3 install rpi_ws281x adafruit-circuitpython-neopixel
```

Colours are done with RGB values from 0 to 255. If you're looking for a specific colour, Google the name of it with 'RGB' appended, and you'll usually find a guide on how to make the colour. The code for this build uses GPIO Zero, but Pygame can also work with the button input.

03 Installing into the costume

In this build, Freya glued the NeoPixel ring to some frosted acrylic in the eye, allowing the lights to diffuse and look more natural. The wires were quite long and she was able to have them all fit into the pocket of the hoodie through a hole on the inside. With everything in place, she just had to press the button in her pocket to change the code sequence.



Stomach shot

01 Set up the hardware

The basic hardware for this is a Raspberry Pi Camera Module, any kind of Raspberry Pi-mountable display, and some cable extenders for the camera cable. All you need to have it do is use the camera preview to have a constant display of what the camera sees, which is set up to start during boot.

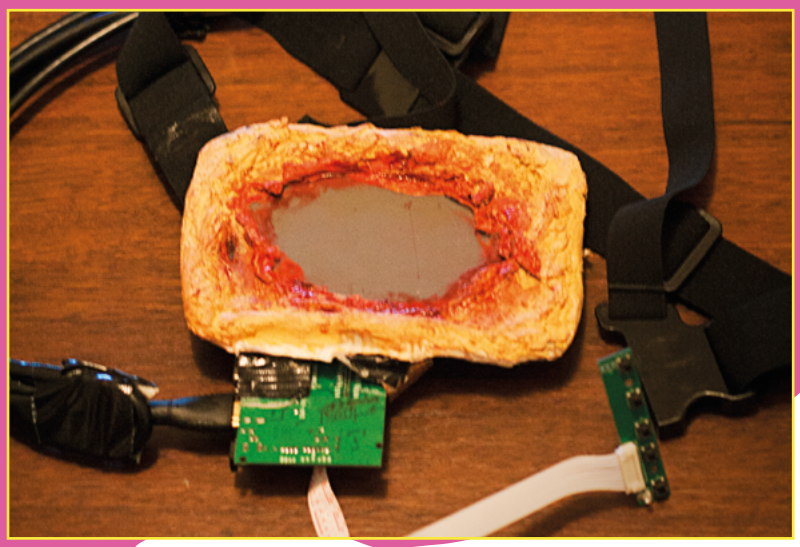
02 Build the prosthetic

Luis went the extra mile and created a latex stomach wound attached to the front of the display, and connected to a harness he could wear under his T-shirt. It's coloured with make-up to look skin-coloured, and paint for the blood and 'insides'.

03 T-shirt time

The last part is the T-shirt. Two holes are cut into it: one at the front that is slightly smaller than the screen, and one at the back so that the camera can see behind you. With the harness on and the T-shirt over it, it's time to turn it all on and try and scare some kids. Although, apparently, they tend to just want to take selfies through the stomach.

With the harness on and the T-shirt over it, it's time to turn it all on



Maker

Luis Martín Nuez

Web

magpi.cc/stomachshot

Project

Inspired by *The Walking Dead*, Luis decided to make a Halloween costume where you could see through him. In TV and movies, there's green screen magic involved during editing. In real life, it's as simple as using a camera and display.

MOVING MOTORS and servos

Robots and robotic armour are cool costumes to make. And sometimes, you need some fake eyes to blink. Using servos and motors, you can really make a costume come alive.

Tentacle hat

Maker
Derek Woodroffe

Web
magpi.cc/tentaclehat

Project
This writhing and wiggling hat is more of a passive accessory. Just turn it on and the tentacles will move of their own accord.

01 Electrical components

The hat uses eight servos, two for each tentacle, which are hooked up to a Raspberry Pi Zero. A sine wave is sent to each servo – an X and Y movement – to try and emulate a more natural movement in the finished product. This build uses an eight-port I2C PWM controller so that everything can be connected properly.

02 Make the tentacles

The build of the tentacles is very smart; the core is made from the spring of a net curtain so it's flexible and moves in curves. Acrylic discs are slotted over it to give the whole thing some structure, before stockings are applied on top. Some latex is applied on top, and some laser-cut 2 mm MDF suckers are glued on. Wonderful.

03 Put your hat on

Cut a hole in a top hat (preferably a spare one from a costume shop) and glue the whole apparatus into the hat. Green latex was applied here to make it look like they were bursting out of a pool of ooze. The only thing left is power which, in this case, is held in a special box for the whole system. You can add it to your belt or secrete all the components in a jacket.

Face-tracking NERF gun

Warning!
Projectiles

Wear goggles when working with face tracking weapons, however soft the darts are

magpi.cc/nerfsafety

Maker
Engineering After Hours

Web
magpi.cc/predatortrack

Project
Taking inspiration from the predators of... *Predator*, this shoulder-mounted NERF gun uses AI to track faces and aim a NERF gun in that direction. Good thing there are no laser pointers attached.

01 Creating a model

Using a Raspberry Pi connected to an Edge TPU, you can start training the model for your AI face tracker. Feeding it with pictures of faces allows it to learn what they look like, although you can always find existing models that will have faces already built in. The more it's used, the better it becomes as well.

02 Tracking

A Pan-Tilt HAT from Pimoroni was used, with the code manipulating the motors to turn the pan and tilt portions of the setup. This is usually used in robots for object tracking, but is perfect for anything that requires a camera to move and track at the same time. The NERF gun needs to be mounted to the Pan-Tilt HAT and the camera mounted to the gun.

The more it's used, the better it becomes as well

03 Wearing a gun

The mount in this project is quite simple – a few pieces of wood are cut to length and attached to a series of straps that keep the gun counterbalanced while wearing it on his back. The whole thing is powered by a series of batteries and, using a simple wireless remote, can be fired at the person it's tracking. Be careful, even with foam darts.

INSTALLING
inputs and sound

Your costume moves? It lights up? Great. Does it have sound effects and interactive elements on the outside? Time to add them.

Social media without the internet

Maker
Tuang Thongborisute

Web
tuangstudio.com

Project
This social interaction project has a lot of buttons and lights and displays everywhere, which the general public can interact with, just like Twitter, Facebook, or Instagram.

01 Components galore
There are several parts to this outfit, controlled by a Raspberry Pi and several Arduino controllers. Conductive rings on the fingers touch during a handshake, a button is used to unfriend and is tied to an LED display, a like is a high five which activates a force-sensitive resistor (FSR), dislike is via a touchscreen, and microswitches and pressure-conductive resistors (PCR) are used for follow/following. It's a lot to connect, but very cool.

02 Jacket attachment
Once all of the complicated circuit stuff is completed, it's time to attach it to a jacket. This can be done with conductive thread to reduce wire usage, along with tape to attach things and glue and whatever else you have. The locations are important for the experiment – the unfollow button is over the heart you could break, and the PCR is on the shoulder.

03 Socially interact
The experiment can now begin – go out into the world and find out what people think about you. Tuang's experience showed that people react differently: "Some may hesitate to ask or interact, but some partake in face-to-face conversation."

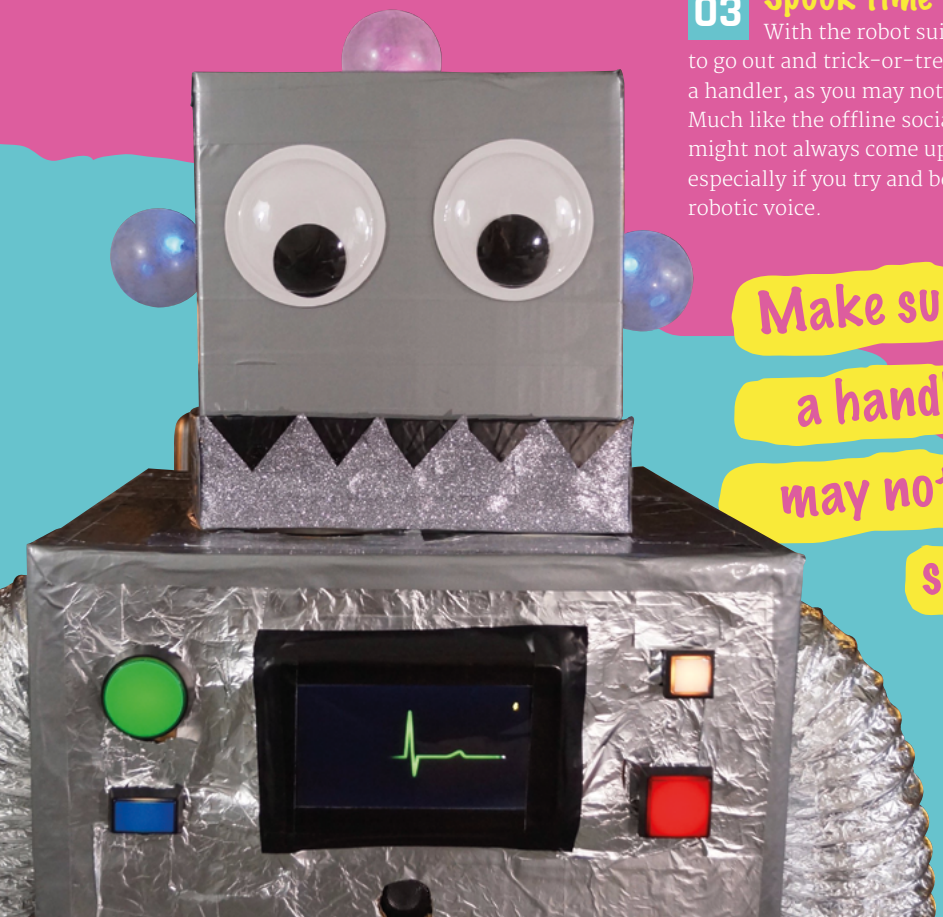
Robot costume

01 Hardware setup

The display on the chest uses Instagram likes to create an electrocardiogram heartbeat, and the buttons control the LED colours, play random sounds, and start playing videos on the display. The voice changer uses Arduino parts as well, and employs a microphone inside the head. The speech activates the servo that flaps the robot jaw.

