MAKE | BUILD | HACK | CREATE

HackSpace

TECHNOLOGY IN YOUR HANDS

hsmag.CC | February 2024 | Issue #75

DIY Gadgets

Build your own phone, camera, watch and more

Full control
Print exactly what you want to print

Fractal art
Enhance your living space with maths

Dinosaurs
3D scanning at the Natural History Museum

Feb. 2024
Issue #75 £6
9 772515 514006 75

ROBOT RUGBY **COFFEE** RUBBER **MUSIC**

Free eBook!

Download your copy from

hsmag.cc/freecadbook

Welcome to HackSpace magazine

As someone who grew up watching *Inspector Gadget*, the ability to create some wacky portable devices was a big part of the reason I grew up loving to build things. To be honest, not much has changed. Gadgets exist in the space between utility, geekery, and fashion – a really good gadget has to have a bit of all three. And our skills as makers can help in each of these areas. We can personalise it to our particular needs, we can nerd-out as much as we like on the technicals, and we can give it our own unique aesthetic.

Join us as we take a look at makers building their own doodads and whatsits. Go go gadget … well, whatever.

BEN EVERARD
Editor ben.everard@raspberrypi.com

PAGE 26

EDITORIAL
Editor
Ben Everard
ben.everard@raspberrypi.com

Features Editor
Andrew Gregory
andrew.gregory@raspberrypi.com

Sub-Editors
David Higgs, Nicola King

DESIGN
Critical Media and Raspberry Pi
criticalmedia.co.uk

Head of Design
Lee Allen

Designers
Sam Ribbits, Sara Parodi, Jack Willis

Photography
Brian O'Halloran

CONTRIBUTORS
Marc de Vinck, Rosie Hattersley, Jo Hinchliffe, Andrew Lewis, Rob Miles, Karl Mose

PUBLISHING
Publishing Director
Brian Jepson
brian.jepson@raspberrypi.com

Advertising
Charlie Milligan
charlotte.milligan@raspberrypi.com

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

To subscribe
01293 312189
hsmag.cc/subscribe

Subscription queries
hackspace@subscriptionhelpline.co.uk

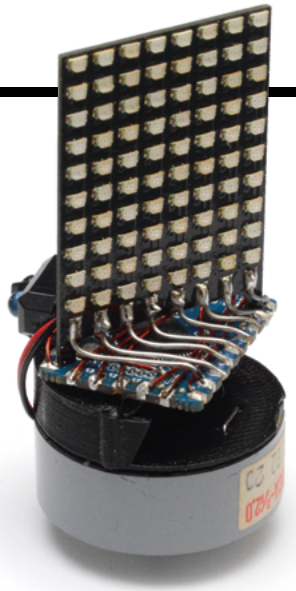
recycle

CC BY NC SA

This magazine is printed on paper sourced from sustainable forests. The printer operates an environmental management system which has been assessed as conforming to ISO 14001.

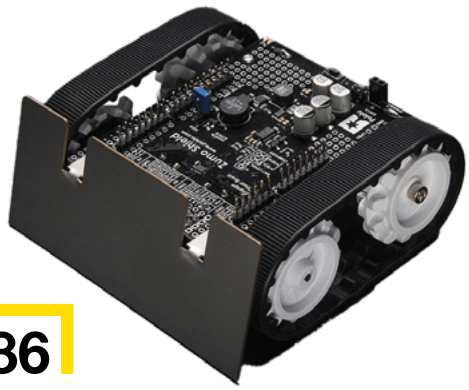
 ISSN: 2515-5148.

Contents



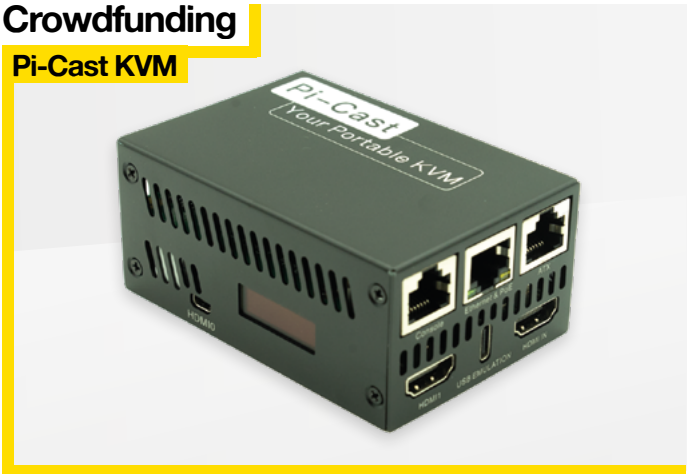
Interview

VEEB Projects



Crowdfunding

Pi-Cast KVM



47 FORGE

85 FIELD TEST

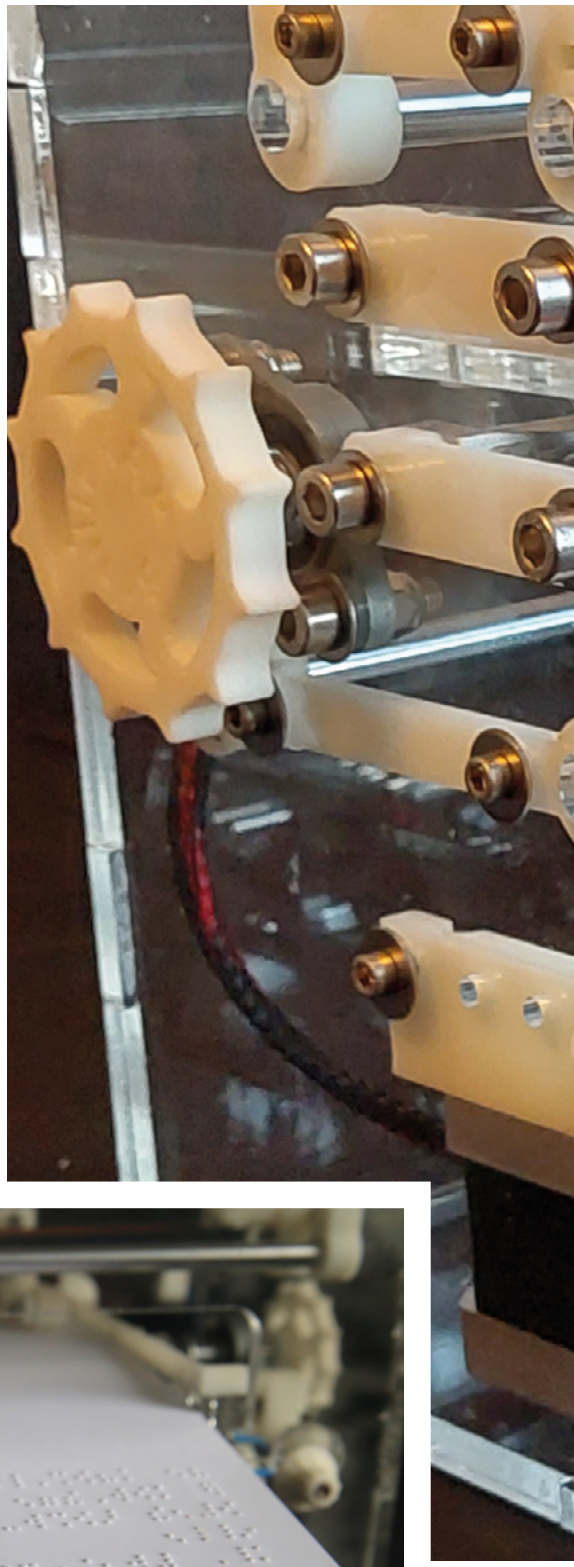
BrailleRAP

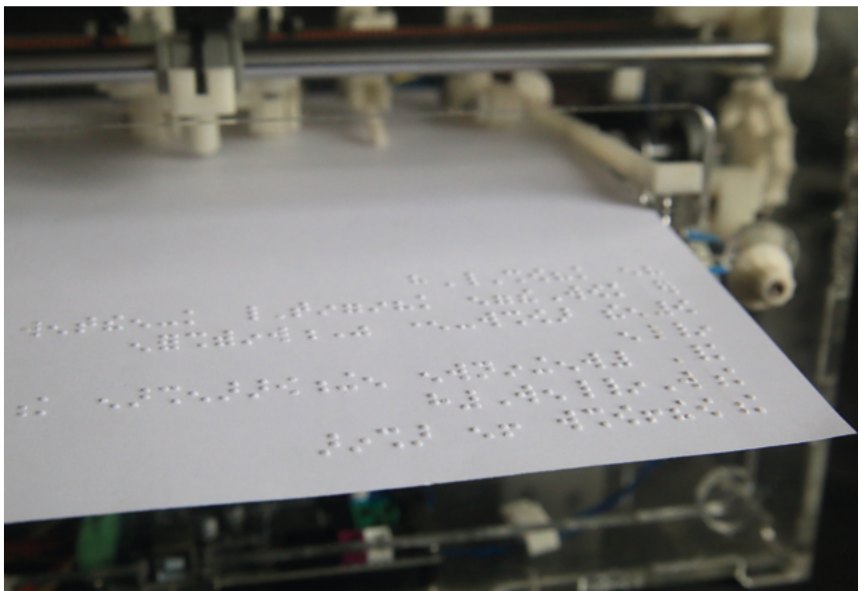
By **Stephane**

braillerap.org

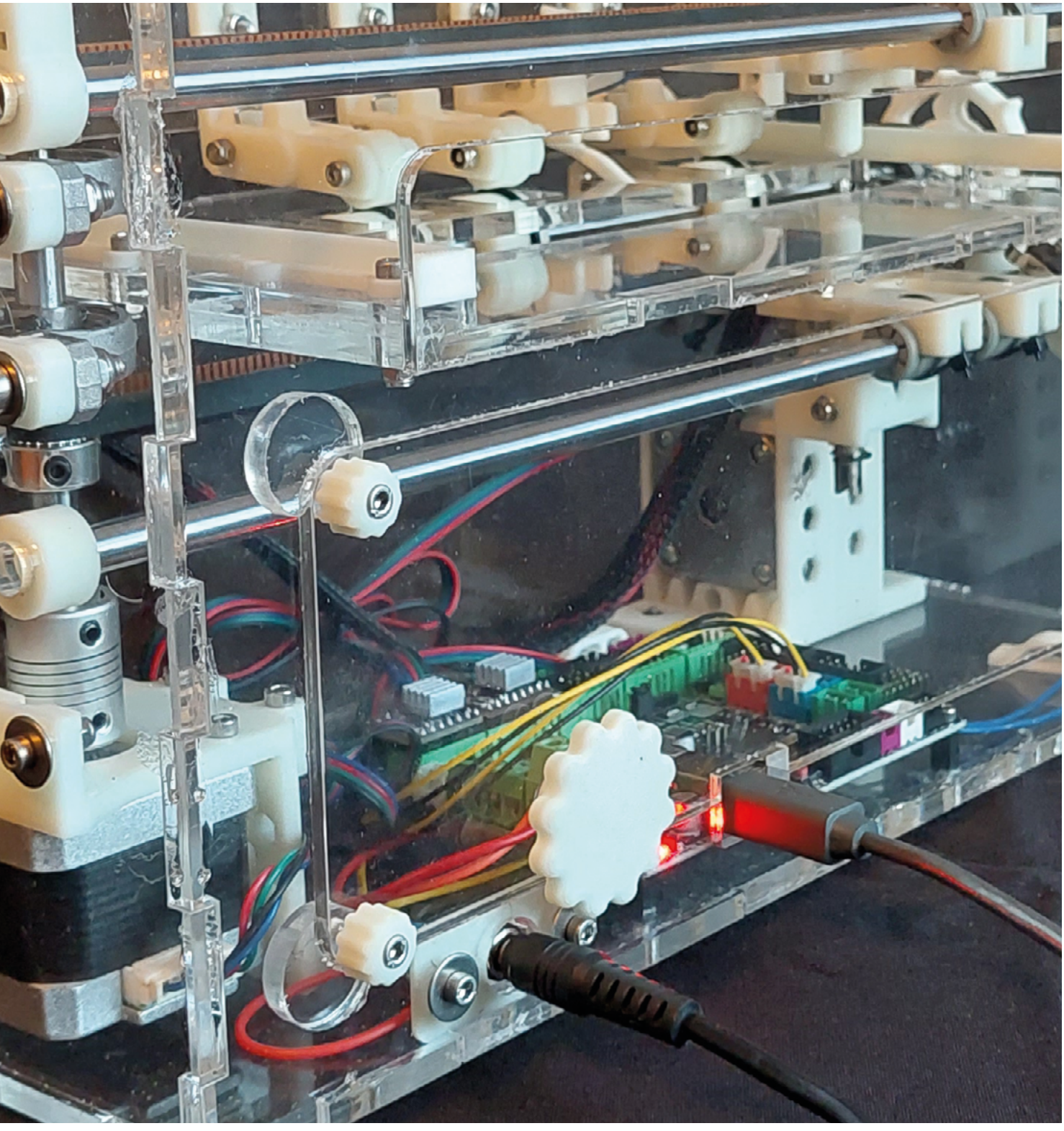
Braille, the tactile alphabet used by blind and partially sighted people, was invented almost 200 years ago by Louis Braille, in 1829. It's still widely used, but if you want to print anything in Braille (actually, there's no ink involved, so the correct term is embossing rather than printing) it's going to set you back a few quid. The Royal National Institute of Blind People (RNIB) has a few on offer on its website, with prices starting at £1760 for the cheapest model. You can get a roughly equivalent home inkjet printer for under £30. The difference is absurd, and it's all because of the much more limited market, and the number of moving parts that machines like Braille embossers need.

This combination – of high price, established tech, and a market that isn't being met by commercial forces – makes Braille embossers ripe for open-source hardware makers. And here, we have one such product: the BrailleRAP, released under the CERN Open Hardware Licence v1.2. We hope to bring you more about this wonderful project in a forthcoming issue of HackSpace magazine but, for now, just enjoy the clean lines of all that laser-cut acrylic, and ponder that the build pictured here cost around $250 to produce – around 10% the cost of the cheapest commercial solution. □





Right
BrailleRAP is an open-source Braille embosser in the spirit of the RepRap 3D printing movement



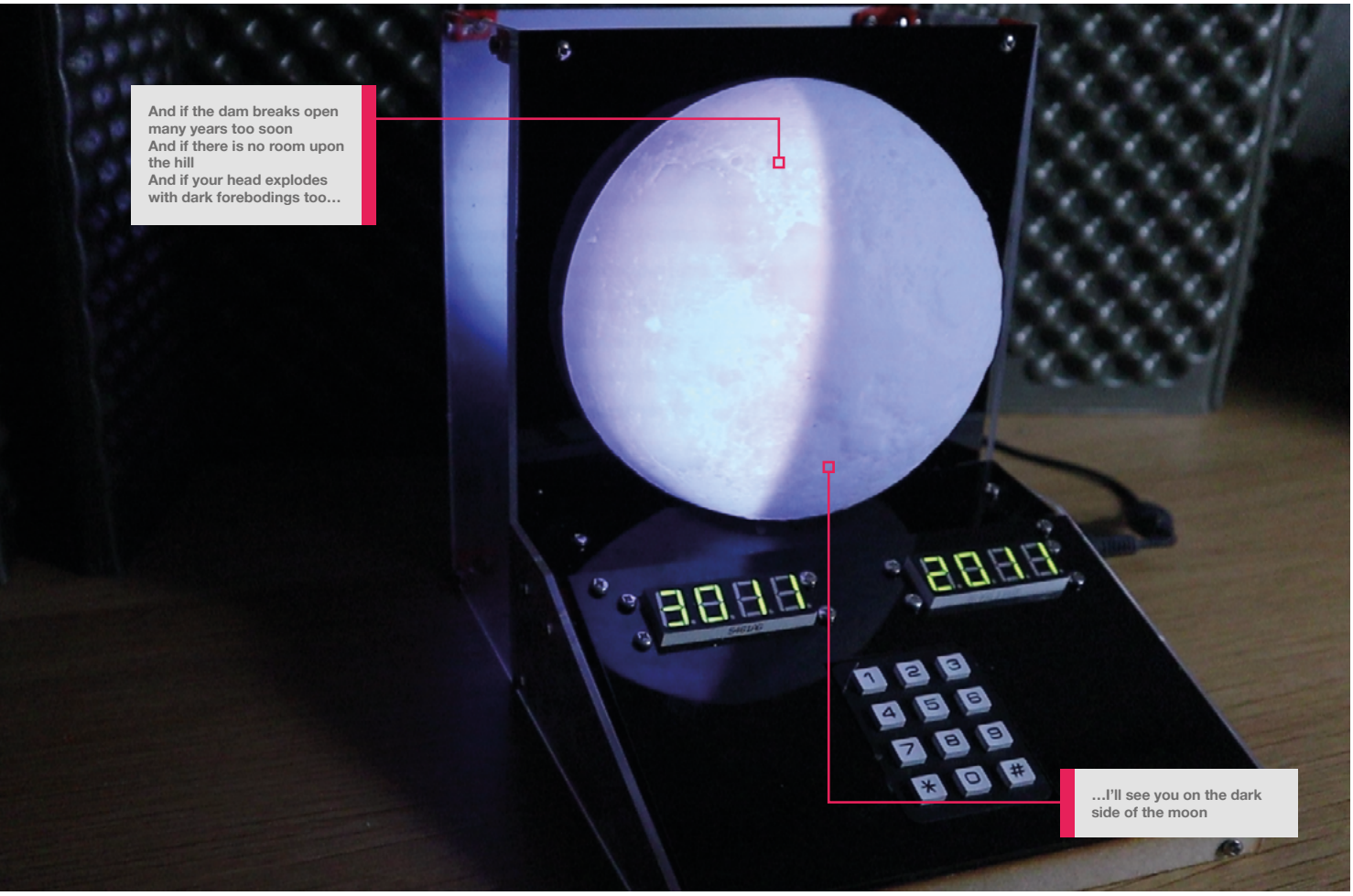
Right
The moving part of this moon display is controlled by a Kitronik motor board

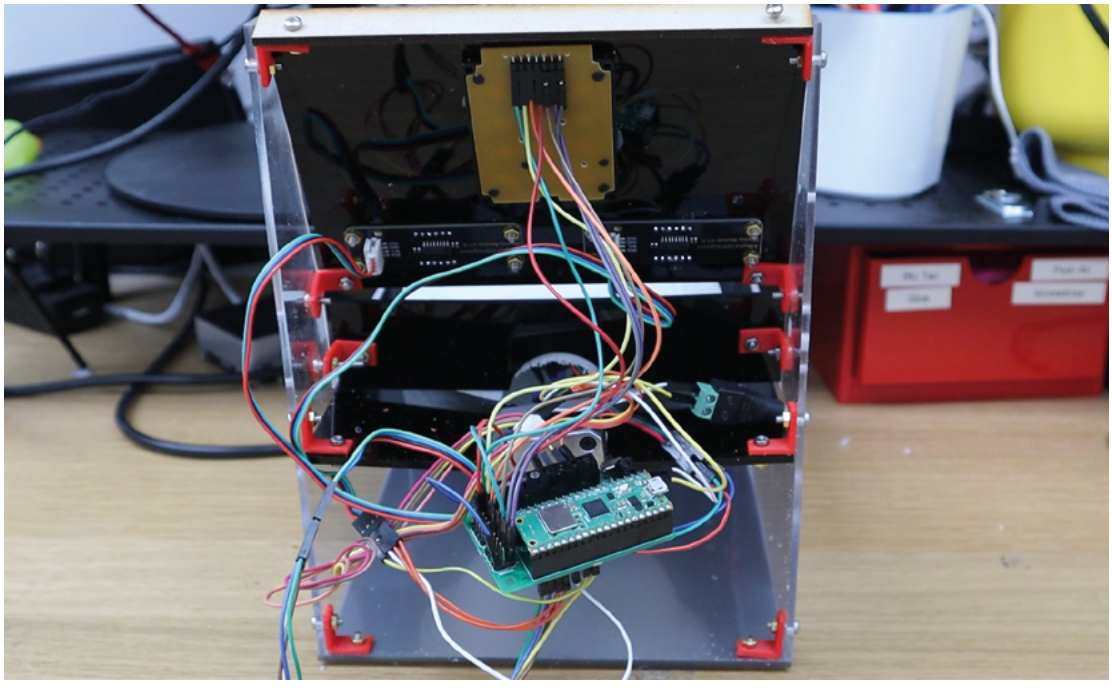
Moon Phase Display

By Lorraine Underwood community.element14.com

The moon orbits Earth every 27.3 days. It doesn't have leap years; it doesn't have some months when it takes 30 days, or 31, like our silly calendar months. Give a person (or, in this case, a Raspberry Pi Pico W) a date in time, present or past, and they'll be able to tell you what the moon looked like on that day using a really simple formula.

If you were to take that knowledge and add a hollow, light-up model of the moon, a motor board from Kitronik, and a Raspberry Pi Pico, you might come up with a device like this by Lorraine Underwood. It's very simple: you just enter a date, and the light inside the moon moves to show the phase of the moon on that date. We don't have any more information than that at this stage, but Lorraine's working on a build video for Element 14 that will reveal all.


And if the dam breaks open
many years too soon
And if there is no room upon
the hill
And if your head explodes
with dark forebodings too…
…I'll see you on the dark
side of the moon



Kinetic PC case

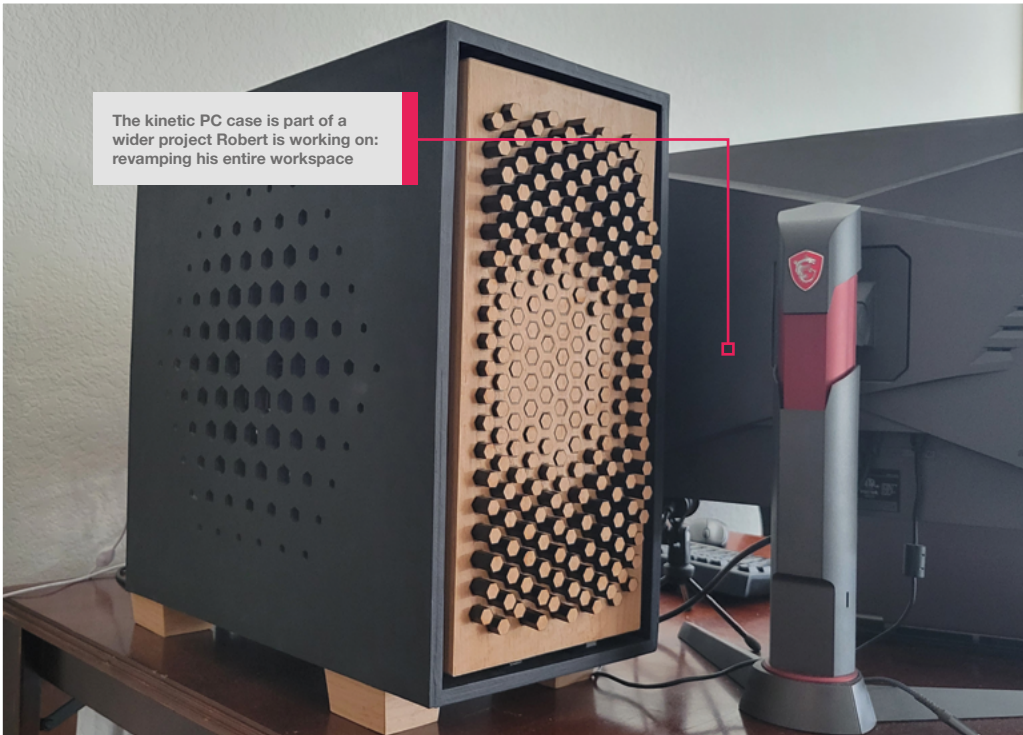
By **Ideal Idea** hsmag.cc/KineticPCCase

This beautiful PC case was inspired by PC retailer CyberPower's kinetic PC case. It moves – it looks a bit like the vents of a jet engine opening – but it's not available to buy. The company teased PC enthusiasts with a glimpse a couple of years ago but, for whatever reason, it hasn't yet come to pass other than as a concept. Robert, the human behind the Ideal Idea YouTube channel, took this as a sign that he should build his own moving PC case, and, wouldn't you know, it's even better than the one that inspired him.

The brilliant thing about this build is how open Robert has been in showing his development process. He initially thought of controlling each hexagon with its own linear actuator, but realised that this would make the coding horrifically complex, and cost at least $600 for the parts. Then he tried powering the linear motion by means of turning a shaft, which turned another shaft, which turned lots of other shafts, but that also presented its own problems.

The eventual build uses an Arduino Uno, a 4488 stepper motor driver, a Nema 17 stepper motor, and gears attached to 3D-printed camshafts. These run vertically up the side of the case, and convert rotational movement into the linear movement of the little hexagons moving in and out along the front panel of the case.

Right
If you've got access to a laser cutter and infinite amounts of creativity, why not try something similar yourself?



The panels hosting the moving parts are modular, and can be swapped out if the user wants to get a different look – Robert's also made a panel inspired by the rose windows of mediaeval stained glass

The camshafts were 3D-printed flat, then bent using a heat gun and a 3D-printed 30-degree jig

Volumetric display

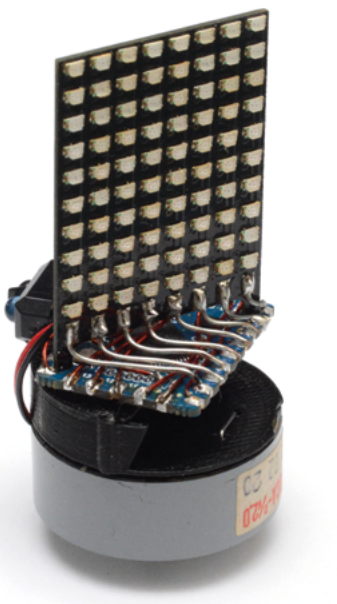
By **Tim Jacobs** Mitxela.com

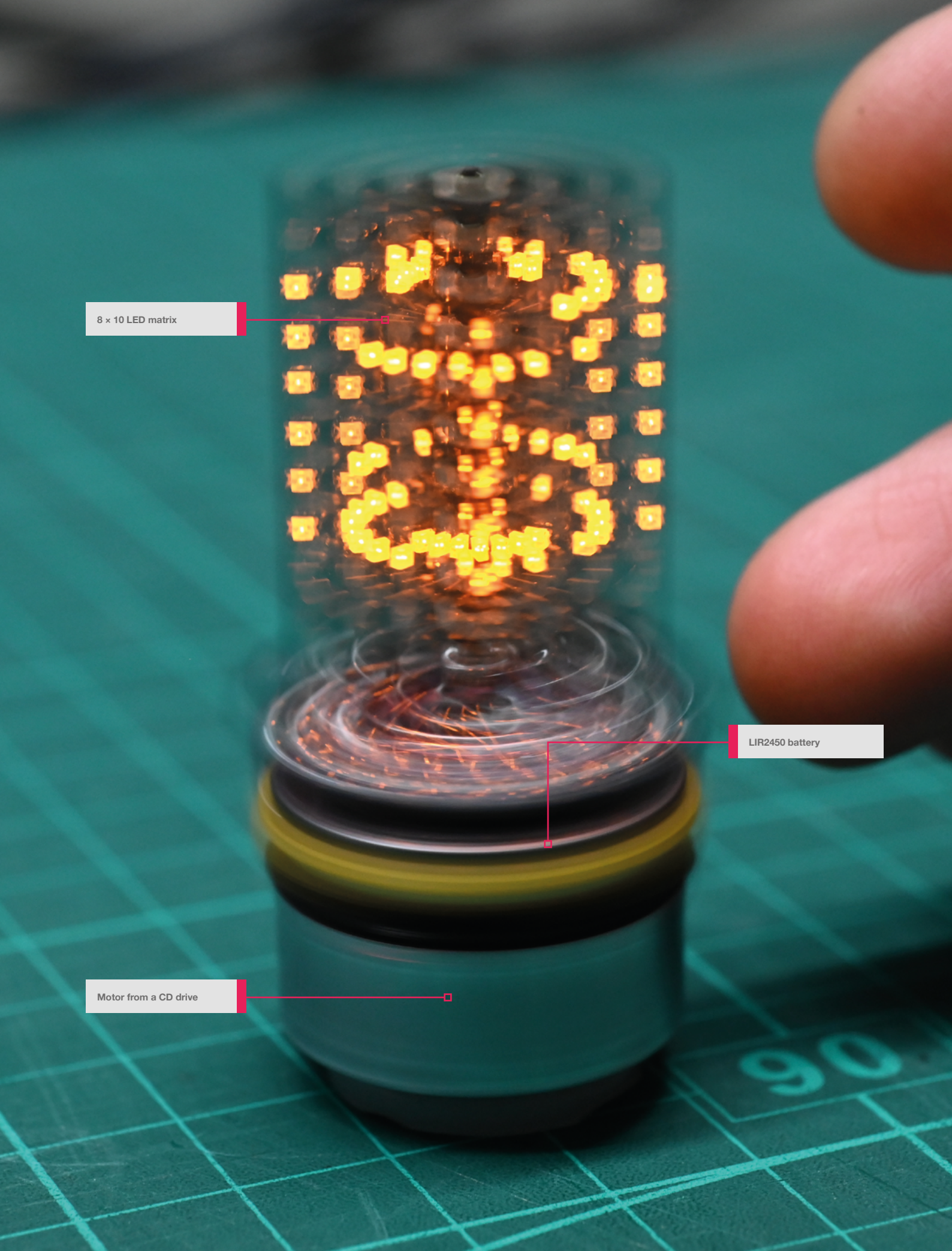
"I **was recently fortunate enough to find myself in the pub with some very creative and talented people.** The discussion turned to electronic candles, and how one might create something that would look like a flickering candle from any angle. I suggested a persistence of vision display, but the general consensus was that those require too much in the way of supporting machinery to make them work: bearings, and probably slip rings and so on."

So begins Tim Jacobs' explanation of his brilliant volumetric display. It's a tiny build (pictured here with a human finger for scale), and it's also quite brilliant in its approach to an engineering challenge: the problem that you normally get when building anything using persistence of vision is getting power to the moving parts. If you have an LED mounted on something spinning, you need some way of getting power to the LEDs without getting the power wires twisted up. You can do this using wireless power, or by using brushes that maintain contact even as they spin, but that naturally increases complexity, cost, and the likelihood that the device will fail at some point. Tim got round this challenge by making everything spin. The display, the electronics, the motor, even the battery all rotate when the motor is running. Problem solved!

So far, Tim's used his display to show fluid sloshing about in a container, a spinning cube, and a candle flame among others. Naturally, you don't appreciate the full glory of the animations when you're looking at a static image, but while we're here, we can all coo over the free-form soldering that he's used to attach the display to the Waveshare RP2040-Tiny, a small board that breaks out some of the GPIO pins from Raspberry Pi's RP2040 chip.

Right
Tim's test image is a "janky low-resolution wire-frame cube", as he puts it. But it works!