02 Cardboard construction

One of the best things about this costume is how much cheap and recycled material it uses. Cardboard boxes, googly eyes, aluminium foil, and air conditioner hoses, among many other things. Be careful when cutting and gluing to make sure you have enough space to actually fit inside.



03 Spook time

With the robot suit completed, it's time to go out and trick-or-treat. Make sure you have a handler, as you may not be able to see very well. Much like the offline social media jacket, people might not always come up to interact with you, especially if you try and beckon them over with a robotic voice.

Make sure you have a handler, as you may not be able to see very well

Maker

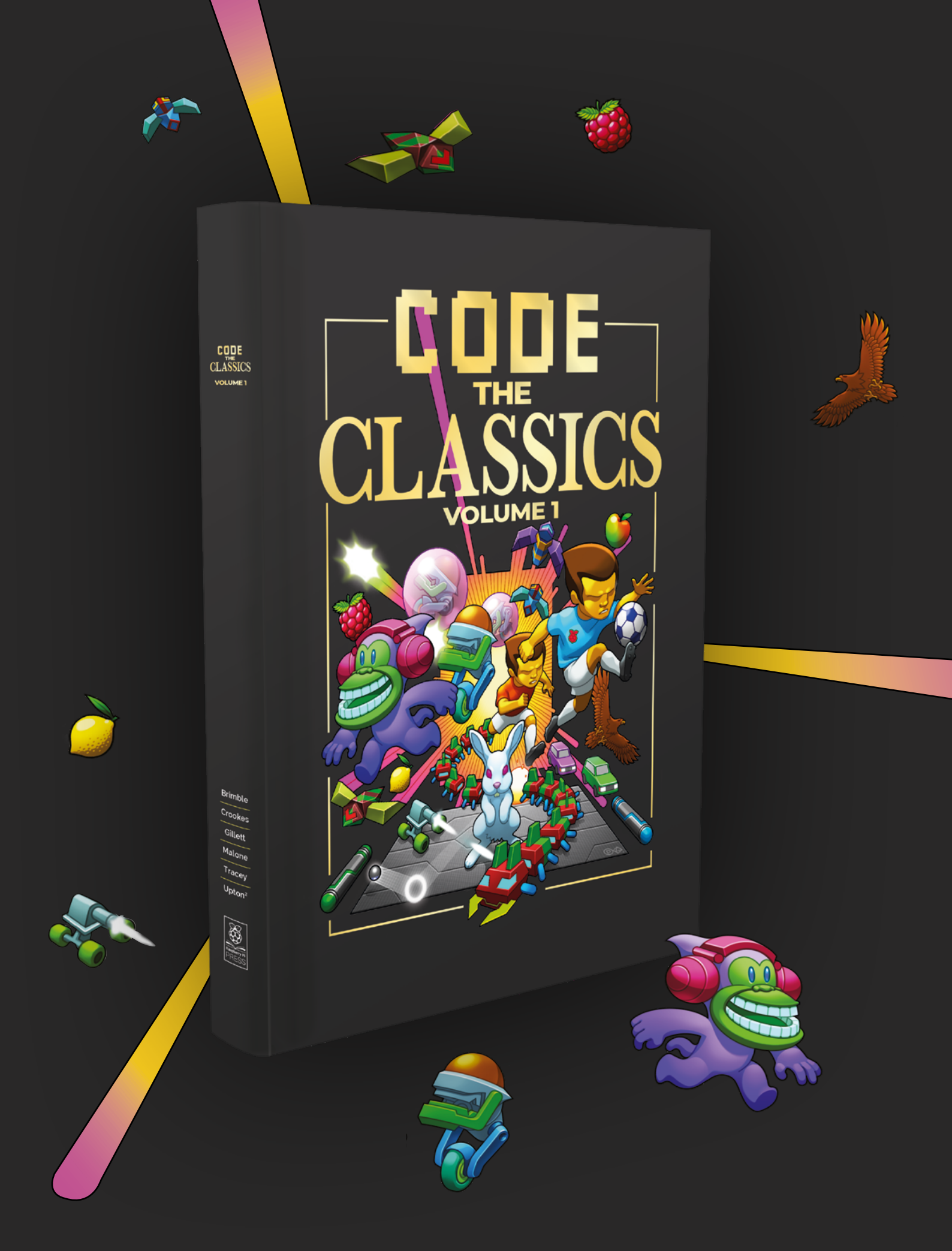
Estefannie

Web

magpi.cc/robotcostume

Project

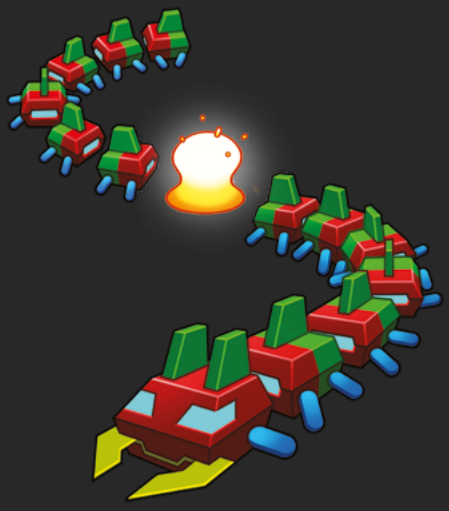
This excellent Halloween costume uses just about everything we've covered so far – lights, displays, buttons, and a moving mouth using servos. There's also a voice changer for some sound integration as well.


CODE
THE
CLASSICS
VOLUME 1
Brimble
Crookes
Gillett
Malone
Tracey
Upton²
CODE
THE
CLASSICS
VOLUME 1

CODE THE CLASSICS VOLUME 1

This stunning 224-page hardback book not only tells the stories of some of the seminal video games of the 1970s and 1980s, but shows you how to create your own games inspired by them using Python and Pygame Zero, following examples programmed by Raspberry Pi founder Eben Upton.

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



Available now

magpi.cc/codetheclassics

StackyPi

SPECS

CONNECTORS: 40-pin GPIO header, 6-pin debug header, micro-USB

STORAGE: 8MB on-board flash, plus microSD card slot

FEATURES: Boot and reset buttons, status LED, 4 × 12-bit ADC channels, PIO, I2C, SPI, UART

DIMENSIONS: 65 × 30 × 10 mm

Verdict

Not a plug-and-play solution for HATs, as you'll need to do some software tinkering, but still a neat little Pico-style board with some bonus features.