8 × 10 LED matrix
LIR2450 battery
Motor from a CD drive

Mailblocks

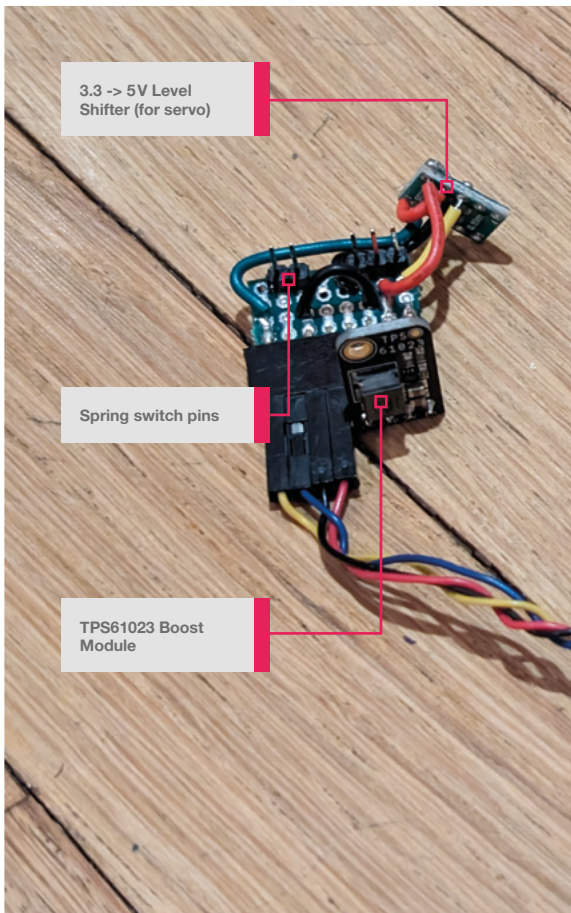
By **Guy Dupont** hsmag.cc/Mailblocks

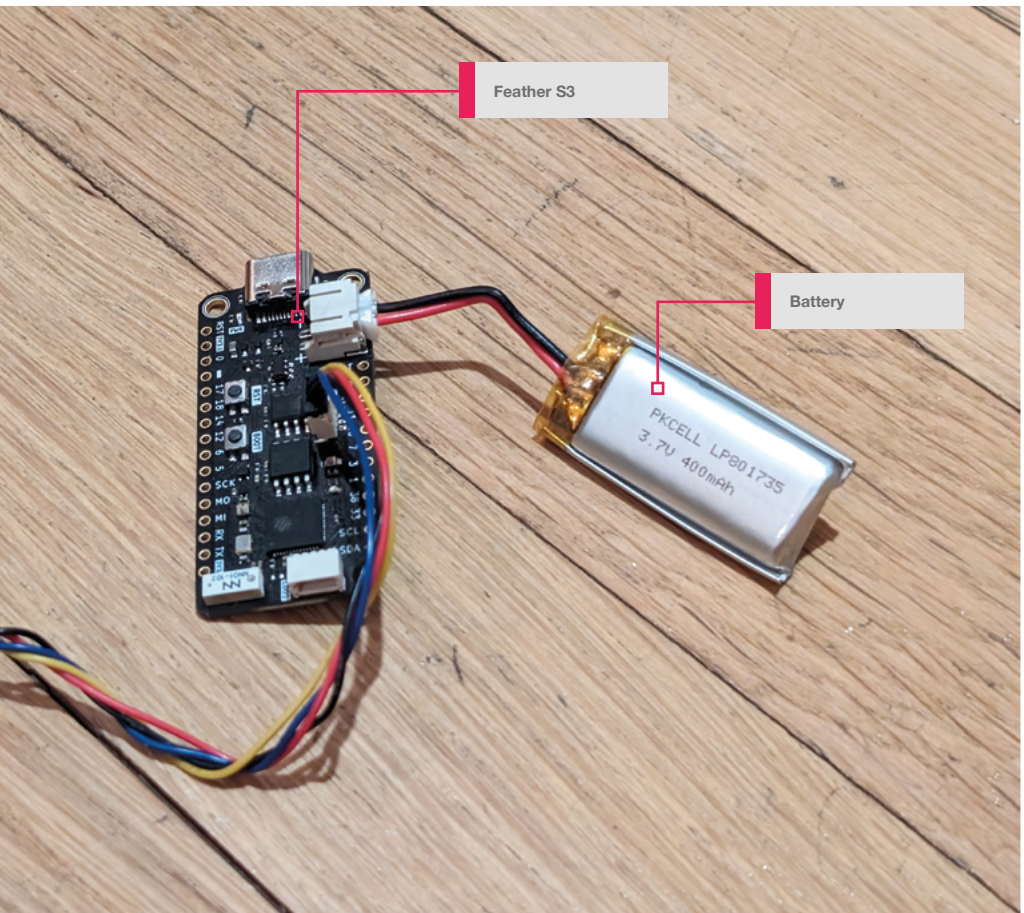
Guy Dupont has a problem: he keeps making brilliant creations that pique the interest of the internet. Every time he makes something interesting on the internet, his phone starts blowing up with notifications. What to do?

Well, in a bit of a meta-twist, he decided to make yet another brilliant device, only this time, one that would block notifications until such a time as he's ready for them. This is the Mailblocks. It's based on the old-fashioned mailboxes of North America, in the days when you'd have to get up and move your body to receive paper messages once a day from the outside world, and it means that Guy's Android phone doesn't bug him with updates unless he chooses to accept them.

Guy uses a Raspberry Pi acting as a Wi-Fi router on his home network, running a custom DNS server that looks for IP addresses associated with push notifications, and redirects the DNS requests to nonsense IP addresses.

That's the blocking part. So far, so good: now, in order to pick up notifications, you need to put it into the mailbox. Inside the mailbox is a mechanical switch that closes when the phone is inserted. The switch is connected to a Feather S3 board, from Unexpected Maker, that tells the Raspberry Pi to stop the monkey business with DNS. The Wi-Fi network reboots and, when the phone looks again for the Google servers, all the previously blocked notifications come through. Congratulations: you are now in charge of your phone, rather than the other way around.





Above
The flag raises when you've got mail; it's activated by a small servo, powered by an Adafruit MiniBoost 5V

Your trust is our goal

From genuine, manufacturer-warrantied components to millions of in-stock parts shipped same day, be confident DigiKey will get you what you need—when you need it.

Visit digikey.co.uk today or call 0800 587 0991.

DigiKey

we get technical

HackSpace
TECHNOLOGY IN YOUR HANDS

LENS

HACK | MAKE | BUILD | CREATE

Uncover the technology that's powering the future

FEATURE

DIY GADGETS

Who doesn't love a good gadget? **Rosie Hattersley** finds out more about some amazing makes

Think of what makes a great gadget and it's hard to escape the idea that they are about kudos, kerb appeal, and communication – not necessarily all in the same package, of course. The smartphone is arguably the ultimate expression of gadgetry, encompassing all three concepts. It also demonstrates the importance of portability. How else to gain that all-important kudos without being able to take it out and about to show it off to our friends? We kick-start our look at the world of homemade handhelds and wearables with two great phone examples of home-brew design before applauding repurposed tech to create incredible cyberdecks and tiny PCs. Electronics play an important part in musical and visual gadgets too, as well as in homages to iconic films and games. With cosplay becoming ever-more indistinguishable from original props and costumes, we highlight some amazing makes along with achievable accessories you can 3D-print or simply customise.

PHONES

Keep in touch or upcycle some old tech

OURS smartphone

If you're taken by handset design rather than upcycling an existing phone, you could do a lot worse than follow Evan Robinson's build instructions for the 4G LTE OURS (open-source, upgradable, repairable smartphone). Running Raspberry Pi OS and primed with familiar smartphone apps such as Facebook, YouTube, and WhatsApp, it has a 5MP camera, microphone, and volume control, as well as an HDMI port to attach an external display alongside the OURS' 4-inch Waveshare touchscreen. Evan created the handset with privacy and security in mind, but it is also great value, with a bill of materials costing roughly 180 euros.

Evan says the most difficult part of the build was getting the power system stable enough to power the machine through boot because it uses a lot of current upfront, and sporadically for activating the 4G LTE module, but then uses less current once the OS is loaded. He spent a lot of time fiddling with tiny potentiometers on voltage regulators, then turning the system on and off over and over again. Since unveiling OURS, the idea has been embraced by dozens of other makers, with plenty of constructive feedback and around 40 enthusiasts actively working with Evan to develop a new model.

hsmag.cc/OurPhone

Antique telephone

Maker Mark Lister bought an antique phone from eBay, took out its existing electronics, and added a Raspberry Pi 3B from 2022 with bespoke GSM HAT and a speaker. The Raspberry Pi also provides internet access. Mark mapped the numbers on the old phone's rotary dial to Raspberry Pi's GPIO pins to create a usable handset that stores numbers and reads out the name of callers it recognises. The SIM card in the GSM HAT recognises the pulses generated by turning the dial as digits, thanks to Mark's specially written code. Further features include a backup UPS battery and, of course, a proper ring. Brrring, brrring!

hsmag.cc/OldPhone

Phone Map

If you're holding a phone handset up to your ear, it might as well deliver something you want to hear. Caroline Buttet decided she'd like to enjoy music from around the world, using the **radiooooo.com** online radio station player, an Arduino, and Raspberry Pi to serve up broadcasts, depending on where on a map of the world she placed her pin. The user can also pick up the rotary phone handset, dial the country code, and select a decade to choose what they hear. Caroline used Raspberry Pi to connect to the radio player app, while an Arduino hidden in a metal panel behind the framed world map controls the country selector.

hsmag.cc/PhoneMap

CLAMSHELL BLACKBERRY CYBERDECK

Because everything's better when it's tiny

The original business smartphone, BlackBerry's main legacy now seems to be the usefulness of its tiny keyboard in retro builds and cyberdecks such as this laser-cut, wood-clad version by Michael Klements. "Most keyboards, even compact and foldable ones, are three to four times bigger than the final build size that I was going for – and you'd still need to add a mouse to that," Michael explains. However, he was delighted to discover Solder Party's BB version on Tindie (**hsmag.cc/BBQ20Keyboard**). The Clamshell Blackberry Cyberdeck features a HyperPixel 4.0 display which hooks up to Raspberry Pi 4 via its GPIO pins. The advantage here is that there's no need for an additional power source for the screen, meaning Michael could keep his build ultra-compact. This aim also involved a tricky design process, so the hinged plywood case, sketched out in Inkscape, is just large enough to accommodate the cyberdeck's components, and allows the screen to tilt back to a 20-degree angle.

hsmag.cc/CyberBB

GOOD FOR GAMING

Taking your entertainment on the road

Fancy Octopus Arcades

Brooklyn-based Shonee Strother and his son Wolf both loved arcade games and the characters found in animations and cartoons, but the constraints of a New York apartment meant their entertainment ideas far outstripped the realities of what might be practical at home. Shonee "saw folks making huge gaming rooms, loading them up with arcade cabinets, and realised that for some of us in smaller apartments, that wasn't a possibility." The pair set about installing RetroPie builds inside portable boom boxes and mini cabinets and customising each 'fancy' arcade with plastic dolls and other props to match each arcade's games offerings. The mini cabinets are Raspberry Pi-powered and "packed to the gills with games and easily plug in via standard HDMI for video audio output." Each design is unique, and the mini arcades are themselves miniature art pieces that look good displayed on a bookshelf or, indeed, shown off on Instagram, which was the main reason Fancy Octopus Arcades began getting noticed. It was also a neat way of Wolf immersing himself in the games history his dad was keen for him to soak up.

hsmag.cc/OctopusArcade

Bluetooth speaker Mini PC

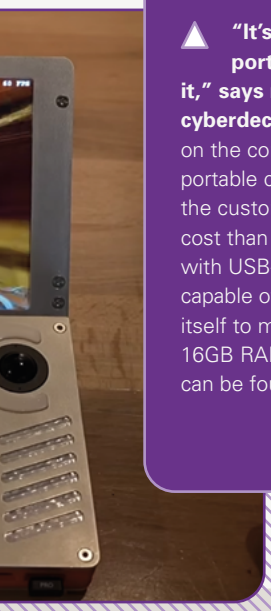
Carter Hurd didn't really want or need a Bluetooth speaker, but having been sent one on spec, he set about making use of it anyway rather than have it sit around gathering dust. The Divoom Ditoo Plus Retro Pixel Art Game Bluetooth Speaker's distinctive looks seemed to lend themselves to re-use as a mini PC, not least because it had several buttons and a navistick already. However, Carter dispensed with these, as well as the original speaker, in order to fit in a screen. He then built a cyberdeck computer using Raspberry Pi 3B and an old BlackBerry keyboard, which was a near-perfect fit. He covered up any rough edges and the surrounding board by 3D-printing a bezel. However, the screen was more of a challenge. "Being square was a good start, but the glass was too large to fit within the speaker housing." He ended up taking a pair of shears to the screen to trim off its edges – a rather risky endeavour – then shaped a piece of thermal plastic into a dome with a hairdryer and placed it over the LCD to give the screen a retro look. While component size versus the Divoom case was an issue all round (Carter says Raspberry Pi Zero would have been a better fit), the end result looks great and runs Raspberry Pi OS "a treat." Had he pre-planned this build, however, he'd love to have found a way of preserving the touchscreen.

hsmag.cc/MiniPC

Framework Cyberdeck

"It's weird, heavy, and looks like it came out of a portal to some alternate timeline, but I personally like it," says maker Ben Makes Everything of the DOOM-playing cyberdeck he built from Framework laptop parts. His take on the concept of a cyberdeck is an individually tailored, portable computer. Framework supplies the basic hardware for the customer to build, repair, and upgrade at will, but at lower cost than other bespoke laptops. It has a modular I/O system with USB-C connections and runs Windows (and is easily capable of running many mainstream games) but also lends itself to modifications. Ben's own designs, based around the 16GB RAM laptop with Western Digital hard drive he created, can be found at **hsmag.cc/BMECyberdeck**.

hsmag.cc/FrameworkCD



COSPLAY AND FILM PROPS

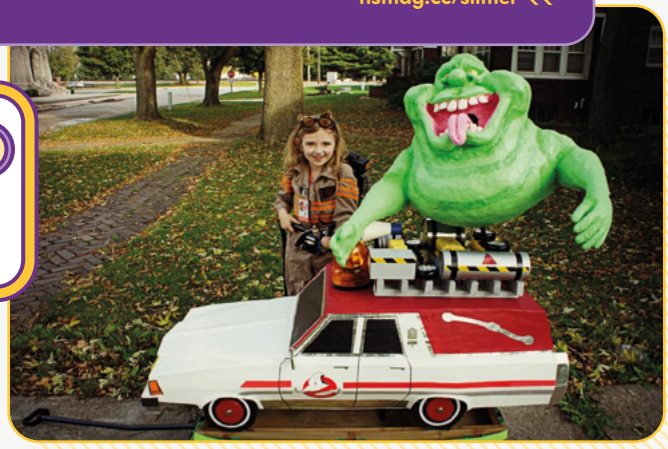
Get dressed up

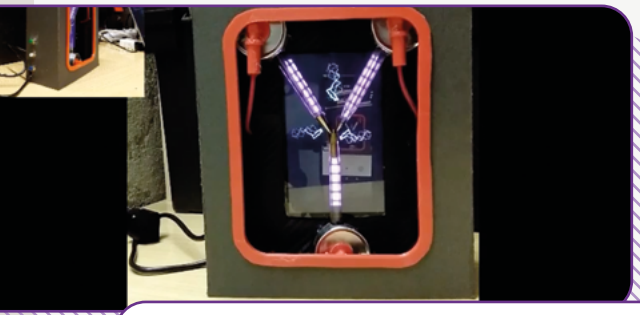
Fancy dress just doesn't cut it these days. Realistic props and intricate cosplay outfits, along with studio-quality make-up, are expected in this age of constant visual scrutiny. If you're new to the world of properly dressing up in character, the Costume Wall website has some good ideas: **costumewall.com/titles**, e.g. **costumewall.com/title/adventure-time**. Perhaps you love *Stranger Things* and yearn to dress like a Demogorgon (**costumewall.com/dress-like-demogorgon**), or want to recreate a battle scene and need to make armour that passes muster, in which case Kamui Cosplay armour tutorials can help **kamuicosplay.com**. Cosplayers also rate Coscraft (**coscraft.co.uk**) for supplies.

Proton Pack Slimer

You can probably source your own *Ghostbusters* boiler suit, but hats off to the parents of Livi, who set about creating this amazing proton pack, complete with ectoplasm. Expandable foam on a piece of dowel was used to make Slimer himself, using a coat hanger to give the body form and foam-filled gloves for hands. The face, and other features, were then carved and the whole beast painted. The proton pack has three LEDs and was made from cardboard, plastic, wires, ribbon cable, stickers, a calculator, random switches and knobs, some hose, pipes, and batteries, explains mum Katrina who wrote up the family project.

hsmag.cc/slimer





Yet Another Flux Capacitor

Ambrogio is far from the first film fan to try and recreate items from 1980s classic *Back to the Future,* hence calling this project YAFC (Yet Another Flux Capacitor). A device that propels us into the near future or past seems a fitting DIY project for a keen paraglider given to flights of fancy. "The internet is full of replicas made of LED strips, but I wanted to follow a different innovative approach, reproducing the 'energy flow' effect in the flux capacitor by means of a video created with a video editor software." He used Raspberry Pi, since it has a built-in VLC player that will work with almost any inexpensive display, and supports Python. "From a maker point of view, [it] is a great language since it does not require complex environment setup," he finds. He attached a Pimoroni Automation HAT to Raspberry Pi 4 and used VSDC video editor to get the flux capacitor effects he was after. The whole thing is hidden within a painted cardboard frame and runs headless, with three large control buttons to initiate the looping flux effect (viewable as a YouTube clip). Find his instructions at **hsmag.cc/GHyafc**.

hsmag.cc/yafc

ELECTRONIC JEWELLERY

Put some spark in your accessories

Watchy

Watchy is a great-looking timepiece with a 200×200-pixel e-paper display, a glare-free 180-degree viewing angle, and ultra-low power consumption. Powered by an ESP32-PICO-D4, it is a certified open-source hardware kit that you can customise with your own watch-face designs. Examples (and code provided) include random Tetris and mandala designs. There is also the option to design your own watch-case with the provided schematic if you don't wish to stick with Watchy's standard version. The kit includes the PCB, battery, double-sided tape, wrist strap, plastic case, and e-ink display.

watchy.sqfmi.com

NeoPixel Cosmic Turtle Necklace

There's a lot to like about this light-up pendant build which makes use of Adafruit NeoPixel Jewel, which has seven controllable LEDs, a Trinket, and your choice of either upcycled or 3D-printed and painted pendant shell. Maker Erin St Blaine chose a pendant shell with the right sort of holes to optimally show off the illumination created by wiring up a tiny breadboard with minute backpack charger, before soldering on two lengths of wire that double as the necklace itself. Adding a micro USB connection means recharging can be done by plugging into a laptop or USB charger when needed.

hsmag.cc/LEDpend

Cyberpunk Fusion Ring

Jewellery that incorporates electronic components can make a great statement, and we've seen versions as earrings, cuff-links, watches, and necklaces. A clever way to upcycle circuit boards and declare your techie leanings, few such accessories combine utility with fashion. However, NFC offers a little something extra (**hsmag.cc/pcbring**). Maker marketplace Etsy lists pendants that can be customised with a personalised message – essentially modern-day lockets (**hsmag.cc/nfcpend**) – while jeweller Eli Yoo meticulously crafts NFC rings that can be used to unlock a phone and make a micropayment. Eli describes his collection (**hsmag.cc/nfcring**) as Cyberpunk Fusion Rings that provide a futuristic twist and embrace the cyberpunk vibe. The precisely CNC-milled brass rings are "adorned by a circuit board, featuring an added NFC chip and tritium tube." Glowing LED vials can be set to blue or orange, while the NFC chip is programmable so the owner of this "wearable piece of the future" can select a range of functions.

Bat Girl Birkin Bag

Dennis Louie, aka Caitlinsdad on Instructables, created the Batgirl Birkin Handbag a decade ago, figuring "What's more exclusive and has more cachet than scoring a Birkin bag? Pow!!! A Batgirl Birkin bag!" The Batgirl bag was a plasticky Birkin knock-off with a helpfully translucent coating that a "megalumens flashlight" can shine through. Customising it involved adding purple, yellow, and black fabric trim, plus a length of cable for the Taser probe and a pink toy phone handset that he wired to work with a smartphone. Caitlin's version features a bat signal, bat proximity alarm, bat taser, bat phone, and plays the *Batman* TV theme. Based on its maker's nostalgia for the 1960s TV series, the accessory is hefty enough to serve as a weapon, though the Batman the theme tune, enabled by Adafruit FLORA, is a little on the quiet side. Dennis suggests listening to its play here: **hsmag.cc/BGsound**.

VISUAL SPECTACLES

Making everything look just right

Miniature observatory

This gorgeous mini observatory is the work of Matt Hough, a retiree with a long-term interest in astronomy who now has time to put the Python and hardware design ideas he's been honing into practice, starting with experiments using Raspberry Pi Pico. He taught himself about stepper motors for use with astronomy cameras, then realised an RP2040 offered the power and control he needed and would work alongside the main Raspberry Pi 4B. "Mechanical and observatory design were something new for me. I'd done some 3D printing before, but never had to design so many parts myself. The observatory, gears, and mechanism are all homegrown. The semi-intelligent motor controller for the telescope is probably the most novel homegrown element. I needed a way for the telescope to move itself whilst the Raspberry Pi was busy taking photographs, so I gave the motors a little brain. I'm sure there are better solutions out there to some of the problems I faced – I hope to still find a few for the next version. It felt like a distraction at the time, but it's a new skill, and it has come in useful elsewhere since." He used CircuitPython for the 'very nice' Adafruit components and made use of Python libraries for the Skyfield, OpenCV, PiDNG, and Astroalign astronomy-specific features (see his GitHub: **hsmag.cc/Pilomar**). Chief among Matt's goals for the next version of the mini observatory is finding a way to have the telescope process images in real-time.

hsmag.cc/MiniObs

489MP DIY Scanner Camera

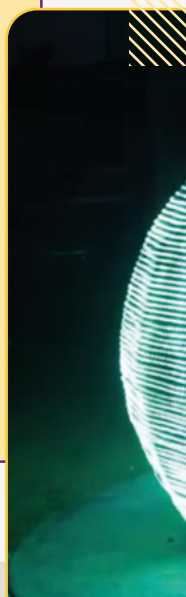
Yunus Zenichowski's prototype 489MP camera might be a bit of a misnomer: it's actually an Epson flatbed scanner capable of capturing high levels of detail. Its CCD moves across a fixed, linear area and processes an image for each line of the image. Having removed the mechanical CCD from the old scanner, Yunus attached an inexpensive projector lens to the front. He admits that this cost-saving choice "comes at a cost of sharpness." However, aside from any 3D printing costs, it means a high-resolution camera can be built from other photography-related parts. Modifying the scanner potentially results in distortion and ghosting effects due to the lengthy exposure times, but could also make for some interesting artistic effects that lend themselves to large-scale printing. An advantage is the scanner has a colour depth of 48-bit, rather than the more common 8 or 24. Yunus details his DIY idea at **hsmag.cc/489MPcam**.

hsmag.cc/megascanner

DIY Vegas Sphere

When DrZzs and GrZzs saw the Las Vegas Sphere entertainment complex, they knew they wanted to build a replica. They settled on an 8-inch Mini Megasphere (a little smaller than the 516ft-wide Vegas version), but are also "pretty far along creating a smaller version that pretty much anyone can 3D-print and build." Challenges included ensuring the pixel density was distributed uniformly around the sphere. After all, if they "just wrapped a matrix of pixels around a ball, they would have ended up with "a bunch of pixels bunched together around the top and bottom." To equally space the pixels, they ended up with 45 pixels in the top ring and 310 around the middle. The frame consists of five horizontal rings, 18 vertical ribs, and 77 rings of pixels. A commercial CNC routing company cut out all the lightweight 'gorilla-ply' parts. Weighing roughly 550lbs, the huge sphere also needed to be protected from the elements. Once painted and waterproofed, 20,000 pixels were attached to 1-inch-spaced pixel straps. Even with the help of "pixel-pushing pliers, a pixel-pushing jig, and wearing work gloves", the process took 50 hours!

hsmag.cc/VegasSphere



GET MUSICAL

Build your own instruments

Fuzz Effect

Fuzz effects can make music seem more raw and alive, rather than regular and controlled, adding reverb and delay to shape the tone to make it more unique and appropriate, in the words of maker Handy Bear. Beloved of rock guitarists, it seems fitting that the grungy image is echoed by the homemade vibes of this DIY distortion pedal build. Using a blank breadboard, some jumpers, LEDs, and capacitors, and encasing the electronics in a metal enclosure, you can make a range of effects. Sites such as **tagboardeffects.blogspot.com** advise on how to make effects reminiscent of particular songs and bands, explains Handy Bear, and a fuzz effect is fairly simple and a good starting point. Spaces for the foot switch, LEDs, potentiometer, and cable all need to be marked out and drilled before the power jack and cable socket are added (and isolated) and the enclosure is wired up as a foot pedal.

hsmag.cc/FuzzEffect

Stepper Motor Piano

Thisinomine's gorgeous instrument works by spinning motors at different speeds. There are a few such examples around, but this one is user-playable, rather than creating random notes or playing a preset song, much like an old-time fairground organ. "I wanted to create something that looked and felt like an instrument, but secretly had an electronic skeleton," says Nomine. The motors used here have 200 steps and can be rotated a single step at a time, so they are highly accurate, and can be set to rotate a certain number of degrees and at a certain speed. This level of accuracy makes them ideal for music-making. Whenever one of the copper keys is pressed, it sets off one of the stepper motors, controlled by an Arduino. The acoustics of the piano's resulting notes are enhanced by the polished African hardwood case in which it's encased.

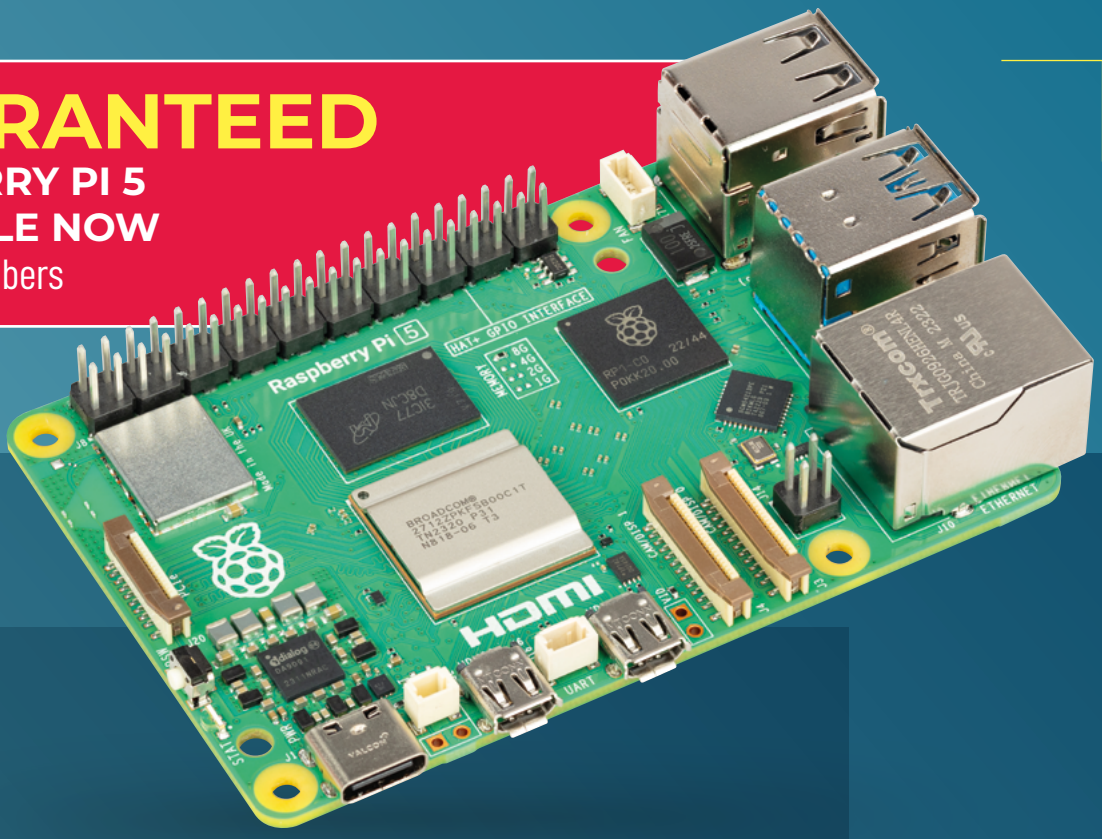
hsmag.cc/MotorPiano

SUBSCRIBE TODAY

GET SIX ISSUES FOR JUST:

£30 UK / **€43** EU / **$43** USA & Canada

GUARANTEED
RASPBERRY PI 5 AVAILABLE NOW
for all subscribers



SUBSCRIBER BENEFITS:

- **Get every issue of HackSpace magazine delivered to your door**
- Beat the crowds with guaranteed Raspberry Pi 5 stock for subscribers
- Early access to the PDF edition
- **Get a free Raspberry Pi Pico W**

hsmag.cc/subscribe

Subscribers will get a voucher giving them the chance to purchase one Raspberry Pi 5 from reserved stock at The Pi Hut (thepihut.com) for full retail price. Reserved stock means that these will be available even if they are out of stock for general purchase.

HOW I MADE

By **Karl Mose**

PIARTFRAME

Bringing mathematical art to halls of residence

About four years ago, in the cold of December, I was boarding a plane destined for the UK. I'd applied for university to study computer science, and part of the interview involved an expectation you'd have some kind of project to talk about. So, I spent the preceding weeks attempting to devise a project that I hoped would impress the interviewers. I'd always been very fascinated with maths and, at the time, fractals more than anything had caught my attention. Fractals are geometric shapes that we can generate with mathematics. Some fractals, when tuned correctly, can be stunningly beautiful. To this day, it still blows my mind how much visual complexity you can have emerge from relatively simple equations. At that time, I spent hours exploring their vast landscapes and adjusting their colours, making it a natural choice for the project.

The project that I came up with at the time was a rather painstaking effort of a couple of months, using a Raspberry Pi to generate the fractals, and a smartphone app that would download them periodically and set them as my phone wallpaper, for me to look at throughout the day. The app

got, at most, a hundred or so downloads, and in the interview I spent, at most, one or two minutes talking about the project, but the admissions team seemed pleased with it and I'd like to think it had some credit in me being granted a place. Last year I finished my undergraduate [course] and left academia, as well as the UK, behind. I started working on a job abroad and enjoyed no longer having to live with a student's budget. However, missing the excitement and curiosity of research, and after several tough decisions, I returned to my old university to start a PhD. Having moved onto research instead of revising, and teaching students rather than being taught, there is some sense of having gone full circle. This feeling was only strengthened by moving back into a university dorm room. Shortly after, I invited some old friends along to mine, who promptly complimented my room as being 'pretty minimalist'. I took the hint, and decided that my room needed →

Above ↑
From the front, there's not much to show that this is electronics

Above Left ↖
The Waveshare driver slots directly onto the Raspberry Pi GPIO pins

FEATURE

MAD

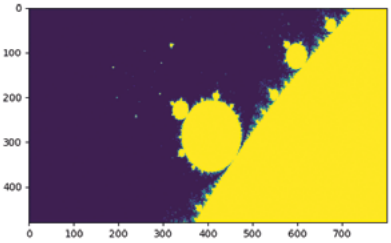
Right →
By continuously zooming in, I get an ever-changing bit of art

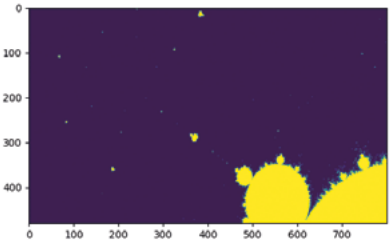
some decoration. And so, I decided to pay a homage to this old project of mine that originally led me to this university path. That's how I came up with the idea for the PiArtFrame: rather than rendering fractals to my mobile phone, I'd turn them into a piece of room decoration. I wanted to build a digital picture frame that, using the Raspberry Pi, would continually update itself with new fractals.

"WE FIRST PICK A SET OF BOUNDARIES"

The first hurdle of the project would be to write code to actually render the fractal. Speed wasn't essential, as the frame just needed to run fast enough to update a couple of times a day, so I opted for Python. The goal of the script was to start at some large-scale section of the Mandelbrot set, and then continually zoom in on increasingly smaller areas that appeared visually interesting. To render a section of the Mandelbrot set, we first pick a set of boundaries. For example, this could range from x = -2 to x = 2 and y = -1 to y = 1. We then divide this section into a grid of pixels and, for each pixel, we compute whether or not it is contained in the Mandelbrot set, using the aforementioned algorithm. For those who would like a more in-depth look at how one computes whether or not a point is in the Mandelbrot set, check out the GitHub repo of the project (**hsmag.cc/PiArtFrame**).

A notable challenge arises from the script continuously zooming into the Mandelbrot set, necessitating increasingly precise floating-point computations. To solve this, I used the Python 3 decimal library, which allows for doing computations at arbitrary precision. As the renderer zooms into a smaller and smaller section of the Mandelbrot set, it increases the precision at which it does decimal computation. As one attempts to render increasingly granular sections of the Mandelbrot set, it takes longer and longer to determine whether or not a given point is in the set or not. The algorithm we use needs progressively more iterations, and to make matters worse, there's not an easy way to calculate *how* many we need. If we do too few, the render will look smooth



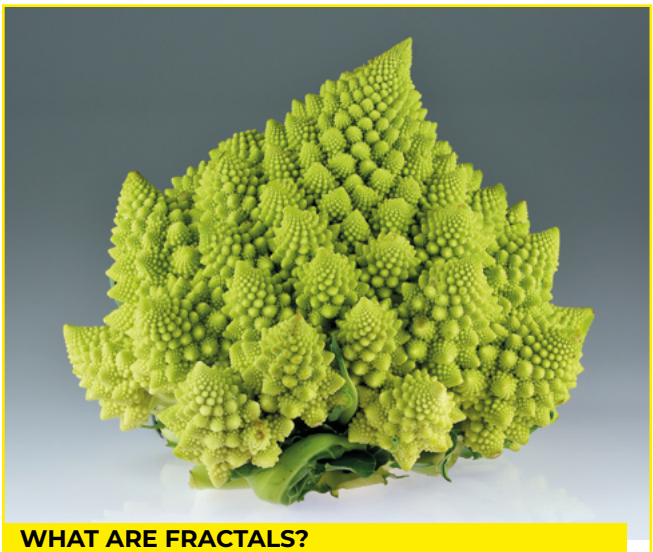


Right →
The same bulbous shape of the Mandelbrot set appears infinitely many times within it

PIARTFRAME

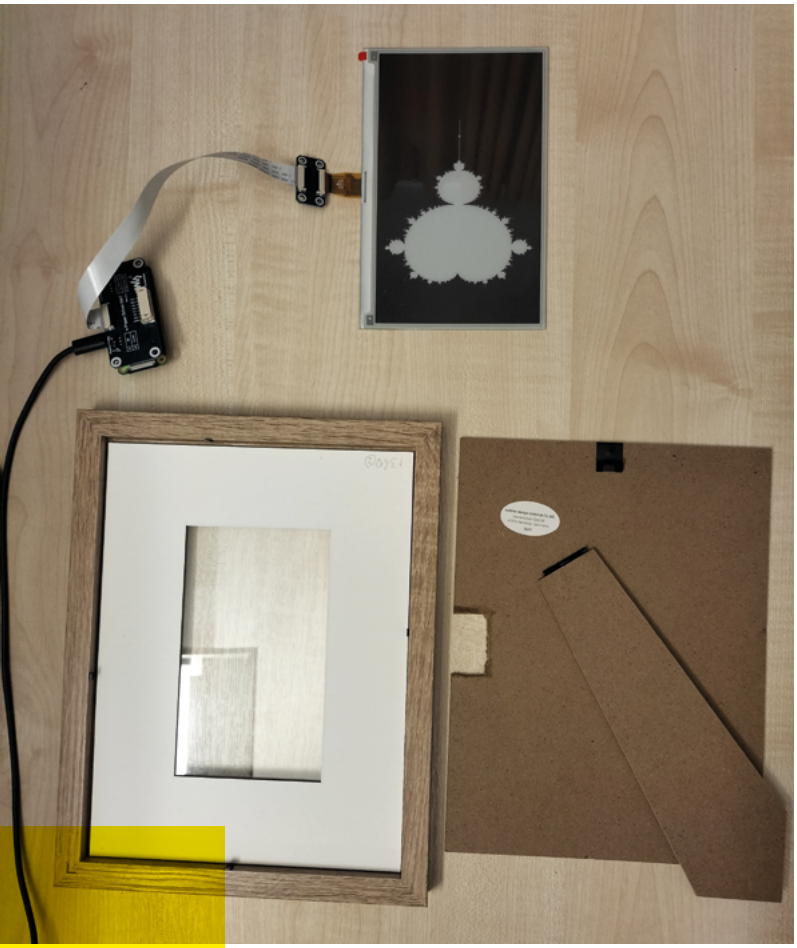
and uninteresting, and if we do too many, it will take too long. I made an approximate equation for the number of iterations for a given zoom-level, but it is inexact and sometimes either over- or undershoots.

One of the hardest parts of the script was designing a method for automatically exploring the Mandelbrot set. The Mandelbrot set is huge, but most parts of it aren't that visually interesting, it's only really around the edge that you see the interesting patterns discussed earlier. The current system that handles exploration works by dividing the most recent render into four squares. It then quantifies the uniformness; that is, how close each of the four quadrants is to being just one colour. Based on this info, it then picks a random quadrant to zoom into, filtering →



Right →
Fractals occur all over the place in nature, most deliciously in the Romanesco – a sort of halfway house between broccoli and cauliflower

Credit: Ivar Leidus creativecommons.org/licenses/by-sa/4.0/



Below →
All the parts you need: the frame, picture mount, back panel, e-ink panel, and the Raspberry Pi Zero

WHAT ARE FRACTALS?

For those who've yet to be initiated, consider this a much too brief introduction into a very exciting and mind-boggling world. Fractals are loosely defined as geometric objects that display detailed structure at arbitrarily small scales (source: Wikipedia). That means that one can keep zooming in on a fractal, without it ever reducing down to a pixellated grid like a traditional image, or just lines and colours like a vector graphic. Many fractals are self-similar – that is, the fractal contains itself in its entirety within some subsection. Fractals sometimes appear in nature, in things like pine cones, snowflakes, or even broccoli! The Mandelbrot is just one example of a fractal, but a rather interesting one.

The Mandelbrot set is defined by an equation which tells us, given a certain two-dimensional coordinate, whether or not that point is inside the Mandelbrot set or not. It's based on a relatively simple equation using complex numbers. If you're familiar with complex numbers, it won't take more than two to three minutes to understand the algorithm by which we figure out if a pixel is included or not included in the Mandelbrot set. There are plenty of good sources online, but I can recommend the YouTube channel Numberphile in particular. If you've not yet heard of complex numbers, I'd highly recommend it – they allow you to take the square root of a negative number, which is quite cool. The Mandelbrot set is self-similar – but often in surprising ways. The Mandelbrot set contains itself an infinite amount of times, but in some cases, it may also be bent or skewed in some odd way! There's loads of other different shapes to discover within the set, with some areas having been dubbed with creative names such as the Sea Horse or Elephant Valleys.

HOW

out any fully uniform quadrants. This algorithm isn't perfect, and sometimes (though not often) leads one into fairly uninteresting areas of the Mandelbrot set. If one sees that the frame has ended up in an uninteresting section, though, all it takes is a quick switch on and off the Raspberry Pi to restart its search. The script can take anywhere from a few minutes to hours or days to render, depending on how far one has zoomed in. This could be sped up (to a limited extent) using a faster Raspberry Pi. That said, part of the charm of the project is that the frame updates only rarely, and considering the superior power characteristics of the Raspberry Pi Zero, it seemed a natural choice for this project.

The project's aim was not to create something large and attention-demanding, but rather something subtle that could be mistaken for an ordinary picture, if not for the power cord. The natural choice for such a project is an e-ink screen. I picked a 7.5-inch e-ink screen from Waveshare, which is sold with a Raspberry Pi header that makes integrating with the Raspberry Pi Zero a breeze. While the e-ink does add to the subtlety, standard e-ink panels still have a grey hue rather than white that distinguishes them from, well, paper. Leaning on a past summer job working in a print shop, as well as advice from friends with better taste than myself, I decided to try to give the frame a more retro look to mask the tech-vibes of the e-ink colours. I paired the screen with a comparatively large 18 × 24 cm light-wood picture frame which, when paired together with a white picture mount, would give the frame a more classic look. Other than aesthetics, the picture mount also provides padding between the glass of the frame and the e-ink panel, which makes me more confident in the durability of the build. I also configured Linux to start the script when the Raspberry Pi's operating system boots.

The build itself is very simple, and can be done with the aforementioned pieces

Left
Only minor modifications to the picture frame were necessary

"THE BUILD ITSELF IS VERY SIMPLE"

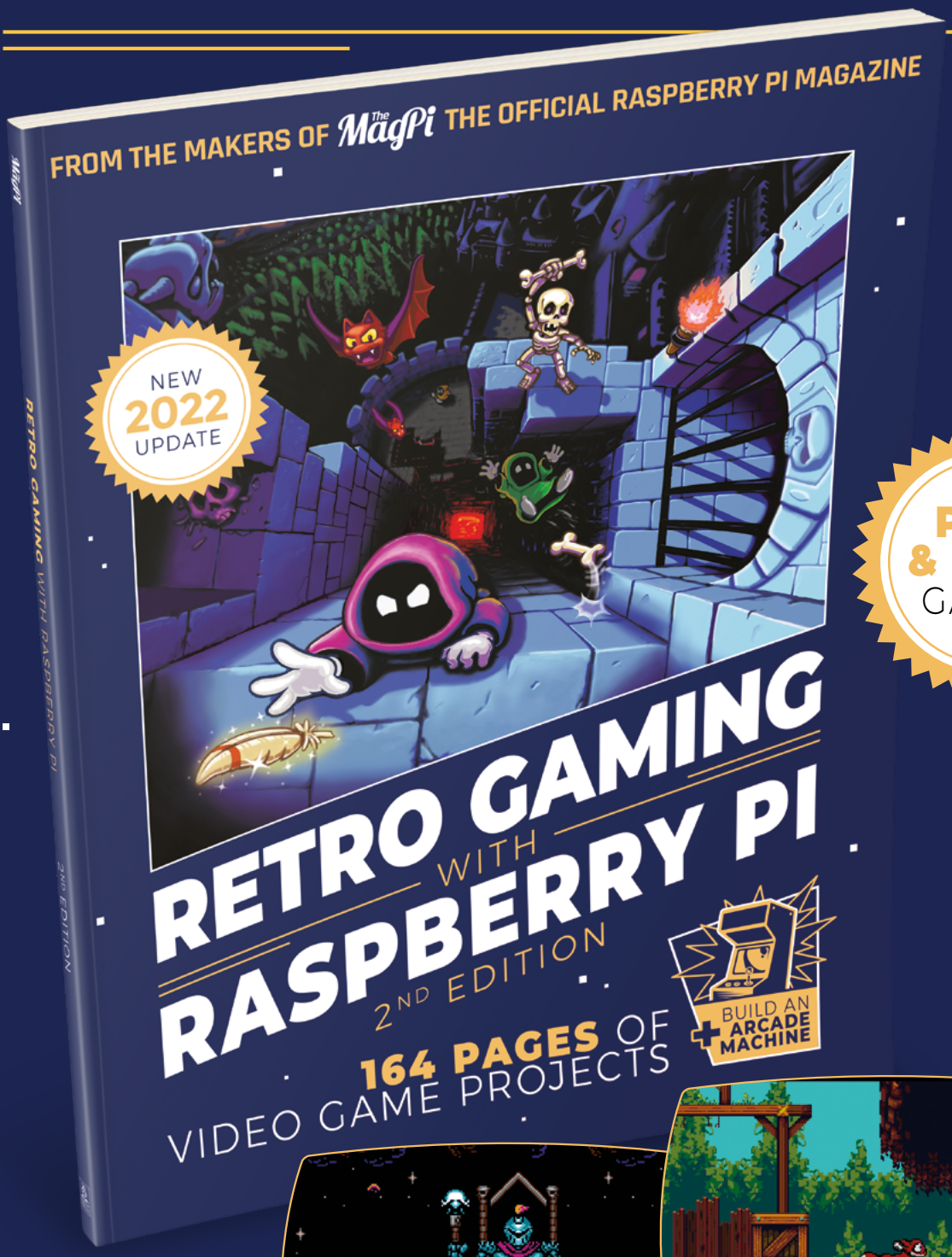
of hardware, as well as a knife (ideally a penknife, but anything sharp will do). Simply insert the picture mount inside the picture frame, and then place the e-ink panel on top. Make sure that it's aligned – if you'd like, you can use tape to keep it in place, but it's not strictly necessary. You'll need to cut a little slit in the back of the cardboard for the e-ink panel's cable to go through. Most picture frame backs are made of cardboard or some kind of fibre, so this shouldn't be too difficult. The slit doesn't need to be large, but ensure that the angle on the frame isn't too sharp. Now all you need to do is insert the back of the picture frame, and squeeze the tabs tightly so that the panel doesn't slip. The mount provides a cushioning for the panel here – if you're not using a mount, I'd recommend taping the panel instead and only pushing in the tabs lightly.

The project is done, for now. In the future, I'd like to improve the exploration and iteration-approximation algorithms further so you're more guaranteed to see an interesting image. I'd also like to experiment with the new colour e-ink panels, as the Mandelbrot set can look stunning with some simple colouring.

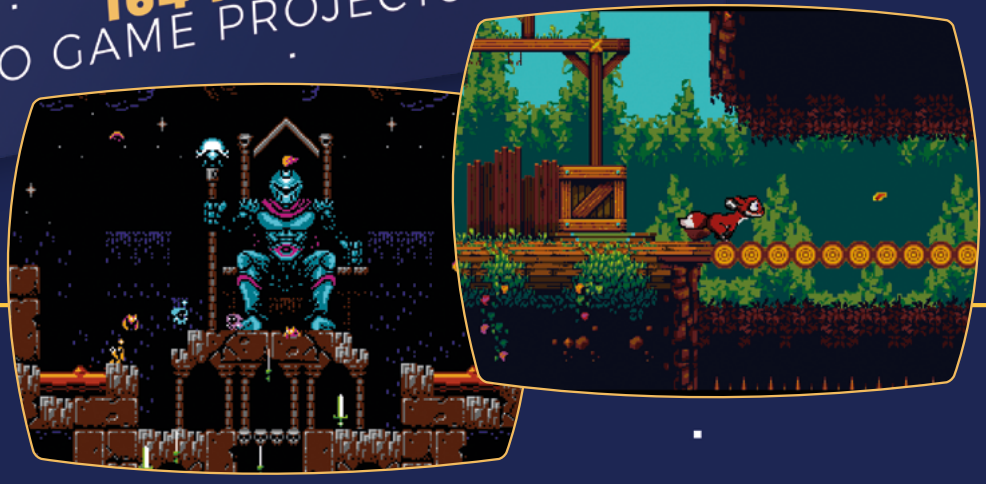
The last challenge was picking a spot for the finished frame. I couldn't decide between the office or my bedroom, but chose the latter. Now, I wake up to new artwork every day, and it's quite exciting when I catch it changing. It reminds me of the intrinsic fun factor in computer science, and why I decided to pursue this degree to begin with.

Above
The frame, as well as the Raspberry Pi Zero, now have a home on my pinboard

Above Left
Good thing that this side is facing the wall



PLAY & CODE GAMES!




T:8 s
F/11
iso
1.8EV
400

HackSpace magazine meets...

VEEB Projects

Design studio in Basel, Switzerland

For a small mountainous country divided between Italian, French and German [and Latin] speakers, Switzerland punches well above its weight in terms of design. Univers, Frutiger, and the mighty Helvetica are all Swiss; HR Giger, the creative brain behind the look and feel of the *Alien* films, was Swiss; even the Swiss flag is neatly minimalist. If you're designing in Switzerland, you're standing on some very large shoulders.

Two such designers are English duo Vanessa Bradley and Martin Spendiff. As VEEB, the duo have been creating devices sometimes simple, sometimes complicated, but always beautifully clean. We caught up with Martin late last year to find out what they're all about. →

INTERVIEW

HACKSPACE Are you designers first and foremost, or engineers, or something else? I ask because your projects are unusually pretty. A lot of Raspberry Pi projects have wires sticking out and bare PCBs exposed, but yours always look so clean.

VEEB We aren't. Our backgrounds are IT (Vanessa) and Mathematics (Martin). The physical form that a project ends up taking is one of the most challenging bits (for us, it is anyway).

Appearance is often tied to the workflow. The 'cyberdeck' (**github.com/veebch/boostbox**) we made a while ago was a good example. It's the 80/20 rule. Getting to a working version was quick and easy, but it took about another year of experimenting with tools that turned a novelty into something that we found genuinely useful. It now gets used for emailing (NeoMutt), writing (Neovim) and compiling MicroPython for some of the hardware we sell.

HS What first drew you to using Raspberry Pi?

VEEB When it first came out, it was excitement at such a small and affordable tool for doing things with, but, over time, it became much more about the community. We tried using <insert name of other fruit> SBCs every now and again, but found that the software was a little too bleeding-edge for our limited skills, and the software was getting in the way of doing stuff.

The real epiphany was the Pico microcontroller, which made us suddenly look at things and think 'we could make one of those.' It also taught us to be better at using tools that were more suited to addressing the thing you're trying to do. There was a case, when we were building something recently, when we realised that the best thing to use was just a simple switch... no microcontrollers, no relay, just a one-dollar switch. I suppose that it's a version of 'just because you can, doesn't mean that you should.'

HS You do a lot of upcycling, adding new functionality to machines that should, by rights, be obsolete. What is it about old objects that inspires you to work with them?

VEEB Three reasons:

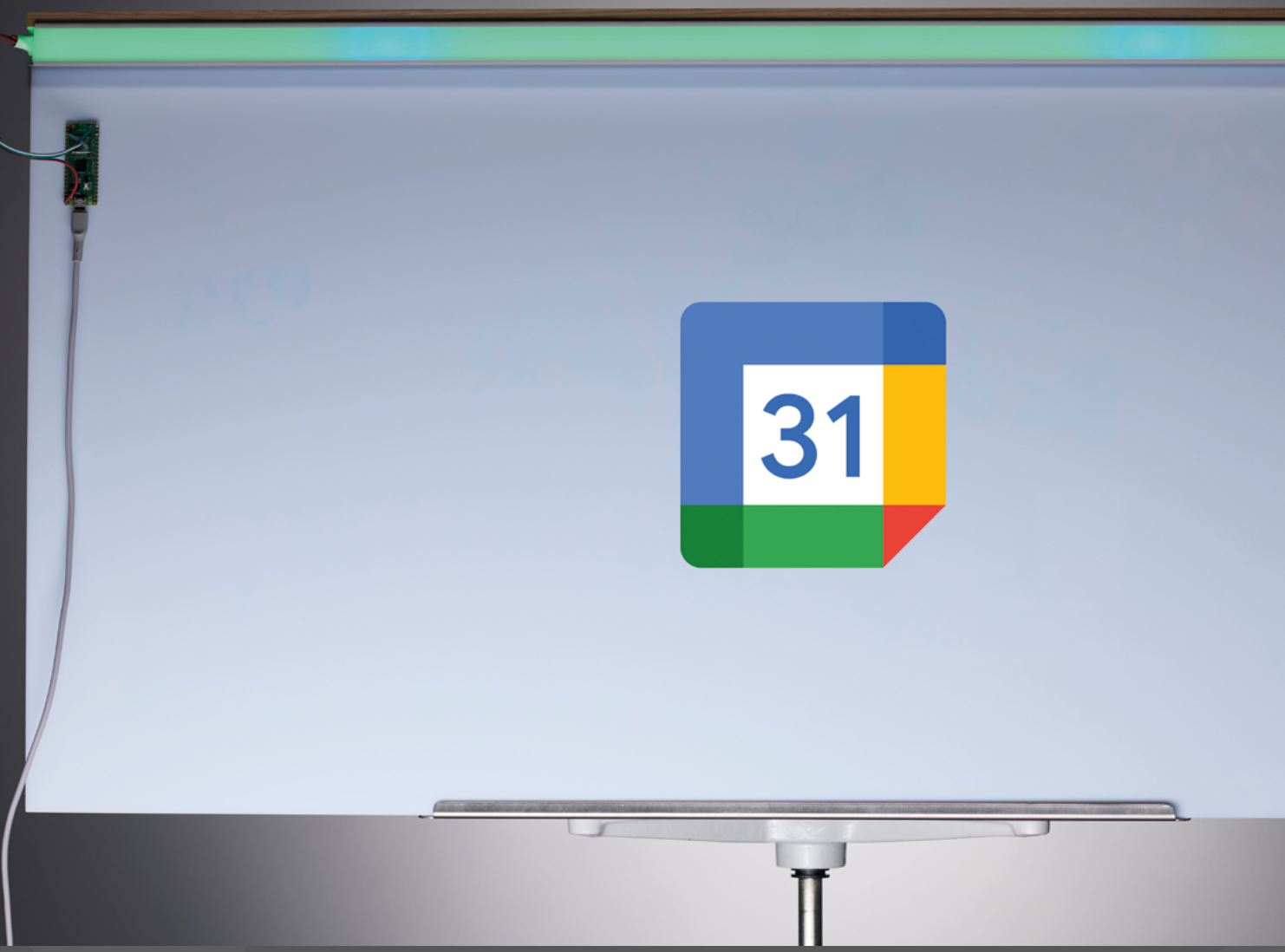
- Maybe it's an age thing, but the constant cycle of upgrading means that you can end up chasing the latest and greatest thing while ignoring what went before. A lot of innovation is working around limitations that exist at the time, and there are some really smart ideas contained in old objects. Those smart ideas should be augmented with new tools, not abandoned.
- There's also an element that is just about waste. There's an old video of someone making a toaster from scratch (**hsmag.cc/ToasterFromScratch**), and it really underlines how much effort goes into making 'things'.
- *Blade Runner* was a really cool film. The retro-futuristic tech in that was pretty spectacular. If we can make things that have that aesthetic, we'd be more than happy.

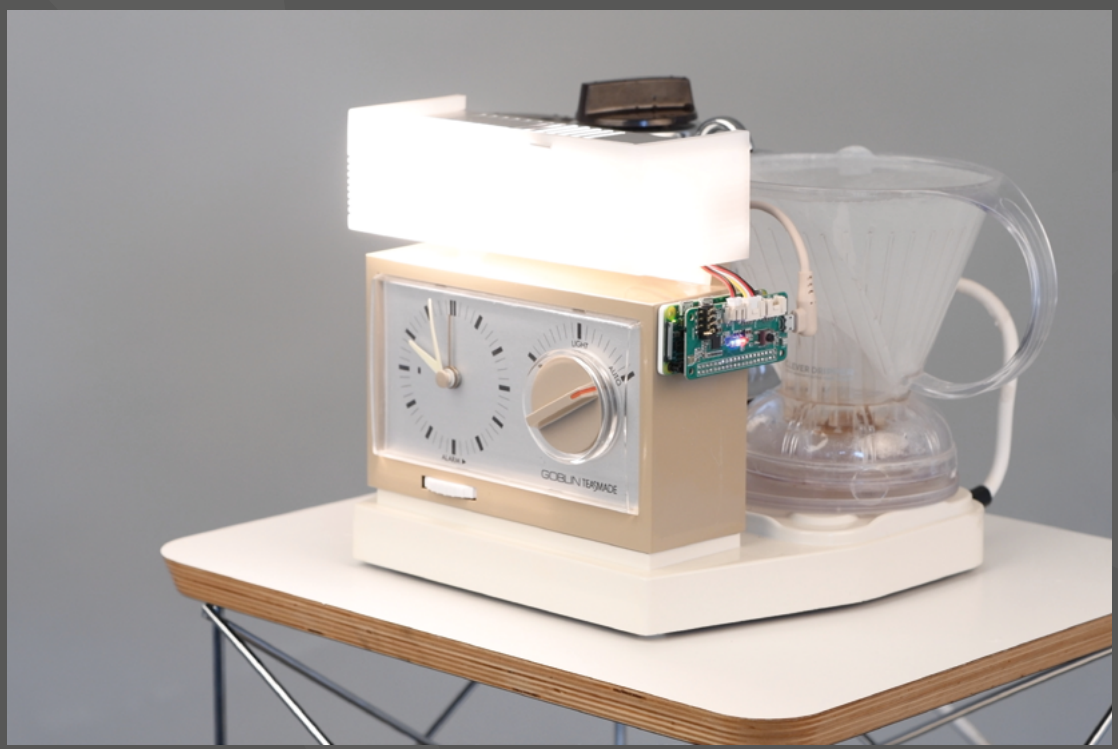
HS Everything you make seems to do one thing, and one thing well. Is that on purpose? →

Left
Endless wisdom, supplied by the r/Quotes subreddit

Right
Infrared photography the pretty way


HN-23
JAPAN

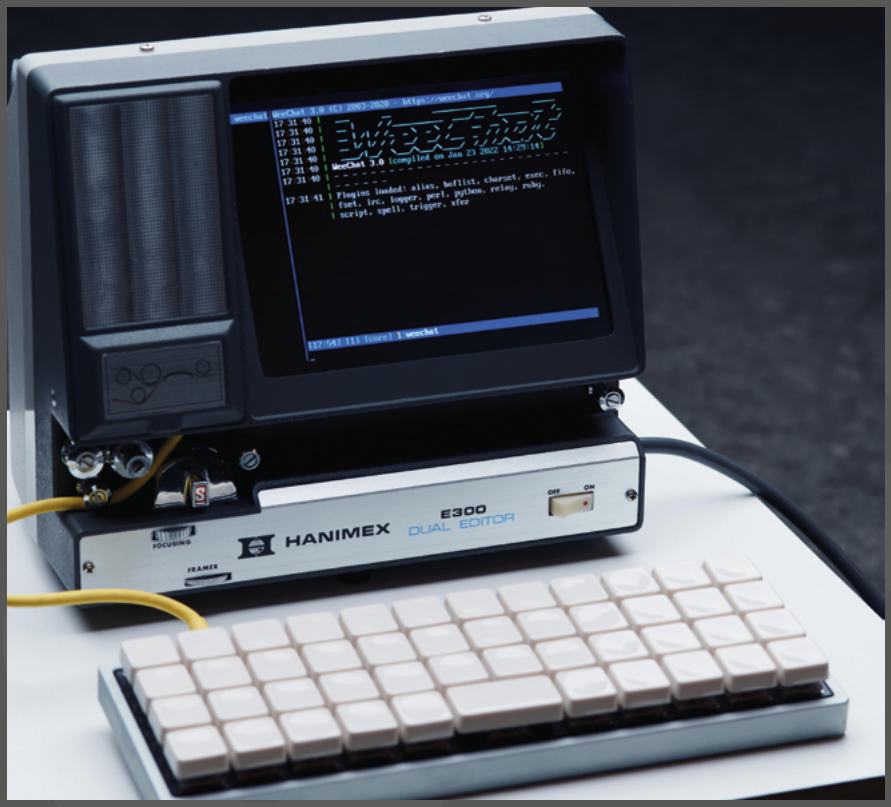




Left
This adapted Teasmade plays *God Save the Queen* when it's time for a brew

Above
Track the progress of the day using a Pico W and some neopixels

Above Right
The Boostbox was originally a project to turn an old Super 8 viewer into a YouTube viewing terminal – it turned into a fully fledged computer



VEEB We buy into the first item of the Unix philosophy which was written in a Bell System technical journal back in 1978:

"Make each program do one thing well. To do a new job, build afresh rather than complicate old programs by adding new 'features'."

Maybe our version would include adding new features to old things that would otherwise be redundant, but that's the general idea.

HS You rehabilitated a German railway clock using a ferrite antenna attached to a Raspberry Pi Pico. That's way beyond most people's ability. Where do you start with knowing how to do this? Is this deep ham radio knowledge?

VEEB Definitely not. That was a rabbit hole that appeared upon wondering how the clocks at train stations worked. We live in a country that is obsessed with things being on time, and just ended up talking to the people that have an encyclopaedic knowledge about that stuff. Pick any subject and, no matter how obscure it is, there's most likely a forum on the internet, full of experts who are keen to grumpily share their expertise.

> We live in a country that is obsessed with things being on time, and ended up talking to the people with an encyclopaedic knowledge about that stuff

HS Also, it's incredibly cool, but most people wouldn't bother with the cool part. What inspired you to take the time from a (presumably Cold War) radio signal coming from an atomic clock?

VEEB It's probably the uncool part! The clocks were really interesting to us because they were gorgeous objects, and there was a time signal there, so we just needed the 'glue' to join that signal to these redundant old things to get them rejuvenated and useful again.

HS You document your build processes on GitHub. Do you get people adding improvements/ suggestions to the way that you do things?

VEEB The best bit about adding things on GitHub is when people use your idea to do something new or better.

There has been more than one person that has made massive progress bars (**github.com/veebch/hometime**) for their classroom. Knowing that people are looking at your code is a good way to make sure it's not *too* terrible. ◻

Objet 3d'art

3D-printed artwork to bring more beauty into your life

The year is 2024. Humans have been subjugated, ruled over by parasitic oblongs that feed on our emotions, trading dopamine for electricity. It is the year of the smartphone.

Some fight against the inevitable – see page 14 to see how Guy Dupont tames the emotional blackmail of the smartphone. Some accept their fate. Matthew Ghost has decided that if he must have a smartphone pinging up notifications every so often, it might as well be in his eyeline on his desk so he can take a quick look, rather than suffer the physical distraction of getting the thing out of his pocket and prodding it to find out which app is craving his attention this time.

He's done a brilliant job of it too. This example is printed in Prusament PLA Prusa Galaxy Black, and the adjustable tilt mechanism makes it a great example of a simple, functional print. □

hsmag.cc/TiltPhoneStand

Letters

ATTENTION ALL MAKERS!

If you have something you'd like to get off your chest (or even throw a word of praise in our direction), let us know at **hsmag.cc/hello**

BARBOT

O.M.G. The cocktail robot in issue 74 is amazing. I can't believe I've never thought of this before. I currently have a cupboard full of weird and wonderful drinks that I bought at one time or another, but never use because I can't remember what to do with them. With one of these, I can just program in all the drinks and then the robot can serve me up whatever strange drinks my heart desires. I suppose I could also get a notepad and write down the recipes, but a robot seems like more fun.

Dave
Glasgow

Ben Says: **Glad you enjoyed it. Cocktail bots have been around for a while, but this one is a nice compact build. Who needs a notepad when you can over-engineer a solution? Actually, that should be the title of this magazine.**

REPAIR

I'm totally onboard with Jude Pullen's fight to repair things. It's an absolute disgrace that it's so hard to keep the things we buy working.

Zac
London

Ben Says: If you'll allow me to stand on my soapbox briefly. Personally, I think that the current disposable culture is a major world problem that needs to be tackled by co-ordinated action at every level. A big part of this has to come top-down from the government. It could be as simple as forcing companies to honour longer warranties which, in turn, would make things more durable. Or it could be more complex, such as introducing a tax on new parts while simultaneously introducing financial incentives for fixing things. The net result of this would be that we're not paying more for our stuff because, even though new things are more expensive, repair is cheaper and more effective.

GLASSES HOLDER

This might sound a bit over the top, but the 3D-printed chimpanzee skull glasses holder might be the coolest thing I've ever seen. Seriously. It's just a shame I don't wear glasses. Maybe I could get a pack of those really cheap reading glasses just to have something to put on my new chimp skull?

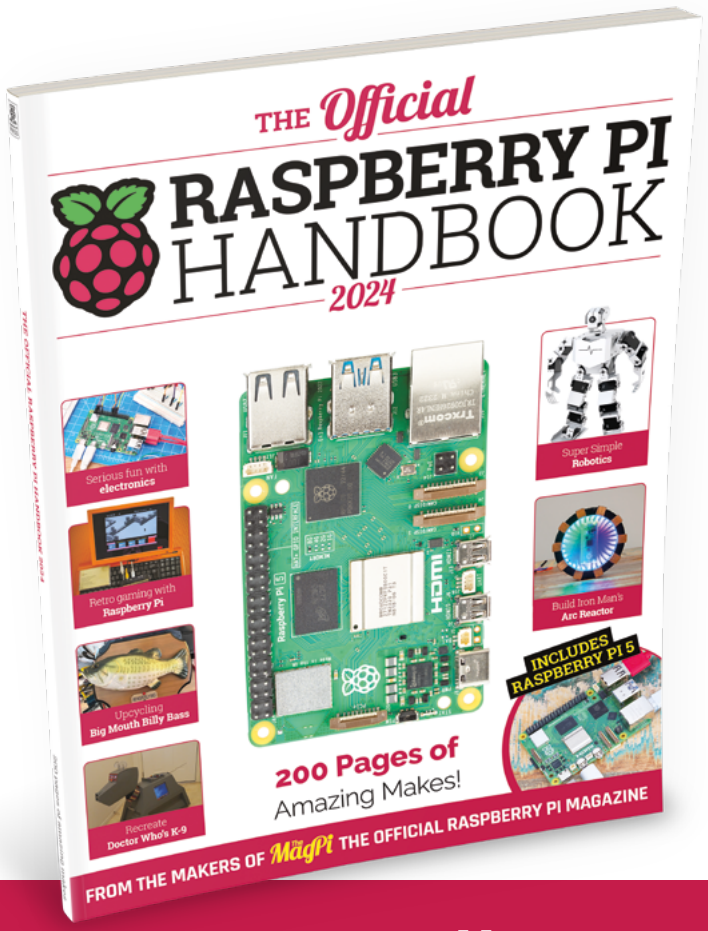
Jane
Norwich

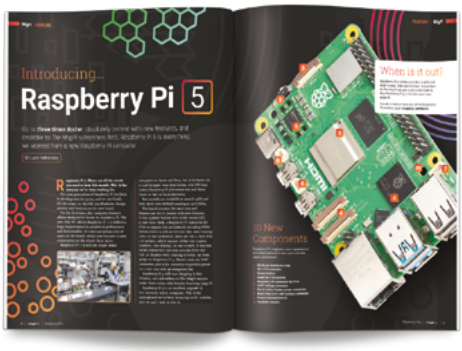
Ben Says: If you think the chimpanzee skull glasses holder is cool, you're not going to believe how it looks with a pair of sunglasses on. Obviously you're free to pick whatever sunnies you like, but pop a pair of aviators on this thing and bask in the glow of the coolest object to grace this planet.