7/10

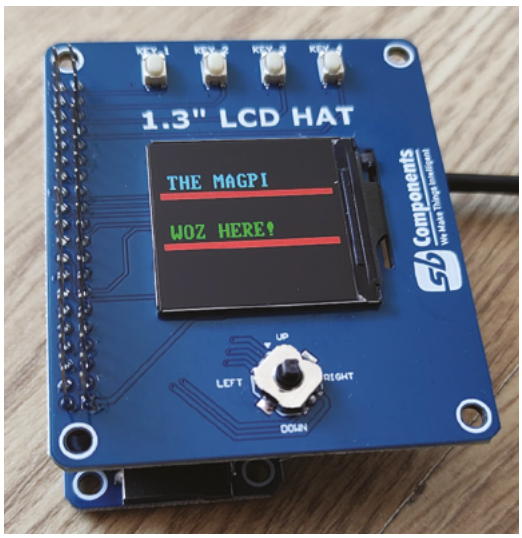
▶ SB Components ▶ **magpi.cc/stackypi** ▶ £14 / $18

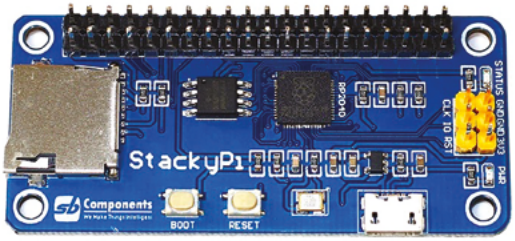
An RP2040 microcontroller compatible with standard HATs. By **Phil King**

Well is it a Raspberry Pi Zero? Is it a Pico?... No, it's StackyPi. While there are a few dedicated add-ons for Pico, if you're a Raspberry Pi fan then you probably have some HATs kicking around that you might like to use with it. Due to Pico's different arrangement of GPIO pins (in two separate rows), however, it's not possible to simply plug in a HAT.

Step forward StackyPi. Based on the same Raspberry Pi RP2040 microcontroller chip as Pico, it offers the same functionality, but with a pre-soldered 40-pin (2×20) GPIO header. It has a very similar form factor to a Raspberry Pi Zero, including four mounting holes. There's even a microSD card slot for extra storage, although StackyPi already has 8MB of on-board flash (compared to Pico's 2MB).

Other features include a standard micro-USB port (for 5V power or connecting to a computer), two tiny push-buttons (including a handy reset), an on-board status LED, and six debug pins.





▲ The size of a Raspberry Pi Zero, the StackyPi is an RP2040-based microcontroller with a 40-pin GPIO header

Plug and play?

That standard 40-pin GPIO header means you can connect any standard Raspberry Pi HAT, pHAT, or other compatible add-on board. The downside is that it's not a case of plug and play. You will need

> You will need to adapt any existing software for your HAT to make it work

to adapt any existing software for your HAT to make it work.

SB Components has adapted software for several of its own HATs for StackyPi, available in a GitHub repo (**magpi.cc/stackpigh**), which also features a handy GPIO pinout comparison chart – it's very similar to that on a Raspberry Pi, although some GPIO pin numbers are different.

As with Pico, you need to connect the board via micro-USB to a computer to flash the UF2 firmware and program it – using MicroPython, CircuitPython, or C++.

Why use this over a Raspberry Pi Zero? Well, it does offer far lower power consumption and four ADC channels. Alternatively, you could use a Raspberry Pi Pico with a Red Robotics Pico 2 Pi adapter for standard HATs.

HackSpace

TECHNOLOGY IN YOUR HANDS

THE MAGAZINE FOR THE MODERN MAKER

SUBSCRIBE AND SAVE UP TO 35% on the cover price

ISSUE #55

OUT NOW

hsmag.cc

Available on the App Store

GET IT ON Google Play

Kitronik Air Quality Datalogging Board for Pico

SPECS

SENSOR:
BME688 – temperature, humidity, pressure, gases

CONNECTORS:
Dual female header for Pico, 3 × ADC inputs, ZIP LED socket, servo output, 2 × high-power outputs, 5 V power input, plus extra solder pads

FEATURES:
128×64 monochrome OLED, 2 × push-buttons, piezo buzzer, 3 × ZIP (WS2812B) RGB LEDs, 3 × AA battery holder

DIMENSIONS:
74×72×27.1 mm

▲ The BME688 sensor is at the top left of the board; its readings can be shown on a small OLED in the middle

▶ The Pi Hut ▶ **magpi.cc/picoairquality** ▶ £39 / $42

Much more than an air quality monitor for your Pico. By **Phil King**

Air pollution is an ongoing issue in many parts of the world, so the Air Quality Datalogging Board for Pico may well come in very useful for monitoring purposes. A fair bit larger than a Raspberry Pi Pico (not supplied) which plugs into its dual female headers, this square board surprised us with just how many features have been crammed in.

For starters, there's a mini OLED screen (128×64 monochrome) to show sensor readings, plus two small push-buttons, a piezo buzzer, and three ZIP (WS2812B) RGB LEDs along one edge.

Then there's a host of input and output options, including three analogue inputs (connected to Pico's ADC channels) for connecting external sensors, a ZIP socket to add more LEDs, a three-pin servo connector, and two high-power outputs (screw terminals) with a max draw of 1A. In addition, there are solder pads for 3V3 power, GND, digital pins, SPI, and UART.

Power-wise, there's a handy switch in one corner and a 3 × AA battery holder on the rear. There's even a 5 V input if you want to charge up the batteries with a solar panel.

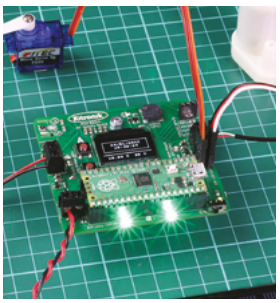
What a gas

The key feature of the board is its BME688 sensor, located in one corner. As well as measuring temperature, humidity, and atmospheric pressure (as on a BME680), it features an AI-enhanced gas sensor that detects volatile compounds and gases to determine IAQ (Index of Air Quality) and estimated CO_2 levels. It takes five minutes to calculate the baseline for the gas sensor when first used, but this a one-off process and ensures greater accuracy for future readings.

All in all, this is a well-thought-out board backed up by some top-notch documentation and software. Multiple online tutorials cover its various functions

> A well-thought-out board backed up by some top-notch documentation and software

in detail, including data logging and analogue input/output control. The GitHub repo includes a comprehensive MicroPython library for the board and numerous code examples to try out.



▲ So many inputs and outputs! Hook it up to a servo, water pump, heating pad, or even a solar panel to recharge the batteries

Verdict

A fully-featured Pico add-on with plentiful input/output options, offering a host of possibilities for projects. Excellent documentation too.

9/10

English not your mother tongue?

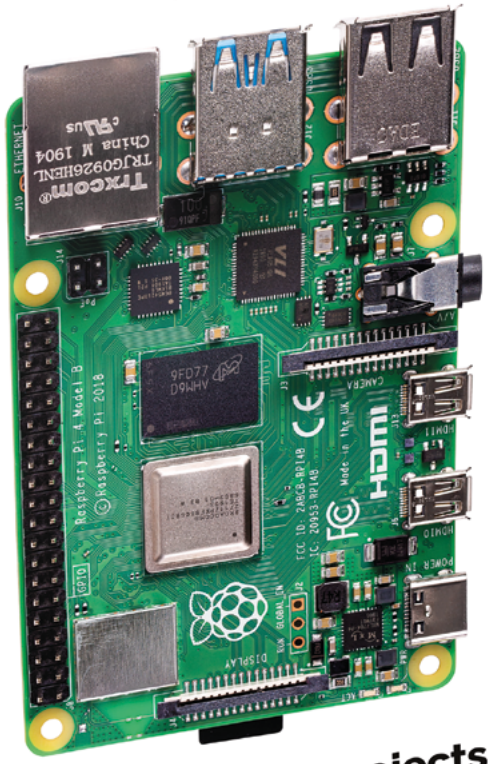
The MagPi is also available in German!

Subscribe to the German edition of The MagPi and get a Raspberry Pi Pico with headers and a cool welcome box **FOR FREE!**

Use the coupon code 115PicoDE

on www.magpi.de/115

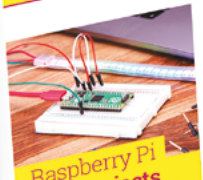
THE Official RASPBERRY PI HANDBOOK 2022



50 Raspberry Pi **Hacks & Hints**

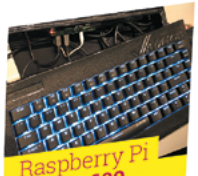
iPod Classic **Spotify Player**

BeccaCam **Digital Camera**



Raspberry Pi **Pico projects**

Build a **Home Assistant**



Raspberry Pi **Amiga 600**

Walking **Bipedal Robot**

Start incredible projects with Raspberry Pi

FROM THE MAKERS OF The MagPi THE OFFICIAL RASPBERRY PI MAGAZINE

THE OFFICIAL RASPBERRY PI HANDBOOK 2022

10 Amazing:

Facial recognition projects

Make faces at your Raspberry Pi with these incredible builds

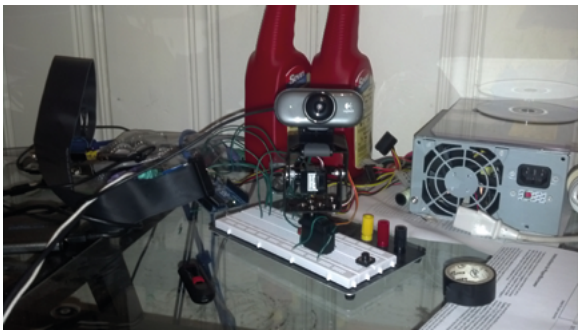
With software like OpenCV, it's easy to create Raspberry Pi projects that can both recognise and track faces from a camera. Here are some of our favourite uses of this kind of smart vision.