THE *Official* RASPBERRY PI HANDBOOK 2024

200 PAGES OF RASPBERRY PI

- QuickStart guide to setting up your Raspberry Pi computer
- Updated with Raspberry Pi Pico and all the latest kit
- The very best projects built by your Raspberry Pi community
- Discover incredible kit and tutorials for your projects





Buy online: magpi.cc/store

HackSpace
TECHNOLOGY IN YOUR HANDS

FORGE

HACK | MAKE | BUILD | CREATE

Improve your skills, learn something new, or just have fun tinkering – we hope you enjoy these hand-picked projects

Gain complete control of your 3D printer

Ditch the slicer and design models in Python and G-code

Ben Everard

Ben's house is slowly being taken over by 3D printers. He plans to solve this by printing an extension, once he gets enough printers.

When you learn 3D printing, you find out about the different types of files that are used. First, there are files that contain details of the objects – usually STL files. We slice these files to produce the second type of file, G-code, that tells the printer what to do and how to move.

This is a good way of doing things, and it means that it's easy to print identical objects repeatedly on a large number of machines. However, it does have a downside. It means that we are at the mercy of our slicers. We can only print things in the way they are sliced. It doesn't have to be like this, though. Our printers are far more capable than our slicing software would have us believe.

FullControl GCODE takes a different approach. It does away with the 3D model file entirely. Instead, we write Python code to create the G-code directly. We'll look at exactly how to do this a bit later on, but let's get stuck in and see what we're able to do.

NO MORE LAYERS

Slicers work by breaking a design into layers (or slices) and assembling your final object like this. However, there's nothing inherent in your 3D printer that means it has to work this way. It's an extruder that can move in three dimensions. Obviously, there are some restrictions physics imposes on us about what happens when molten plastic comes out of an extruder, but these aren't quite as severe as slicing software would have you believe.

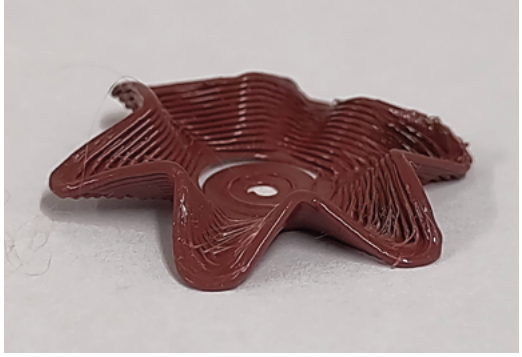


Figure 1
This spacer doesn't have layers in the traditional sense – the print head moves round while going up and down

Let's take a look at the spacer design of **Figure 1** at **hsmag.cc/NonplanarSpacer** (you can manipulate and modify the design in your browser).

In traditional terms, this is one layer. The print head moves around, sticking each line to the edge of the next. However, this single layer isn't flat. It's wavy because the print head moves up and down as it goes around.

FIT THE GEOMETRY

Sometimes it makes sense to design the G-code output for the particular geometry you're printing. A particularly good example of this is nuts and bolts. It's perfectly possible to print nuts and bolts using conventional slicing. However, the results can be mixed. The thread moves upwards at a slight angle, but the layers remain stubbornly horizontal. What should be a smooth thread has a series of contours from the layer lines.

With FullControl, we can print in a spiral. This is much like vase mode on a traditional slicer but, as we have more control, we can match the angle of the spiral to the angle of the threads on the bolt, meaning that you get a perfectly smooth thread. **Figure 2** shows an example: **hsmag.cc/NutsAndBolts**.

In this case, the infill design is lined up with how forces are applied when screwing and unscrewing, so you get a lot of strength for very little filament.

NEED FOR SPEED

Slicing is a general-purpose approach that isn't necessarily the best way of printing a particular design. Taking a specific approach can sometimes be faster. A simple example of this is vase mode. This custom way of slicing is a very quick way of

Figure 2
The star pattern in the bolt's head gives strength in exactly the directions needed

printing large things quickly. However, not all models can be done in vase mode. There's another method called 'snake mode', which is similar but prints a wall. The print head moves one way, then moves up a layer and moves backwards in a zig-zag pattern. This prints a whole other class of objects very quickly, but isn't available in any slicer – but we're not using a slicer! Take a look at the example at **hsmag.cc/SnakeModeSoapDish** for one model that prints this way.

RIPPLES IN THE POND

By default, slicers try to give you the smoothest possible surface for your print. This is often what we want, but not always. An example is the 'fuzzy mode' some slicers offer, where they imitate a furry texture. However, we can be far more creative than this if we create our own G-code. The ripple texture demo creates a surface that ripples, and you can adjust the way it does this (or indeed, you can create whatever surface finish you like).

You can see this in action in **Figure 3**, which is taken from **hsmag.cc/RippleTexture**.

GOING OUR OWN WAY

So far, we've looked at code that other people have created. There are a few parameters we can adjust, but it's basically pretty fixed. Let's take a look now at how we can create our own designs.

It's a little late for Christmas, but this fir tree model we created for a little winter cheer gives a good introduction to creating your own models (**Figure 4**).

We wanted to create a pine branch effect by drooping loops of filament down the side of a cone. It's not impossible to design something like this as a 3D model that can be sliced, but it's not very easy,

Figure 3
You can make the outer surfaces any texture you want

as you have to persuade the slicer to interpret a particular model in a particular way. However, with FullControl, it's easy. If you want filament to droop, just move the extruder to the position you want and squirt filament.

FullControl is a Python module that you can install and run locally. However, if you just want to have a play around with it, the easiest way of getting started is with a Google Colab workbook. Log in at **colab.research.google.com** (you'll need a Google account) and click to create a new notebook.

In Colab, you can build a notebook out of blocks. Each block can either be code or text. Code blocks can be run at any time by clicking on the arrow next to them. Each block runs in the same Python environment as the others, so if you import something, or create a variable, you can use it in the other blocks. The code is run in the order you run it, so a block earlier in the notebook is not necessarily run before a block lower in the notebook – you →

USE AT YOUR OWN RISK

It is possible to damage your printer with G-code, and there's no guarantee that the code output from FullControl won't do some harm. It's possible that it could try and move your printer beyond its limits, overheat it, or otherwise cause a problem.

Modern 3D printers are pretty good at detecting any movement that could damage them and not doing it, but if you're going to use FullControl, you have to accept that the power you have to manipulate the code comes with this risk.

You have been warned, so don't blame us if anything goes wrong.

have to click on the Run arrow to actually run a block. However, you can automatically run all the blocks sequentially by going to Runtime > Run all.

That is a very brief overview of what Colab can do, but it should be enough for you to run our notebook. If you just want a Christmas tree, you can go straight to a completed notebook at **hsmag.cc/xtree**. Because of the way Google handles permissions, you may need to go to this in a private browsing tab.

Our Colab notebook has a Python environment, but this doesn't have FullControl installed, so the first thing we need to do is create and run a code block to install FullControl:

```
if 'google.colab' in str(get_ipython()):
  !pip install git+https://github.com/
FullControlXYZ/fullcontrol --quiet
```

You'll need to run this each time you get a new runtime. This means each time you visit the site or each time the runtime disconnects (you'll get a warning message pop up if this happens).

Figure 4
By extruding above empty space, we can get filament to droop down

With FullControl, you build up a list of steps. Each step is generated by a FullControl method, and when you're finished, you can convert these into G-code.

Our Christmas tree is:

```
import fullcontrol as fc
from math import tau
import random
steps = []
for j in range(360):
  for i in range(60-int(j/6)):
    steps.append(fc.polar_to_point(centre=fc.
Point(x=100, y=100, z=(j*0.2)), radius = 15-
(j/24), angle = (i)*(tau/(60-int(j/6)))))
    if random.randint(0,11) == 10 and j>10:
      steps.append(fc.polar_to_point(centre=fc.
Point(x=100, y=100, z=(j*0.2)), radius = 15-
(j/24)+10, angle = (i)*(tau/(60-int(j/6)))))

fc.transform(steps, 'plot', fc.PlotControls(neat_
for_publishing=True, zoom=1))
```

We use two loops: an outer loop that increments each layer, and an inner loop that ticks around to create a circle. With each iteration of the inner loop,

MACHINE-SPECIFIC G-CODE

FullControl G-code is machine-specific, as most G-code files are. There are preset defaults for some printers, but not all.

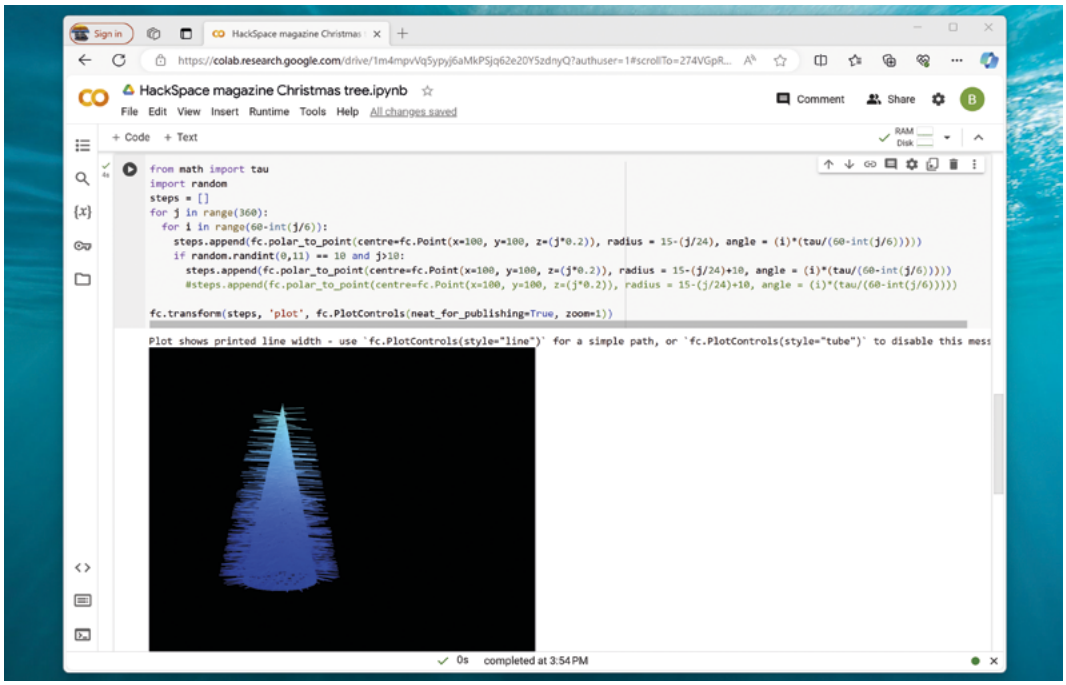
There is a subreddit at **reddit.com/r/FullControl** where you might find other users who have the same printer as you.

In truth, there's not a lot of difference between the G-code for different printers. Obviously there are limits on the size of the print volume, the speeds that can be reached, etc. You'll need to ensure that your code respects these. Other than this, there are slightly different initialisation functions that run at the start of the printer.

For example, on the Prusa MK4, you need to tell the printer the size of the object so that it does the mesh bed levelling in the right way.

You can find out the initialisation G-code by slicing an object as you normally would and taking a look at the G-code produced. You might need to copy and paste some of this over into the G-code produced by FullControl.

You can do this in a more robust way by creating your own printer profile. Take a look at the existing printer profiles in the GitHub repository to find out where to start: **hsmag.cc/FullControlXYZ**.



Left
You can run FullControl in your browser via Google's Colab programming environment

we do one of two things. Either we draw a line segment to the next point or – one out of every eleven times – it draws an extra line an additional 10 mm out of the side. Since there's nothing below this line, it will sag down against the side of the tree.

We build up the steps with a list of **`Point`** objects which you can create in two ways. You can do it directly with code such as:

```
fc.Point(x=100, y=100, z=(j*0.2))
```

However, most of the time we use the **`polar_to_point`** helper function. This lets you use polar coordinates, which is really useful for round objects such as this. The method takes the following parameters:

- **`centre`** – The origin of the polar coordinate system as a Point object
- **`radius`** – The distance from the centre
- **`angle`** – The angle that you travel out from the centre.

It returns a **`Point`** object that corresponds to the correct place in 3D space.

The final line in the code block creates a 3D render of the print, but it doesn't attempt to predict how the filament will flow once it's out of the printer. For that, we actually have to print it.

Each time you run the code, you'll get a different output because it's based on random chance, so each Christmas tree is unique.

If you're happy with your code, you can run the following to download it:

```
from google.colab import files
printer=prusa_i3

gcode_controls = fc.GcodeControls(
    printer_name=printer,
    initialization_data={
        'primer': 'front_lines_then_y',
        'print_speed': 200,
        'nozzle_temp': 210,
        'bed_temp': 60,
        'fan_percent': 100,
        'extrusion_width': 1,
        'extrusion_height': 0.2})
gcode = fc.transform(steps, 'gcode', gcode_
controls)
open(f'{filename}.gcode', 'w').write(gcode)
files.download(f'{filename}.gcode')
```

You'll see that we selected **`prusa_i3`** as the printer. This is for Prusa machines before the MK4. Other options are generic, **`ultimaker2plus`**, **`prusa_i3`**, **`ender_3`**, **`cr_10`**, **`bambulab_x1`**, **`toolchanger_T0`**, **`toolchanger_T1`**, **`toolchanger_T2`**, **`toolchanger_T3`**. If your printer isn't one listed, then take a look at the box, opposite, on how to get your printer working.

We've covered the very basics of how to get started with FullControl; next issue, we'll delve a little deeper into how to create some weird and wonderful prints.

Part 01

Raspberry Pi mecanum robot

Create a robot which can go forwards, backwards, sideways, diagonally, and turn on the spot. The mecanum wheels allow the robot to navigate the tightest of spaces

MAKER

Stewart Watkiss

Also known as Penguin Tutor. Maker and YouTuber that loves all things Raspberry Pi and Pico. Author of *Learn Electronics with Raspberry Pi*.

penguintutor.com

twitter.com/stewartwatkiss

Take your Raspberry Pi on the move by designing and making a wheeled robot This tutorial will explain how you can design and make your own robot. Start by designing your own chassis and mount mecanum wheels with full direction control. The robot will then need four separate motors individually controlled with a H-bridge driver for each wheel. Take control by learning how to use AntiMicroX to control the robot with a gamepad.

01 Mecanum wheels

Mecanum wheels are a type of omnidirectional wheel which can be used to move a robot vehicle in multiple directions. They are particularly good at getting into tight spaces as the wheels can propel the robot forwards and backwards as normal, but can then also move diagonally or sideways without any forward or backwards movement.

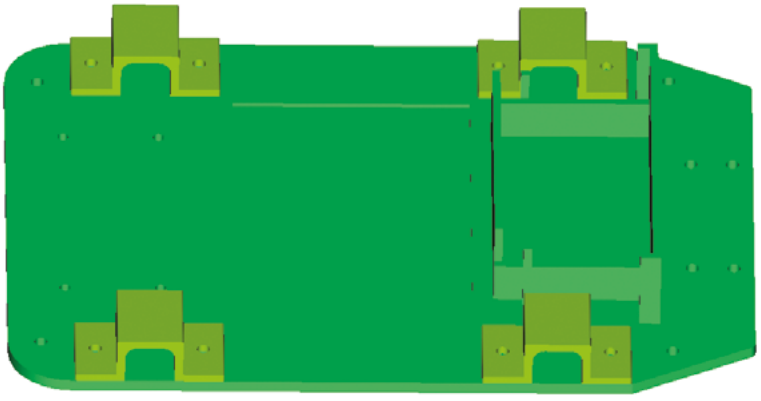


Figure 1: Robot chassis design created in FreeCAD. Includes four motor brackets which are printed separately

This is achieved by having rollers angled around the wheel. The rollers of diagonally opposite wheels need to be in the same direction. Turning the wheels in a certain combination will determine the direction. This is easiest to understand through watching the video on YouTube: **magpi.cc/mecanumrobotyt**.

02 Creating a chassis

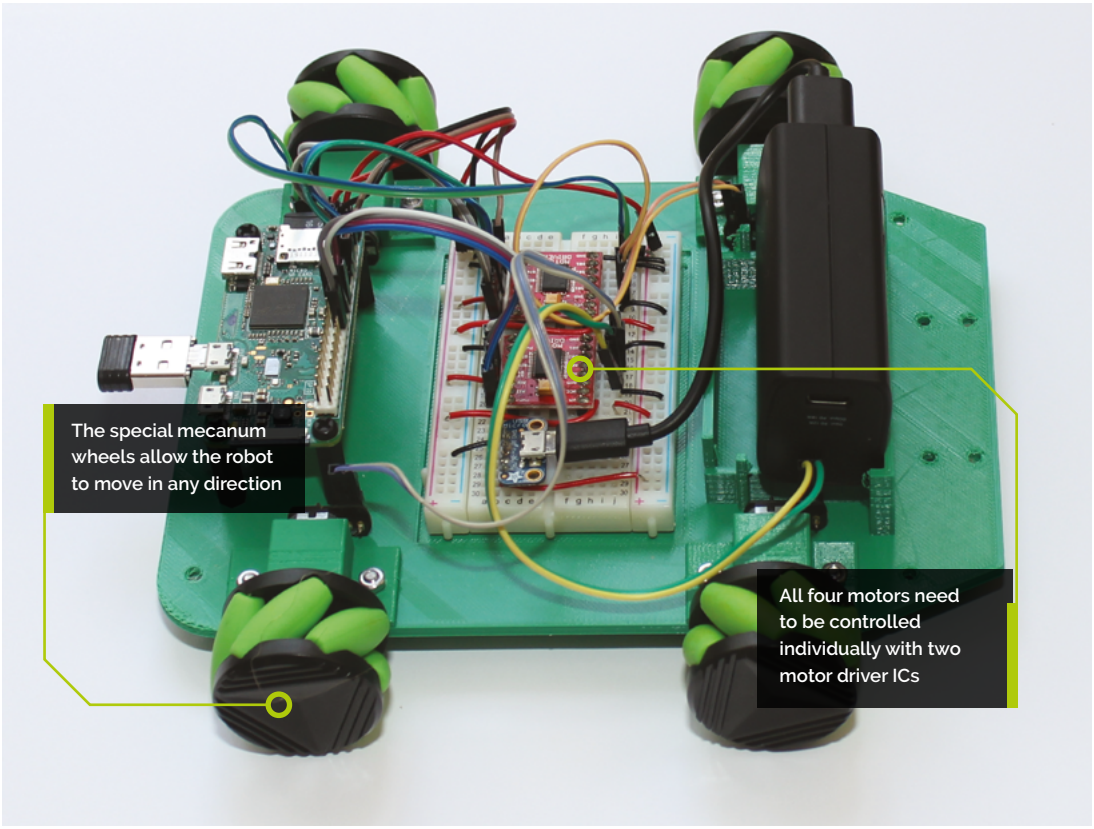
For any kind of vehicle, you will need a chassis to mount the motors and electronics. You can use any method you choose. The base should be thin enough to accommodate the size of the wheels, but otherwise most materials can be used. You could use plywood, acrylic sheet, or even thick cardboard.

If you have access to a 3D printer then you can design your own 3D-printed chassis, or use the one available at **magpi.cc/robotmecanum**. This is shown in **Figure 1**.

You will need some way of fixing the motors. Brackets are included in the 3D printer files, or you can purchase suitable brackets which are included in the component list.

03 Raspberry Pi Zero W

Whilst any model of Raspberry Pi can be used, a Raspberry Pi Zero W or a Raspberry Pi Zero 2 W are well-suited for a robot vehicle. They are small, making them easier to fit on a mobile robot, and they also have lower power requirements than some of the other models. The wireless capability is useful when programming the robot

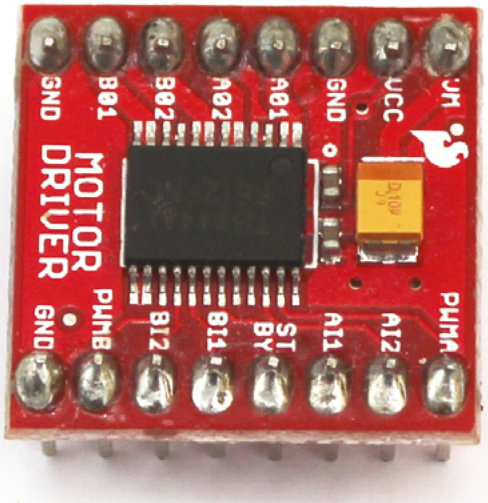


and for getting the controller working. A micro-USB to USB adapter is needed for connecting the controller. A shim adapter is also recommended.

04 Powering the robot

The motors are designed for 6V, whereas a Raspberry Pi needs a 5V power supply. In practice, you can normally use a single power supply for both your Raspberry Pi and the motors.

The different options are a 6V power supply (4 × AA batteries) or a USB power bank. The breadboard layout shows both options. Raspberry Pi is powered through GPIO 2 using a diode which protects against reverse voltage and drops the voltage slightly. If using the USB power bank and the voltage is too low, then the diode bypassed if required. In either case, power must NOT be connected to your Raspberry Pi power input socket when powered through the GPIO.



▲ **Figure 2:** The motor driver is an SMD IC soldered onto a breakout board, allowing it to be used with a breadboard

05 H-bridge motor controller

To allow the motors to go both forwards and backwards, they each need an H-bridge. A good choice is the TB6612FNG driver. This driver can handle up to 1.2A of current, and has two H-bridge circuits on a single SMD integrated circuit. To be able to use these with a breadboard, they are available on a SparkFun motor driver board. This is shown in **Figure 2**.

Along with the inputs for forwards and backwards, each H-bridge needs a PWM signal to set the speed of the motor. A single PWM output can be used from your Raspberry Pi, which will ensure that all the motors run at the same speed.

You'll Need

- 2 × TB6612FNG motor drivers **magpi.cc/TB6612FNG**
- Mecanum wheels **magpi.cc/mecanumwheels**
- 4 × geared motors **magpi.cc/gearmotor**
- Motor brackets **magpi.cc/motorbracket**
- USB power bank **magpi.cc/nanowave**
- Micro USB shim **magpi.cc/microusbshim**
- Wireless gamepad **magpi.cc/wirelessgamepad**
- Micro USB breakout board **magpi.cc/usbbreakout**

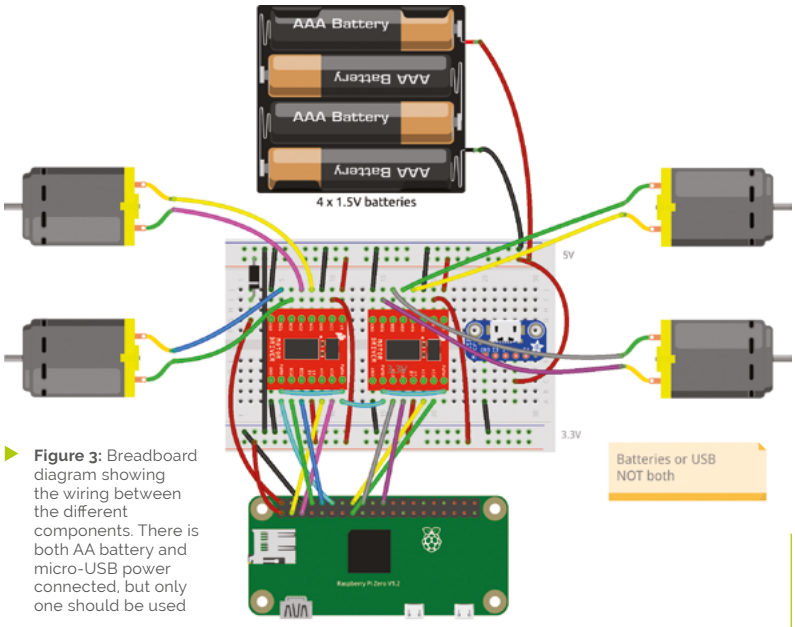


Figure 3: Breadboard diagram showing the wiring between the different components. There is both AA battery and micro-USB power connected, but only one should be used

06 Wiring up the circuit

The wiring diagram is shown in **Figure 3**. This is just a case of wiring up the motor driver inputs with appropriate pins from your Raspberry Pi. Note that the breadboard has two different power rails. The top rail is connected to the 6V or 5V supply for the motors, whereas the bottom rail is connected to the 3.3V pin on your Raspberry Pi to run the motor driver at 3.3V (so that it is at the same voltage as the GPIO pins).

A schematic diagram is shown in **Figure 4**. Note that the labels beginning with D indicate connections to the motor driver input, and M is used for the motor connections.

07 Setting up the Raspberry Pi

When setting up your Raspberry Pi, you may find it easier to enable VNC, which you can do through the Raspberry Pi Configuration tool. This will allow you to connect from another computer using a VNC client. Using VNC, you won't need to connect a keyboard or mouse, allowing the only USB port to be used for the USB dongle for the gamepad controller.

If using a game controller, then the AntiMicroX software is needed, which can be installed using:

```
sudo apt install antimicro
```

This will be configured later in the tutorial.

08 Programming motor control

The motors are controlled using GPIO Zero. There is an instance of `Motor` for each of

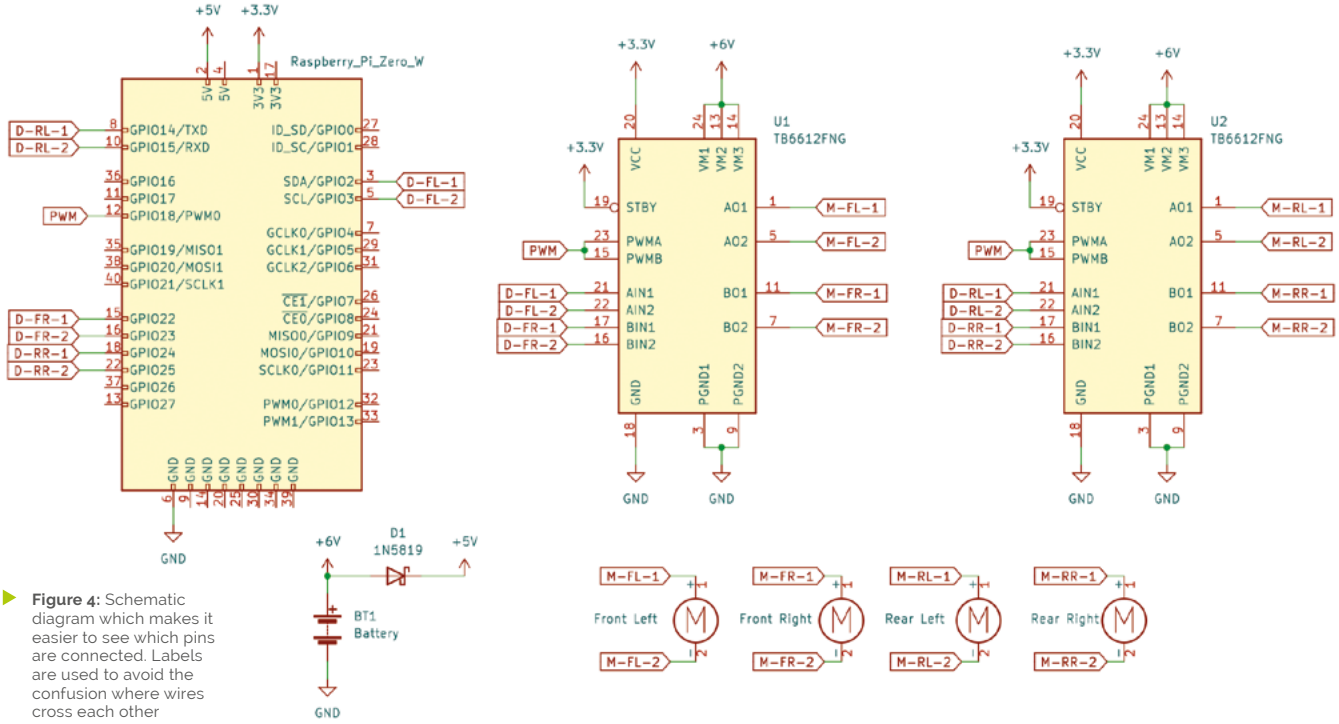


Figure 4: Schematic diagram which makes it easier to see which pins are connected. Labels are used to avoid the confusion where wires cross each other

Figure 5: A wireless USB gamepad. This is detected by Linux as an Xbox 360 controller

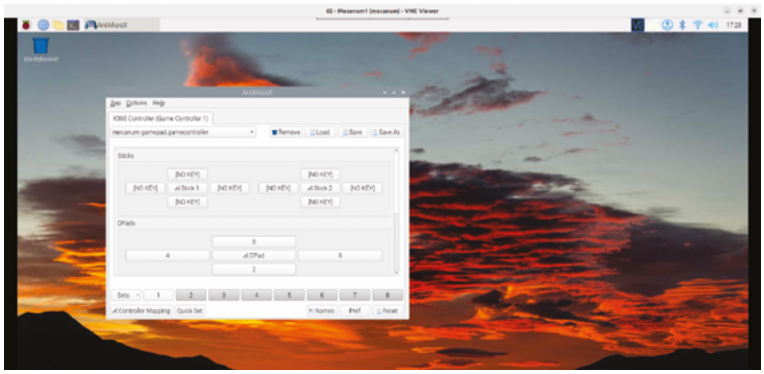
the motor drivers with a pin for forwards and reverse. There is then one more pin for the `PWMOutputDevice` which is used to control the speed of all the motors.

The dictionary `direction` holds the direction of all four motors based on the selected direction. These are `1` for the motor to go forward, `-1` for the motor to go in reverse, and `0` to stop. It does not provide speed control, which is included in the code available from the GitHub repository at **magpi.cc/mecanumrobotgit**.

09 Adding keyboard support

One thing about Python is that it doesn't directly provide a way to read input characters except when the **ENTER** key is pressed. There are various different ways around this, the one used here is a `getch()` function which provides similar functionality to the C/C++ `getch()` function. This will pause the `while` loop until a key is pressed, and then return the key into the `ch` variable.

The `direction` dictionary provides mapping between the numerical keys and the selected direction.



A wireless USB gamepad will be used, which needs to be mapped to the appropriate keys. An example gamepad is shown in **Figure 5**.

10 Using gamepad with AntiMicroX

AntiMicroX is an application which converts gamepad instructions into keyboard or similar instructions. Start AntiMicroX from the accessories menu. This will add a game controller icon to the top right of the screen. Clicking on that will allow

Figure 6: AntiMicroX converts game controller instructions to key presses. It can be configured using the GUI application, pressing the controller will highlight the appropriate button

you to map each of the buttons to the relevant keys, and this is illustrated in **Figure 6**.

You can map each button on the gamepad, or import the suggested layout which is included in the GitHub repository.

11 Troubleshooting

The gamepad must be recognised by Linux to work with AntiMicroX. Some controllers appear to work consistently, whereas others need reconnecting once or twice before they will work with AntiMicroX.

If the gamepad doesn't work after a reboot, then you may need to try disconnecting and reconnecting the USB dongle after AntiMicroX is running. Then, choosing Update Joysticks from the menu.

12 Future upgrades

The robot can be controlled using a game controller, or using a keyboard.

With some extra electronics and code, the robot can do more. There are holes included in the front of the chassis which can be used to add an ultrasonic distance sensor to detect and avoid nearby objects. Or, you could add a line sensor to allow your robot to follow a line drawn along the floor.

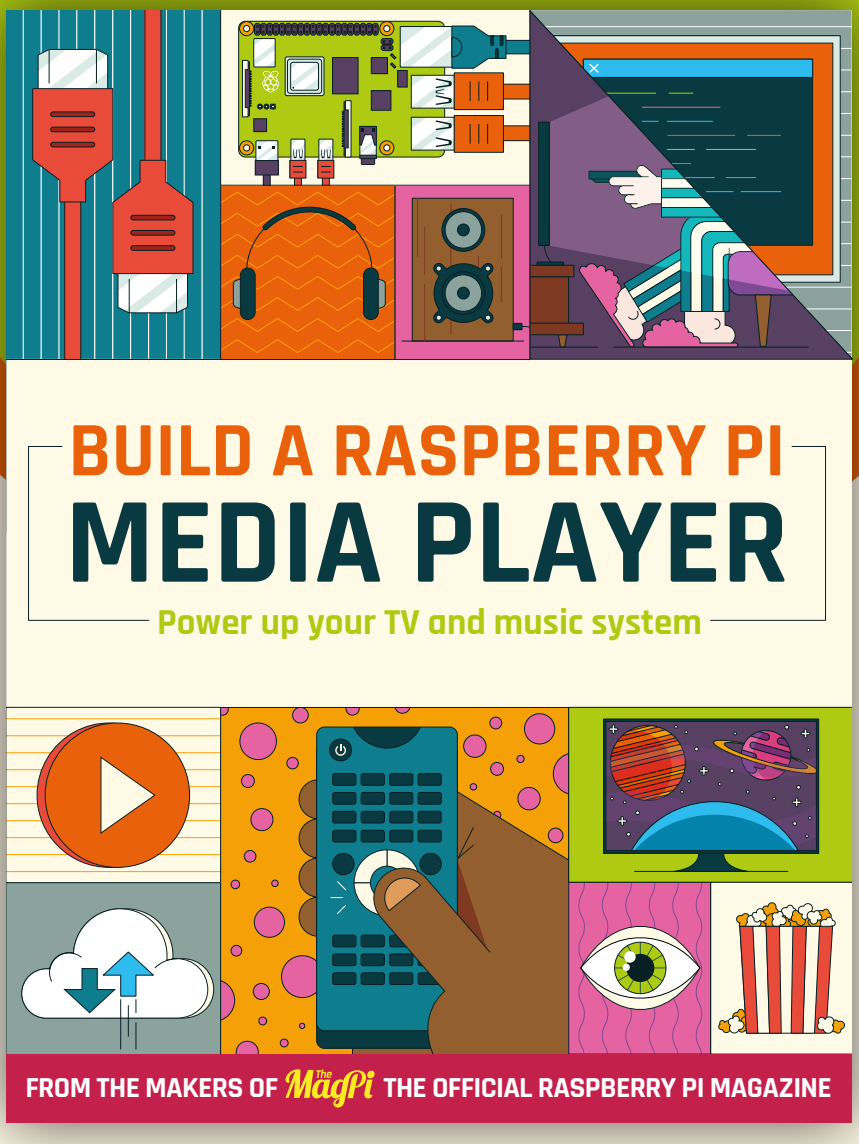
mecanum1.py

DOWNLOAD THE FULL CODE: magpi.cc/mecanumpy

> Language: **Python 3**

```
import sys, tty, termios
from gpiozero import PWMOutputDevice, Motor

motors = [
    Motor(2, 3, pwm=False),    #Front left
    Motor(22, 23, pwm=False), # Front right
    Motor(14, 15, pwm=False), # Rear left
    Motor(24, 25, pwm=False)  # Rear right
    ]
pwm_out = PWMOutputDevice (18)

# get a character from the command line
def getch() :
    fd = sys.stdin.fileno()
    old_settings = termios.tcgetattr(fd)
    try:
        tty.setraw(sys.stdin.fileno())
        ch = sys.stdin.read(1)
    finally:
        termios.tcsetattr(fd, termios.TCSADRAIN, old_settings)
    return ch

# list to convert key into motor on/off values to correspond with direction
direction = {
    # number keys
    '1' : (-1, 1, -1, 1),    # Turn left
    '2' : (-1, -1, -1, -1), # Backwards
    '3' : (1, -1, 1, -1),    # Turn right
    '4' : (-1, 1, 1, -1),    # Left
    '5' : (0, 0, 0, 0),      # Stop
    '6' : (1, -1, -1, 1),    # Right
    '7' : (0, 1, 1, 0),      # Diagonal left
    '8' : (1, 1, 1, 1),      # Forwards
    '9' : (1, 0, 0, 1)       # Diagonal right
}

current_direction = "stop"
# speed is as a percentage (ie. 100 = top speed)
speed = 50
pwm_out.value = speed/100

print ("Robot control - use number keys to control direction")
while True:
    # Get next key pressed
    ch = getch()
    if (ch == 'q') :                    # Quit
        break
    elif (ch in direction.keys()) :     # Change direction
        for i in range (0, 4):
            if direction[ch][i] == -1:
                motors[i].backward()
            elif direction[ch][i] == 1:
                motors[i].forward()
            else:
                motors[i].stop()
        print ("Direction "+ch)
```

Your FREE guide to making a smart TV



magpi.cc/mediaplayer

TUTORIAL

Laser-cut rubber stamps

Carve out images using pure light

Ben Everard

Ben's house is slowly being taken over by 3D printers. He plans to solve this by printing an extension, once he gets enough printers.

Rubber stamps are different things to different people. To some, they're a labour-saving device – why write the same thing over and over again when you don't have to? To others, they're a way of adding a bit of colour to personalised objects. To children, they're a way of adding a level of detail to a picture that they may not yet have the coordination to add otherwise. In some countries, they're still used in place of signatures on official documents.

However you plan to use one, though, the process for making one is the same. You need to start with a flexible material, and remove any bits that you don't want to transfer ink onto the paper. Technically, the material doesn't have to be flexible, but you'll get a more even ink transfer if it is.

Above
When stuck on wood, it's easy to handle these and stamp them anywhere you want to add a little detail

When it comes to removing material, there's one common hack- or maker-space tool that's quicker and more accurate than most: the laser cutter. With one of these, you can quickly blast away unwanted material to leave you with a perfect rubber stamp.

You do have to be a bit careful about what material you use. While there are loads of softish, plasticky materials around, some of them can be unpleasant to laser-cut. Some simply smell awful when burned, but others are outright dangerous. The easiest solution here is to stick with materials that are labelled as laser-safe, and many laser cutter stockists sell laser-safe rubber for precisely this purpose. The biggest decision is the thickness of the sheet you use. We used a 2.3 mm sheet – this let us laser off about 1 mm and still leave a decent thickness.

The technique is really simple, with one potential pitfall. It is a pitfall that everyone makes, so don't worry about it when you fall into this particular pit. We'll warn you about it right at the start, but this won't stop you from making the mistake. We're pretty sure it's impossible to laser-cut rubber stamps without making this mistake once. The thing these warnings do is make you feel more foolish for making the mistake. Sorry about that.

REMEMBER TO MIRROR THE IMAGE AS STAMPS COME OUT BACKWARDS.

With that warning out of the way (don't worry, you'll forget), let's take a look at the process.

This really does depend a bit on the laser cutter software you use. Our laser cutter runs on Lasercut 5.3. This software is quite old and janky, but it works. The easiest way to etch with this software is

Left
We could have cut these out on the laser cutter, but it seemed easier to cut them out by hand than dial in the cut settings

to import a black-and-white image. Anywhere black will be removed.

To do this etching, then, we need an image that's white where we want the ink to remain, and black everywhere else. This is slightly inconvenient because almost every source of clip art that we could base our designs on is black where we want the ink to go and white everywhere else.

You can manipulate your images using whatever image software you're comfortable using. We opted for the GNU Image Manipulation Program because it has the ability to both invert and flip images.

First, acquire your black-on-white images, either by creating them yourself or by finding what you want in a stock image or clip art gallery. These can be text, pictures, or whatever you like.

Once you have your image the way you want it, you need to invert it. In the GNU Image Manipulation Program, go to Colours > Invert. Then, flip it by going to Tools > Transform Tools > Flip. Now you can export it as a PNG file that Lasercut 5.3 can import.

Finally, you can import this into the laser cutter and etch it out. You might need to adjust the power settings a bit to get it right. We'd recommend starting low and then increasing because you can always do a second pass if it's not deep enough, but you can't put material back on if you take too much off.

> We're pretty sure it's impossible to laser-cut **rubber stamps without making this mistake once**

Now you've got your rubber out of the laser cutter, you can cut out the bit you want using a pair of scissors and glue it onto a bit of wood to act as a handle. Dab it in some ink, tap it onto some paper, and realise that you forgot to flip the image. Finally, go back to your image editing software, flip the image, and try again. At least you know the correct power settings this time.

The real challenge with these stamps isn't the process of creating them – once you've dialled in the power, then it's fairly straightforward. The challenge is coming up with interesting and unique designs. We'd love to see what you come up with.

TUTORIAL

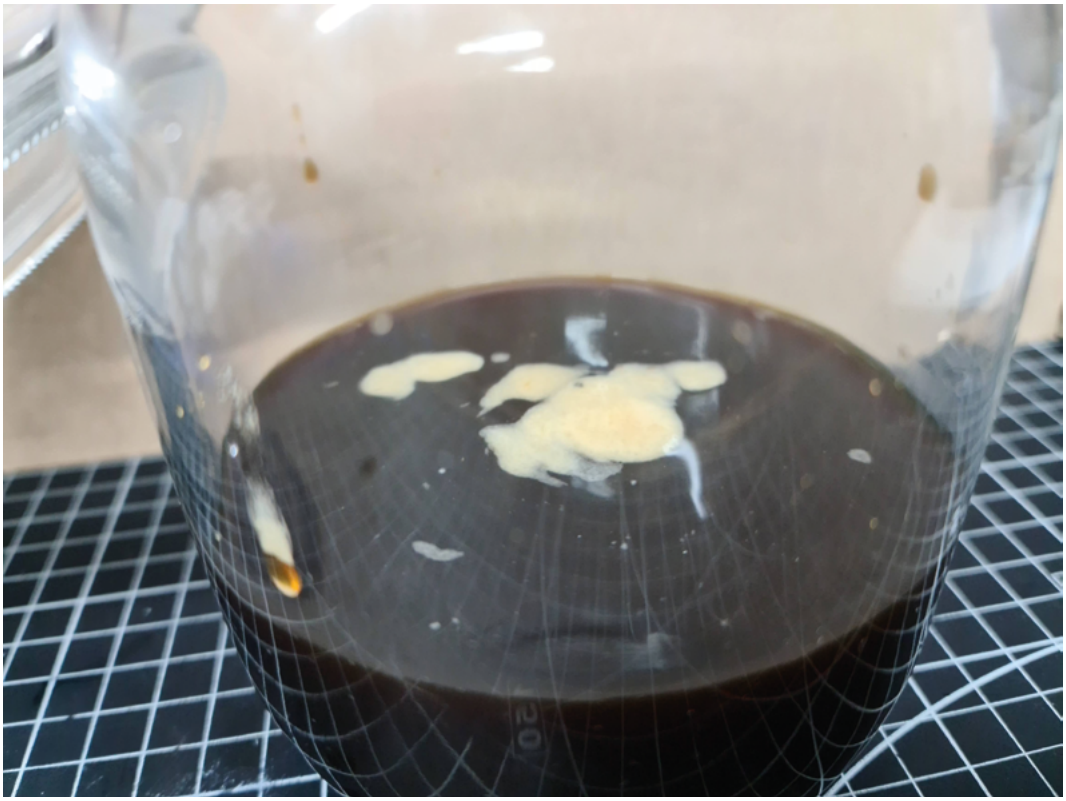
Instant coffee camera chemistry

Use household chemicals to develop film

Dr Andrew Lewis

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

Thirty years ago, most people came back from a holiday, or a special event, with a collection of little plastic containers. Those containers would get put into an envelope and taken to the local pharmacy, where you'd fill out a form and hand over some money to the person behind the counter. A few days later, you'd collect another envelope with your (or occasionally someone else's) photographs in, and then you'd visit your friends and bore them to tears by showing them all, one after another. A few years later, digital cameras came of age. Social media gave us the opportunity to put photographs directly onto the internet and bore all of our friends at once from hundreds of miles away. In the wake of these technological advances, the interest in traditional photography reduced, and the number of services available for developing traditional 35mm or 120mm film shrank down to almost zero. At the same time, the cost of film processing and the chemicals themselves has increased. In this article, you'll learn the basics of caffenol-based black-and-white film processing, mixing your own photographic chemicals from ordinary household items.



YOU'LL NEED

- **Instant coffee** (not decaf)
- **Soda crystals** (sodium carbonate, also called washing soda)
- **Vitamin C powder**
- **A glass container with a lid to mix and store the developer**
- **A developing tank** (commonly a Paterson tank) to load and process the film

> **While none of the chemicals we'll be using are particularly harsh,** gloves and eye protection are advised

PROCESS IN STAGES

Film processing has a few discrete stages. You might have seen these referred to as Develop, Stop, Fix, and Rinse. Caffenol replaces the chemicals in the developing stage, but you still need to do the other stages to get usable negatives. Stopping the film is fairly self-explanatory, and is the process of stopping the developing process from progressing any further. This can be done very precisely using a chemical stop bath, or it can be done by simply removing the developer from the processing tank and rinsing away any residue from the film with water. The fixing process strips off any remaining light-sensitive chemicals from the film, stopping it from reacting to light. Fixing is usually done with sodium thiosulfate (also called hypo) or ammonium thiosulfate. If you're feeling particularly brave, you can use brine to fix the film, but this can take days to process and isn't always successful.

DEVELOP YOUR PHOTOGRAPHY SKILLS

Mixing your own film chemicals is a very rewarding process, and it's one of those projects where you can let go with your creative side and experiment to get unique results. In this article, you'll learn the basic recipe for caffenol developer, but you could exchange the coffee in this recipe for tea, juice, or even beer if you prefer. While none of the chemicals we'll be using are particularly harsh, gloves and eye protection are advised to keep you safe.

Begin by mixing two heaped tablespoons of cheap coffee with about 150ml of warm water, and stir it until the coffee is dissolved. Let the water cool to room temperature. Add one heaped tablespoon of soda crystals to 200ml of water and stir it until the crystals dissolve. Get comfortable, this will take a while. Mix the two solutions together, and then add a generous teaspoon of vitamin C powder. Stir the mix until everything dissolves.

Pass the whole mixture through a fine coffee filter to remove any undissolved solids, then let the mixture settle undisturbed for 24 hours. Check for any solids at the bottom of the mixture. If you do see any silty residue in the container, gently pour off the top ¾ of the mixture into another container. Discard the last ¼ of the mixture with the residue in.

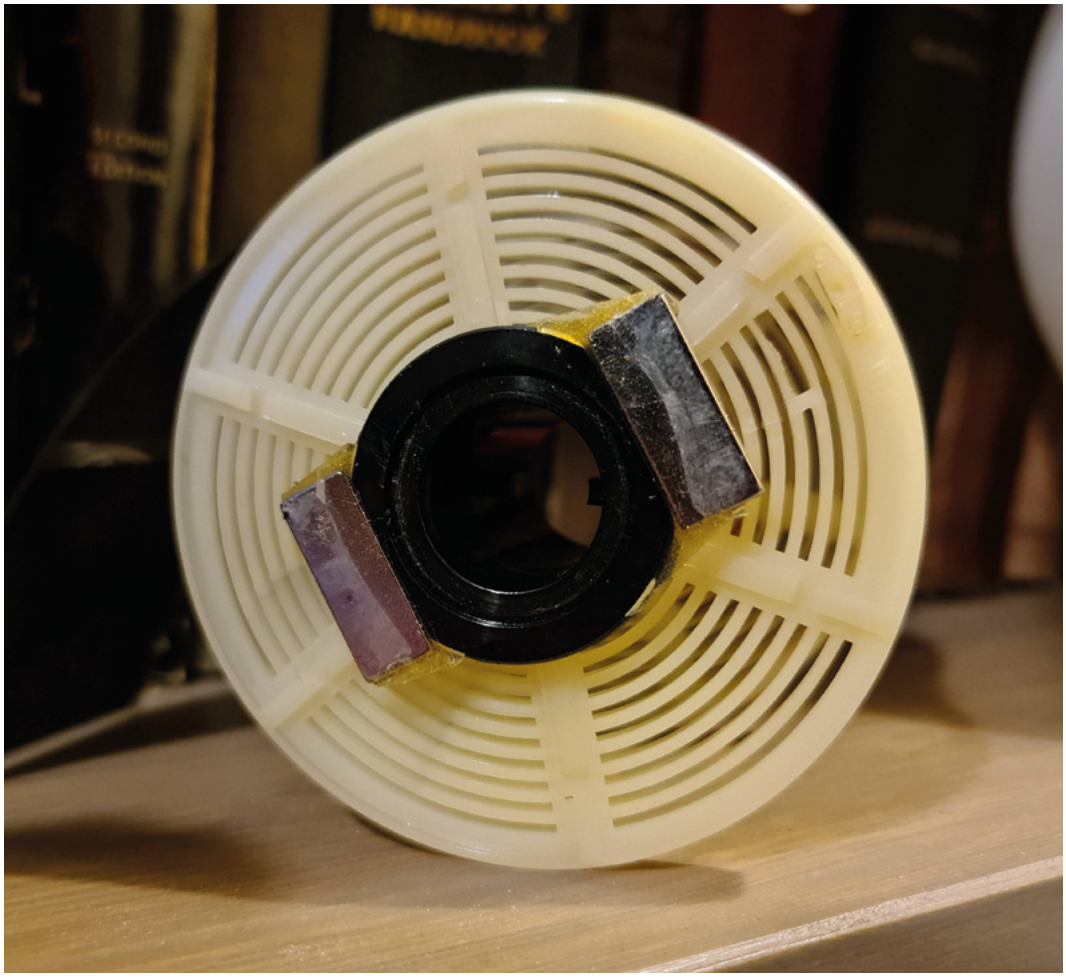
Get into your film adventure by taking a few black-and-white photographs, then load the film into a →

Above
A few basic ingredients from the hardware store and kitchen are all you need to develop your own black-and-white negatives, provided you already own a Paterson tank. The tank shown here can process two 35mm films at once, or one 120mm film

Left
It's normal for the developer to fizz a bit when you add the vitamin C powder. It's also the point when the mixture starts to smell awful

QUICK TIP

Developing negatives can be a stinky process. Work in a well-ventilated area, and be prepared to explain strange smells to any visitors.



Right
If you're going to be processing a lot of films, you can add some magnets to the bottom of your spool and build your own magnetic stirrer using a low-rpm motor and some matching magnets outside the tank. If you're only processing a few films, this isn't really worth the time, but if you've got a lot to get through, it can be a big time-saver

SHUTTER AND EXPOSURE

If you're confused about how to set up your camera to take a good photo, there are plenty of light meter apps for mobile phones that will tell you the correct shutter and exposure setting for the ISO film you are using. If you're a bit more old-school than that, you can use the 'Sunny 16' rule. This rule states that the shutter setting in bright sunlight, with the aperture set to f/16, is roughly the inverse of the film speed. This sounds mind-boggling at first, but with a little bit of practice it's easy to get the hang of guessing settings. As an example: if your aperture is set to f/16 and you are using ISO 100 film, then the correct exposure should be about 1/100th of a second. The closest shutter setting to this is usually 1/125th of a second, which should be fine. For well-lit indoors environments, open up to about f/5.6 and apply the same rule. It won't be exactly right, but with computers to manipulate the image after developing, you should get close enough.

Paterson tank. Pour in the developer and gently agitate the tank for ten minutes (assuming you're using Ilford HP5 film and the developer is about 20°C; other films may take longer to process). Paterson tanks usually have a short handle that lets you spin the spool around inside the tank without opening it. Do this slowly. Gentle agitation means that you do not shake the tank or swish the film around from side to side. You absolutely do not want to get air bubbles between the fluid and the film, because this will make the chemical reaction happen unevenly. Just slowly turn the film in the solution until the ten minutes are up.

Pour the developer out of the Paterson tank, and rinse the film thoroughly a few times. Don't remove the lid; the film isn't fixed yet. Empty and fill the tank until the water coming out is completely clear. If you want to stop the developing process immediately, you can use a stop bath instead of water for the first rinse. A 2% solution of white vinegar will do the trick

Left
The results from caffenol developer can be quite impressive, particularly once your image has been processed and printed onto quality paper. If you're very brave, you can have a go at making your own photographic paper. It isn't as impossible as you might think, but it is well outside the scope of this article

– the developer requires an alkali solution to work, so dumping some acetic acid onto the film should stop the process immediately.

The film is now developed and rinsed, so all that remains is applying fixer and another rinse. Depending on the type of fixer you use, the time for fixing the film will vary. A standard hypo fix should take about five to ten minutes of gentle agitation. Other rapid fixers might take as little as two to three minutes, depending on temperature and concentration. Rinse the film with water and remove it from the tank. Hopefully, you'll have a roll of exposed negative film that you can cut into strips and hang up to dry.

One of the easiest ways to get your negative photographs reversed and printed onto paper is to scan them into a computer. Some scanners have a back-light built in that makes scanning easy, while others might need an adapter made from some silver card. Fold the card into a right angle that's double the width of the film slide you want to scan, and place it over the slide like a tent.

The light from the scanner will be bounced behind the slide by the reflective card. Once the negative is scanned into the computer, you can use software, like darktable (**darktable.org**), to edit and print the file.

PATERSON TANK

Loading film into a Paterson tank can be a bit awkward if you've never done it before. It's really the most difficult part of the developing process, and requires some practice to get right. It's best to start with a spoiled film and work in the light so you can see what you're doing, then start practising the task with your eyes closed.

The spool from a Paterson tank is designed with two small bearings that grip the edge of the film. You slide the film into the start of the spool, and push it until the bearings grip the film. The spool is designed to work like a ratchet, so you hold one side of the spool steady while you rotate the other side to pull in the film, repeating the action as many times as you need to completely load the film into the spiral channel in the spool. You may need to apply very gentle pressure to the edge of the film as you ratchet the spool, to stop the film feeding backwards.

This sounds easy enough to accomplish, but when you're working blind in a dark room, a changing bag, or underneath a pile of blankets, it can feel impossible the first few times that you attempt it. Have a look at the YouTube video at **hsmag.cc/LoadingFilm** for a demonstration of how to load the film. Once the film is on the spool, put it in the Paterson tank and screw on the lid. The tank is light-fast once the lid is on, so you can then crawl out from your darkened hiding place for the rest of the developing process.

QUICK TIP

Use distilled water for the final rinse of the film, with a wetting agent (a spot of liquid soap, for example). This will reduce streaking and spots forming on the film as it dries.

TUTORIAL

PART 02

Making a connected Hull Pixelbot

Make a network-connected robot you can program from your phone or use in team games such as Robot Rugby

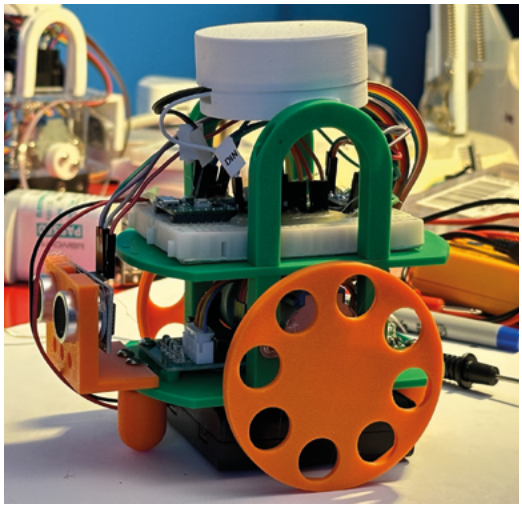
Rob Miles

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

In a previous article, we discovered how to create a little desktop robot called the Hull Pixelbot. Now we are going to give our robot connections. We'll start by making it host a web page we can use to program it, and then we'll move on to look at a network server that can be used to host robot sports events. You can find all the code for the robot and instructions for setup at the GitHub repository here: **hsmag.cc/HullPixelbotGH**.

HOSTING A WEB PAGE IN A ROBOT

A non-networked Hull Pixelbot is programmed via its USB connection. To do this, you plug the robot into a computer with a USB cable and visit this link **hsmag.cc/HullPixelbotEditor** in your browser. This website hosts an editor for program code written in the language 'Python-ish' which can send code to the connected robot. This works, but it is not very convenient. It would be nice if you could program your robot remotely over Wi-Fi. To do this, we can make the robot host an editing website rather than visiting a site on the internet.



Right
This version of the robot is powered by a Raspberry Pi Pico. You can use an Arduino Uno instead

The website in **Figure 1** is hosted by a C++ program running in the robot. The robot is hosting a website on the local network at the address **pico.local**. You can use this site to send new programs to the robot at any time, even if it is running another program. You can also use the site to run and stop the program in the robot and load pre-written sample programs.

CONNECTING UP

Before the robot can host the website, it needs to connect to the local network. The statements below fetch the definitions of the libraries which will be used to make everything work.

```
#include <Arduino.h>
#include <WiFi.h>
#include <WiFiClient.h>
#include <WebServer.h>
#include <LEAmDNS.h>
#include "WebPage.h"
```

Once we have all our libraries available, the next thing is to set up a network connection for the robot. You will have to enter your network details into the program code in the position of the placeholders below before you compile the program and send it to the robot.

```
const char *ssid = "your network name";
const char *password = "your network password";
WiFi.mode(WIFI_STA);
WiFi.begin(ssid, password);
Serial.println("Connecting WiFi");
startBusyPixel(0,255,255);
```

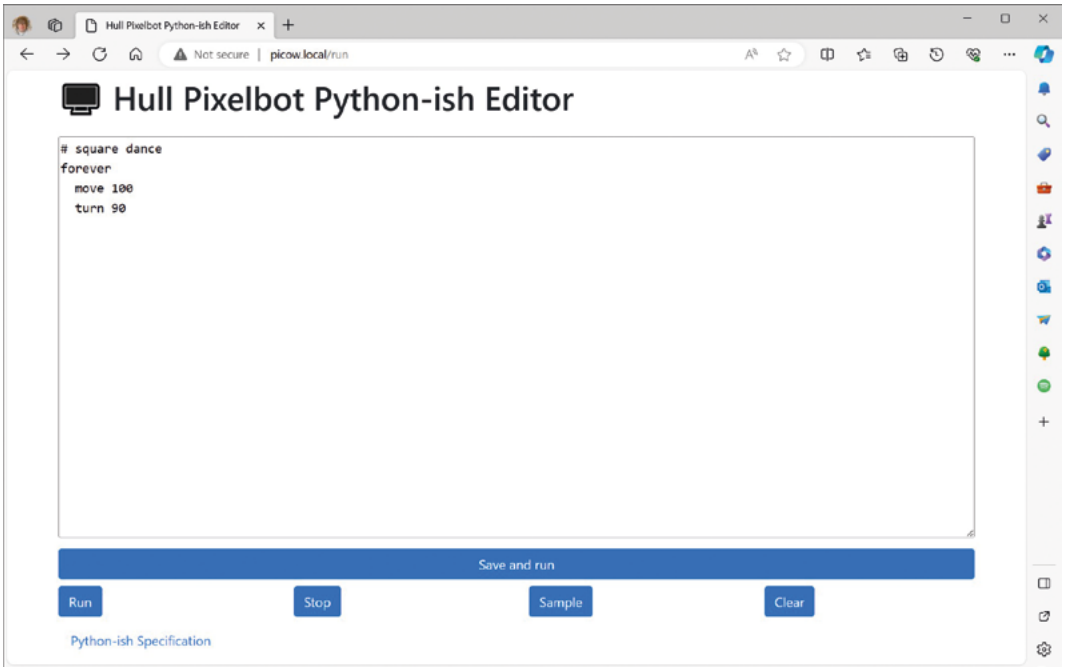


Figure 1
The program being edited makes the robot move in a square. Can you change it to make a robot move in a hexagon?

The code above tells a Pico to become a Wi-Fi station and to connect to the local Wi-Fi network with the given network name and password. The final statement starts a 'busy pixel' display of a moving pixel in the ring of pixels on top of the robot. The three parameters to the `startBusyPixel` function set the red, green, and blue intensity values of the colour of the busy pixel. The pixel above would be cyan (all the green and all the blue). The pixel is displayed during the network connection process to keep the robot user informed of what is going on.

```
// Wait for connection
while (WiFi.status() != WL_CONNECTED)
{
  updateBusyPixel();
  delay(500);
}

stopBusyPixel();
```

The code above waits for the Wi-Fi to connect. Every 500 milliseconds (half a second), the busy pixel is updated to the next position. The user will see a dot move around the ring of pixels. When the connection is made, the while loop ends and the busy pixel is turned off. Now that we have a network connection, we can start running a web server inside the robot.

A SITE FOR SORE EYES

The robot will use the WebServer library from the Arduino framework. The server object is declared at global level so that it can be used by several functions in the robot code.

> Now that we have a network connection, **we can start running a web server inside the robot**

```
WebServer server(80);
```

The code above creates a web server object in the robot code with the highly appropriate name of `server`. This server listens on network port 80, which is traditionally the port for web requests. Remember that what is happening here is that the robot is acting as a web server (the kind of device that you reach when you type **robmiles.com** into your browser). It's a tiny web server that is only visible on the local network (i.e. the network the robot is connected to).

When you try to open a website at a local address (for example **picow.local**), your browser finds a local device with that name and sends a `GET` request using HTTP (Hypertext Transfer Protocol) to port 80 on that device. The web server in the robot then returns a file containing HTML (Hypertext Markup Language) text describing the page to be displayed. In this case, the requested page will contain the HTML for the Python-ish editor home page. →

YOU'LL NEED

- A Pixelbot powered by an Arduino Uno or Raspberry Pi Pico
- If you are using the Uno to control your Pixelbot, you will need an ESP8266 device (the WeMos D1 mini works well) to add the network connection

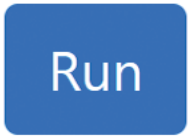


Figure 2
The Stop button is defined in a similar way and has a route which sends the command "*RH" to the robot to make it stop running its internal program

```
server.on("/", handleRoot);
```

Above, you can see the code that runs in the robot and handles the request for the editor home page. You might think that the statement above turns the server on, but this is not quite what is happening here. It is best to think of this as meaning 'on receiving a message asking for something at this route, call this function'. The Pico web server will contain a **server.on** statement for every route that is available on the website hosted by the robot. The root path is the one at the 'top' of the website. The robot web server will also have routes for the GET messages that request the robot to stop and start. The statement above tells the server to call a function called **handleRoot** when a request is made for the route to the home page of the site.

```
void handleRoot()
{
  server.send(200, "text/html", webPage);
}
```

The **handleRoot** function is very simple. It tells the server to send the contents of the string **webPage** back to the browser. The method **server.send** is used to do this. The **send** method accepts three parameters. The first parameter to **send** is the status value of the reply to be sent back. The value **200** tells the receiving browser that the server has managed to find the web page and that the page contents are in the response. This tells the browser it has a page it can display. The second parameter is a string describing the type of the reply. In this case, the item being returned is a text page encoded in HTML. The third parameter is a string containing the text of the web page to be returned to the browser. In the robot code, the page text is stored in a character array variable called **webPage**, which was created by pasting the entire web page file into the program source:

```
const char *webPage = R"(
<!DOCTYPE html>
<html lang="en">
... rest of web page
</html>
)";
```

The code above puts the entire home page for the editor into a single C++ character array referred to by the **webPage** variable. The web page was designed and tested and then incorporated into the C++ code.