▲ Archimedes

Handsome owl

Using the AIY Vision kit, this robo-owl looks around for happy people and allow you to take a sticker. Very cute.

magpi.cc/archimedes



▲ Pan / tilt face tracking

Simple movement

A simple way to use face tracking – literally just track the face with a camera. This can be useful for robot projects or motion tracks, or even CCTV.

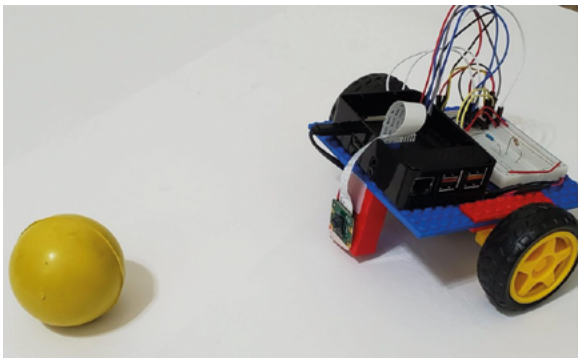
magpi.cc/pantilt

▶ EmpathyBot

Emotional robot

This Dexter Industries project will respond differently to different emotions – a fun little robot that hopefully will not be used during the inevitable automaton uprising.

magpi.cc/empathybot

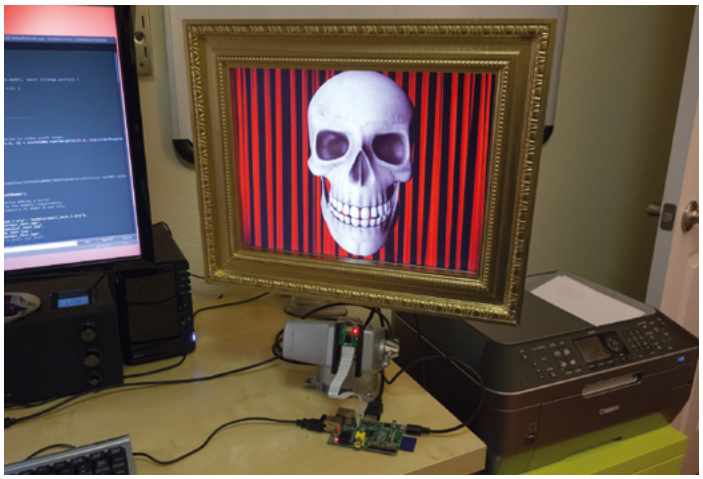


▲ Object-tracking robot

Follow the ball

Getting robots to follow specific things is a great way to implement automation tasks – it's also how robot racing league Formula Pi used to work.

magpi.cc/objecttrack



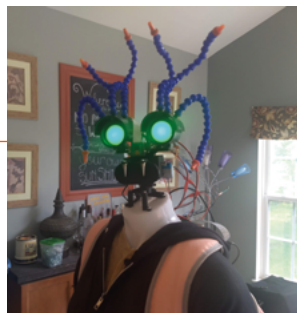
Creepy face-tracking portrait

Halloween trick

This classic illusion has been updated for the digital age by having the face actually move as you do, albeit very subtly. Next step, VTubers.

magpi.cc/creepyportrait

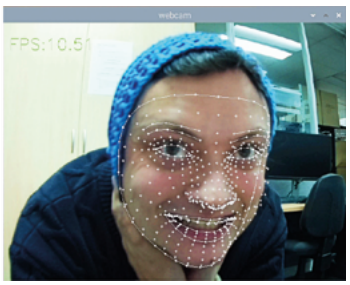
Face-tracking robot mannequin head



Nightmare fuel

A pan-and-tilt style project, although this time attached to a robot that acts as an alarm system – not a very good one though.

magpi.cc/robothead

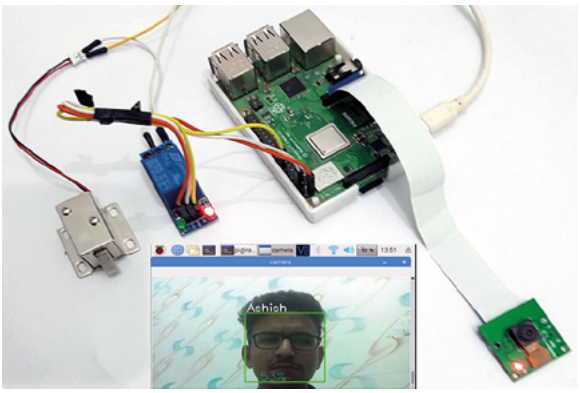


Face landmark tracking

Expression tracking

Using the same tech as those creepy animated memoji things, and less creepy cartoon VTubers, this is a cool and powerful use of Raspberry Pi face tracking.

magpi.cc/landmarktrack



Face recognition door lock system

Stare to enter

Using similar facial recognition tech as before, this one is connected to an actuator that will unlock a door. It will also get better the more it's used.

magpi.cc/facelock

Face mask detector

Don't spread it

This clever facial recognition model knows if you're wearing a face mask or not. It's inspired by a shopping centre in Thailand that would only let you in while wearing a mask.

magpi.cc/facemask

Face recognition

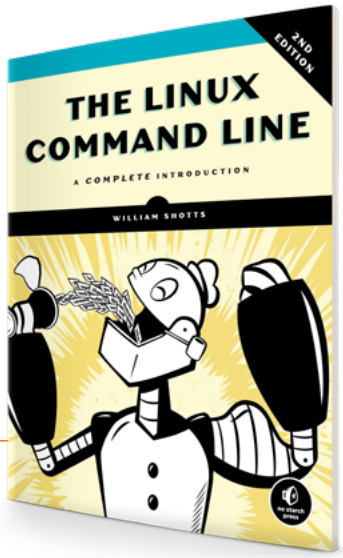
Who's there?

With the right training model, it's fairly easy to teach a Raspberry Pi to recognise people and objects. With this, you can get started with that.

magpi.cc/recognition

Learn Terminal with Raspberry Pi

Unleash the power of the Linux command line with these resources. By **Phil King**



The Linux Command Line

AUTHOR

William Shotts

Price:
£34 / $40 (Free PDF version)

magpi.cc/linuxclbook

Like most Linux systems, Raspberry Pi OS has a user-friendly graphical user interface, but opening up a terminal window enables you get under the hood. Its command-line interface can often prove quicker and more efficient than a GUI – once you get to know it and learn a few useful commands.

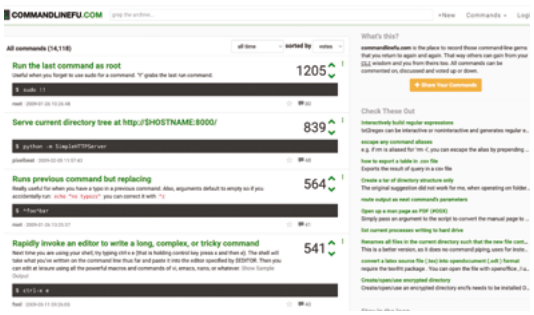
Over 500+ pages, *The Linux Command Line* takes you from entering your first command right through to writing programs in Bash shell script. Part one covers topics such as navigating the file system, manipulating files and directories, advanced keyboard tricks, permissions, and processes. The second part introduces text editors, with a detailed section on vi, and prompt customisation.

Part three covers common tasks and essential tools, including package management and regular expressions. Finally, part four is all about writing shell scripts.

The book is also available as a free PDF download from **linuxcommand.org**, where there's also a stripped-back online guide to get you started.

Websites

Useful web resources for the command line



COMMAND LINE FU

This searchable repository of terminal commands is a goldmine of information. Users share their most useful commands, as can you. It can also be used to store your personal favourites.

▶ commandlinefu.com

RASPBERRY PI DOCUMENTATION

This section of the Raspberry Pi online documentation gives an overview of the terminal and the basics of how to use it on Raspberry Pi, including a selection of useful commands.

▶ magpi.cc/terminaldocs

BASH REFERENCE MANUAL

Bash is the standard shell, or command language interpreter, used in Linux. This comprehensive resource covers a wide range of features and functions beyond the entry-level commands.

▶ magpi.cc/bashref

Learn the Command Line

AUTHOR

Codecademy

Price:
£32 / $40 per month
(Free trial)

magpi.cc/codecadamycl

If you just want to quickly learn the basics, this course is great – to get started, you'll just need to sign up to Codecademy and start a free seven-day Pro trial (just remember to cancel it later). It features an interactive virtual terminal to enter commands and navigate a system – not quite the same nor as satisfying as using a Linux system terminal directly, but fine for learning purposes.

The first section takes you through navigating the file system. Next comes manipulating files (copy, move, remove), then redirecting the input and output from commands and programs (pipe, cat, grep). The final part, 'configuring the environment', covers aliases, environment variables, the Bash profile, history, and exporting. Cheat sheets and quizzes reinforce the learning.

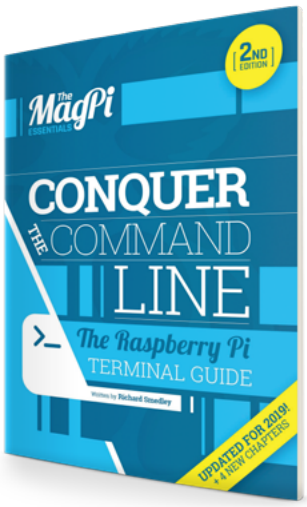
Conquer the Command Line

AUTHOR

Richard Smedley

Price:
Free

magpi.cc/clibook



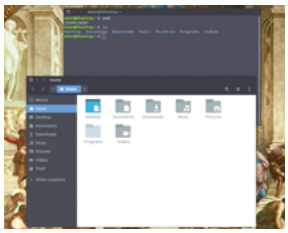
It's Raspberry Pi's own guide to the command line! While the print edition is now sold out, you can still download the PDF version for free.

Naturally, it's aimed at Raspberry Pi users and helps you to unlock the power of the command line. While the current edition was written for an earlier version of Raspbian/Raspberry Pi OS, and some of the directories have been changed (e.g. there's no MagPi directory in /home/pi any more), you could always substitute them with others.

Beginning with navigating the file system, it moves on to editing files, installing packages with APT, and manipulating text. Other topics include creating users, connecting removable storage, networking, stopping a process, remote access with SSH, compiling software, and even browsing the web from the command line.

Online courses

Study the terminal online with these courses



LEARN THE LINUX COMMAND LINE: BASIC COMMANDS

Comprising 17 video lectures, this free Udemy course will help you to become familiar with many essential commands. It even shows you how to create your own custom commands.

▶ magpi.cc/udemyclbasics

BASH FOR PROGRAMMERS

Accessible on a free seven-day trial, this comprehensive course from Educative features 49 lessons, eight quizzes, and 66 playgrounds for interactive hands-on practice.

▶ magpi.cc/educativebash

AUTOMATION SCRIPTS USING BASH

Learn how to write Bash scripts to automate long and tedious tasks on the command line in this two-hour project-based course featuring a virtual interactive terminal.

▶ magpi.cc/courseraautobash

Kevin McAleer

A master of robotics, Kevin has built many kinds and can also teach you how to make your own, all with Raspberry Pi

- Name Kevin McAleer | Occupation Freelance Project Manager
- Community role Maker and educator | URL magpi.cc/kevinmcaleer

One of our most passionate readers is Kevin. He always has something to share with us on #MagPiMonday, and notices little bits and pieces we put in the magazine, including accidental patterns in the colour of the spine. He's also an excellent maker and teacher of robotics on YouTube.

"I've always been into making things as far back as I remember," Kevin tells us. "From making things with cereal boxes (the best source of robust and available cardboard) to creating robots and futuristic worlds in Lego, at age eight! There was probably a gap of about 30 years before I got back into making and practical computing; I bought an Arduino from Maplins and made some LEDs blink. However, I've always been passionate about computing and making things, so building robots is the ultimate Venn diagram, with my happy place right in the middle."

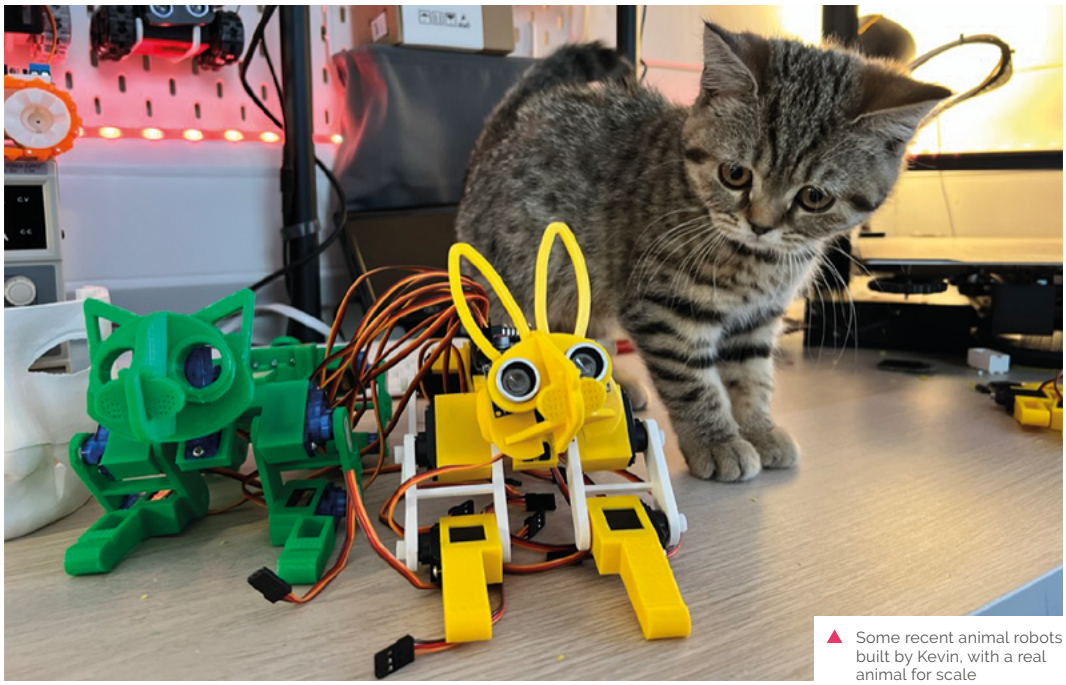
Kevin getting his original Raspberry Pi signed by Eben during the tenth anniversary

What is your history with making?

When I was growing up, our house always had electronics and disassembled TVs and things as my father was a TV repair engineer. He had a lab full of repair equipment such as an oscilloscope, spectrum analyser, and a soldering station. So, understanding how electronic things work and how to repair them when they don't is part of my DNA. I specifically remember reading a book on superheterodyne receiver theory from his room during high school and following along with it.

When did you first learn about Raspberry Pi?

Early 2012 – I checked my inbox and can see that I ordered my first Raspberry Pi back in June 2012 from RS Components!



Some recent animal robots built by Kevin, with a real animal for scale

I remember my uncle asking if I'd heard about this credit card-sized computer that could run Linux and was around £20 (£21.60 – I just checked!). Having recently bought the 8-bit Arduino Uno for more than that, this seemed too good to be true, so I placed my order. That first Raspberry Pi has now been signed by Eben Upton. Eben kindly autographed it at the 10th Anniversary Raspberry Pi exhibit launch early this year at The National Museum of Computing.

> My first robot was a 3D-printed SMARS robot designed by Kevin Thomas, an engineering student

What was your first robot project?

My first robot was a 3D-printed SMARS robot designed by Kevin Thomas, an engineering student. (Check out my fansite for this little robotic marvel – **smarsfan.com**.) It uses an Arduino Uno, a motor shield to connect up the two N20 motors, a 9V battery for power, and uses an ultrasonic range-finder to detect objects in front of it.

This is an excellent robot for getting started with robotics; it doesn't take too long to 3D-print out all the parts and is easy to get up and running with some simple code.

What is your favourite project?

My favourite project is a Python-based AI assistant I created for Raspberry Pi. The software is now embodied within the robot I call Isaaca (named after Isaac Asimov, my favourite robotics author). I created a couple of videos about this project on YouTube, and it's also the most popular one I've made to date. The AI assistant can listen to spoken commands and perform a range of actions. This includes telling jokes, telling you the weather forecast, adding things to a to-do list, and adding items to a calendar (that you can sync to your phone).