> In the robot code, the page text is stored in a **character array variable called 'webPage'**

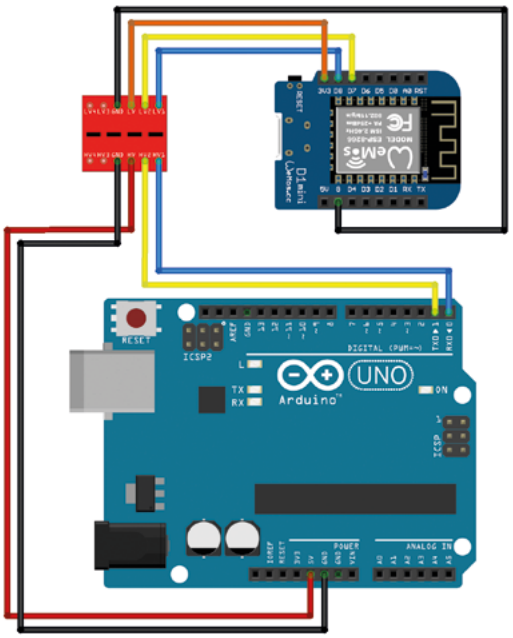


Figure 3
You can replace the Pico W with an Arduino Uno and an ESP8266. You'll need a level converter is used to convert between the 5-volt levels on the Arduino Uno and the 3.3-volt levels on the ESP8266

ROOTING FOR ROUTES

A route is a place that a browser will request something from. There are four different routes supported by the Pico-hosted program editor. We've seen one of them, which requests the home page of the server. Now we can look at the others:

```
server.on("/run", []()
{
  Serial.println("Got a run request");
  sendLineToRobot("*RS");
  handleRoot();
});
```

Remember that the word 'on' in the code above means 'on this route being requested'. The code above uses another form of the **on** method which accepts two parameters: the route that is being requested (in this case, the route **"/run"**) and an 'anonymous' function that runs when the browser requests a page from that route. This makes the code slightly more compact. There are three statements in the function.

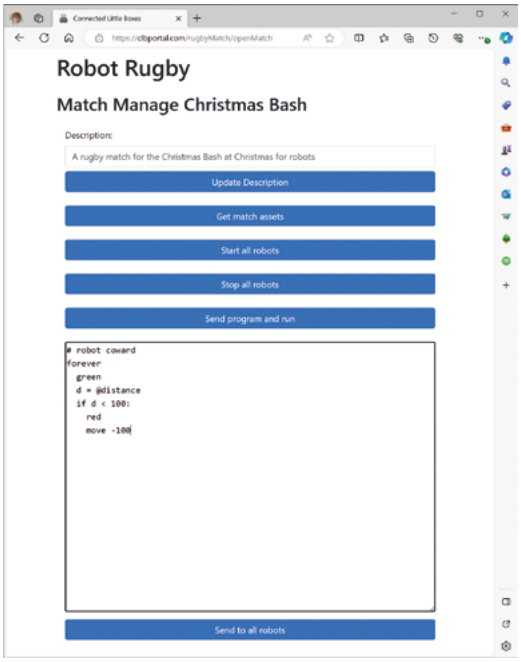


Figure 4
If we click 'Send to all robots', the robots are all turned into cowards

The first prints a message, the second sends the line **"*RS"** to the robot, and the third calls **handleRoot** to redisplay the editor home page. The **"*RS"** message tells the robot to run its stored program. The run route is called from the HTML in the editor when the user clicks the Run button in their browser.

```
<a href="/run" class="btn btn-primary mb-2">Run</a>
```

The code above is HTML text in the web page. The text describes a reference to the **"/run"** route which will be followed when the button is clicked. The browser will render this as a button.

Figure 2 shows how the button is displayed on the web page. When the user clicks the button, the browser will follow the **/run** route, the server will recognise this and the code we have seen above will run to send the command **"*RS"** to the robot. If we wanted to send the robot more commands, we could add more buttons to the edit page which access additional routes. Then we would create route handlers in the robot that respond to these routes.

GETTING THE CODE

The Run and Stop buttons send a message to the robot, but the 'Save and run' button must update the program in the robot by sending the edited text from the browser to the server running inside the robot. To do this, we use a route which accepts a POST message from the edit page.

BY THE POWER OF THE BOOTSTRAP

Websites can be greatly improved by using style sheets to set the format and behaviour of the page elements. Creating good style sheets which work on a wide variety of different devices (from desktop PC to mobile phone) is hard work – one of the many things that the author is afraid of. So, he has made use of the Bootstrap framework (**getbootstrap.com**). This provides a powerful set of styles and reactive page elements. With this framework you don't have to include any of these style definitions in your web page text, you just tell the browser to go and get them when it loads the page. You can add it to the head of your HTML with:

```
<head>
  <title>Hull Pixelbot Python-ish Editor</title>
  <meta name="viewport" content="width=device-
width, initial-scale=1.0">
  <link rel="stylesheet"
    href="https://cdn.jsdelivr.net/npm/
bootstrap@4.3.1/dist/css/bootstrap.min.css"
    integrity="sha384-CbMQv3Xipma34MD+dH/1fQ784/
j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
    crossorigin="anonymous">
</head>
```

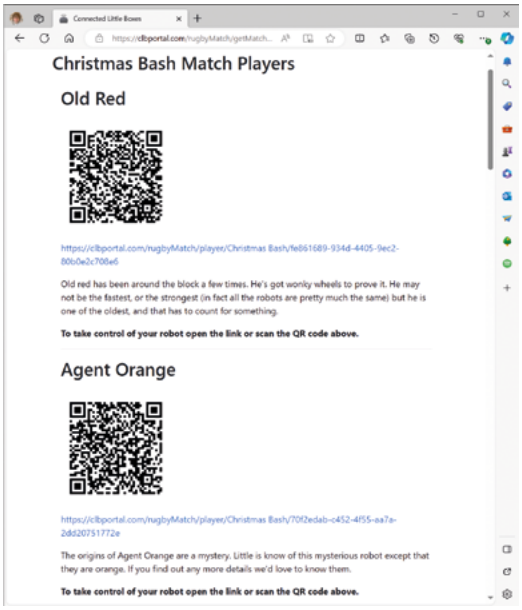
```
<form method="POST" action="/save">
  <textarea id="codeTextarea" name="codeTextarea"
   style="font-family:monospace;" class="w-100"
rows="20"
   cols="120">
  </textarea>
  <input type="submit" value="Save and run"
   class="btn btn-primary mt-2 mb-2 btn-block">
</form>
```

The code above shows the form defined in the editor page. An HTML form contains input items. In this case there are two. The first is the text area where the robot program is entered. The second is a button labelled 'Save and run'. You can see how this looks in the browser in **Figure 1**. When the button is pressed, the form will perform its action. It will use the HTTP POST action to send a message to a route called **"/save"**.

```
server.on("/save", []()
{
    // code is the only argument
    String robotCode = server.arg(0);
    sendLineToRobot("begin");
    sendStringToRobot(robotCode);
```

QUICK TIP

If we wanted to send more items of data into the robot from the web page, we would add the input items to the form and then the code running in the robot would pick them up as arguments 1, 2, 3, and so on.



Right
Each robot has its own back story

QUICK TIP

The robot has an address on your local network – picow.local. If you have multiple robots that you want to control, you can give each of them its own network name.

```
    sendLineToRobot("end");
    handleRoot();
});
```

The code above shows the handler in the robot server for the **"/save"** route. The server exposes items from the message, including any arguments added to the post. The form in the web page only contains one input item, which is returned as arg (argument) 0. The **robotCode** variable is set to this value and then sent to the robot, enclosed in **begin** and **end** messages that mark the start and end of the program being sent into the robot to be stored. The robot will store this program code and then run it. Once the program has been sent to the robot, the **handleRoot** function is called to redisplay the edit page. Now that the server has been configured, we can start it running:

```
server.begin();
```

This will start the server. However, the program must repeatedly call a method in the server to keep it updated. The robot does this in the **loop** function which is called by the Arduino framework:

```
server.handleClient();
```

The above statement is in the update function for the robot web server. It is called at regular intervals and will process incoming requests and serve out pages as appropriate.

MAKING A NETWORK PRESENCE

When the robot connects to the local network, it will be assigned a network address. We can connect to the robot directly using this address. Unfortunately, we don't know what the address is. This is a bit like the situation when a friend gets a new phone. But the good news is that we can get the robot to advertise a name that we do know. This is like your friend sending everyone they know a text message saying 'It's Dave here. Got a new phone'. The network service that will broadcast names and addresses for us is called the 'multicast Domain Name System' (mDNS).

```
#include <LEAmDNS.h>
```

The statement above imports the mDNS responder code into the program in our robot. Now we can start the mDNS responder:

```
if (MDNS.begin("picow"))
{
  Serial.println("MDNS responder started");
}
```

The code above starts an mDNS responder which broadcasts the name 'picow.local' and responds to requests for the network address at this name. If you look at the web address of the edit page in **Figure 2**, the browser is showing '**picow.local/run**' which means that the user has just pressed the Run button and used that route to run the program in the robot. All the network names created by the MDNS server have '.local' on the end of the name to indicate that the device is local to the current physical network. In other words, someone living next door will not be able to use this address on their home network to connect to a robot on your home network. Like the web server, the mDNS responder needs to be updated regularly so that it can respond to requests:

```
MDNS.update();
```

The statement above will update the mDNS responder. This will cause it to send out regular broadcasts giving the name and respond to requests for the network address for that name.

ADDING A SECOND BRAIN

The program web editing works well on the Pico W, but what do you do if you are controlling your robot using an Arduino Uno which has no network connection? You can do this by connecting a WiFi-

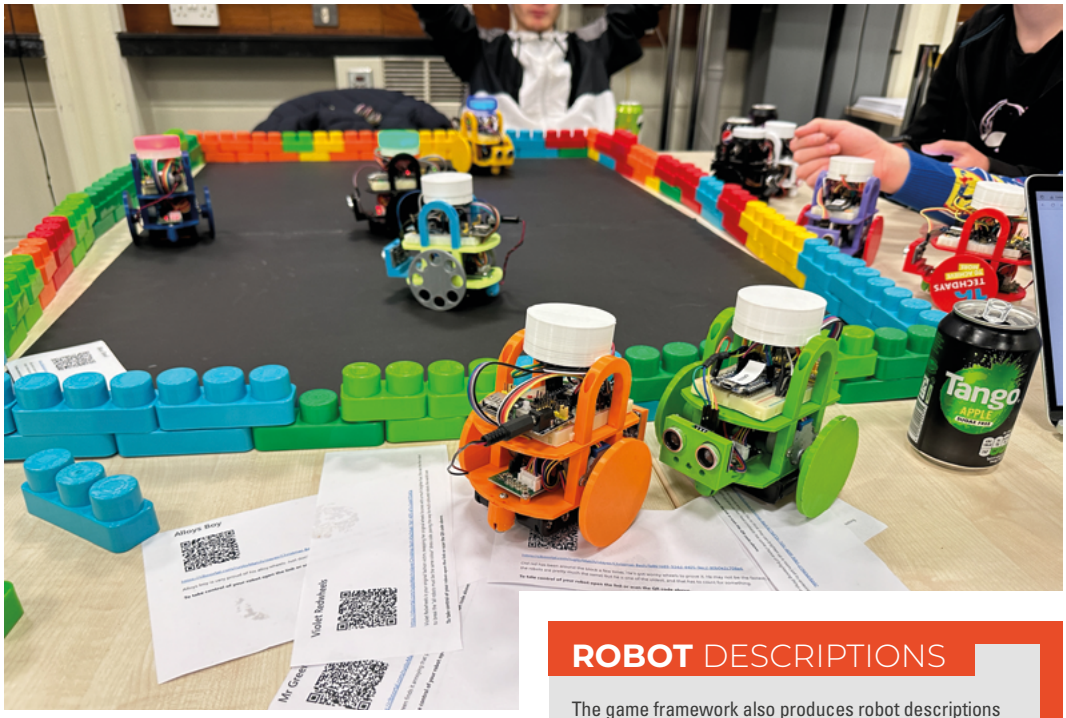


Figure 5 The wall around the robot arena is a very necessary part of the event

enabled device to the Uno and using that to connect to the network and host the edit web page. The Uno will accept commands from the network via its serial port as if they were being entered by the user.

Figure 3 shows how the ES8266 and the Uno are connected. The network device can send commands to the Uno as if they had been received via the serial port. The network device is an ESP8266 computer packaged as a WeMos D1 mini. The author is a big fan of the ESP8266 processor. It is the precursor of the highly popular ESP32. It provides WiFi connectivity but not Bluetooth. It can be programmed in C++ in the same way as an Arduino and is available very cheaply. The author happens to have a drawer full of them, which makes him like them even more. The software running in the ESP8266 is just the web server part of the robot code. The rest runs inside the robot itself. You can find this software on GitHub here at this link: **hsmag.cc/UnoNetEditor**.

You can use this technique to add a web control page for any device which has a serial port that can be used to control it. Just edit the web page to include the inputs that you need and then update the route handling code in the web server to send the appropriate messages.

SERVING OUT ROBOT RUGBY

You can create a team of Pixelbots which each have addresses on your local network. You can then have fun programming the robots. But the robots are only visible on the local network. The author wanted to create a robot rugby game which could be live-streamed and played by anyone in the world. So, he created a game framework to manage a rugby match.

The match manager can create a match and then manage it using the page shown in **Figure 4**. Each robot is a member of a team. The robot controllers each enter a Python-ish program into their robot and, when the 'Start all robots' button is clicked, the server downloads the program into all the robots. The robots run for a while and are then stopped with the 'Stop all robots' button for the teams to reprogram their robots for the next play. After five plays, the team with the most robots furthest into their opponent's half is the winner. The server and the robots use MQTT to communicate, and the messaging and device management is built on the Connected Little Boxes project (**connectedlittleboxes.com**). The full details of how the gameplay works are still being thrashed out, but it is thought this would be a great way to introduce people to programming.

ROBOT DESCRIPTIONS

The game framework also produces robot descriptions which can be printed out and given to the players. The descriptions contain a QR code which is generated by the server and can be scanned by the player to obtain a link to the code edit page for that robot. The page also contains a long-form of the link and a small back story for that robot.

Figure 5 shows the robots in action at the first match at the Hull University Computer Science Christmas Bash. There was a lot of interest in the robots and making them do things, and we've got plenty of ideas for making the gameplay even more involving.

QUICK TIP

If you want a very small version of the Arduino Uno that you can put inside devices, you should look at the Arduino Uno Pro Mini. There is even a 3.3-volt version of this device, which means you don't need a level converter to connect it to the ESP8266.

Polyvinyl butyral (PVB)

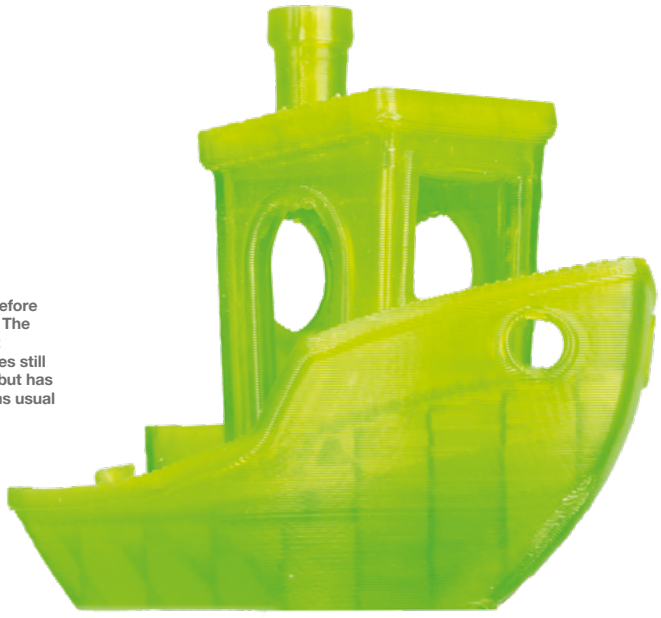
Get glassy-smooth prints with PVB and alcohol

Ben Everard

Ben's house is slowly being taken over by 3D printers. He plans to solve this by printing an extension, once he gets enough printers.

Solvent smoothing is one of those things that seems great in theory, but can be a pain in reality. The idea is simple. All 3D prints have layer lines, but you can remove them by dissolving the outer layer of plastic in a small amount of solvent. The now-liquid plastic will flow into the crevices between the layers, and the solvent will evaporate, leaving the plastic that was dissolved in it to fill any gaps and produce a shiny surface.

The main problem with it is that it usually involves harsh chemicals. PLA, in particular, needs substances like chloroform, which requires equipment beyond our workshop setup to do it safely, so we've been unable to test it out. ABS is a little more straightforward as it can be smoothed with acetone. However, even this is still unpleasant to work with.



Right
A Benchy before smoothing. The transparent filament does still look good, but has layer lines as usual

PVB is soluble in isopropyl alcohol (IPA). While there are still issues to be considered when working with IPA, it's pretty safe to assume that most people who 3D print are comfortable using this particular chemical. However, don't assume that you can easily do this safely because you're used to using IPA for other things. Take a look at the Safety box (opposite), and make sure that you're properly considering your own experience and equipment.

Solvent smoothing aside, the only other reason we can think of to use PVB is because it's transparent. It's a little clearer than other transparent filaments we've seen, but only slightly. Unless you need absolute maximum transparency, you will probably find it easier to use transparent PLA.

ABSOLUTE MELT

That's the theory; let's see how it works in practice. PVB is reasonably easy to print. It requires a slightly hotter bed than PLA, but it's nothing that will cause problems for most modern printers.

The biggest problem with printing PVB is that it is very hygroscopic, meaning that it absorbs moisture from the atmosphere. If you plan on printing it quickly, you might be able to get away with keeping the filament in a sealed bag containing a desiccant sachet. However, ideally, you should have a filament dryer.

When printing something that's going to be transparent, as most PVB prints will be, you need to consider the inside of the print – in other words,

POLYSHER

Polymaker makes a device specifically for smoothing PVB objects with IPA called the Polysher. We've not tested this out, so we can't comment on how well it works, but it should make the process a little more straightforward and less messy.

Left
After smoothing, the layers are still visible to the eye, but the surface is smooth to the touch

> When printing something that's going to be transparent, **you need to consider the inside of the print**

the infill. This pattern can ruin the look of your transparent object. Some prints you can make completely hollow (such as with Vase mode), but if this isn't possible, you can completely fill it. You can do this by adding 100% infill or by upping the number of perimeters to something far bigger than would actually fit (such as 999). We'd recommend the latter because it will lay down the filament concentric with the outside perimeter, which is less likely to produce visible issues. Obviously, filling up an object with plastic is going to have a pretty big negative impact on both the amount of filament that it uses and the amount of time it takes to print. →

SAFETY

It's easy to get complacent about a chemical you use regularly, but it's important to remember the dangers of IPA because when applying it with a brush, you're more likely to have a problem. We recommend reading the material safety data sheet (this should be available from the supplier). Particular risks are:

- Inhalation
- Eye contact
- Fire

All of these are significantly more likely when brushing it on a model than when wiping it on a print bed. Use proper eye protection. It's recommended that you use nitrile gloves when handling IPA.

Remember that safety is your responsibility, and you must fully understand the risks associated with a process before starting it. Proper PPE is a requirement for this process, not an optional add-on.

Above
In Vase mode, there's no infill to see, so the transparency isn't disturbed. This photo is before smoothing

If you have a sufficient flair for 3D design, you could probably make something interesting using the voids inside the print.

There are two main brands of PVB printer filament: Prusament PVB and Polymaker PolySmooth. We tested out Prusament, but the same process should work with PolySmooth.

Print out your part as you would normally. Since we're going to be smoothing this, there's no need to worry too much about layer lines, so go big for a fast print. This might cause problems with small details, but smoothing can cause problems with small details anyway, so this process is best suited to objects that can be printed with a large layer height. If you've got a big nozzle, this can be a good time to use it.

Once you've got your model, you need to apply the IPA to the surface. It should be at least 70% IPA. You can do it in a few ways:

- immerse it
- squirt it
- brush it
- vaporise it by applying it to a cloth; putting both cloth and 3D print in an enclosed space, such as Tupperware

Whichever way you do it, the key is to get a uniform coating, and leave it to dry. When drying, you want to place it on something that minimises the number of contact points, as these will leave blemishes on the surface. In most cases, you'll need to repeat the process several times in order to get the desired effect.

We tested this out by brushing on IPA – this is probably the most approachable option for people trying this out. The method is really simple. Take

VASE MODE

You can get great results with PVB in Vase mode. The resulting prints are almost see-through and have a glass-like shine. However, you do need to be careful because the walls are very thin. The vases can become very weak. We accidentally dissolved a hole in one of the test pieces.

While with Vase mode, you can only have one perimeter, this perimeter can be very thick. By default, PrusaSlicer sets the external perimeter wall to 0.45mm on a standard 0.40mm nozzle, but you can adjust this. We upped it to 1.00mm and had good results.

Left
We heavily smoothed this one and the layers are only just visible. However, it has distorted the vase slightly. Corners are more rounded and finer detail is lost

> There's something very appealing about the **transparent, shiny prints you can get with PVB**

your print, a paintbrush, and a small amount of IPA. Paint the IPA onto the print and try to get as smooth a coat as possible. You need to be quite liberal with the IPA to get a good effect. You'll find that the IPA will spread itself out along the layer lines. We'd recommend using nitrile gloves when doing this.

Once you've got a good coat, leave the object to dry on a suitable surface – this will take about half an hour. If the object you're making is hollow – like a vase – you can swish some IP around inside to get an even coat. Pour out any excess and leave to dry upside down.

It can take about five to ten applications of IPA to smooth away the layer lines.

RESULTS

We are quite partial to the look of layer lines, and they're not something we generally feel the need to sand or post-process away. That said, there's something very appealing about the transparent, shiny prints you can get with PVB. When done well, it can be almost glass-like.

There are, however, downsides. The process that removes layer lines will also remove other fine details (just as sanding, painting, and other post-processing would). It's a bit tricky to apply the IPA smoothly and evenly.

While Vase mode does work well, it does produce parts with thin walls, and the IPA will remove material from some parts – this can leave your print weak in places.

There are certainly prints that can look excellent when printed in PVB and smoothed. Things with gentle curves that are prone to false contours can smooth particularly well. We'd love to see your prints. Email us at **hackspace@raspberrypi.com** or tag us on social media.

Exploring PCB services

In this part of the ongoing KiCad series, let's explore working with a range of PCB services

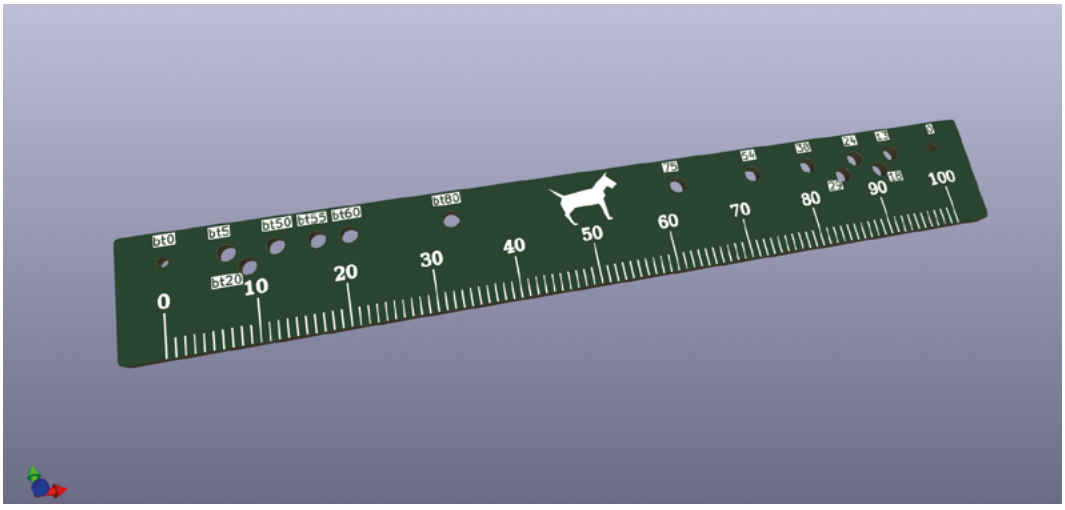
Jo Hinchliffe

Jo Hinchliffe is a constant tinkerer and is passionate about all things DIY space. He loves designing and scratch-building both model and high-power rockets, and releases the designs and components as open-source. He also has a shed full of lathes and milling machines and CNC kit!

In the last part of this KiCad series, we looked at the different substrates and other hardware options that are available across a range of PCB fabricators and PCBA services. In this section, we are going to generally look at the range of services that are available, and look at what some of the services and companies need from us to make our PCB projects. We'll also mention a few of the quirks we've found in some services that had us scratching our heads.

Each service is well-established and capable of creating quality PCBs and/or assembled PCBs. However, they each have different specifications and tolerances. In fact, that's often a good place to start when comparing or looking for a service to make your project. Questions to ask yourself include: what minimum clearances and track widths do I need? What is the smallest hole diameter? How accurate do I need things?

Over on OSH Park, they have a great collection of documents relating to the services they provide. The Drill Specs page details the minimum and maximum hole sizes, sizes for annular rings, and the via plating specification. Beware that all these specifications are different across OSH Park's various services, changing, for example, between the two-, four-, and six-layer options. The minimum track width specification on the OSH Park site is listed as 0.006", with 0.006" clearance on the standard two-layer boards, moving down to a very accurate 0.005" in the four-layer board offerings (it's common to



Right
Our small PCB ruler project provided some interesting challenges to PCB services

OSHPARK SERVICES SUPPORT OSH PARK DOCS / DESIGN TOOL HELP / KICAD / DESIGN RULE SETUP

Design Rules > Constraints

These vary slightly based on production service.

Setting	2 Layer Services	4 Layer	6 Layer
Minimum Clearance	6mil (0.1524mm)	5mil (0.127mm)	5mil (0.127mm)
Minimum track Width	6mil (0.1524mm)	5mil (0.127mm)	5mil (0.127mm)
Minimum Connection Width	6mil (0.1524mm)	5mil (0.127mm)	5mil (0.127mm)
Minimum Annular Ring	5mil (0.127mm)	4mil (0.1016mm)	4mil (0.1016mm)
Minimum Via Diameter	20mil (0.508mm)	18mil (0.4572mm)	16mil (0.4064mm)
Copper to hole clearance	5mil (0.127mm)	5mil (0.127mm)	5mil (0.127mm)
Minimum Through Hole	10mil (0.254mm)	10mil (0.254mm)	8mil (0.2032mm)
Hole to hole clearance	5mil (0.127mm)	5mil (0.127mm)	5mil (0.127mm)
Minimum uVia diameter	20mil (0.508 mm)	18mil (0.4572 mm)	16mil (0.4064 mm)
minimum uVia Hole	10mil (0.254mm)	10mil (0.254mm)	8mil (0.2032mm)
Silkscreen Min Item Clearance	user preference		
Silkscreen Min Text Height	user preference		
Silkscreen Min Text Thickness	5 mil (0.127mm)	5 mil (0.127mm)	5 mil (0.127mm)

Left Technical details of the OSH Park PCB services

see limits given in thousandths of an inch because, apparently, you can have both metric and imperial at the same time if you try hard enough). You can find this and all the other details on the OSH Park KiCad Design Rules page at **hsmag.cc/oshparkrules**.

If you have questions about the OSH Park services, they have an excellent track record in communication. You can email the support email address and they will get back to you offering excellent advice. We have even had the OSH Park team open our KiCad project file and they have fixed problems and then taken the time to teach us solutions.

Different PCB manufacturers put the information in different places. Another PCB service, DirtyPCBs, bundle all their specifications and tolerances information on their About page. They similarly list a 0.006" minimum track width and clearance across both their two- and four-layer boards and cite their other specifications and tolerances. The DirtyPCBs service was designed as a minimal service with its emphasis on 'cheap', therefore, it has no chat service, and it's difficult to contact the company to ask questions. If you have challenges here, it can be a better option to use search engines and find forum conversations about the service to try and get the answers you need! →

> **You can email the support email address and they** will get back to you offering excellent advice

2/4 Layer Capabilities

ITEM	CAPABILITY
Material	FR-4 0.6mm-2.0mm 1oz copper ('standard' PCB material is 1.6mm thick, but we default to 1.2mm)
Layer number	4L
Maximum size	600*600mm (60*60cm)
Shape	Almost anything! We'll send it and see if they accept it!
Min internal slot	32mil (0.8mm)
Min core thickness	4mil 0.08mm
Min core thickness	16-96mil(inner) 16-118mil(out)
Min w/s	6/6mil(I/L) (increased from 5/5 due to poor yield)
Min w/s	6/6mil(O/L) (increased from 5/5 due to poor yield)
Min silkscreen line	0.15mm
Min BGA size	Oblong:10*13.5mil/circle:12mil
Min SMD width	8mil
Min solder dam	3mil(green)/3.5mil(black)
Min dielectric thickness	2.5mil
Min diameter of finished hole	12mil
Tolerance of drill position	+/- 2mil
Tolerance of finished hole size	PTH +/- 3mil
Tolerance of finished hole size	NPTH: +/- 2mil
PTH hole copper thickness	0.6~1.4mil
Max A.R of PTH	8:1
Surface copper thickness	1oz
Routing dimension tolerance	(Z0) Impedance control 4mil
V-cut/V-groove	80mm (8cm) minimum, 380mm maximum. **Optional, extra charge**
(Z0) Impedance control	+/- 15%
Ionic contamination	< 6.4ugNaCl/inch2
Surface treating	HASL (hot air surface leveling, not PB free unless special request)/AuSn/AgCN/Electrogilding/Ni/OSP*/G.F
*OSP	Organic solder-ability Preservatives

For best results don't torture the board house! Be conservative. After all, these are crappy PCBs!

Left The DirtyPCBs PCB specification is in amongst a large About page

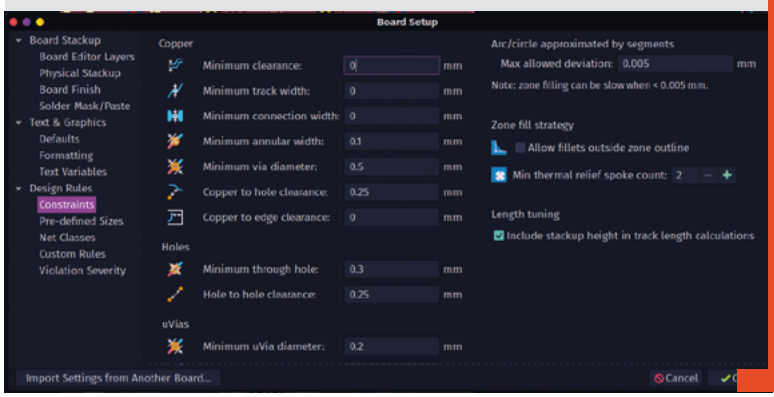
RULES RULE!

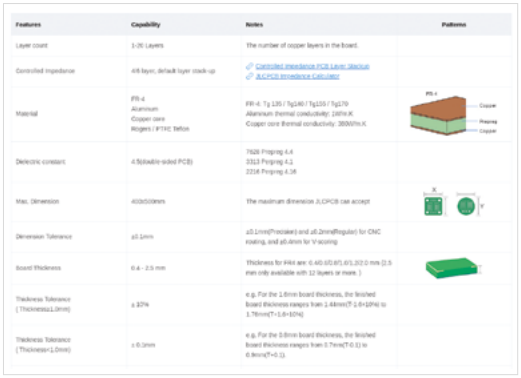
If, while developing a PCB project, you have a particular fabrication service in mind, you can ensure your board design's compatibility by setting up the Design Rules for your board to match the service. The Design Rules is nested under the Board Setup window we used in the previous section of this series to set up the different physical stackup characteristics for different substrates. In the PCB Editor, navigate to File > Board Setup and then open the drop-down menu labelled Design Rules. In the first section, Constraints, you can set up limits for the minimum clearances, minimum track widths, and other limits relating to the copper regions. You can also adjust the minimum via sizes, hole sizes, and clearances, and also the minimum dimensions for text objects on the silkscreen layer. The next item down in the Design Rules drop-down menu is the Pre-defined Sizes tab. We've used this earlier in the series to set track widths for projects. However, as a reminder, you could set this up at the beginning of a project whilst considering any limitations or constraints your target PCB service has. Also, notice that we can import the settings from a previous project – this is an excellent function if you set up a project for a particular PCB service specification.

Jumping to the bottom of the Design Rules drop-down menu, we can see the Violation Severity section. This section sets up how the Design Rule Checker (DRC) tool responds if any of the rules set for a project are broken. As a primary word of warning, think very carefully before setting any of these to 'ignore'. It may well be that for a current project you don't mind some of these issues, but it's possible under different circumstances or projects, an ignored error could be critical.