I liked this project the most because, at first, it was a stretch for me to write. I was then able to simplify the code to the extent I could teach others how to build it, too – it's much easier than you'd think, and I've shared the code on GitHub. I share all my code on GitHub; I'm a huge advocate of open-source projects.

A cluster computer built from Raspberry Pi Zero and based on the Cray-2

Find Kevin online!

YouTube
magpi.cc/kevinmcaleer

Website
kevsrobots.com

Twitter
@kevsmac

Instagram
@kevinmcaleer28

TikTok
@kevinmcaleer6

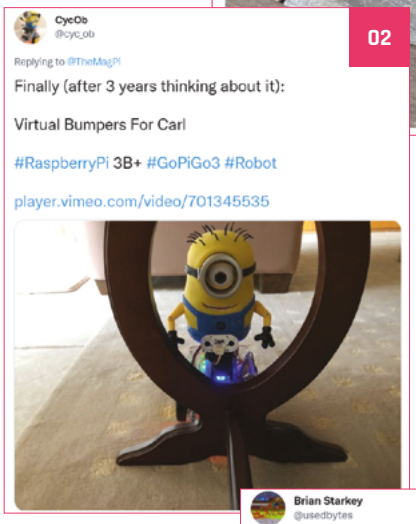
You can also find him on Pimoroni's website doing product intro videos.

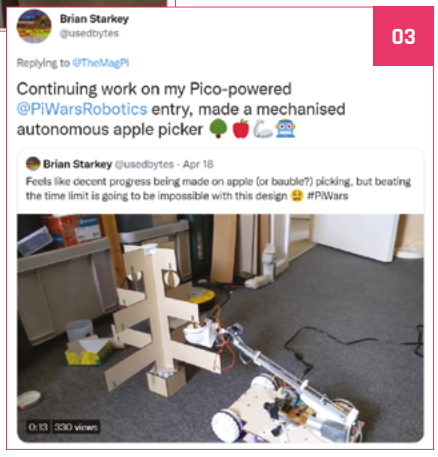
MagPi Monday

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 remember to follow along at the hashtag #MagPiMonday!!





01. This very cute camera build is also a fun bit of upcycling

02. Carl can now move around without hurting themselves, only furniture

03. Slow or not, we're really impressed with this apple-picking robot design

04. People seem to be making cool gaming things with Pico this month

05. Quite a simple looking robot controlled over serial is quite cool

06. A good use of automation and LEDs: model railways

07. This is exceptionally useful for those wanting to control NeoPixels remotely from Raspberry Pi

08. This is an exceptionally compact project for a great display

09. A fairly simple board for a very usable MIDI controller

10. Running BASIC on Raspberry Pi Pico is no mean feat!

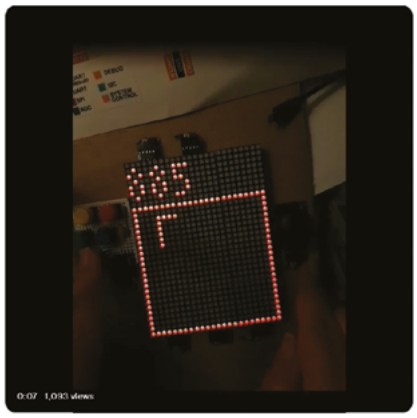
11. This is an incredible achievement! And the graphics aren't half bad either

04

Pierre Poliakoff
@ON4LPF

Replying to @TheMagPi

#picovision : a cardboard retro like gaming console base on #raspberry pi pico

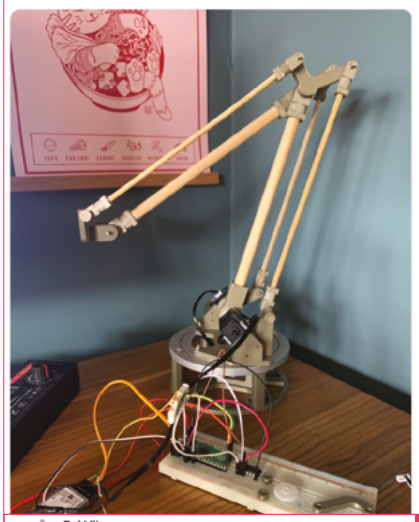


05

Pete
@mnstrmnch

Replying to @TheMagPi

A robot arm with a Pi Pico controlling @sb_ltd serial servos



06

Andy Proctor
@ProctorAndy

Replying to @TheMagPi

Still in development but I'm using a @Raspberry_Pi to run the points and lights on my sons @hornby #railway #OOGauge

07

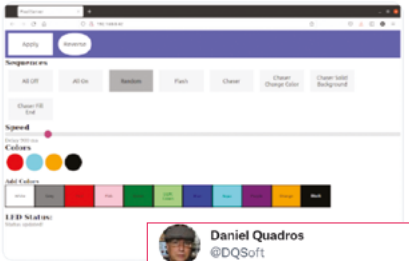
Stewart Watkiss
@stewartwatkiss

Replying to @TheMagPi

I've been adding new features to my pixel server. A Raspberry Pi based web server for controlling NeoPixels.

Only small changes today, ready for a new related project coming soon.

penguintutor.com/projects/pixel...

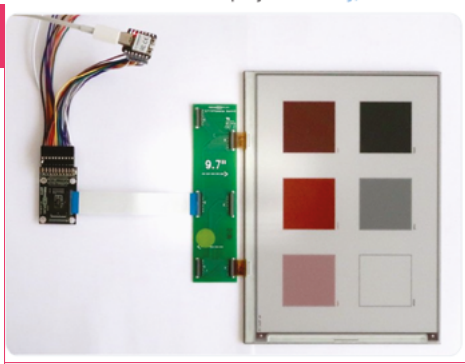


08

Rei Vilo
@rei_vilo

Replying to @TheMagPi

Yes, a large e-paper display driven by a compact RP2040-based board! Full project at bit.ly/3rJYshe



09

Leonella Rodríguez Dotti
@leonedott

Replying to @TheMagPi

twitter.com/leonedott/stat...

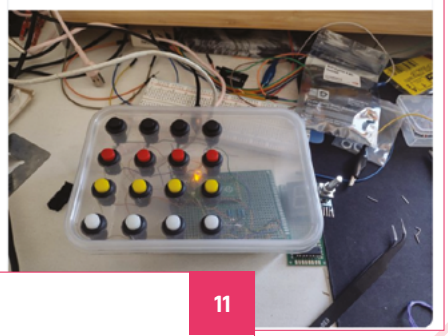
This MIDI controller

Translate Tweet

Leonella Rodríguez Dotti @leonedott · Apr 24
"Terminé" este proyecto que tenía un año parado. Es un controlador MIDI.

Pongo que terminé entre comillas porque quiero agregarle más funciones, pero si me pongo a pensar en eso desde ahorita, nunca lo voy a terminar.

Show this thread



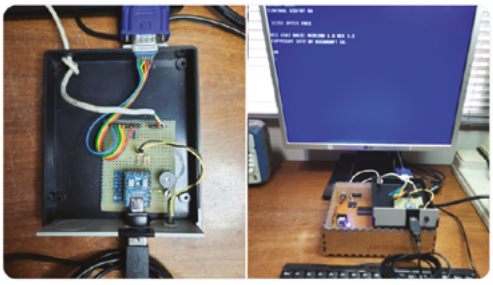
10

Daniel Quadros
@DQSoft

Replying to @TheMagPi

RP2040 based serial terminal

Translate Tweet



5:43 PM · Apr 19, 2022 · Twitter for Android

11

Harishankar
@codetiger

Replying to @TheMagPi

Harishankar @codetiger · Apr 26
Finally completed building a 2d platform #gameengine close enough to run NinjaFrog Game on RPi Pico based GameConsole. Runs at 40FPS on 320x240 LCD. Thanks to #RP2040's DMA
@PixelFrogStudio @Raspberry_Pi @waveshare00 @pimoroni @robulabs
github.com/codetiger/Game...

RaspberryPint
@RaspberryPint

12

Replying to @TheMagPi

I got my LU tube map clock working. It displays real time updates on the status of each line (good service, minor delays, closures, suspensions, etc. changes the LED flash pattern). Cat for scale. 5cm e-paper screen displays time.

Rob Miles
@robmiles

13

Replying to @TheMagPi

Raspberry Pi Chocolate Synthbox now running and talking to the PICO MIDI Cheesbox and Crackers controller.....