Back in the PCB Editor window, to run the DRC at any point in your project, you can either click the 'Show the design rules checker window' tool icon or you can select 'Design Rules Checker' from the Inspect drop-down menu. Once the window is open, you can then click the Run DRC button for the PCB to be checked against the defined design rules. Note that getting an error or a warning doesn't always mean that your PCB project isn't working, but they are simply indications that there is something that hasn't met the design rules.

Rules don't always have to be followed, but it's always good to check which rules are broken so you can be sure that any that are broken are broken intentionally. For example, on our ruler PCB design, we got numerous silkscreen errors as the ruler sat over the edge cut geometry, and we also got lots of courtyard errors where the courtyard areas of the mounting hole footprints we had used had overlapped. The actual distances between the mounting holes were all over the minimum clearance from each other, so neither of these sets of issues actually mattered – it's definitely worth checking though. The DRC, when run, will add small red arrows or markers on your PCB design, highlighting where the issues are located. If you close the DRC window, the markers still remain on your design. When selecting a marker when the DRC window is closed, the issue that the marker relates to will be shown in the lower toolbar on the PCB Editor. You can reopen the DRC window and delete single markers or all markers using the relative buttons. Obviously, if you don't make changes to the PCB design and run the DRC again, removed markers will be replaced.

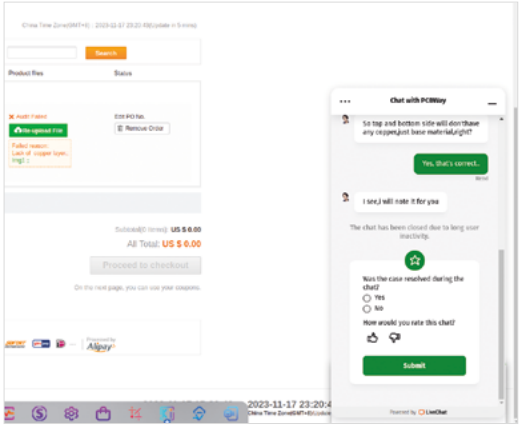




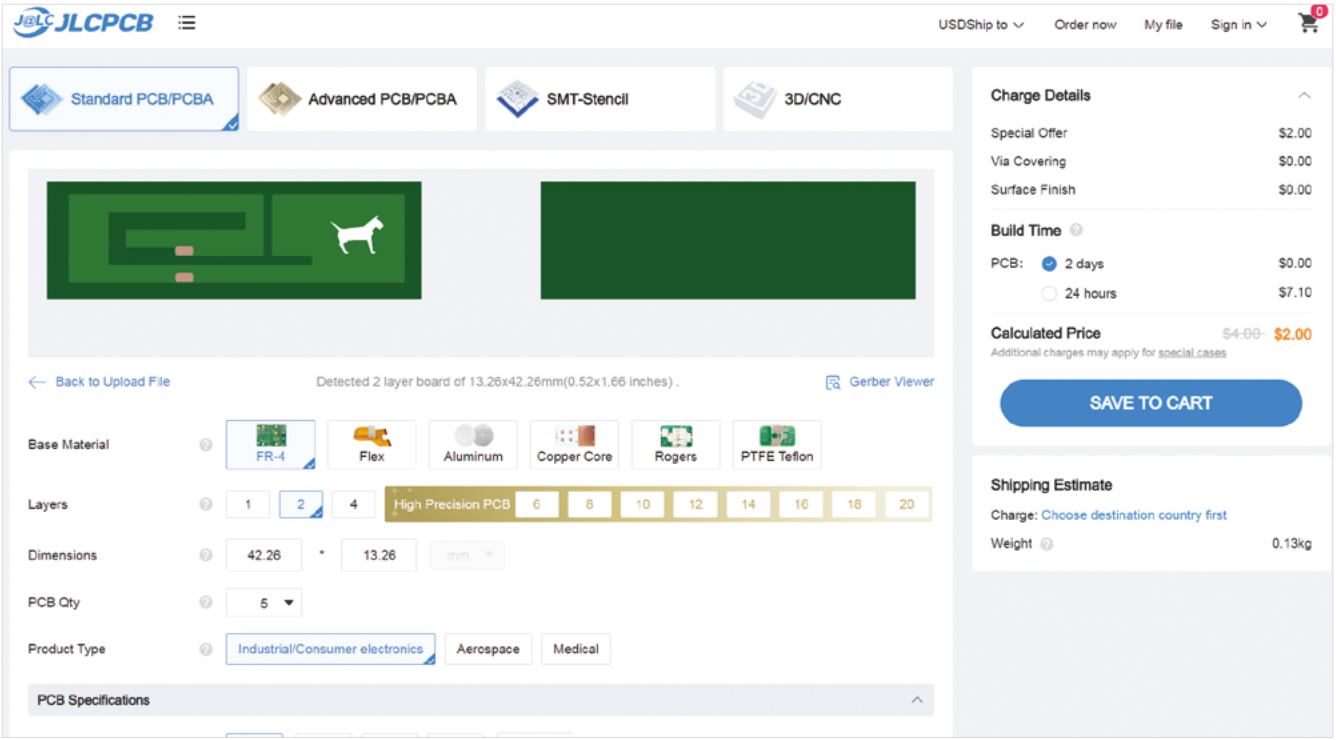
Above
JLCPCB's specifications are found on their Capabilities page

We have, of course, used JLCPCB/A a fair amount in this series. JLCPCB have a pretty exhaustive list of their specifications over on their Capabilities page. They can offer up to a whopping 20 copper layers in their PCBs, with track widths and clearances a default minimum of 0.005" in the two-layer offering, moving to 0.0035" for four-layer options and more.

Similar specifications are available from PCBWay. An advantage of this service is the maximum dimensions of PCBs they can fabricate are 1100 × 500 mm, whereas, for example, JLCPCB are 500 × 400 mm. One thing of note about PCBWay is that you specify the board and the board dimensions and add it to your shopping cart prior to uploading Gerbers, which can seem slightly counter-intuitive compared to other services.



Above
Many of the services offer online chat portals. These can be useful when trying to negotiate problems or challenges with a service



Above
An interesting issue occurred where JLCPCB didn't detect the edge cut geometry on the flex antenna design, which should have rounded corners

With all of these services, it's definitely worth contacting them using the chat function or emailing if you want to check specifications.

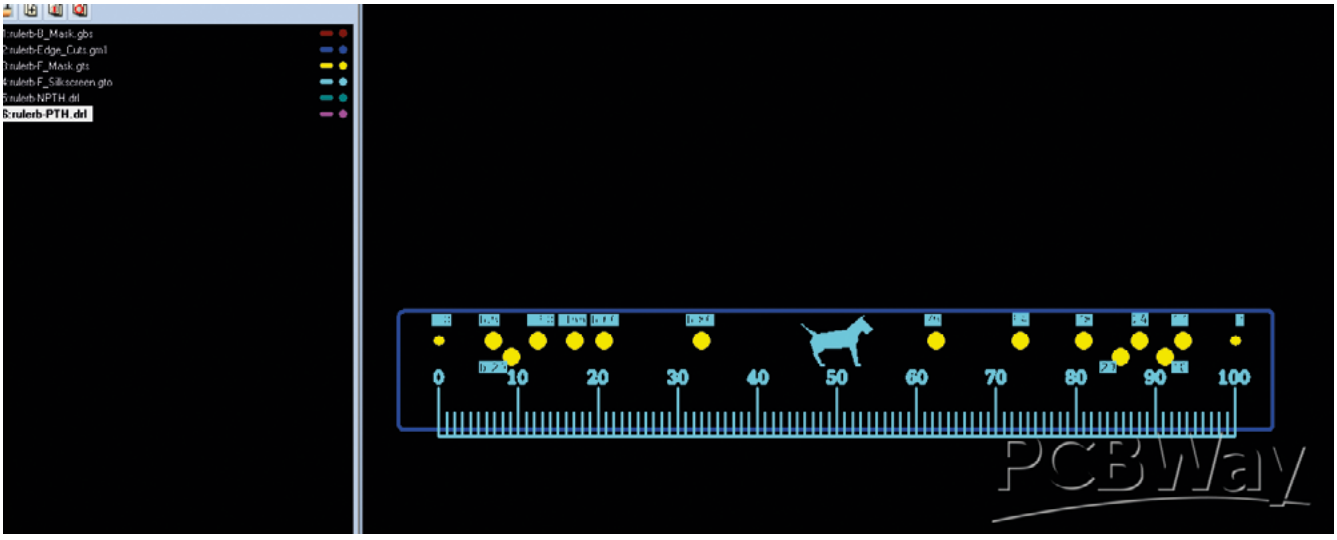
DIVING DEEPER

We have often featured OSH Park as a go-to company for PCB fabrication in HackSpace magazine. They have an excellent track record in supporting and promoting open-source projects, and they have that iconic purple solder mask finish which is very visible across lots of maker/hardware hacker projects. However, probably one of the main reasons that OSH Park have often featured as a service is that you can directly upload KiCad PCB files to their website – you don't have to go through the process of making compatible Gerber files. This makes the service really easy to use. If you are reading this after a new milestone version of KiCad has been released (for example, a future version 8), you might find that it takes a little while for the OSH Park website service to become compatible, but rest assured, you can also upload a zip file of Gerbers in the same way as other services in the interim. OSH Park's guidance on Gerber set up and requirements is available at **hsmag.cc/Gen_Gerbers**.

Beyond the standard OSH Park purple offering, there are options for the 'After Dark' finish (black substrate and a clear solder mask), a lighter 0.8mm board with a heavier 2oz copper layer, and a flex option. Whilst they offer good quality, they are incredibly affordable when working with smaller PCB designs. Finally on OSH Park, they are excellent at communications. If you need to raise a ticket to ask a question, they go above and beyond many services.

> **Whilst they offer good quality, they are incredibly** affordable when working with smaller PCB designs

Different fabrication houses have different needs around the files and file formats that you upload. One area we have noticed creating issues is the DRL or drill files. In KiCad, you can create either a pair of DRL files, one containing the non-plated through-holes and another containing the plated drill holes, or you can merge these two files into one. JLCPCB wants these files supplied as a pair, whereas if you upload Gerbers with two separate DRL files to →



Above
Whilst PCBWay rendered the ruler project correctly from Gerber files, it couldn't initially process an order due to an error being created as there was zero copper in the copper layers

OSH Park, you get an error message from OSH Park, but it conveniently will merge the two files online and also solve the issue for you. There can be other little glitches involving drill files.

Some fabrication houses want there to be drill files even if the PCB has no drill holes in the design, for example, in the design of a single-sided PCB. This caused an interesting issue when we designed the flex PCB antenna example in the last part of this series. We exported the design Gerbers and DRL

> **Some fabrication houses want there to be drill files** even if the PCB has no drill holes in the design

files even though the design contained no drilled holes – this was just because we wanted to upload the Gerbers to a range of services to see how they rendered and get quotes.

With JLCPCB, when we uploaded the zipped Gerber file, the preview would ignore the edge cuts geometry, so the curved corners of the design would disappear and, incorrectly, the board would appear as having square edges. Chatting to the online chat service, they confirmed that they could see the edge cuts layer and assured us that if we placed the order, the board would be cut correctly. We tried playing around with the Gerbers and we also posted the issue on the KiCad forum for discussion. It seemed that others generating their own Gerbers from our project would get a correct render on the JLCPCB site. The difference we spotted was that they weren't including any drill files. Re-uploading without drill files and the correct board outline and edge cut geometry rendered correctly. As part of this process, we discovered that OSH Park didn't have this issue and rendered the board correctly at upload. We placed the order with them.

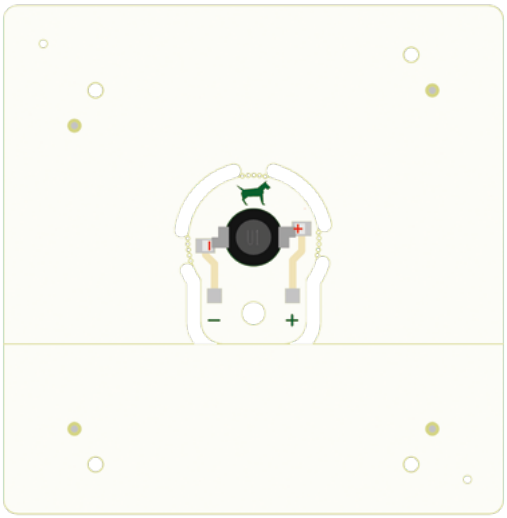
The moral of this particular story is that Gerbers aren't standard, so be sure to check what you need, and be prepared to talk to the PCB manufacturer if things don't look right.

We had another issue relating to empty layers when looking at different services to fabricate the ruler design. When uploading the design to PCBWay, it would throw an error with the upload as the Gerber file for the copper layers contained no copper. Obviously this is very unusual for a PCB as copper is usually the conductive layer connecting components and more. The ruler project shows that with a more artistic use of PCB fabrication, it's possible to cause headaches for fabrication houses.

We had a problem with one of our recent projects when submitting the aluminium substrate LED module project to JLCPCB services. The 1-watt COB LED we had identified in the JLCPCB parts library had a diagram on the datasheet of the LED which had polarity markings on the two semi-symmetrical flat SMD pin connectors. We'd actually seen these LEDs in real life – they have an etched - and + in these metal connectors.

When we uploaded to JLCPCB, the website rendered the PCB with the components placed and the JLCPCB 3D model of the LED had no polarity markings. We presumed it would be correct, and as only ordering a small number and the LED is a large part, it wouldn't be too onerous to swap them around if they arrived incorrectly. After ordering, the process was halted, and JLCPCB contacted us to discuss and check the polarity of the LED. We had to point out that it was an issue with their 3D model that meant it was impossible to tell if it was rotated correctly, and an engineer at their end would only be able to tell when they physically went and looked at the package. In the end, the aluminium LED PCBs were correctly manufactured.

Our main takeaway point for working with any PCB service is that most things are achievable with good communication, which increasingly becomes, in combination with the physical specifications we require, a valuable deciding factor in choosing which service to use.



Above
In the original render preview, the LED polarity couldn't be ascertained as JLCPCB had a problem with the 3D model. After an engineer inspected the part, the correct orientation was confirmed

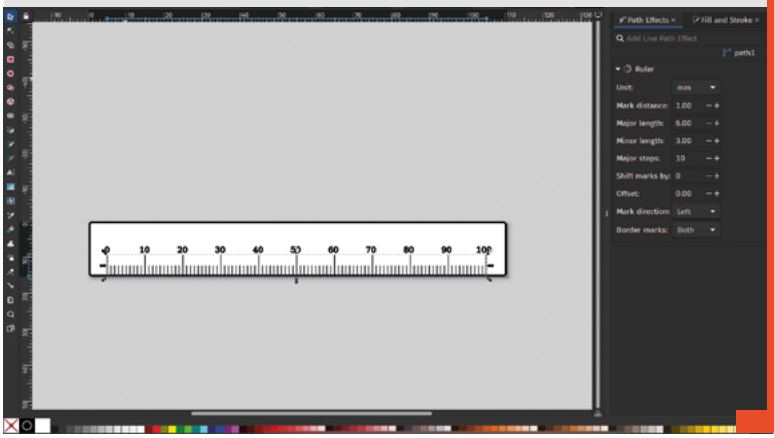
RULERS RULE!

Making a PCB ruler is almost a right of passage in the PCB-making communities. They can be a simple, useful tool, a good business card, or perhaps even perform some extra function. Having an interest in model and high-power rocketry, we made a ruler which has some accurately placed holes in it, into which you can place a pen. You can then pin the hole at the 0 or 100mm marker and draw circles that match common rocket motor diameters or common Estes rocket body tube diameters. Handy for impromptu cardstock or balsa rocket component-making. It might seem strange to use a PCB manufacturer for this, but it's actually a very affordable way of getting very accurate 2D designs made.

One slightly tricky aspect of creating a PCB ruler in KiCad is how to actually draw a graduated line to give the ruler its measuring graphic element. Again, we've used the excellent open-source Inkscape to solve this in combination with KiCad 7's SVG import abilities.

In Inkscape, draw a straight line using the pen tool and use the height and width settings to set it to the length of the ruled section you require. We wanted a compact ruler PCB, so we went with a 100mm length. Next, select the line and then click Path > Path Effects. The Path Effects dialog box should open on the right-hand side of the screen – there should be a search bar at the top of this dialog. Type in 'ruler' and select the Ruler Path Effect that appears in the results. This, in turn, should launch the Ruler Path Effects dialog. Set the 'Units' to 'mm' and then set the 'Mark Distance' to '1'. This should then add a graduation line one millimetre apart along your line. Next, you can set the length of the major length for the longer graduation lines and the minor length for the shorter lines. Finally, set the 'Major Steps' to '10'. You should now have a ruler graphic with the commonly used idea of a longer marker every 10mm.

You could use KiCad to add the text to mark the numbers on your rule graphic, but we went ahead and did this in Inkscape as it's pretty easy to use the align and distribute tools to bring a line of text labels into alignment with the ruler. We then resized the document using document properties to the size of the design and saved it as an SVG to be imported to the silkscreen layer in KiCad in the PCB Editor. We've covered this before in the series, but as a reminder, it's as simple as File > Import Graphics, and then, in the dialog, navigate to the SVG, set it to import to the correct layer (in this case, the front silkscreen), and make sure the scaling is set at '1'. It's worth setting the PCB Editor grid to '1 mm' so that you can align other elements to the ruler design well. Finally, we were lazy and left the baseline in our ruler graphic and then used the edge cuts geometry to remove it – this ensures that the silkscreen ruler lines run right to the edge of the PCB. However, you can remove the original line back in Inkscape. When you have your path effect applied, you can select the entire ruler graphic and then use Path > Object to Path to convert the path effect into regular paths. Then, using the node selection tool, you can select the bottom line and delete it. A tip, as it's difficult to grab the nodes at the end of the baseline, is to zoom in and then bend the line away by dragging the baseline in between two of the graduation line nodes. Once the line is bent away from the graduations, you can click it and delete it.



TUTORIAL

PART 03

Pico modular: A bit of polish

Better sound output, cases, and a tune-up

Ben Everard

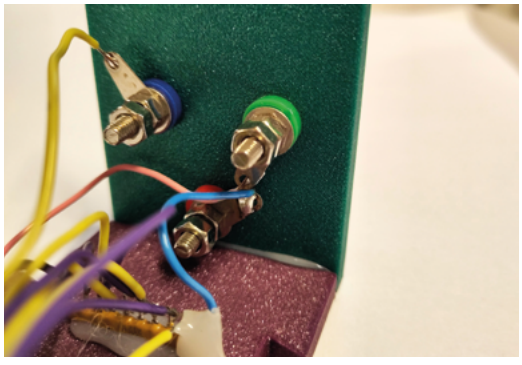
Ben's house is slowly being taken over by 3D printers. He plans to solve this by printing an extension, once he gets enough printers.

In part three of this series on building a budget modular synth, we're going to tidy up a few of the niggles in our nascent synthesizer system. We've skipped over a few bits in order to get some output, even if it's not perfect output. First, let's make sure we can get sound out and into a pair of headphones with something a little more robust than a crocodile clip.

So far, we've been taking the sound output directly from the R2R ladder. This works but it's very quiet. A much better solution is to add an amplifier, and you can get headphone amplifier modules cheaply. The one we got takes a 5V power supply, and left and right inputs that it amplifies. As with everything in this (or any other audio) build, there are a wide range of parts at different points on the price/quality spectrum. We're unashamedly aiming to build an affordable synth, so we're starting on the low price end.

This is an experiment and, at the moment, we're not creating any particularly complex sounds. Once we have a more complete system, we'll test out some other headphone amplifiers and see if they have much of a bearing on quality.

Unfortunately, cheap headphone amplifiers don't come with much identifying information, so it's hard for us to give much of a guide to the one we have, other than what it looks like.



Right
There's just a single connection on banana sockets, so just bolt them in and solder them on

Connecting it up is just a case of plugging the right bits into the right bits. There's a knob to twiddle for the volume and that's about it. It has an on-board jack that we can plug headphones straight into, but the layout of the board means we can't mount it so that we can access both this and the knob. We're working on the mounting for this; while we do that, let's take a look at how we're mounting the rest of the synth.

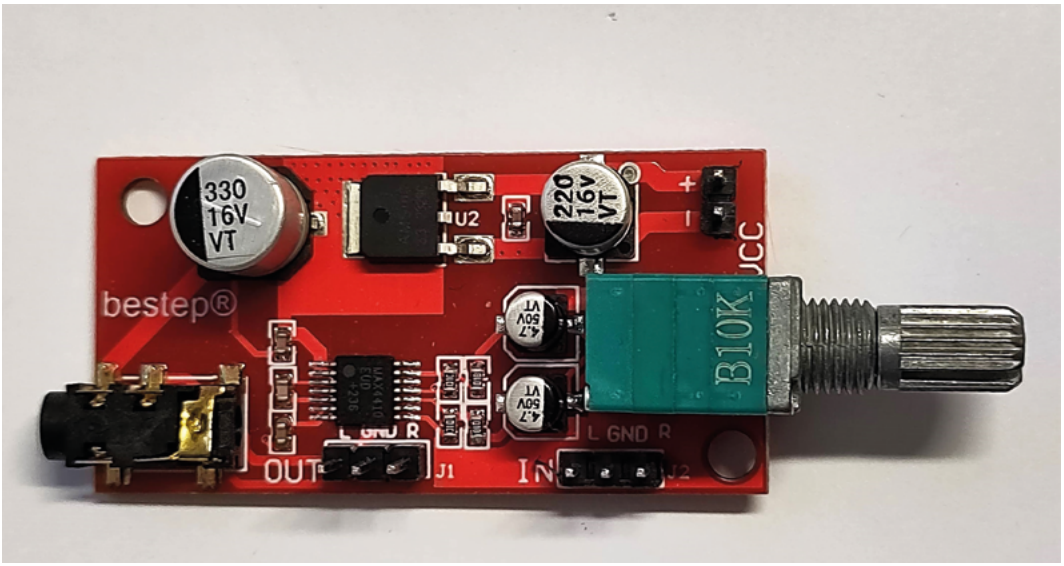
MOUNTING

Just having circuits isn't enough. We need a way to keep these tidy and out of the way. In an effort to keep costs down, we're trying to not use anything other than standard parts and off-the-shelf modules. That means no custom-made PCBs, or other things that could add to the cost. We're trying to mount everything on 3D-printed cases.

Our requirements for this are that it's as 3D-printed as possible, as hackable as possible, and as expandable as possible. The solution we've come up with may not be perfect, but it's working for us so far.

First, we'll take a look at how we've made it hackable. We wanted a case design that you could tweak and modify for your own modules without having to dive into the murky world of CAD. So, what we've done is created base files that you can carve up and modify using PrusaSlicer. This has basic modification tools that let us add holes and text to our creations. You can find these (as well as some examples) at **hsmag.cc/pmparts**.

The bases have a jigsaw-like fitting that you can slot together to build up a synth that's an arbitrary number of modules long. There are two widths – normal and wide – and you can pick whichever best suits your needs. Each case is an L-shape stretched out, and it could be printed in one go, but we decided to split it up into two parts. This means that it prints faster and the resulting part is stronger. The two parts can either be glued together or attached with M2 bolts (8 to 12mm long work best).



Left
There are many similar modules around, but this one is working fine for us at the moment

The base plate contains mounting holes for a Pico (Pico Ws also fit) and, again, these are attached with M2 screws. There are also screw holes for attaching the upright. Other than this, it's blank. If you want to attach additional hardware, there are a couple of options:

1. Glue it on
2. Add holes

The first one is the method we've used for most electronics hardware. Both circuit boards and PLA work quite well with hot glue. However, for a more permanent solution, you can add screw holes.

PrusaSlicer has changed a bit recently, so we'd recommend updating to the latest version before going any further.

Import the STL into PrusaSlicer and make sure it's active (it will be green). You can now right-click on it and select Negative volume > cylinder. This will add a cylinder with a diameter of 28.2mm. Unless you happen to be adding a part exactly this size, use the Size [World] boxes on the right-hand side to make it the size you want. You can either place it by eye, by dragging it to the place you want, or you can use the position boxes to place it precisely. We usually use a combination of both by dragging one hole to a place that looks right, and then using the positioning boxes to place others at the correct position away from this.

We've also experimented with different ways of connecting the modules together. In the first test, we used springs into which you can fit wires. To be honest, this was mostly because we were nostalgic for the electronics kits of the 1980s and 1990s. The connections were, at best, inconsistent. We've now ditched these in favour of banana plugs. These are push-fit connectors with a slight expansion on the pin so they fit snugly and lock into place, but can still be easily pulled out.

Banana plugs are easier and cheaper than jack sockets that are more commonly used in modular synths, and we'll look more at the other options when we get to interfacing between our modular system and a more traditional system. Attaching these is simply a case of creating a hole and bolting the socket in place, and then soldering it on. There are a few different styles of banana socket around, so make sure you have the right-sized hole.

We're still using springs for the power connection, and it seems to be working reasonably well, but we'll investigate this and more options when we start to get a larger system.

As well as the physical setup of the system, we've added writing. This helps us identify the module and remember what the inputs are. Again, we can do this directly in PrusaSlicer. Highlight the part so that it's green, and right-click it but, this time, select Add text. You can edit the text as well as the size. We've had the best results using the default Segoe UI font, but in bold with a minimum letter height of 7mm. Any smaller than this and we've found it prone to snapping off or being illegible. The default depth is 4.5mm, which is a bit deep for us – 1 or 2mm is plenty.

When you slice the file, you should see a pop-up with: 'Sliced object looks like a logo or sign. Apply colour changes automatically?' If you click on this, it will pause the print when it gets to the text to let you change the filament to a different colour. You can make different bits of text different colours by giving them different depths. We'd recommend about 1mm between each colour. Now that we can hear our sounds better, and mount them better, we should actually make sure they sound good. Let's take a look at getting the right pitches from our synth. →

IN TUNE

So far in this series, we've not worried too much about the exact frequency a given voltage emits. This is fine for testing, but really we want to make our synth play the right notes. There are basically two ways of doing this – we could do all the calculations and make sure everything is correct in theory. Or, alternatively, we could test what it is in practice and then adjust our code to fit what we detect. We opted to go with the latter approach.

We created a VCO using the code we developed in part one of this series, and connected it to a MIDI-to-voltage bridge (from part two of the series). However, we don't want to actually use the MIDI code. Instead, we used CircuitPython's interactive mode to adjust some values and see how this affects things.

In the Mu editor, we opened a serial connection and pressed a key to enter interactive mode. We entered the following:

```
>>> import board
>>> import busio
>>> import adafruit_mcp4728
>>> i2c = busio.I2C(board.GP1, board.GP0)
>>> mcp4728 = adafruit_mcp4728.MCP4728(i2c)
>>> gate = digitalio.DigitalInOut(board.GP15)
>>> gate.direction = digitalio.Direction.OUTPUT
>>> gate.value = True
>>> mcp4728.channel_a.value = 4000
```

This sets the first channel to a particular voltage (given as a 16-bit number). We connected our headphones and used a tuning app on our phone to find the actual pitch at this point. We noted down the pitch and then repeated the last line with a new value, noted down the frequency, and so on. These points can be used to create a curve that plots the relationship between the DAC value on the MIDI-to-voltage converter and the actual frequency played.

The closer together these points are, the more accurately you'll be able to hit a note.

There are many ways you can convert this set of points into a function to calculate the correct DAC values for frequency. Some involve fancy mathematics with exotic names, like sliding cubic splines. However, we did a very simple linear interpolation which basically plots a straight line between two points on the graph and takes that. It's not the most accurate method of calculating the value, but it's fast and straightforward. If you want more accuracy, it's easier to take more data points rather than do more fancy mathematics.

The code for this is as follows:

```
import array
import time
import math
import digitalio

import board
import usb_midi
import adafruit_midi
from adafruit_midi.midi_message     import note_
parser
from adafruit_midi.note_on          import NoteOn
from adafruit_midi.note_off         import NoteOff
from adafruit_midi.control_change   import
ControlChange
from adafruit_midi.pitch_bend       import PitchBend

import busio
import adafruit_mcp4728

i2c = busio.I2C(board.GP1, board.GP0)
mcp4728 = adafruit_mcp4728.MCP4728(i2c)

dacs = [mcp4728.channel_a, mcp4728.channel_b,
mcp4728.channel_c, mcp4728.channel_d]
gate_pins = [board.GP15, board.GP14, board.GP13,
board.GP12, board.GP11, board.GP10, board.GP9,
board.GP8]

gates = []

for pin in gate_pins:
    gates.append(digitalio.DigitalInOut(pin))

for gate in gates:
    gate.direction = digitalio.Direction.OUTPUT
    gate.value = False

midi_channel = 1
midi = adafruit_midi.MIDI(midi_in=usb_midi.ports[0],
                          in_
channel=(0,1,2,3,4,5,6,7))

basenote = 60
max_val = 65535 #the module takes a 16 bit number
even though it's a 12 bit dac


octave_size = int(65535/3.3)
notes = []

current_val = 0
inc = int(octave_size/12)
for i in range(int (3.3*12)):
```

Left
This is still the old front for PicoMidi. As you can see, setting the font too small can cause bits to break off

```
    notes.append(int(i*inc))

def noteToFreq(note):
    a = 440
    return (a / 32) * (2 ** ((note - 9) / 12))

frequencies = [96.1, 108, 124.4, 142.6, 164.7, 190,
219.1, 251.3,
               289.2, 334.3, 385.8, 442.8, 510.5,
587.1, 673.3, 775.2, 892]

dac_vals = [ 0, 4000, 8000, 12000, 16000, 20000,
24000, 28000, 32000, 36000, 40000, 44000, 48000,
52000, 56000, 60000, 64000]

def frequency_to_dac_val(want_freq):
    if want_freq < 96.2: return 0
    if want_freq > 892: return 64000


    for number,freq in enumerate(frequencies):
        print("number, freq: ", number, freq)
        if want_freq < freq:
            print("number: ", number)
            break

    gap = frequencies[number] - frequencies[number-1]
    position = want_freq - frequencies[number-1]
    proportion = position / gap

    dac_gap = dac_vals[number] - dac_vals[number-1]
    dac_position = dac_gap * proportion

    return int(dac_vals[number-1] + dac_position)

def note_to_dac_val(note):
    print("note: ", note)
    print("freq: ", noteToFreq(note))
    print("dac val: ", frequency_to_dac_
val(noteToFreq(note)))
    return frequency_to_dac_val(noteToFreq(note))

last_note = [0,0,0,0]

while True:
    msg = midi.receive()
    if isinstance(msg, NoteOn) and msg.velocity !=
0:
        if (msg.channel < 4):
            dacs[msg.channel].value = note_to_dac_
val(msg.note)
            last_note[msg.channel] = msg.note

        if (msg.channel < 8):
            gates[msg.channel].value = True

    elif (isinstance(msg, NoteOff) or
          isinstance(msg, NoteOn) and msg.velocity
== 0):

        if(msg.note == last_note[msg.channel]):
            gates[msg.channel].value = False
        else:
            pass
```

With this in place, we can accurately control our VCO from our MIDI controller.

We've now got cases, better sound out, and a MIDI system that can actually play in tune. In other words, we've more or less got a working MIDI synth. The only problem is that it's playing square waves and doesn't have any effects. Starting next issue, we'll get creative with our sound and look at ways of making it sound more interesting.

DON'T MISS THE **BRAND NEW** ISSUE!

SUBSCRIBE FOR JUST £10!

- **THREE!** issues of The MagPi
- **FREE!** Raspberry Pi Pico W
- **FREE!** delivery to your door

+ FREE RASPBERRY PI PICO W*

Three issues and free Pico W for £10 is a UK-only offer. Free Pico W is included with a 12-month subscription in USA, Europe and Rest of World. Not included with renewals. Offer subject to change or withdrawal at any time.