Kevin Just
@justKevin87

14

Replying to @TheMagPi

Finally printed a stand for my pico clock including a pico LiPo, a RTC, and a unicorn pack from @pimoroni

not the final design yet, but at least it stands now

JamHamster
@RealJamHamster

15

Replying to @TheMagPi

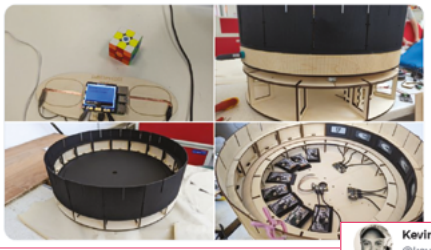
There's this, but it might take a while to finish.

Brian Corteil
@CannonFodder

16

Replying to @TheMagPi

Once again been working on my Zoetrope Pico & Raspberry Pi project, but mainly been helping Tom, my lad on his #CoolestProject entry, a timer for speedcubing.
#MagPiMonday



Kevin McAleer
@kevsmac

17

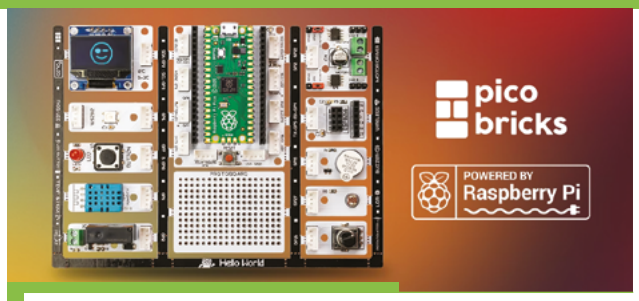
Replying to @TheMagPi

#MagPiMonday I made a 3d printed mecanum robot using an RP 2040 powered board. Looks kinda a cute don't you think?! youtu.be/JsYGHDRF-VQ #robotics #robot #robots #STEM #micropython #3Dprinting #Fusion360

12. We love the mixture of 1960s futurist tech with the map, and modern e paper screen for tracking times
13. Another MIDI box or two, and a controller in this case? We like the naming convention
14. A very simple stand that does the job perfectly
15. This compact, portable looking TV console build reminds us a bit of the Nintendo GameCube
16. One of Brian's first Raspberry Pi projects was a zoetrope, and this new Pico version looks incredible
17. This 3D-printed robot is very cute!

Crowdfund **this!**

Raspberry Pi projects you can crowdfund this month



PicoBricks

"Pico Bricks is an electronic development board + software which is designed for use in maker projects. With ten detachable modules included, Pico Bricks can be used to create a wide variety of projects. It also includes a protoboard that you can use to add your own modules."

▶ **kck.st/3M9Q036**

SQ1 case

"SQ1 is a Raspberry Pi case you want to put right beside your headamp, DAC, and media centre. The elegant design is meant to be seen but not be distracting. Great for Raspberry Pi aficionados and audiophiles."

▶ **kck.st/3MGQHRl**

RadSense

"The WeatherSense RadSense is built around a Geiger counter that counts the number of energetic particles hitting the gas inside the tube. Software on the Grove Mini Pro Plus board reads these counts and produces a CPM count. The computer then encodes the data and sends it via a 433MHz radio signal (no radio license required!) to your Raspberry Pi."

▶ **kck.st/3LG5USK**

Your Letters

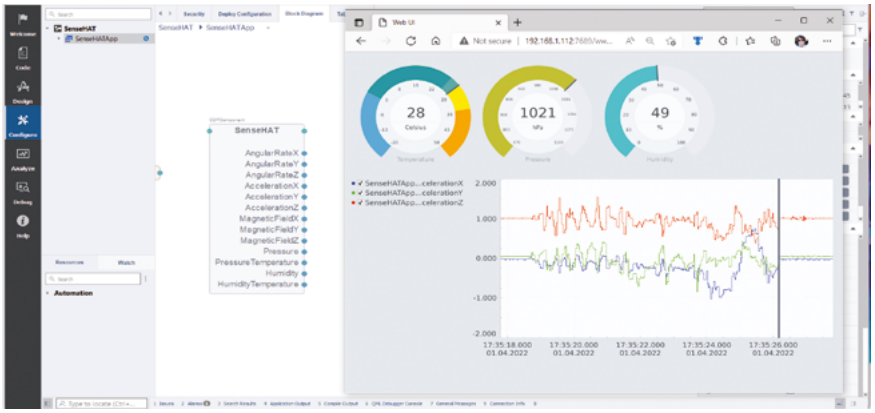
Parts unknown

I noticed in issue 117 the article CDP Studio – apparently this is part two but, for the life of me, I cannot find which issue part one was in? I've spent about 30 mins looking at the contents of each issue on the website going back about a year but can't find it… can you give me a clue as to which issue it's in?

Dave via email

The first part was in issue 116 on page 56 – in the contents it's labelled as 'LED patterns with web GUI', but in the tutorial it is called part one of a CDP Studio series. We're sorry this ended up being confusing! That particular series is complete, and you can finish the whole thing now.

▼ Here's what you can make with the two-part CDP Studio series



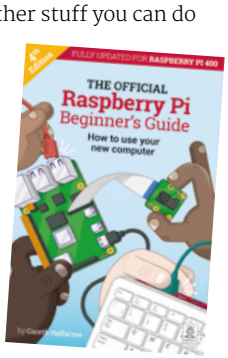
Model B query

I recently purchased a Raspberry Pi 4 Model B to show announcement slides on a flat-screen TV in the foyer of our church. I was curious about how to get it started fresh from the box? Also, I was wondering if Raspberry Pi was unnecessarily sophisticated for my project, and wanted to know if you thought a simple thumb drive, with ten slides downloaded from PowerPoint, would be able to loop on a smart TV without the tech of Raspberry Pi?

David via email

It is exceptionally easy to get a Raspberry Pi working out the box – you can either check out this link (**magpi.cc/imager**) on how to install the operating system, or grab the *Raspberry Pi Beginner's Guide* as a book or as a free PDF here: **magpi.cc/beginnersguide**.

The book will guide you through initial setup, along with other stuff you can do with Raspberry Pi.



As far as we're aware, smart TVs don't generally work with PowerPoint presentations, so Raspberry Pi is probably the best thing to use in this instance!

▲ The contributions page allows for several payment options – and also lets you just skip it too

PDF over paper

While I don't want to pay for the paper edition of *The MagPi*, I'd gladly make a contribution for the free PDF version, but I don't see any possibility on your site (I'm probably not looking at the right place).

Mehren via email

When you click on 'Download free PDF' from our website and on the back issues page (**magpi.cc/issues**), it will come up with a screen asking if you'd like to make a contribution to the magazine. If you've done this before and want to do it again, you may need to clear your browser's cache for our website. You can also make a recurring contribution if you wish. You can also just continue on and get the PDF for free.

Contact us!

- Twitter **@TheMagPi**
- Facebook **magpi.cc/facebook**
- Email **magpi@raspberrypi.com**
- Online **forums.raspberrypi.com**

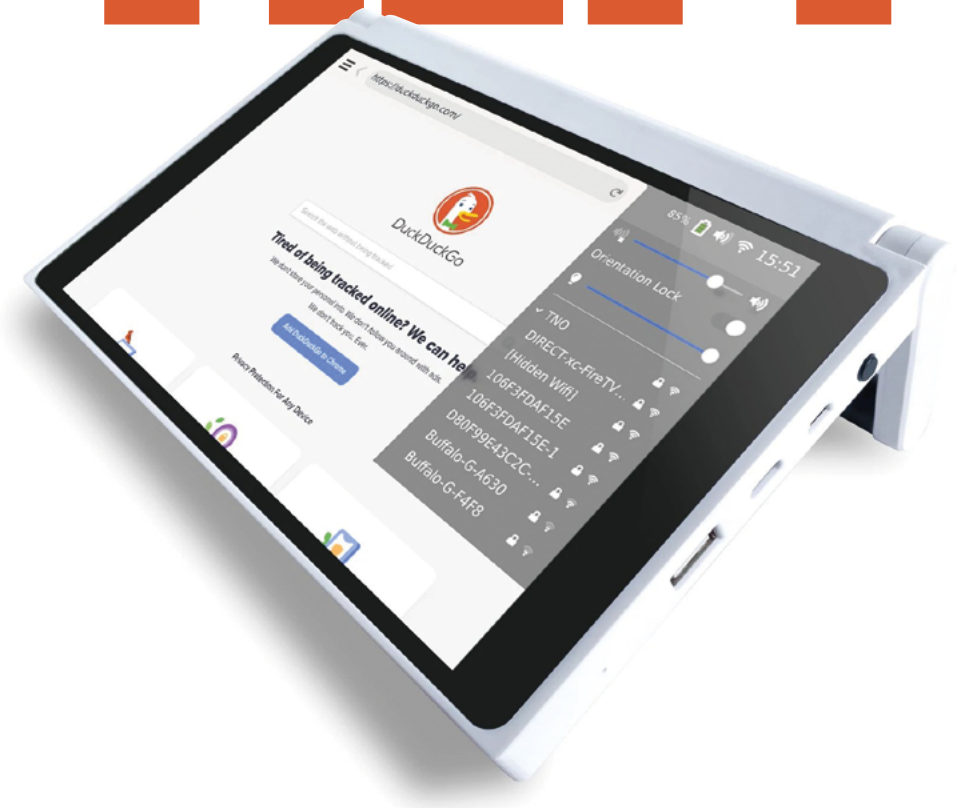
Wireframe

Join us as we lift the lid on video games

Visit **wfmag.cc** to learn more

WIN A CUTIEPI

We reviewed this excellent Compute Module-powered tablet last issue. It's powerful, very user-friendly, and even has a neat handle that can be used to prop it up. Now is your chance to win one.



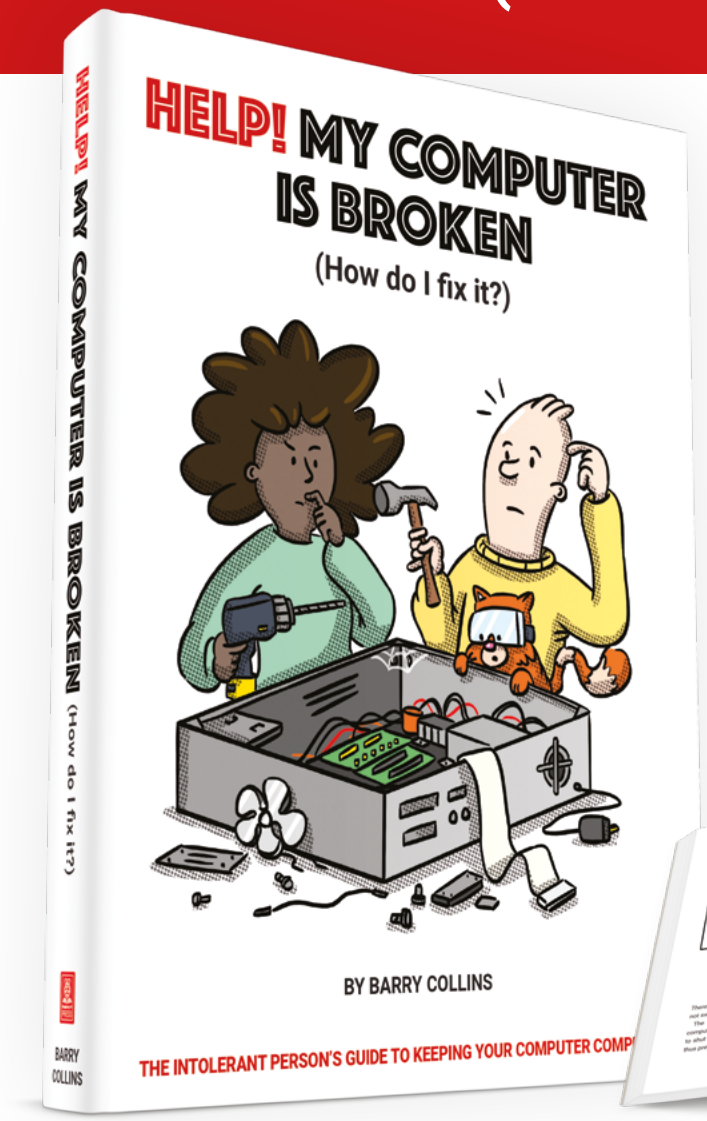
Head here to enter: magpi.cc/win | Learn more: cutiepi.io

Terms & Conditions

Competition opens on **25 May 2022** and closes on **30 June 2022**. Prize is offered to participants worldwide aged 13 or over, except employees of Raspberry Pi Ltd, the prize supplier, their families, or friends. Winners will be notified by email no more than 30 days after the competition closes. By entering the competition, the winner consents to any publicity generated from the competition, in print and online. Participants agree to receive occasional newsletters from 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.

HELP! MY COMPUTER IS BROKEN

(How do I fix it?)



Help! My Computer Is Broken takes the most common computer problems and tells you how to fix them. It's as simple as that! If you've ever wondered why your laptop won't turn on, you can't get a WiFi connection, your printer isn't printing, or why everything is so slow – well, this is your book...

BUY ONLINE: magpi.cc/helpbook

REMARKABLE ROBOTS!

THE MAGPI #119
ON SALE 30 JUNE

Plus!

Reuse an old Raspberry Pi

Make KITT real with AI

Building your ArtEvolver

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

Illustrator
Sam Alder

CONTRIBUTORS

David Crookes, PJ Evans, Rosemary Hattersley, Edwin Jones, Nicola King, Phil King, Sean McManus, Nik Rawlinson, Stephen Smith

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.

ISSN: 2051-9982.

Everybody needs a **hobby(ist)**

The Raspberry Pi community makes it all worthwhile. By **Lucy Hattersley**

One of the big talking points at the moment is Raspberry Pi for hobbyists and Raspberry Pi for industrial experts.

Of course, these two things are rarely separate. My father-in-law, Richard, is a practical sort that likes to have all kinds of projects around the house. Sure, they may not be all finished yet, but they are interesting.

He's built an incredible garage with a dual-action sliding door, and renovated an old wooden canoe with his daughter Rosie, who is insistent we will eventually bring it down to our terraced house in London. There's no space; unless we put it on the roof!

He has a stone dragon gargoyle on his roof and a train layout, for which he built several landmark buildings from scratch.

In his younger days, he inspected and reported on nuclear power stations and MOD nuclear installations, a very important job that required extreme focus to keep his nation safe. These power stations are handy, and I suspect we could use a few more of those at the moment.

He is also one of the few people I know who will listen to me when I start talking about NAND gates and Z80 and ARM Assembly and Raspberry Pi. He hasn't the faintest idea what I'm talking about. Which often makes two of us. But he appreciates a detailed-orientated chat, especially if I include some historical connection to the pre-digital era that he can more easily connect with.

I suspect a few years younger and he'd have been into computing and Raspberry Pi. As it stands, his hobby is steam trains. So much so, he rescued a long-disused Belgian Cockerill Steam Tram with a fellow rail enthusiast (**magpi.cc/cockerill)**, and brought it over to the UK by sea and road on the back of a lorry for renovation.

> "Practical people do practical things"

It's currently powering up and down the Mid-Suffolk Light Railway teaching a younger generation about the steam age. We showed Richard a recent YouTube video of it in action on Easter Weekend (**magpi.cc/mslrYT**).

Practical people do practical things.

Full steam ahead

Somebody who is wholly into a hobby watches the business side of things. Steam train fans are railway fans, and the industrial and consumer network is something they know an awful lot about. I chat chips; he chats coal.

The big tech-chat at the moment is the supply chain. Last issue we mentioned a website called RPI Locator (**magpi.cc/rpilocator)** that provides alerts for when stores have Raspberry Pi in stock. We're happy to see it joined by Hardware Locator (**magpi.cc/hwlocator**).

It's something of a relief to note that Raspberry Pi isn't the only company with more customers than it can supply. On the other hand, it's disheartening to hear of our readers' struggle to pick up, for example, Coral USB Accelerator kits to go with their new Raspberry Pi computers.

I hope this situation picks up soon. In the meantime, you can get a Raspberry Pi by signing up with RPI Locator for an alert. Or, of course, by subscribing to *The MagPi* magazine (**magpi.cc/subscribe**).

We will continue to make the finest magazine for the best hobbyist computer around. The hobbyists make Raspberry Pi what it is. More than just a useful tool, but a loved computer. Long may they tinker.

AUTHOR

Lucy Hattersley

Lucy is back in "that there" London after a brief stint visiting family in the North. She's torn between home and hearth.

magpi.cc

HiPi.io

PiKVM

Manage your servers or workstations remotely

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

PiKVM HAT
for DIY and custom projects

Pre-Assembled version

- Real-time clock with rechargeable super capacitor
- OLED Display
- Bootable virtual CD-ROM & flash drive
- Serial console
- Open-source API & integration
- Open-source software

Available at the main Raspberry Pi resellers

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

Pi-Shop.ch · ThePiHut · PiShop.us · CloudFree · PiShop.mx

Reseller suggestions and inquiries:
wholesale@hipi.io