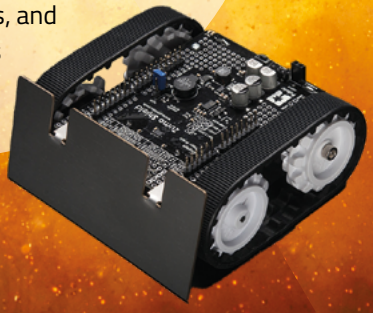
magpi.cc/subscribe

HackSpace
TECHNOLOGY IN YOUR HANDS

FIELD TEST

HACK | MAKE | BUILD | CREATE

Hacker gear poked, prodded, taken apart, and investigated



ONLY THE BEST

Bots! More bots and accessories

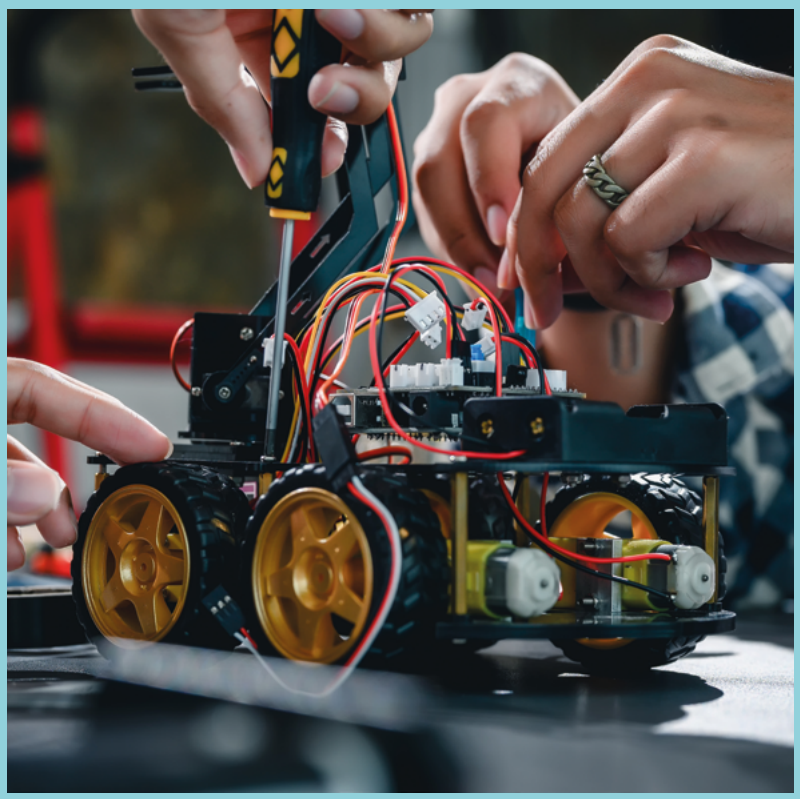
A collection of robotic-related kits and accessories

By **Marc de Vinck**

It's time to look at robots and robot accessories again. In the past, I have covered robotic arms, wheeled robots, walking bots, general robotic rovers and, speaking of rovers, I even covered robotic dogs and cats. Robotics has been around for a long time. It's a popular topic since it's also how many of us got into the DIY community. There are plenty of kits out there, from beginner to advanced, and hundreds, if not thousands, of different accessories.

I remember my first robot. It was a BEAM bot. If you aren't familiar with this type of simple robot, go online and do some research – you're in for a treat! Generally speaking, BEAM bots are robots that use simple analogue circuits rather than microprocessors for control. They are fun, simple to build and many times, when recycling an old piece of equipment like a VCR or other techno trash, you might find all the components needed to build one.

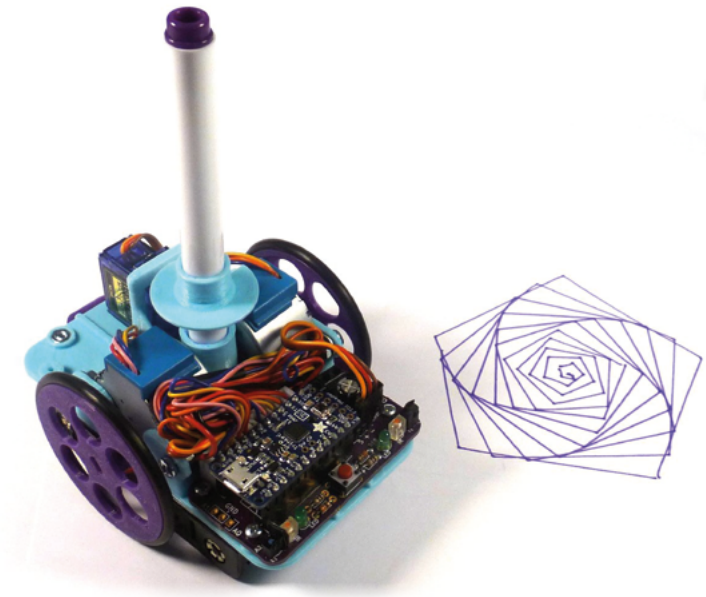
Once you build something like a simple BEAM bot, you'll inevitably want more control or functionality with your bot. Fortunately, there are so many tutorials online for building more advanced bots. It's a robust community, with lots of online resources and support. In this Best of Breed, I'll be looking at some of the robotic kits and accessories that didn't make it into my previous articles. There are plenty of them!

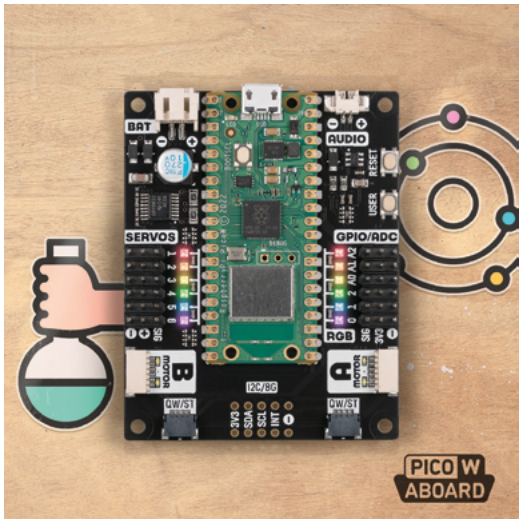


Open Source Turtle Robot Kit vs Inventor 2040 W

TINDIE $95 | tindie.com **PIMORONI** $37 | pimoroni.com

After making some BEAM bots, my next robot was in fact a turtle bot. I believe it was based on a PIC microcontroller. It wasn't easy to program, compared to modern-day microcontrollers, but it was still a lot of fun. This Open Source Turtle Robot Kit by MakersBox reminds me of my first turtle bot. However, this bot comes preloaded with an Adafruit ItsyBitsy M4 microcontroller with Python firmware. That sure does make it a lot easier to program! The main body of the robot is 3D-printed, and it includes everything you need to get up and running. It's easy to build and, once built, you can have fun changing the code and customising it. Made specifically for beginners, we think almost anyone will have fun with the Turtle Robot Kit. It's great being able to draw patterns and write words autonomously. See a video of the bot in action at **hsmag.cc/TurtleBotYT**.





The Inventor 2040 W from Pimoroni is a perfect board for making just about any kind of robotic contraption that you can imagine. You can power multiple motors, up to six servos, add a speaker, or interface with countless sensors. All this is thanks to the integrated Raspberry Pi Pico with its Dual Arm Cortex-M0+ running at up to 133MHz with 264kB of SRAM.

This is the kitchen sink breakout for the Raspberry Pi Pico. In addition to everything mentioned earlier, you can power your project with AA or AAA batteries, or even add a LiPo battery, thanks to its integrated circuitry. It also features twelve addressable LEDs, found on each of the servo breakouts and the GPIO/ADC channels. It's difficult to condense all the features into one review, so head over to the website to learn more.

VERDICT

Open Source Turtle Robot Kit

A classic bot to get started.

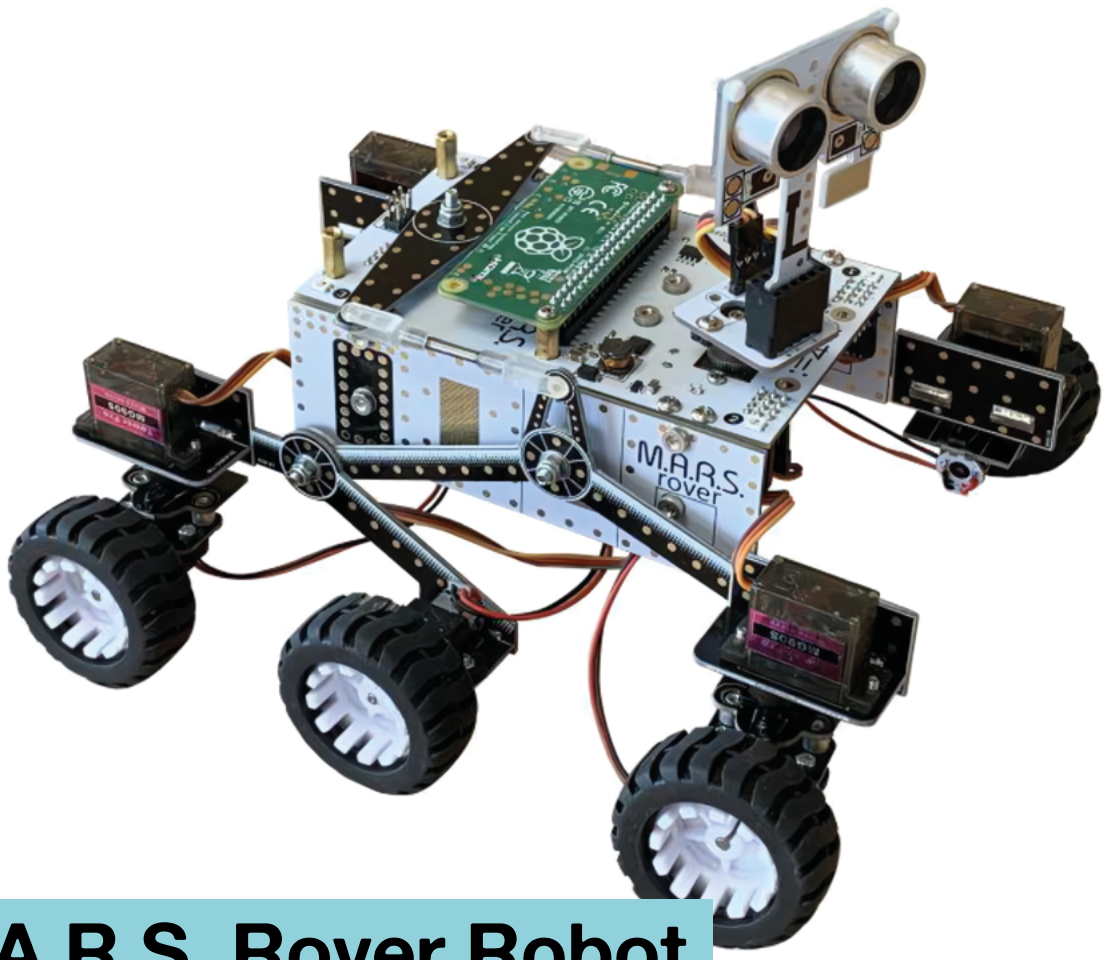
10/10

Inventor 2040 W

A great accessory for Pico owners.

10/10

BEST OF BREED



M.A.R.S. Rover Robot

PIMORONI $162 | pimoroni.com

The Mobile Autonomous Robotic System, aka M.A.R.S, is a Raspberry Pi Zero or micro:bit-based six-wheel-drive robotics kit. This robust robotics platform is capable of crawling over large obstacles, and navigating its environment, thanks to an ultrasonic distance sensor and steerable mast for additional sensors. The kit includes six motors, four servos, and eleven different PCB boards. Assembly takes a couple of hours. And once you've built it, you'll have a capable little bot that looks cool too. The design is inspired by the Mars Curiosity 2020 Rover from NASA/JPL. Go to the website to learn more, and remember to pick up a Raspberry Pi Zero or a micro:bit to go along with this kit, because they are not included.

VERDICT

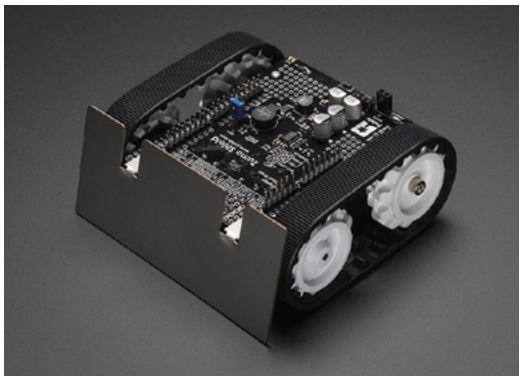
M.A.R.S. Rover Robot

A very cool-looking bot!

9/10

Zumo Robot for Arduino

ADAFRUIT $110 | adafruit.com



The Zumo Robot is an Arduino-based tracked robot that is less than 10 cm × 10 cm, which qualifies it for mini sumo competitions. Power is provided by two micro metal gearbox motors, and it features a stainless steel bulldozer-like blade for pushing around the competition. It has six infrared sensors for line following or edge detection, an accelerometer, a magnetometer, gyroscope, and a buzzer, allowing you to play your favourite victory song. No soldering required – just add four AA batteries and an Arduino, and you'll be all set for some robotic sumo wrestling.

VERDICT

Zumo Robot for Arduino

A clean-looking bot.

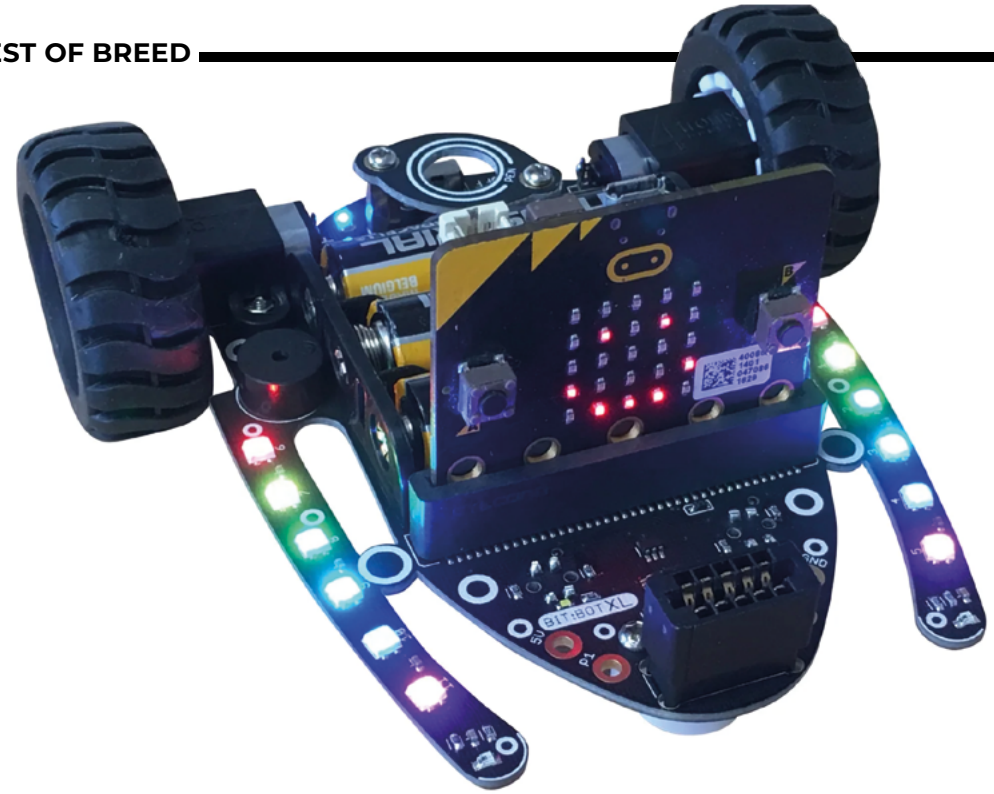
9/10

LEGO COMPATIBLE CONTINUOUS ROTATION SERVO

PIMORONI $6.35 | pimoroni.com

If you've got LEGO, and you want to start building bots, this Continuous Rotation Servo might be perfect for you. Unlike a typical DC motor, there is a servo hidden inside this plastic enclosure. This allows you to control both speed and acceleration. All you need is a microcontroller and some LEGO, and you're well on your way to building a bot.

BEST OF BREED



Bit:Bot XL Robot for micro:bit

PIMORONI $47 | pimoroni.com

The Bit:Bot XL Robot for micro:bit is a colourful little turtle-like robot kit based on the original, albeit much smaller, Bit:Bot. Just grab your micro:bit, add it to the bot, and get started coding in Microsoft MakeCode. It doesn't get much easier to build a bot. The bot is powered by two enclosed micro metal gear motors, and you can control speed and direction via the micro:bit. The robot has a front ball caster in lieu of traditional front wheels, allowing it to turn in any direction or even spin on its own axis. It also features twelve smart RGB LEDs, two digital line-following sensors, two analogue light sensors, a buzzer, and an integrated AA battery holder. It's a fun bot, especially for anyone just getting started with programming.

VERDICT

Bit:Bot XL Robot for micro:bit

Got a micro:bit? Grab this kit for some fun!

9/10

MECANUM WHEELS

PIMORONI $25.40 | pimoroni.com

Are you building a bot and trying to figure out how to integrate steering? How about a robot with mechanical parts related to steering? Just alter the speed and direction of the motors attached to the mecanum wheels, and you can turn your bot in any direction. It adds a bit of software complexity in lieu of mechanical complexity. And, if you've ever seen a robot with these types of wheels, you'll know just how cool it looks too!

Alpakka

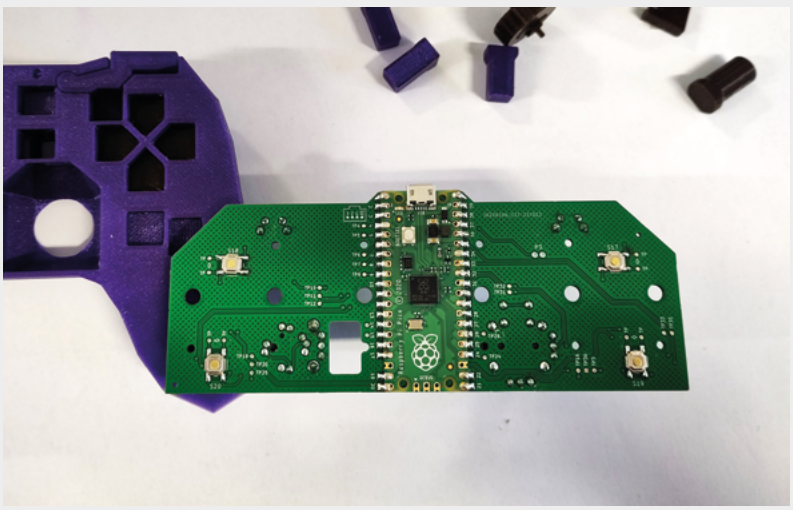
3D-print your own controller

INPUT LABS ◆ £VARIES | inputlabs.io

By Ben Everard

Making a games controller is easy – you just need a handful of buttons, a Raspberry Pi, and about 20 lines of CircuitPython. We've done it many times. However, making a good games controller is a different challenge entirely. Subtle changes in the form factor affect how it feels in your hand. The code has to relay the inputs accurately and with very low latency. What's more, if you want to make it reproducible by other people, you need a design that's easy to copy and good documentation to go with it. The team at Input Labs took on this challenge with the Alpakka gaming controller.

The design files are open-source, so you can start completely from scratch if you like, but we took the easier option of buying the base PCB that comes with one side of surface-mount components already soldered on. We also got the additional components directly from Input Labs to make sure we got the correct parts.



Below ◆
You have to solder this side of the PCB, but it's pretty straightforward

The remaining soldering is a mix of surface-mount and through-hole, but it's all very straightforward. There are no parts that should tax someone who's soldered before, and it would also make a reasonable first project, though given the cost of the PCB, it'd probably be prudent to get a practice board for your first few joints before taking this on. The assembly instructions take you through the process, and there's even a 3D-printable jig to hold the PCB steady while you solder.

With everything soldered into place, the next step is 3D printing the physical parts. The parts are designed to be printed in two different colours, but obviously this is up to you. A couple of the parts need specific slicer settings, and this is in the documentation. There is one part that should be printed in conductive filament because it acts as a touchpad. Printing in conductive PLA isn't particularly difficult; however, it is expensive to get even a small spool for just one part. There are a few hacks people have used, including putting a bolt through the case somewhere else and soldering this onto the touchpads, or covering a non-conductive PLA part in tinfoil. You can also purchase the part printed in conductive PLA but, at 16 euros, it's not cheap.

Tolerances on the 3D-printed parts are tight, and there aren't currently any different fit options for anything other than the touch-sensitive part. We found we had quite a bit of difficulty getting the

Above
The brown filament is recycled PLA coloured with algae

parts together. It took some shaving, sanding, and a bit of lubricant to get everything moving nicely. The problem's made worse by the fact that some of the switches that move the 3D-printed parts have very little push-back, so unless everything is moving freely, the buttons will stick. If you're thinking of building one, we'd recommend printing out a few of the parts (perhaps the bottom section and the trigger buttons), and see how the tolerances work on your printer.

Once you've bought, soldered, and printed all your parts, it's just a case of screwing everything together, and again, here the documentation walks you through the process.

> **It took some shaving, sanding, and a bit of** lubricant to get everything moving nicely

IN YOUR HANDS

Once you've built an Alpakka, it feels great in your hand. It's sturdy and just weighty enough to feel solid. The most unusual feature of the controller is the gyroscope. If you touch the conductive hexagon, the controller emulates a mouse, with the tilt of the controller defining the mouse position. This is pretty intuitive, and we found it quick and easy to get used to. This kind of sits the controller between a traditional gamepad (which typically has two thumb-sticks) and a keyboard-and-mouse desk setup. We found we got much more control with the gyro setup than a regular gamepad, but mouse-and-keyboard players might find it harder to get the same level of control.

Alpakka is almost endlessly hackable. Everything is based around the PCB, so unless you plan on respinning that, the button position is fixed, but the software and 3D design is all available for you to do what you want to with it. You don't have to dive all the way into the source code to do that – there are profiles that allow you to set different functions to different features on the controller.

Perhaps the most obvious missing feature from Alpakka is wireless support. This is something the team are working on with Pico W. If this is important to you, then it might be worth holding back and watching development until this is ready.

We love the idea of an open, hackable controller, and the Alpakka is well on its way to being this. Once wireless support is added, it'll compare favourably to any of the commercial controllers we've used.

VERDICT

A great, hackable controller. We'd just like a few tolerance options.

8/10

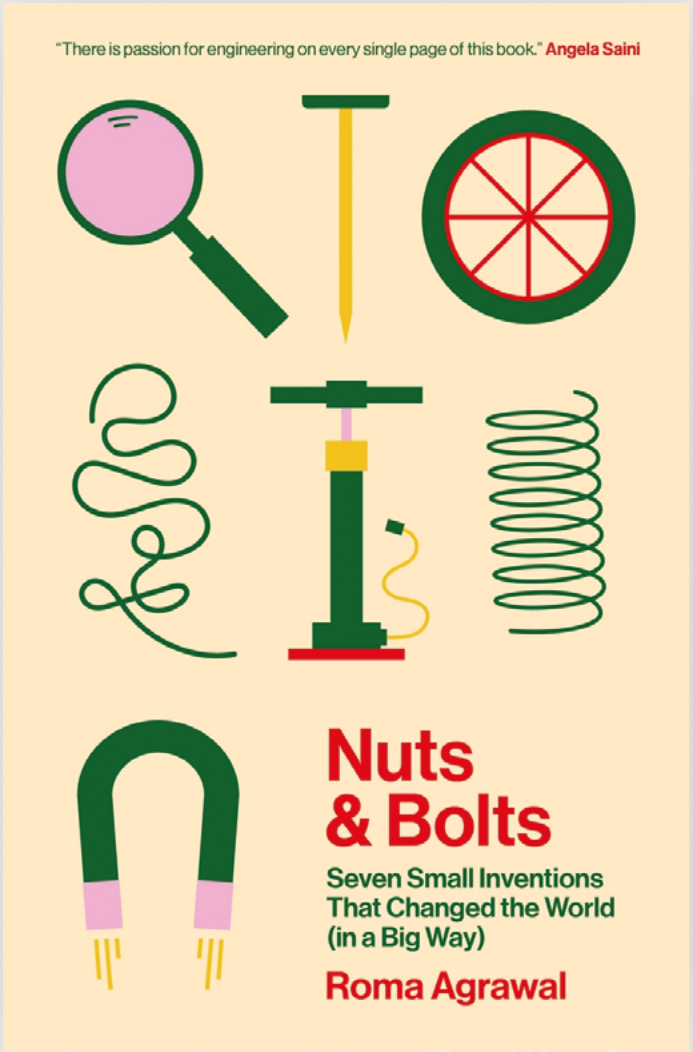
REVIEW

Nuts and Bolts

A brief history of engineering

ROMA AGRAWAL ◈ £22 | hsmag.cc/nandbbook

By Ben Everard



The life of a tech reviewer is thinking deeply – perhaps too deeply – about progress. Is version N+1 of a widget really better than version N? Yes, it has more gigabytes or transistors or fancier graphics, but what does that mean? Is it actually more useful? Will it result in an appreciable difference to your life?

Yes, there is some value in novelty and enjoyment to come from having a new shiny thing, but is it really worth spending the monetary equivalent of tens or hundreds of hours of work to upgrade?

Nuts and Bolts is the antidote to this. It claims to look at seven inventions that changed the world, but we're not sure this is quite accurate. It looks at seven lineages of inventions that have changed the world. For example, while the book gets its name from nuts and bolts, the actual invention talked about in the chapter is nails. Roma Agrawal follows this thread through rivets and screws until she reaches nuts and bolts in just the final three pages of the chapter.

Agrawal takes us through technology by thread, rather than by timeline, so the chapters cover similar time periods, though in many cases the technology is being developed in very different parts of the world. Chapter one takes us from nails, through rivets and screws, to nuts and bolts. Chapter two looks at perhaps the most famous invention of all time: the wheel. While this may seem like a single invention, as Agrawal points out, it has in fact been reinvented many times and, contrary to popular belief, reinventing the wheel is not always a bad thing. A spinning disc has powered everything from the chariot Boudica rode when fighting against the Roman legions, to the gyroscopes that help control the International Space Station.

Next, we look at the spring. This has had some obvious and not-so-obvious uses. As energy storage mechanisms, they have enabled devices from watches to archery bows, yet Agrawal also points out the significant role they play in enabling modern cities to exist through their vibration-isolating abilities. We simply wouldn't be able to live and work in close proximity if we had to live with all the noise we created echoing through the air.

> **It's a whirlwind tour rather than a** deep-dive into particular technologies

The magnets chapter probably travels the furthest in technological terms, as it goes from crude lodestone-based compasses in Song dynasty China to the World Wide Web. It's certainly arguable that this stretches the concept of a single invention beyond breaking point. However, there is a single thread of narrative that brings these together via Roma's relatives, so we'll allow her this indulgence in family history.

Two chapters on lenses and string follow, perhaps the most straightforward line of progress. While most have improved in many ways since their inception, a Neanderthal would still recognise modern thread, and scientists from the Middle Ages would recognise a modern lens. They are still primarily used to join stuff and to see things, respectively. An innovation here is in manufacturing and materials that allow them to perform their roles better while retaining the same basic form.

The final chapter follows the development of pumps. This is perhaps the outlier because, while the same – or at least similar – problem is being solved, there isn't a single thread of development between, for example, the irrigation system at the Hanging Gardens of Babylon and artificial hearts.

This book is well-written and excellently researched. It links together scientific traditions across the world in a coherent narrative. Given the scope of the material (which is most of technological progress over the past couple of thousand years), it is necessarily brief. This is perhaps best exemplified by the scant two and a half pages on nuts and bolts. It's a whirlwind tour rather than a deep dive into particular technologies. This speed lets you see how the forces that fundamentally change technology work, and focuses on the pivotal moments when technology takes a leap forward. It allows us to think about what technological improvement looks like without the marketing spin. By looking back, we can see how particular developments actually changed lives in a way that's hard to do when looking at new developments. Like much of history, it is an interesting story about the past that gives us useful lessons to incorporate into our daily lives – at least, it does if your daily life involves being a technology reviewer.

While it is quite self-indulgent to review the book from our own perspective, it does get to the heart of what this book is about: what actual technological progress looks like. As you're reading this, you're obviously the sort of person who reads tech reviews. Hopefully, it'll help you detect when we reviewers get a bit too caught up in the hype. □

VERDICT

An entertaining summary of what engineers have been up to for the past 2000 years.

10/10

CROWDFUNDING NOW

Pi-Cast KVM

Access your computer from anywhere

From $199 | hsmag.cc/picast | Delivery: June 2024

Not to be confused with Kernel-based Virtual Machines, the KVM in this product stands for Keyboard, Video, and Mouse. This device uses Raspberry Pi's USB Gadget mode to appear to a computer that it's plugged into as a Keyboard and Mouse (and also other USB Gadgets, such as mass storage).

You can plug this into any computer that supports USB and HDMI, and it will take video and give input to that machine. This means that you can control the computer from the PiKVM device which, in turn, can be controlled over the internet. This gives you full remote access to a computer – not just in the usual way via the operating system, but as much control as if you were sitting there. You can interact with BIOS, emulated plugging and unplugging USB mass storage devices, and more. Pi-Cast KVM is based on PiKVM, an open-source project that also releases hardware. However, the official PiKVM boards are closed-source, while Pi-Cast is open (at least as far as schematics). We haven't tested any of them out, but they have a similar feature set. ◻

BUYER BEWARE

When backing a crowdfunding campaign, you are not purchasing a finished product, but supporting a project working on something new. There is a very real chance that the product will never ship and you'll lose your money. It's a great way to support projects you like and get some cheap hardware in the process, but if you use it purely as a chance to snag cheap stuff, you may find that you get burned.

Sake Hack

Drink in the trees

From $76 | **hsmag.cc/SakeHackProject** | **Delivery: September 2024**

The idea behind this is simple. Take a spirit of your choice, and a wood of your choice, and infuse the spirit with the wood in just half a day. This sounds like a great idea. We have some questions though. Firstly, how? As far as we can see, you get a bottle and a bit of wood. It takes a lot longer than half a day to get the flavour from wood into alcohol. We know, we've tried. Unless they're using some clever trickery they don't talk about, we can't see how you'd get much flavour from this.

Secondly, why this? If it is indeed a bottle and a block of wood, why do I need this particular bottle and block of wood? Timber prices may have gone up in recent years, but they're not quite at this level yet.

We're a bit cynical about this, but we haven't tested it out, so perhaps we're missing something. It would be lovely to experiment with wood-flavoured drinks without having to wait years, and perhaps the good folks at Sake Hack have discovered how to do that, but we just wish that they'd tell us a little more about it.

next month

issue #76 ON SALE **22 FEBRUARY**

THE ULTIMATE WORKSHOP

ALSO

- RASPBERRY PI PICO
- 3D PRINTING
- LASER CUTTING
- ELECTRONICS

AND MUCH MORE

—

DON'T MISS OUT

hsmag.cc/subscribe

Fractal fun

Fractals are everywhere. The branches on trees exhibit fractal-like behaviour, as the veins on leaves, the blood vessels that supply our organs, the ice crystals within snowflakes, and the endlessly undulating coastline. It turns out that our teachers were right: mathematics really is beautiful, if only you're looking in the right place.

This issue, Karl Mose has brought the beauty of fractals inside using an incredibly simple build: just a Raspberry Pi Zero, a screen, and a frame of your choice are all it takes to build a portal into an ever-changing, endlessly fascinating world. And it beats the hell out of algebra.

PiKVM

Remote control **redefined**

Manage your servers or PCs remotely!

POWERED BY Raspberry Pi

PiKVM V4 Mini

Small, cost-effective, and powerful!

- Power consumption in idle mode: just 2.67 Watts!
- Transfer your mouse and keyboard actions
- Access to all configuration settings like UEFI/BIOS
- Capture video signal up to 1920x1200@60 Hz
- Take full control of a remote PC's power

PiKVM V4 Plus

The most feature-rich edition

- More connectivity
- Extra storage via internal USB 3.0
- Upgraded powering options
- More physical security features
- Extra HDMI output
- Advanced cooling solution

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

Available at the main Raspberry Pi resellers

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

Pi-Shop.ch · The PiHut · PiShop.us · CloudFree · KUBii · Cytron Marketplace

ROBU.IN Your ideas, Our Parts · NETTOP COMPUTERS & ELECTRONICS · netsket mall · DIREKTRONIK · RaspberryPi.dk

HiPi.io

No reseller in your country?
Check shop.hipi.io (import fees might apply).

List of official resellers by country: