

MAKE | BUILD | HACK | CREATE

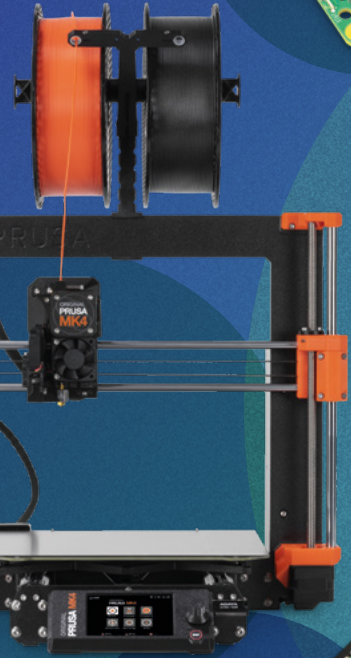
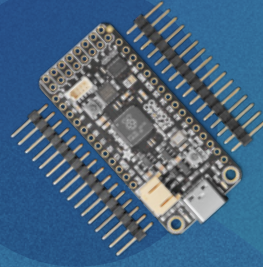
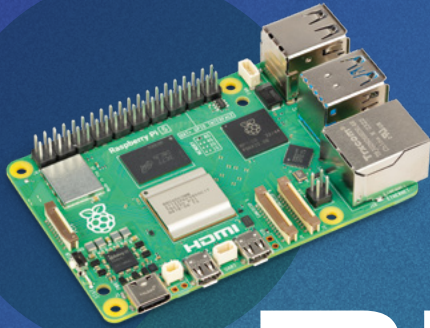
HackSpace

TECHNOLOGY IN YOUR HANDS

hsmag.cc

January 2024

Issue #74



BEST MAKER TECH

essential new toys for
makers and hackers

Cocktail Robot

Meet the Arduino
that mixes drinks

MIDI Music

Generate sound
on small devices

Bike Lights

Demand attention
with programmable LEDs

Jan. 2024
Issue #74 EG



PIXELBOT SKULLS PCBS CAMERAS

Free eBook!



Download your copy from
hsmag.cc/freecadbook



Welcome to HackSpace magazine

What do you need to be a maker? If we're honest, nothing except a fertile imagination and a willingness to get stuck in. However, life isn't always about need, so this month we're looking at the stuff we want. We've got all the latest shiny gizmos and gadgets that a maker could want, from dev boards to CNC machines.

Behind the scenes, a lot has happened in the industrial world this year, but the short version is that it's now much easier for manufacturers to get the parts they need to make things, and we've seen a flurry of new hardware because of this. Everything from 3D printers to dev boards seem to have taken a leap forwards in the last twelve months.

Will they make you a better maker? Maybe. Are they fun to play with? Definitely.

BEN EVERARD

Editor ben.everard@raspberrypi.com

Got a comment, question, or thought about HackSpace magazine?

get in touch at hsmag.cc/hello

GET IN TOUCH

hackspace@raspberrypi.com

[hackspacemag](#)

[hackspacemag](#)

ONLINE

hsmag.cc



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, Jo Hinchliffe, Andrew Lewis, Rob Miles, Sven Kroesen, Phil King

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



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.

HackSpace magazine is published by Raspberry Pi Ltd, 194 Science Park, Cambridge, CB4 0AD. The publisher, editor, and contributors accept no responsibility in respect of any omissions or errors relating to goods, products or services referred to or advertised. Except where otherwise noted, content in this magazine is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported (CC BY-NC-SA 3.0). ISSN: 2515-5148.

Contents

17

LENS

- 18 **Best maker tech**
The ultimate toys to improve your making
- 28 **How I Made: Cocktail machine**
Let an Arduino mix your drinks this Christmas
- 36 **Interview: Jude Pullen**
Why single-use vapes portend the apocalypse
- 44 **Objet 3d'art**
A mechanical computer made of plastic
- 46 **Letters**
The youth of today

Cover Feature



Tutorial

3D printing



Browse the Natural History Museum from the comfort of home



Interview

Jude Pullen

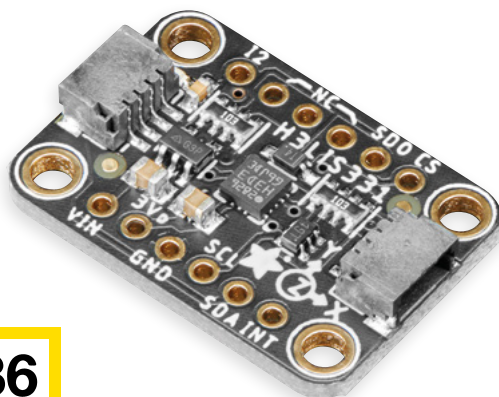


36 It's like the Beastie Boys told us – you've got to fight for your right to repair

49

FORGE

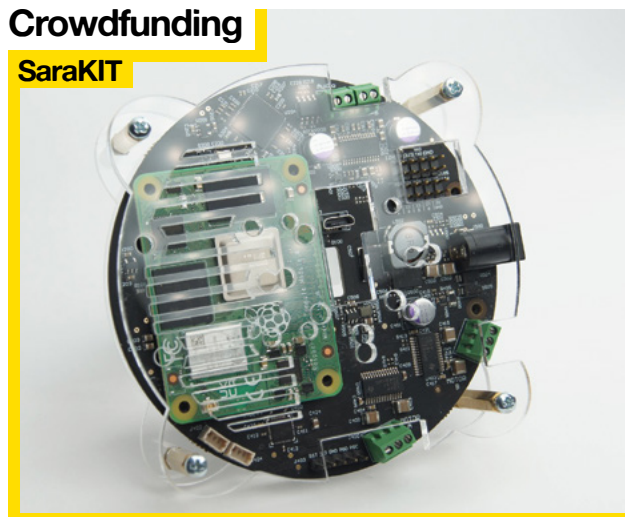
- 50 SoM Modular Pico**
Add MIDI input to a DIY synth
- 54 Tutorial Raspberry Pi**
Do more with a Raspberry Pi Camera Module
- 60 Tutorial KiCad**
Explore difference PCB substrates
- 66 Tutorial 3D printing**
The ethical way to surround yourself with skulls
- 68 Tutorial Camera modification**
Use new tech to restore old tech
- 72 Tutorial Hull Pixelbot**
A robotic programming platform
- 80 Tutorial Bike lights**
Stay safe with RGB LEDs



86

Crowdfunding

SaraKIT



96 Break out the power of the Raspberry Pi Compute Module

85

FIELD TEST

- 86 Best of Breed**
Circuits to make rockets fly straight and true
- 92 Review Porthcurno recycled nylon filament**
Print with old fishing nets
- 96 Crowdfunding**
Control the way you apply ink to T-shirts

Some of the tools and techniques shown in HackSpace Magazine are dangerous unless used with skill, experience and appropriate personal protection equipment. While we attempt to guide the reader, ultimately you are responsible for your own safety and understanding the limits of yourself and your equipment. HackSpace Magazine is intended for an adult audience and some projects may be dangerous for children. Raspberry Pi Ltd does not accept responsibility for any injuries, damage to equipment, or costs incurred from projects, tutorials or suggestions in HackSpace Magazine. Laws and regulations covering many of the topics in HackSpace Magazine are different between countries, and are always subject to change. You are responsible for understanding the requirements in your jurisdiction and ensuring that you comply with them. Some manufacturers place limits on the use of their hardware which some projects or suggestions in HackSpace Magazine may go beyond. It is your responsibility to understand the manufacturer's limits. HackSpace magazine is published monthly by Raspberry Pi Ltd, 194 Science Park, Cambridge, CB4 0AB, United Kingdom. Publishers Service Associates, 2406 Reach Road, Williamsport, PA, 17701, is the mailing agent for copies distributed in the US and Canada. Application to mail at Periodicals prices is pending at Williamsport, PA. Postmaster please send address changes to HackSpace magazine c/o Publishers Service Associates, 2406 Reach Road, Williamsport, PA, 17701.

ScottoWing

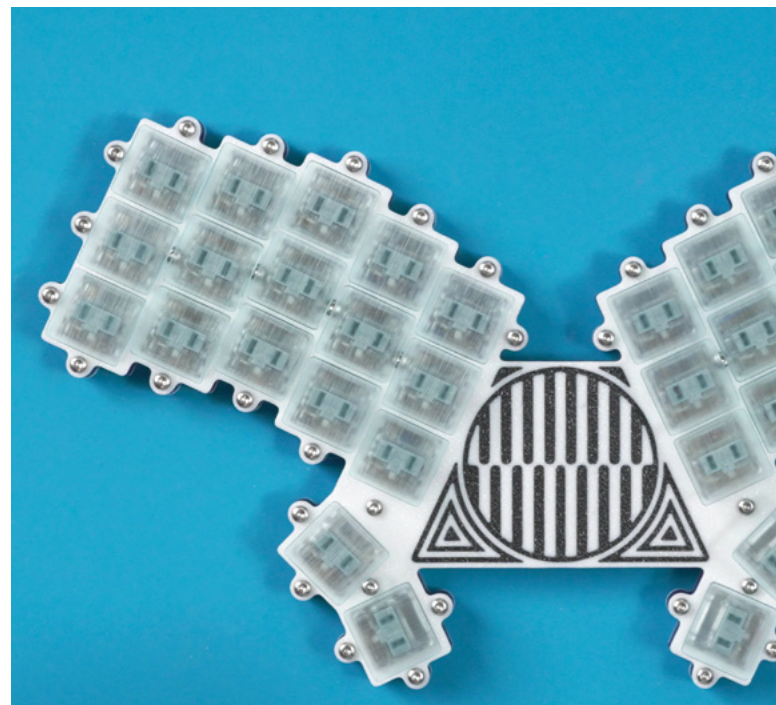
By Joe Scott

hsmag.cc/ScottoWing

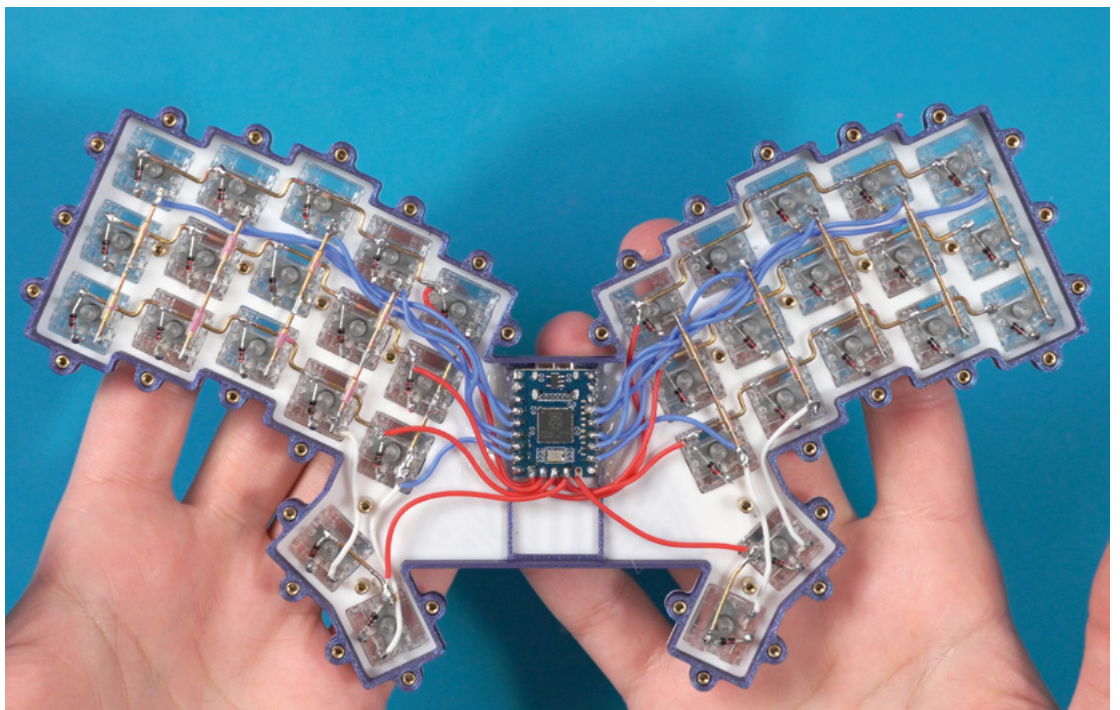
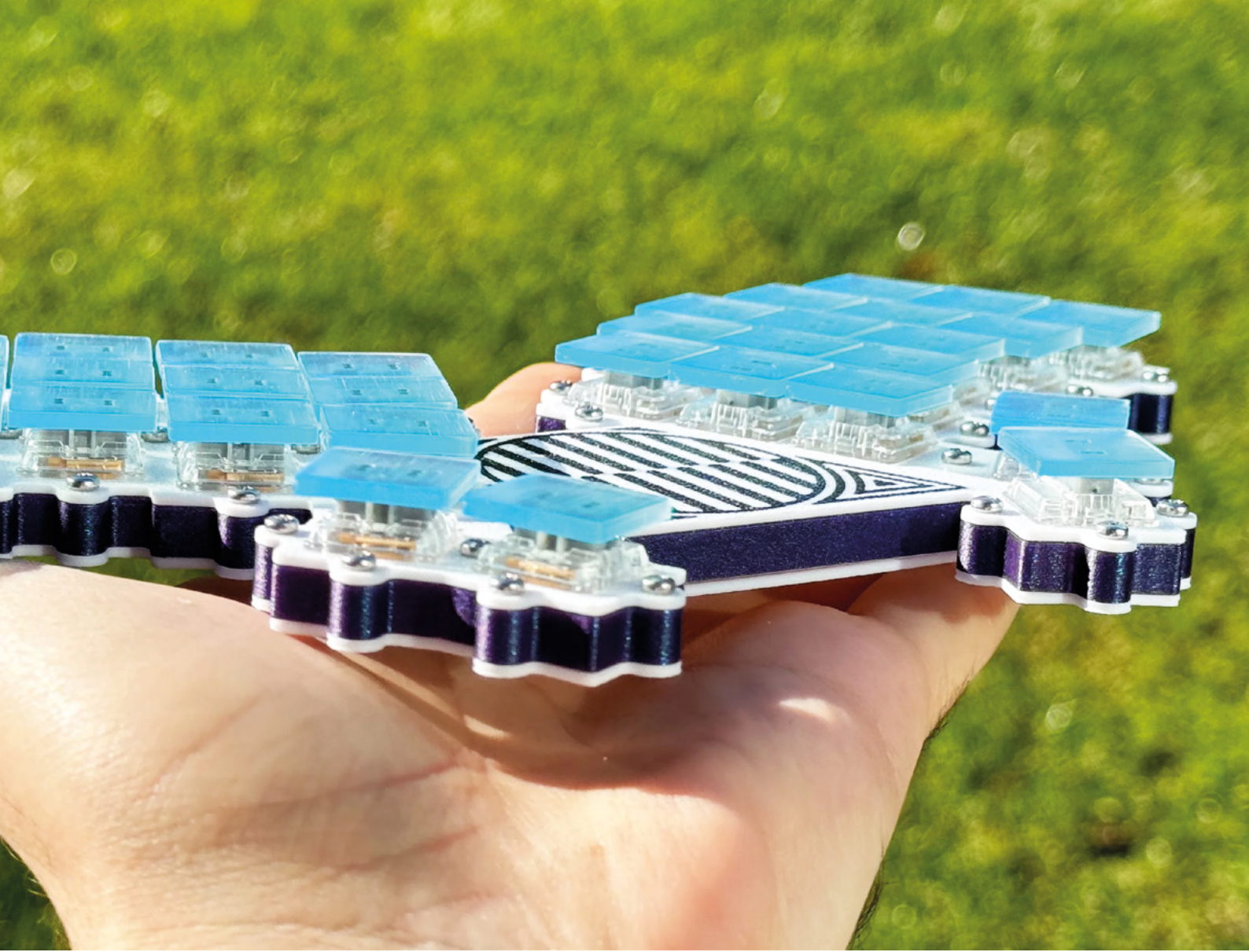
Having just made the plunge and bought our first mechanical keyboard, we're amazed by the scope and variety of what are essentially identical objects. To our previously unenlightened minds, a keyboard was a keyboard was a keyboard.

The idea that people would obsess over layout, or switches, or back-lighting, or programmability was a little bit weird. Why bother?

But that was then: this is now. Using a mechanical keyboard makes writing more fun. In the same way Jude Pullen talks about loving a drill in our interview on page 36, we've come to love our keyboard. It brought us some joy to discover the work of Joe Scott who, under the nom-de-making ScottoKeebs, is making some gorgeous mechanical keyboards. This latest, the ScottoWing, is a fully customisable ortholinear keyboard (that means that keys aren't staggered, but arranged directly above one another). Scott's made the design files available, so anyone with access to a decently equipped maker space should be able to make their own – and check out that beautiful hand wiring on the back. □



Right ♦
That's an RP2040 Zero on the back making sense of the key switches



SnowByte

By cele9999

hsmag.cc/SnowByte

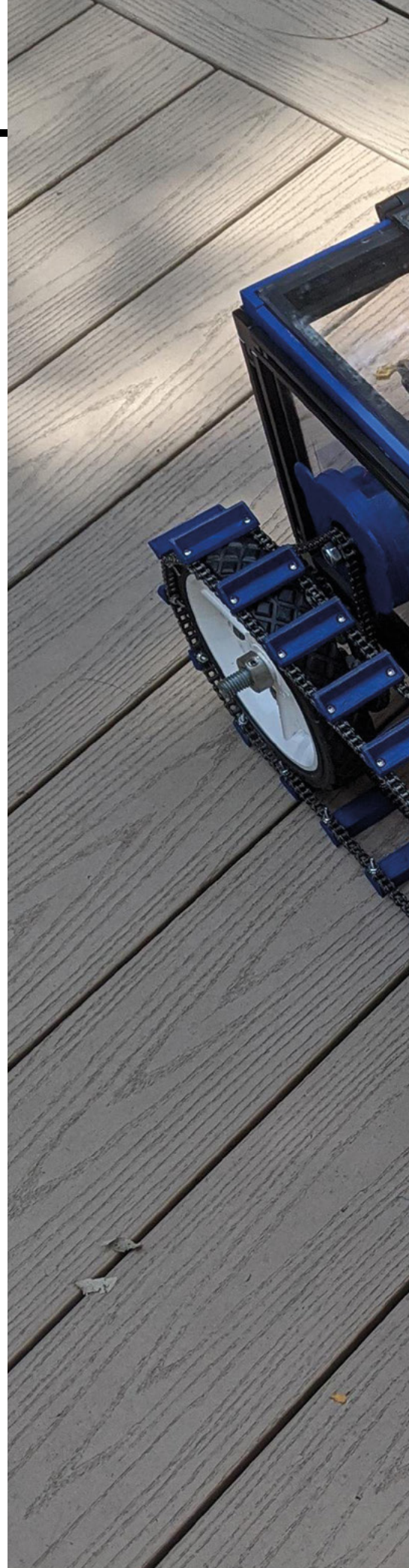
What do you do when you can't shovel snow off your drive because of medical reasons?

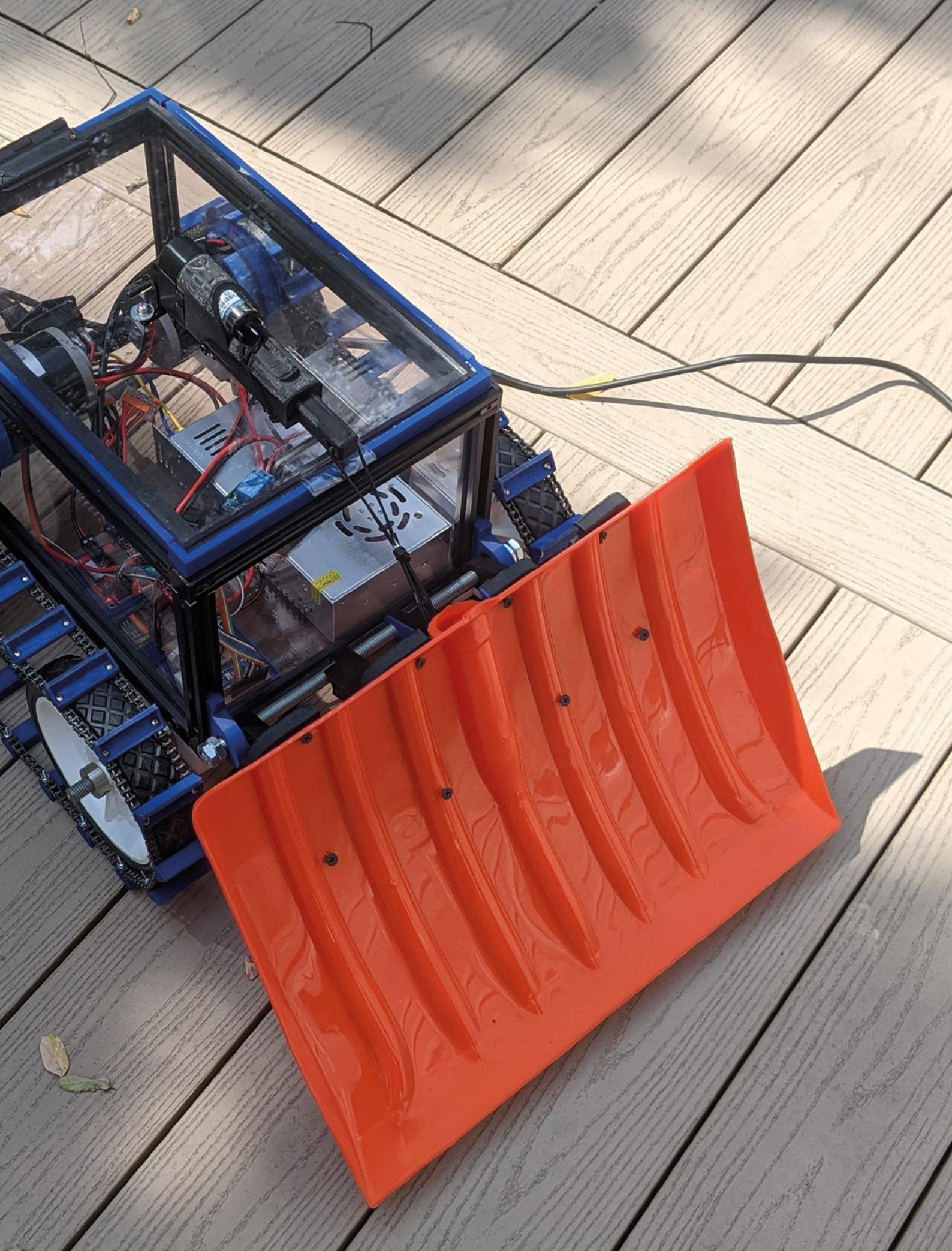
You build your own snowplough, that's what! There are other solutions on the market, but none of them were quite right for the maker of this brilliant, versatile robot. At around 0.4 horsepower, it's powerful enough to shift a decent amount of the white stuff, and as it's not designed to go far, it's suitable for mains power (the files are open-source, so if you want to make your own, do be aware that it's spec'd to run on 120V – your mains power may vary).

The maker has also designed 3D-printable files for converting the SnowByte to other tasks, such as mowing the lawn and, thanks to the magic of open source, you're welcome to design your own. □

Right □

The materials for this project cost \$645 – that sounds like a lot, but you have to remember that there are some chunky components in there that deal with high voltages, such as car fuses





Yozh robot

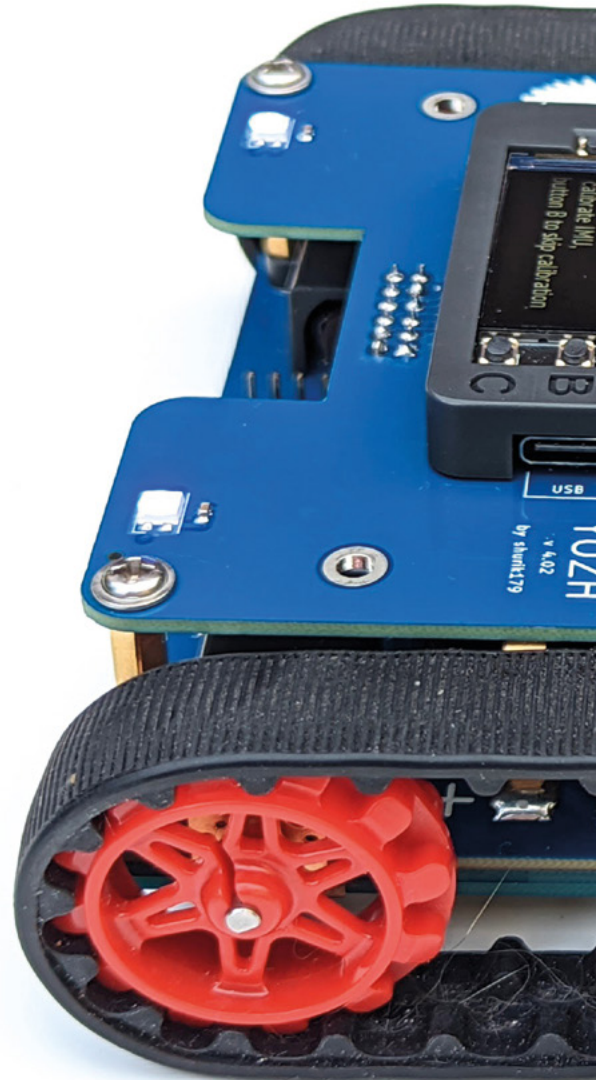
By Alexander Kirillov

hsmag.cc/Yozh

Alexander Kirillov wanted a robot that he could use as a teaching platform at SigmaCamp, a science-based summer camp in the USA. He looked at Poll's Zumo robot, DF Robots' Maqueen Plus, and Pimoroni's Trilobite – all fine platforms, but none of them satisfied all of Alexander's requirements. He wanted something compact, extendable, with a decent number and variety of sensors, and programmable in Python. Unable to find a device that matched these requirements, he decided to build his own.

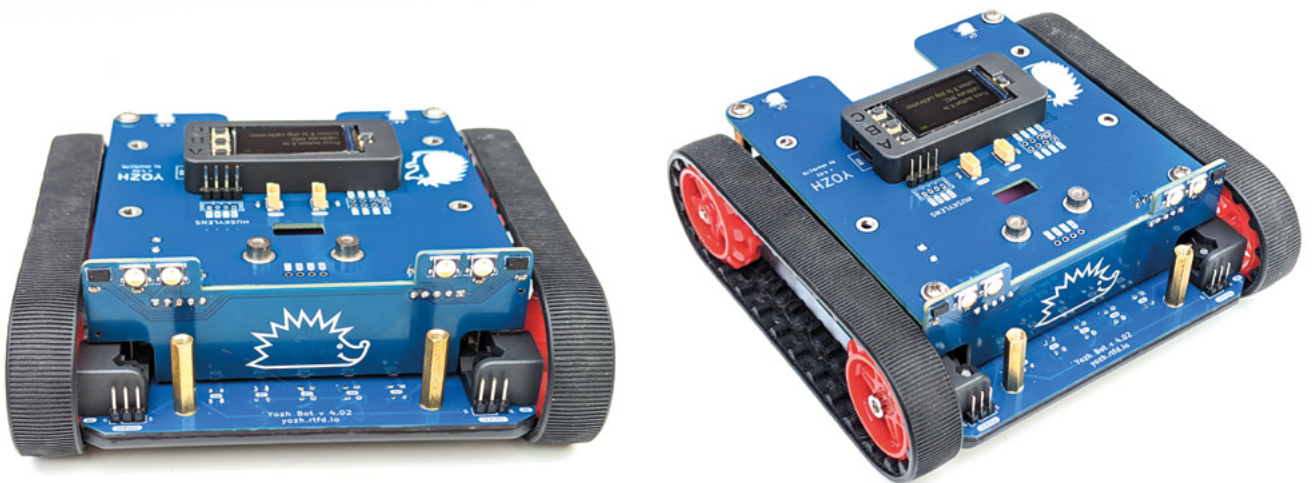
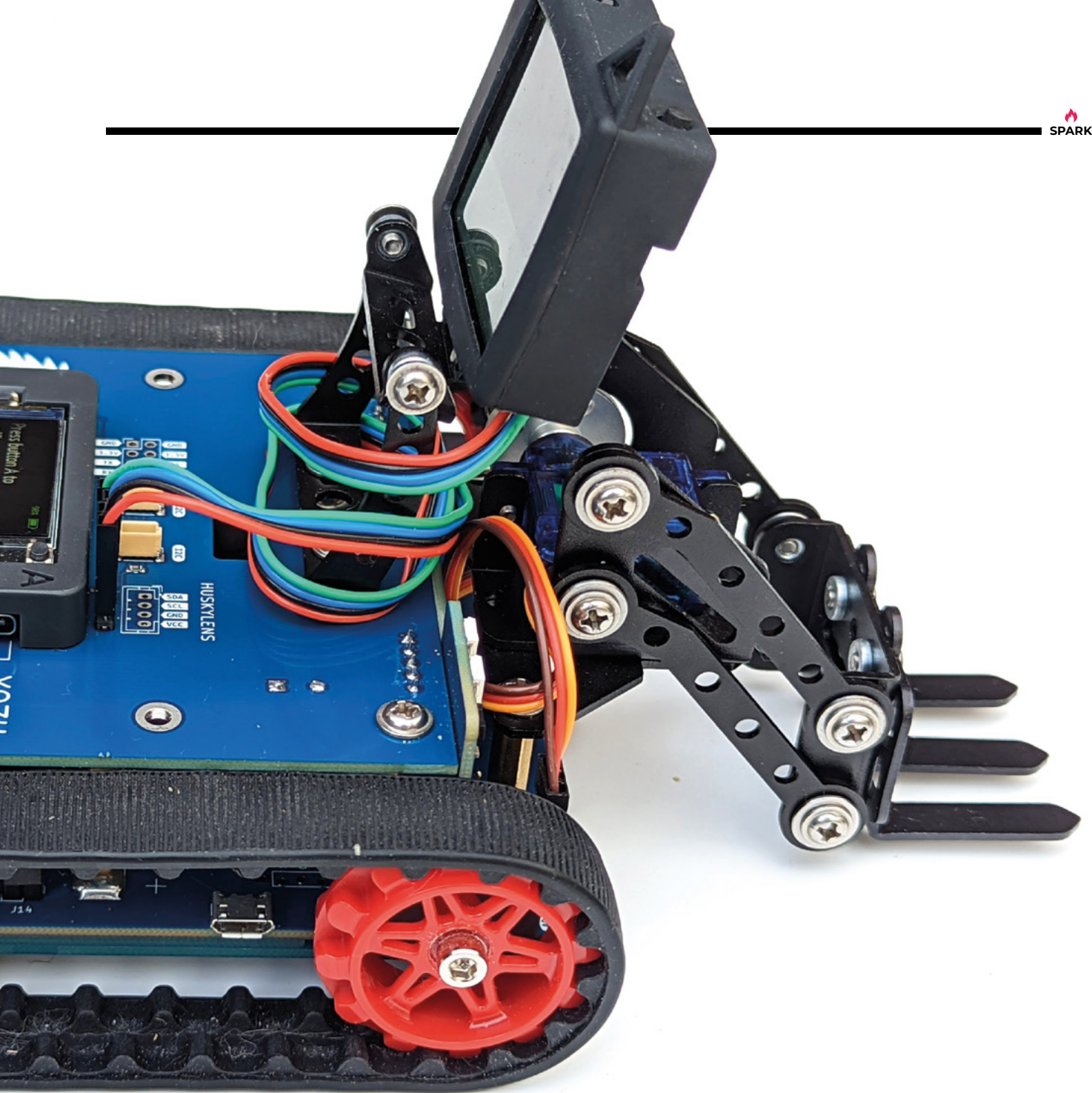
He's used silicone tracks and 6V, 75-gear ratio micro metal gear motors, both by Pololu, an ESP32-S3 Feather board by Adafruit as the main controller, and designed a custom board to handle low-level operations such as maintaining consistent motor speed.

The sensors are where things get really interesting: among other things, there's a 6 DOF Inertial Motion Unit (IMU), which can be used for determining robot orientation; two front-facing laser time-of-flight sensors, for detecting obstacles; and no fewer than seven sensors facing the ground, to enable line-following tasks. □



Right □

One of the design requirements for Yozh was that it should be neat – no spaghetti of wires sticking out of it



Stepper motor piano

By [thisisnomine](#) hsmag.cc/StepperMotorPiano

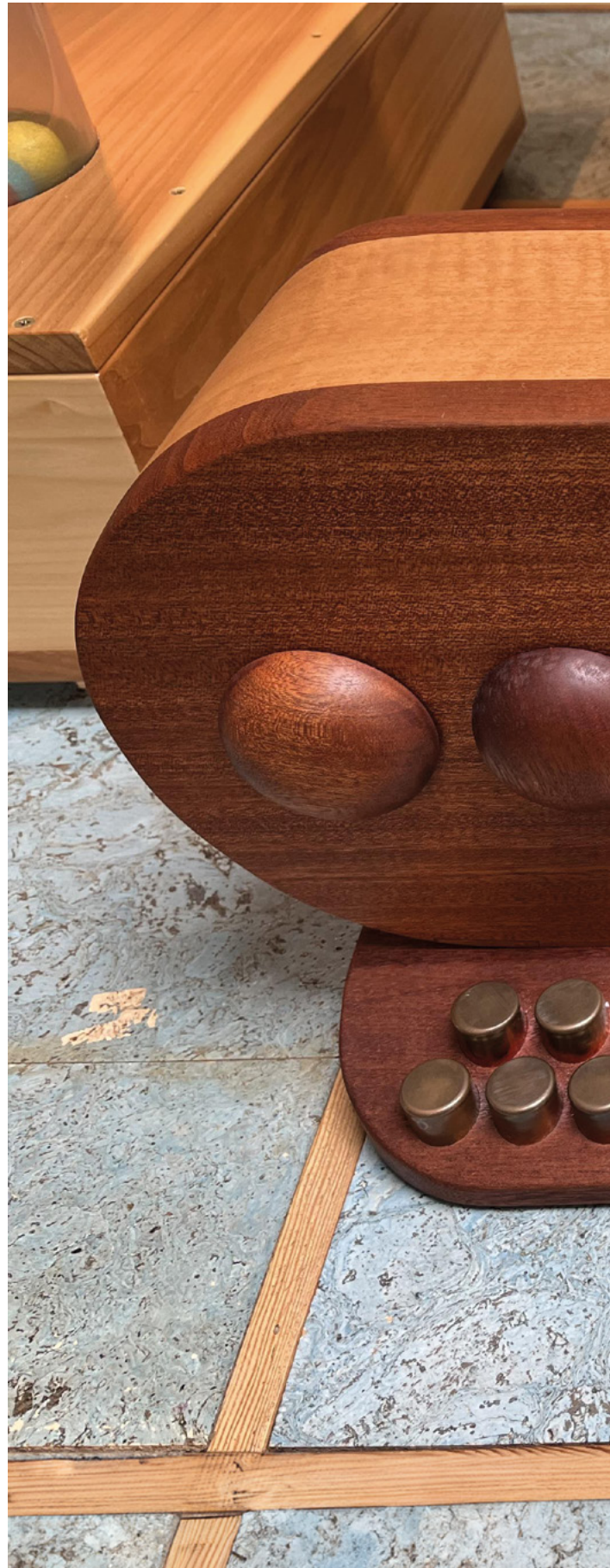
Just because you're experimenting with stepper motors, perfboard, and a whole nest of wires, doesn't mean you shouldn't try your best to make your creations beautiful. Take this thoroughly modern musical instrument by Instructables user thisisnomine:

in the back it's your typical maker project, with PCBs, breakout boards, and wiring. From the front, though, it's a feast of mid-century hardwood, with mother of pearl detailing and a combination of two African hardwoods (Anigre, the light coloured wood, is used here as a veneer, with the rest of the housing made from sapele, which is similar to mahogany).

The music comes from the four stepper motors. These are usually found in robotics projects, as the degree, speed, and direction of rotation can all be controlled in software. The rotation creates noise; when controlled, the noise becomes a note, which the user can change using the capacitive touch keyboard. It's quite simple, and all the more brilliant for that. □

Right □

The copper keys of this instrument are attached to two capacitive touch sensor breakout boards, which, in turn, tells the Adafruit Metro Mini how fast to spin the stepper motors





Strider robot V8

By Moonounation

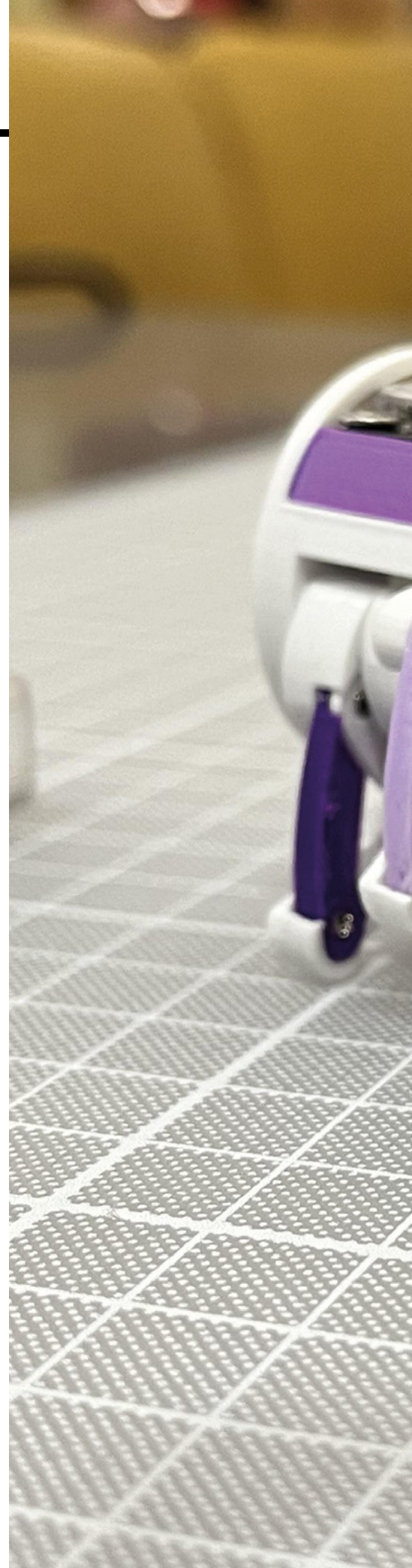
hsmag.cc/strider

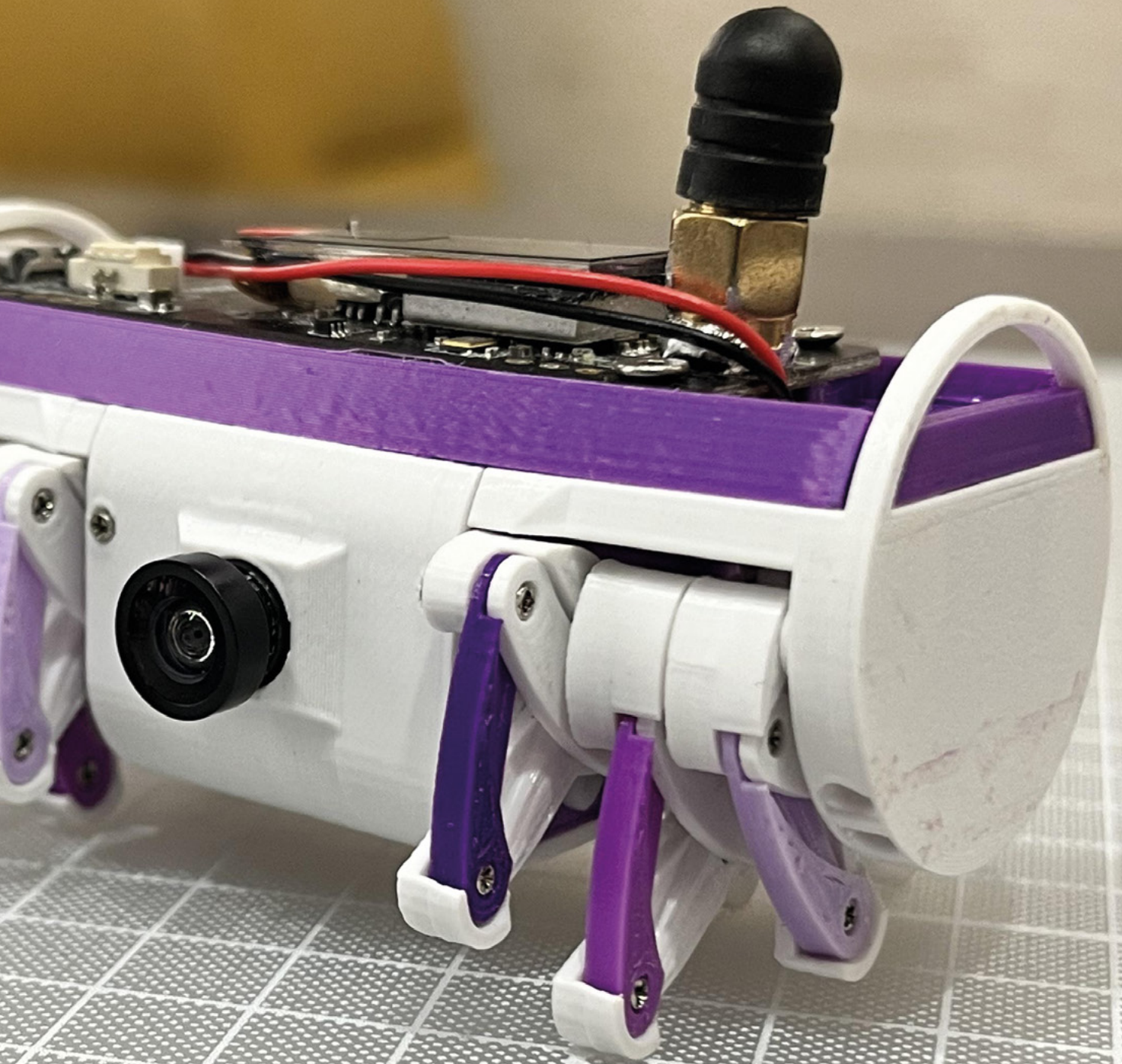
We're used to the creepy, cam-driven walking action of Theo Jansen's Strandbeests. It seems like wherever there's a 3D printer and a human with an interest in mechanical movement, a design will pop up that turns rotation movement into walking. In one sense

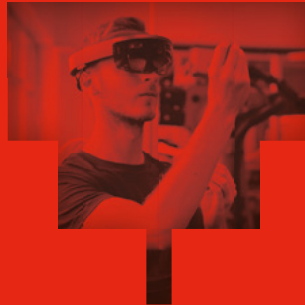
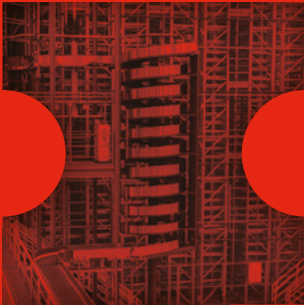
this robot is no different; it uses two servos to turn a crank-shaft, which enables the robot to move around.

What grabbed us about this project is that, as well as 3D printing, there's an older and more primitive technology at use here: a bit of bent metal. Rather than 3D printing the crank-shaft, maker Moonounation used a bit of 1 mm steel rod, bent so as to allow movement to transfer to the legs. In a triumph to progress, the maker recommends 3D-printing a crank-shaft for reference, to check that you're bending the metal in the right place. ▣

Right ▣
With its transversally mounted engine, the Strider robot V8 is the direct descendant of Alec Issigonis' original Mini car







Your trust is our goal

From genuine, manufacturer-warranted 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

DigiKey is a franchised distributor for all supplier partners. New products added daily. DigiKey and DigiKey Electronics are registered trademarks of DigiKey Electronics in the U.S. and other countries. © 2023 DigiKey Electronics, 701 Brooks Ave. South, Thief River Falls, MN 56701, USA

 **ECIA MEMBER**
Supporting The Authorized Channel

LENS

HACK | MAKE | BUILD | CREATE

Uncover the technology that's powering the future

PG
28

HOW I MADE: COCKTAIL MACHINE

Take the effort out of making a Long Island iced tea with an Arduino-controlled bartender



PG
36

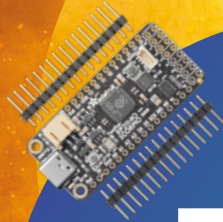
INTERVIEW: JUDE PULLEN

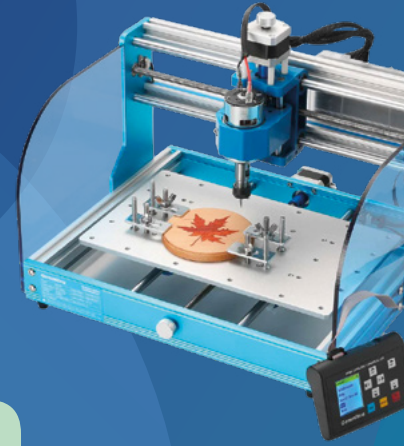
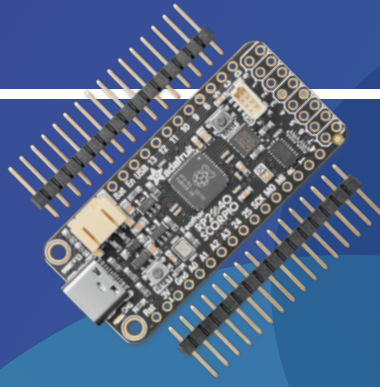
How one man's quest to change a dead battery turned into a fight for the right to repair

PG
18

BEST MAKER TECH

The gear that every maker wants to get hold of in 2024



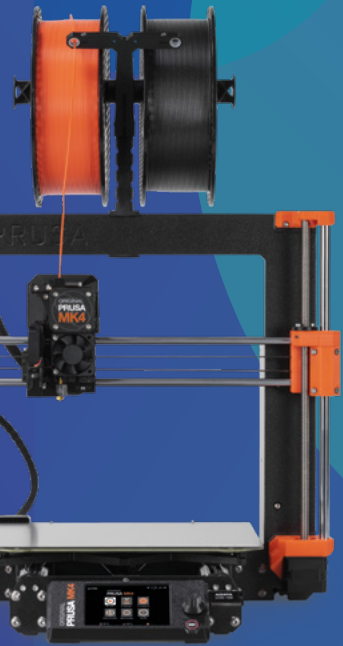


BEST MAKER TECH

The best new devices, machines, tools,
and accessories for makers

BY PHIL KING

Choosing exactly what to use to power projects and be creative is a conundrum for any maker. To make it a little easier, we've rounded up the best recently released maker tech. From microcontrollers and single-board computers to breakout boards, accessories, 3D printers, laser cutters, and CNC routers, we've got it all covered. →



MICROCONTROLLERS & SINGLE-BOARD COMPUTERS

Base your projects around these general-purpose and specialist boards

Arduino Uno R4

The long-awaited new addition to the popular Uno range of Arduino boards, the R4 – available in WiFi and Minima versions – has a few surprises up its sleeve.

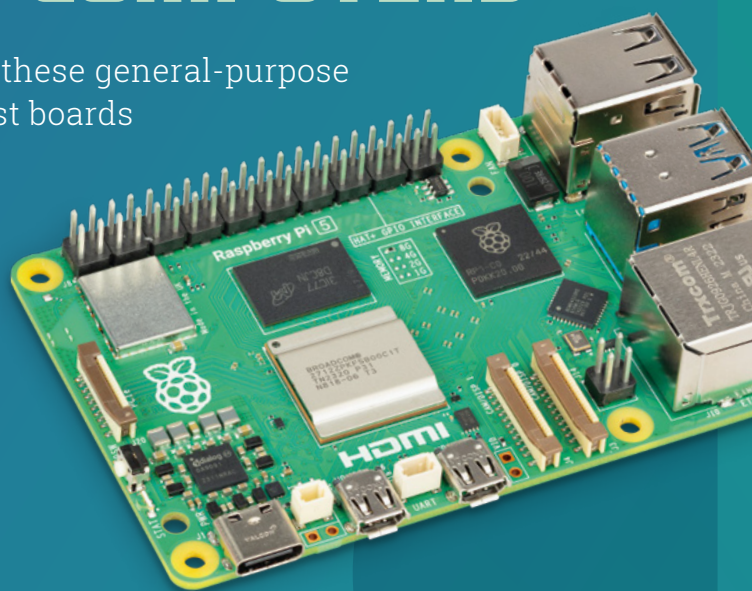
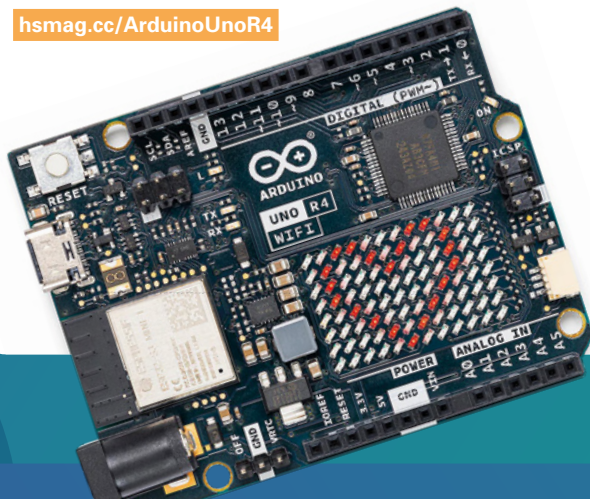
Possibly the biggest is the switch from an AVR-based ATmega 328P microcontroller to an Arm-based Renesas RA4M1 chip, which is superior in every respect: faster, more RAM, more storage, and extra on-board peripherals – including a 12-bit DAC, OpAmp, and CAN bus.

The change in chip architecture does mean that not all R3 code will work on an R4, but on the whole, it's fairly compatible. The pinout is also the same, and the board still operates at 5V. The WiFi version of the R4 also features a pair of 3.3V GPIO pins in the Qwiic I2C connector. In addition, the WiFi model adds a 12x8 red LED matrix and, as the name suggests, a wireless network controller (for WiFi and Bluetooth) – naturally, it's compatible with the Arduino Cloud for IoT projects.

Both R4 versions have a 24V tolerance for power input, which is useful for integrating motors, LED strips, and so on. Another neat feature that we appreciate is the transparent plastic base plate that clips onto the rear of the board, which has matching screw holes for easy mounting and also side clips for securing a breadboard.

It all adds up to a powerful, feature-packed microcontroller board that should ensure the Uno form factor's continued popularity with makers and educators.

hsmag.cc/ArduinoUnoR4



Raspberry Pi 5

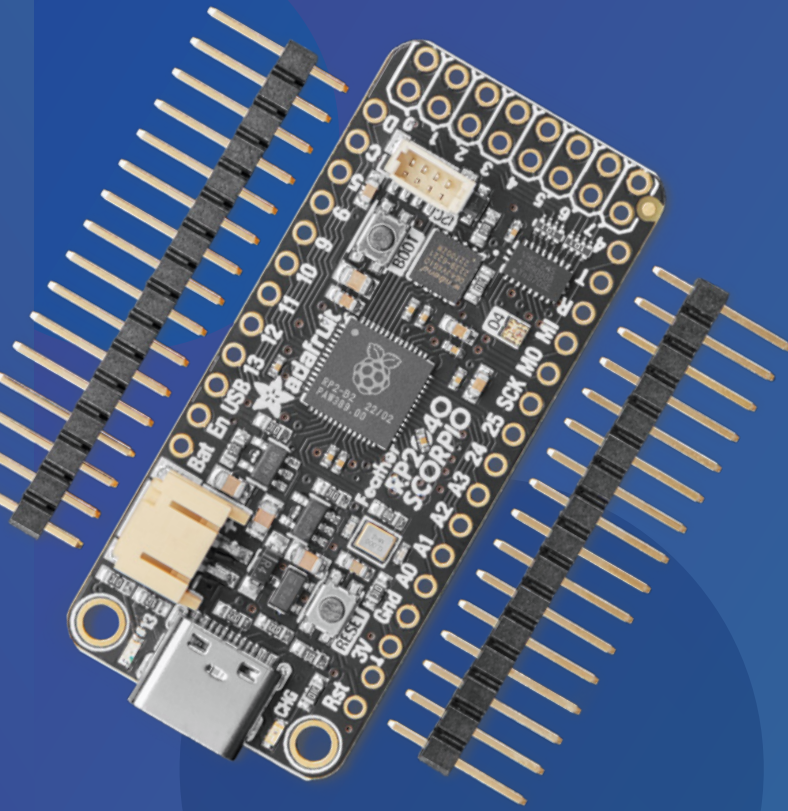
Since HackSpace is published by Raspberry Pi Ltd, we might be a bit biased, but there's little doubt that the latest Raspberry Pi model has a lot going for it. It's the most powerful Raspberry Pi single-board computer yet: two to three times faster than a Raspberry Pi 4, thanks to a more powerful Broadcom BCM2712 SoC based on a 2.4GHz quad-core 64-bit Arm Cortex-A76 CPU.

That's joined by an upgraded VideoCore VII GPU, a brand new RP1 southbridge chip that handles the bulk of the I/O, and a Renesas DA9091 power management IC. The memory bandwidth has also been boosted with a 32-bit LPDDR4X SDRAM subsystem.

While the board retains the same dimensions as other credit-card-sized Raspberry Pi models, it crams in quite a few extra features. There's a PCI Express 2.0 connector for high-speed peripherals such as SSDs – via an M.2 HAT coming soon. Two MIPI connectors can each be used for connecting a camera or display. Then there are JST connectors for an RTC battery, UART Arm debug, and fan – in the official case or Active Cooler (with heatsink), to help keep the Raspberry Pi 5 cool.

All in all, it's a major upgrade on the Raspberry Pi 4, and the extra processing power should prove useful for heavy workloads such as in machine learning projects.

hsmag.cc/RaspberryPi5



Adafruit Feather RP2040 Scorpio

Based around Raspberry Pi Pico's RP2040 microcontroller chip, this Feather board is designed to address some of the problems associated with driving strips of addressable WS2812B LEDs, aka NeoPixels.

For starters, it has eight LED channels, so you can split your lights into sections and drive each one independently, which solves the issue of lag in long LED chains. By default, it uses a level shifter to output 5V from its LED pins, to ensure NeoPixels are adequately powered. Each LED output also has a ground connection, which simplifies the wiring. On the downside, there's no on-board WiFi or Bluetooth.

Software-wise, the NeoPXL8 library supports eight outputs in both Arduino and CircuitPython code. The Arduino-only HDR subclass uses dithering for displaying 16-bit colours.

hsmag.cc/Scorpio

“It has eight LED channels, so you can split your lights into sections and drive each one independently”

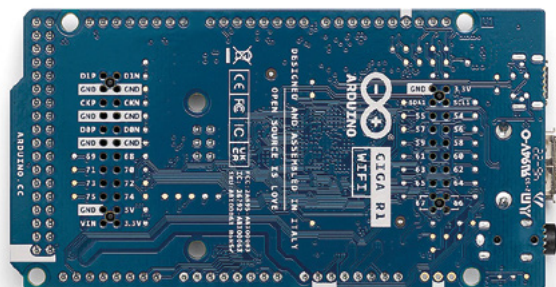
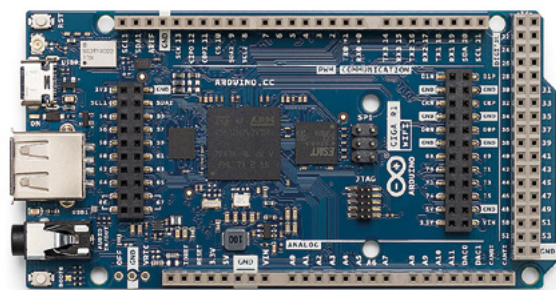
Arduino Giga R1 WiFi

Based around an STM32H747X1 dual-core microcontroller chip, this is the first Mega form factor board to offer WiFi and Bluetooth connectivity. It even expands the I/O options of a standard Arduino Mega with a grand total of 76 pins. Whether you'd ever need all of them is debatable, but you'll never run short! They include 14 analogue inputs, 13 PWM-capable pins, and a couple of DAC outputs.

Processing power also seems OTT for a microcontroller, although it may come in handy for machine learning or audio processing. As mentioned, there are two cores: an Arm Cortex-M7 running at 480MHz, and an Arm Cortex-M4 running at 240MHz. You can upload code sketches to either and get them to call functions from each other.

The Giga is a uniquely powerful microcontroller board that can crunch data at an impressive rate and control a phenomenal amount of I/O. The only downside is a hefty price tag.

hsmag.cc/Giga



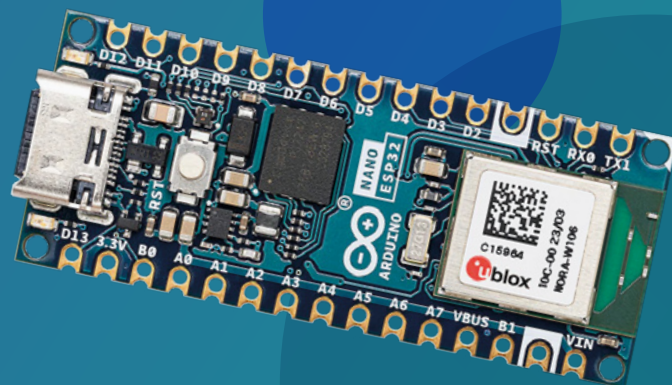
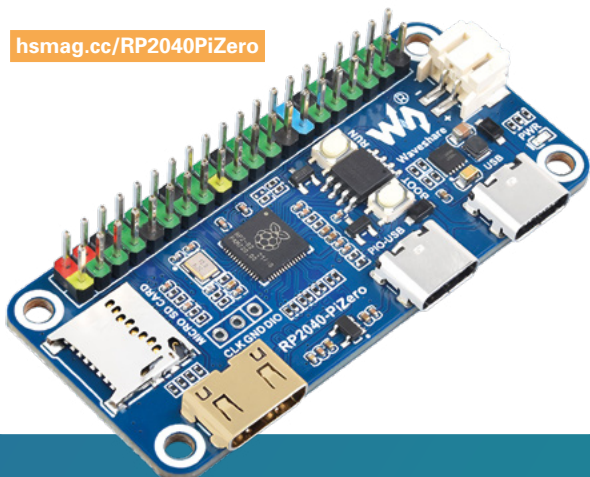
Waveshare RP2040-PiZero

Waveshare has taken the RP2040 microcontroller used in the Pico and put it in a board with a Raspberry Pi Zero form factor. The flash storage has also been expanded from 2MB on the Pico to 16MB, expandable via a microSD card slot. There's no wireless connectivity, though.

A mini-HDMI DVI port enables the board to output video to a standard HDMI monitor. Another notable feature is the configurable PIO-USB port that makes use of the RP2040's eight PIO state machines. Note that these two features can't be used concurrently, and both require the use of C/C++ code.

You might well ask the question: why not just use a Raspberry Pi Zero? The Waveshare board isn't as powerful as a Raspberry Pi Zero W, never mind a Raspberry Pi Zero 2 W. On the upside, it's cheap (\$10) and has the bonus of on-board lithium battery charging (via a two-pin header) along with low-power sleep and dormant modes.

hsmag.cc/RP2040PiZero



Arduino Nano ESP32-S3

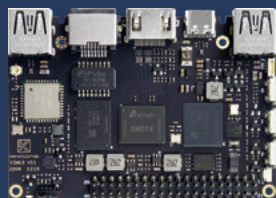
The smaller Nano form factor is ideal for embedded IoT projects. This new official Nano board does away with the need for a secondary chip for wireless connectivity, as the 240MHz dual-core ESP32-S3 handles that as well as running your code sketches. While this does help to lower the cost, it's still pricier than many unofficial ESP32 boards.

The pinout is similar to other Nano microcontrollers, with 14 digital I/Os, five of which can use PWM, and eight analogue input channels. It supports I2C, SPI, and UART communication protocols.

When it comes to programming the board, you can still do it in C/C++ with the Arduino IDE, but it also comes with an official port of MicroPython. Your existing ESP sketches should also run on it without requiring much, if any, modification (depending on pin mapping).

For IoT projects, it works with the Arduino Cloud. This sophisticated online platform allows you to share data between devices, set up online dashboards, use machine learning tools, and more.

hsmag.cc/NanoESP32



Khadax VIM1S

An OOWOW embedded service makes this SBC easy to set up, selecting from OSes including Ubuntu and Android to install on its 16GB eMMC. Powered by an Amlogic SoC with 2.0GHz quad-core Cortex-A35 CPU, its other features include a 40-pin GPIO, 4K60 video, WiFi, Bluetooth, and a dual-channel IR receiver.

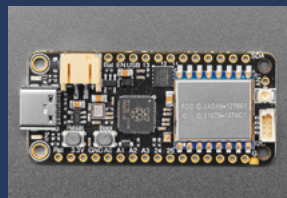
hsmag.cc/VIM1S



ZimaBlade

Want to run a personal cloud server? This Intel Celeron-based SBC is a good option. Along with 32GB of eMMC on-board storage (with Linux pre-installed), it has two SATA ports and a PCIe connector to add drives. By default, it has just Gigabit Ethernet connectivity – upgradable to 2.5 or 10 Gigabit, plus WiFi, with optional add-ons.

hsmag.cc/ZimaBlade



Adafruit Feather RP2040 RF95 LoRa Radio

Based on Pico's RP2040 chip, this Feather crams in a 900MHz radio module for use in LoRa (long range, up to 2km here) radio networks which use less power than WiFi. Along with plenty of GPIO pins, the board has a uFL connector to attach a larger antenna.

hsmag.cc/RP2040LoRa



PicoVision

As well as incorporating a Pico W and extra RAM, this pocket-sized digital video stick adds a second RP2040 chip to act as a GPU to handle big-screen visuals. MicroPython libraries and code examples help you to get started. Possible uses include digital signage and retro gaming – you can even play DOOM on it.

hsmag.cc/PicoVision

BREAKOUTS, ROBOTICS & ACCESSORIES

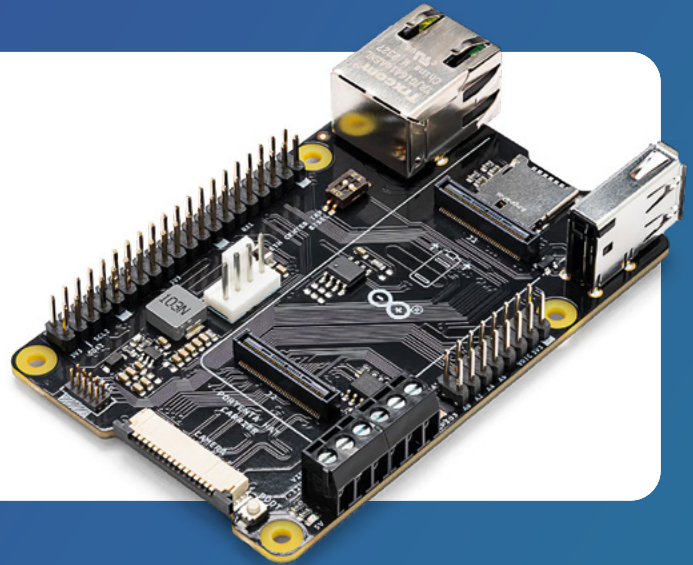
Add these to your SBC or microcontroller

Portenta HAT Carrier

This clever adapter board enables you to use your Arduino Portenta (X8, H7, or C33) with Raspberry Pi HATs. It's an intriguing mash-up to combine the popular microcontroller and single-board computer ecosystems.

As well as a 40-pin GPIO header to mount HATs, the Carrier adds single USB-A, Gigabit Ethernet, and MIPI Camera Serial Interface ports. Other features include a microSD card slot, CAN bus transceiver, fan connector, and 16-pin analogue header.

hsmag.cc/PortentaHATCarrier



Pimoroni Inventor HAT Mini

This add-on board for Raspberry Pi computers crams a multitude of features into a small footprint. While it's a versatile HAT, robotics is an obvious use case due to the four three-pin servo headers and two six-pin motor ports (for use with Pimoroni's encoder-equipped motors). Alternatively, you can connect standard motors to a header on the rear. With its size, it's ideal for a small robot.

Four extra GPIO channels, each with a three-pin header, are ADC capable, so can be used as analogue inputs. There's also a Qwiic/STEMMA QT port, audio output, external power input, UART header, and lots of LEDs. An on-board Nuvoton microcontroller handles all the I/O, so the HAT only uses a few GPIO pins – and it has a pass-through header too, so you could stack another board on top.

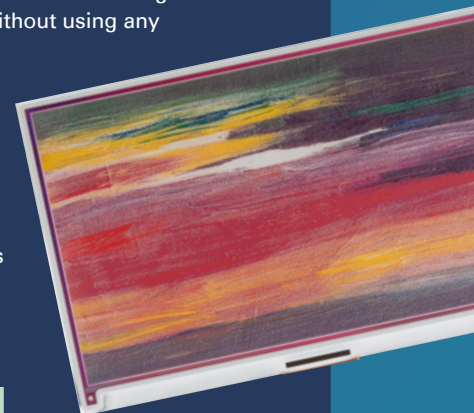
hsmag.cc/InventorHATMini

Pimoroni Inky Impression 7.3"

The Inky Impression range of seven-colour e-paper Raspberry Pi HATs comprises various sizes, and this is the biggest. Alternatively, you could opt for one of the standalone Pico W-based range of Inky Frames, which offer similar functionality.

Like all colour e-ink screens, it does take a while to refresh (about 40 seconds), but the advantage is that an image can stay displayed without using any power. The super-size dimensions make it perfect for a digital photo frame or for showing artwork. The 800x480 pixel resolution enables dithered colour images to be rendered in detail. Bonus features include four user buttons, plus I2C and SPI breakout pins.

hsmag.cc/InkyImpression7

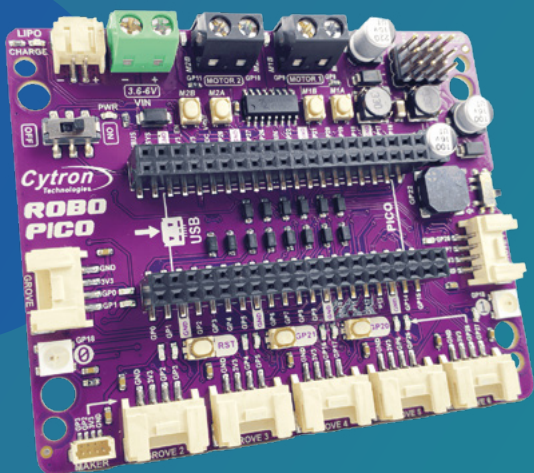


Cytron Robo Pico

Plug your Raspberry Pi Pico or Pico W into this purple PCB and you're all set for robotics projects. The \$15 Robo Pico is particularly suitable for educators and younger makers, as it's equipped with seven Grove ports for easy connection of servos and other components. Motors are connected to two sets of screw terminals, but test buttons enable you to spin them without even needing any code – very handy. In addition, there's a single Qwiic/STEMMA QT port.

You can power everything via Pico's micro USB port, voltage-in terminals, or a LiPo battery – there's an on-board power management system for safe charging. Two breakout headers enable direct connection to Pico's unused GPIO pins. Finally, there are 13 status LEDs and a couple of RGB NeoPixels. If you prefer micro:bit as your microcontroller platform, Cytron makes a similar robotics board for that: the Reka:Bit (hsmag.cc/RekaBit).

hsmag.cc/RoboPico



Pironman

There's a plethora of protective cases available for the Raspberry Pi 4, but this one is a bit special. Once assembled (a little fiddly), surrounding the Raspberry Pi 4 with aluminium and acrylic plates, it looks like a mini PC tower. It's not just for show, though: it includes an ice cooler tower with a 5mm copper pipe and a fan to prevent your Raspberry Pi 4's SoC needing to engage thermal throttling, so you should get better performance under heavy loads. Add in a 0.96-inch OLED display, RGB light strip, GPIO extender, power button, IR receiver, and an on-board USB to M.2 SATA SSD port and this is one super-cool case.

hsmag.cc/Pironman



ElecFreaks WuKong 2040

A multifunctional breakout board for Raspberry Pi Pico, the WuKong 2040 packs a whole load of features for just \$10. Ideal for robotics projects, its row of three-pin connectors can be used to control up to twelve servos, or used as general GPIO pins. There are also two motor interfaces for up to four motors.

For portable power, in the middle of the board there's a holder for an 18650 battery (not supplied) that can be recharged via the micro USB port. Other features include two RGB LEDs, a couple of user buttons, and a buzzer. Underneath, a studded frame can be used to add the WuKong into your LEGO builds.

hsmag.cc/WuKong



3D PRINTING, LASER & CNC

Get creative with a 3D printer, laser cutter, or CNC router

Anycubic Photon Mono M5s

This mid-size resin 3D printer with 12K pixel resolution enables you to produce prints with stunning detail and smooth edges. It's also the first consumer-grade resin printer with auto-levelling: a mechanical sensor on the cantilever arm detects the fit between the printing platform and the floating levelling module and automatically adjusts it.

Coming fully assembled, the Mono M5s has smart features, including detecting print failures and how much resin is in the vat. If you use Anycubic's AFC release film along with its resin, you can achieve speeds of 105mm per hour (though it will also work with other films and resins).

At \$539, the Mono M5s is an attractive proposition, offering speed and smart features in a consumer resin printer.



hsmag.cc/MonoM5s



Genmitsu 3018-PROVer V2

This upgraded version of the popular and affordable CNC router kit features a redesigned control board with extra add-on ports for extending its functionality. Other improvements include a redesigned Z axis carriage with a tool-setting touch probe. In addition, assembly is made easier by more accessory parts coming preassembled. There's an optional laser module kit, too.

hsmag.cc/PROVerV2

Prusa MK4

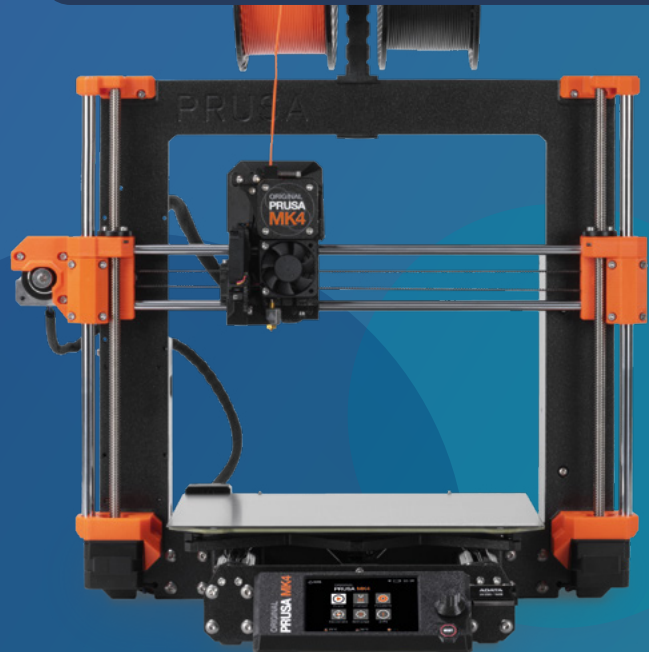
The long-awaited new 3D printer from Prusa delivers on all fronts. While it looks similar to the MK3 model, almost every aspect of the machine has been reworked. One of the most notable is the load-cell based bed levelling system that eliminates the need for first-layer calibration, even after nozzle changes or print surface changes.

A new extruder is lighter and the mass is now close to the X axis, so it can be controlled precisely at high accelerations.

Its powerful processor enables input shaping to be used to prevent the usual wavy edges caused by momentum after changes in direction of the bed, allowing faster printing.

While the MK4 is fairly expensive (\$1099 fully assembled; \$799 for a kit), there are upgrade kits available to retrofit earlier models with the new functionality.

hsmag.cc/PrusaMK4



3DMakerPro Lynx 3D Scanner

Starting at \$459, this entry-level handheld device aims to bring high-res 3D scanning to a whole new audience, enabling you to scan objects and create 3D models that can then be 3D printed.

Good documentation, along with the easy-to-use JMStudio app, helps new users to get to grips with the device. With a precision of 0.1 mm, you can use it to scan large objects – up to 5 × 5 × 5 m. An optical stabiliser and advanced visual tracking eliminate the need for markers and reduce the chance of misalignment.

Note that you will need to pay extra for optional upgrade kits such as a tripod, turntable, and extra-long 4 m device cable. The 'luxury' package also includes a colour texture-mapping kit – used in conjunction with a DSLR camera, this can produce scans with accurate colour surfaces.

hsmag.cc/Lynx3DScanner



xTool D1 Pro

Also available in 10W, 20W, and 40W variants, the xTool D1 is a powerful diode laser cutter and engraver that's relatively affordable yet suitable for serious crafters and makers.

Unlike some fully enclosed laser engravers, such as the more expensive Glowforge machines, it comes as an open-frame kit. Nor is there a UI screen.

It does benefit from high-grade aluminium construction, however, along with all-important fire prevention safety features. Limit switches prevent the laser from hitting the sides of the frame, while a flame detector will turn it off in the event of a fire.

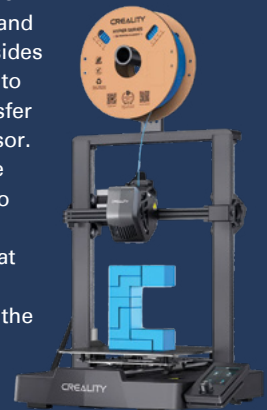
Free online software and tutorials enable you to quickly get laser cutting or engraving. The laser has a precision of 0.08 mm, while the print speed of up to 400 mm/s is faster than that on most budget lasers. It works with many different materials and can cut through wood up to 10 mm thick.

hsmag.cc/xToolD1Pro

Creality Ender-3 V3 SE

If you're looking for a reliable entry-level 3D printer offering decent speed and user-friendly controls, this is it. For an incredible \$199, you get a maximum print speed of 250 mm/s, plus CR Touch automatic bed levelling and pressure sensors in the bed to set the Z height. The machine even comes mostly preassembled, so you just need to screw in a few parts and plug in the cables – it only takes 10–15 minutes.

The V3 SE's 'Sprite' direct drive extruder works well with PLA, PETG, and flexible TPU, while filament loading is automatic. A rigid dual Z axis reduces Z wobbling, while linear rods guide the Y axis movement, improving stability and print quality. A couple of downsides are the lack of WiFi (you'll need to use old-school SD cards to transfer files) and a filament runout sensor. The PC-coated build plate, while offering good adhesion, can also make it tricky to remove prints. Even so, the V3 SE is a bargain at this price and a good way for newcomers to dip their toe into the world of 3D printing.



hsmag.cc/Ender3V3SE

SUBSCRIBE TODAY

GET SIX
ISSUES
FOR JUST:

£30 UK / €43 EU / \$43 USA & Canada



**FREE
Pico W**
for subscribers!

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

HOW

By Sven Kroesen

I

MADE

ARDUINO COCKTAIL MACHINE

Automated, not stirred!

Around a year ago, I bought an Arduino Uno after fiddling around with them at college for a few weeks. I already

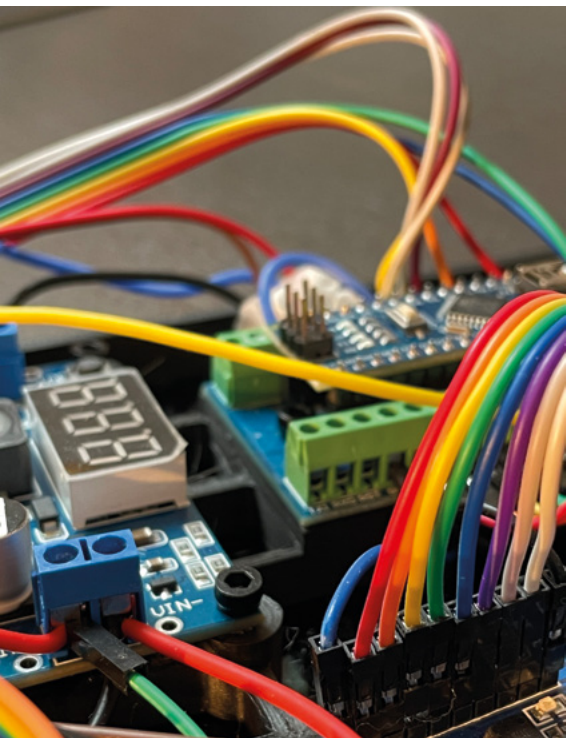
owned a 3D printer, so this seemed like a great addition to the hobby. After completing a couple of simple projects, it was time to try something more complex. On the weekends, I enjoy making cocktails for myself and my friends. A Long Island iced tea has always been our favourite, but with six ingredients, it's quite a hassle to make.

So I thought, why spend a few minutes making a drink when you can spend a few months automating it? So I did.

THE MACHINE

In essence, the cocktail maker is simply seven peristaltic pumps controlled by an Arduino Nano, on which all the recipes are stored. These peristaltic pumps are super-handy components that allow you to accurately dose liquids without them coming in direct contact with the pumps' mechanism. When you know the amount of liquid that is pumped through per unit time, it's very easy to use delays in your code in order to pour drinks accurately.

Previous projects of mine have always exclusively used 3D-printed parts and off-the-shelf components. Because I wanted to try something new this time, and since it would



turn out to be so big, I had the biggest parts laser cut. Doing so helped give the machine a much nicer look. In addition, LED rings hidden under the bottles and glass finish the machine's look.

My goal was for anyone that comes over to be able to use the machine without assistance. So, controlling the machine had to be very simple. For this reason, the only way to interact with it is by using the rotary encoder that's placed under the TFT display. When the cocktail maker is turned on, this screen immediately displays a list of available recipes. Using the knob, the user can navigate through the different drinks, change what bottles are stored, and access various machine settings. Telling the Arduino which bottles are stored as well as their locations helps it determine if all necessary ingredients are present to make a cocktail. If that is not the case, the cocktail is greyed out in the menu. It can still be prepared, but the missing ingredient will be omitted.

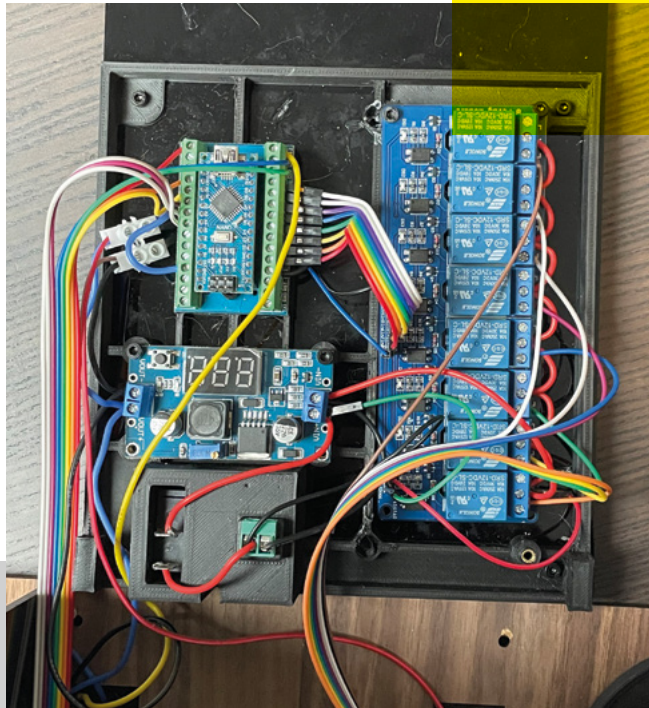
To make a drink, the user simply pushes the button with their drink of choice selected. A list of ingredients is then displayed on →



Above ↑
You can add whatever drinks you like and let the Arduino know where they are

Above Left ↖
The finished machine ready to serve up a delicious drink





Right →
The Arduino controls relays that switch the pumps on and off

cutting, all you need is a 2D projection of your design which you can turn into a DXF file with any modelling program.

All I had to do after that was upload them on the website of a local laser cutting place and wait two weeks. It turned out to be surprisingly cheap, too. Numbers are relative, but all together, it was cheaper than the sum of the electronic components. To keep this price down, it helps to design the parts in such a way that they fit snugly next to, or even within, one another. This way you can cut them all out of one big piece. The cocktail maker is cut out of two such panels: one made out of a reflective black plastic, and another made from wood. These parts don't only look nice, but they also provide rigidity to the structure. On their own, they cannot be held together, though. In order to do this, I tried another new technique.

Brass inserts. These little things allow you to put metal taps into your own 3D-printed objects, which is very handy. At first, I tried to simply use tiny nuts and bolts with brackets in the corners, but the further along the build got, the harder it became to reach in certain places, making it necessary to try a different technique. Using the inserts is very easy if you have a soldering iron handy. If you're going to copy this project, you will probably need one anyway. To use them, simply put circular holes in your designs that are just snug enough. Put the insert on the top of a heated soldering iron and slowly press it into place, making sure no plastic gets on the inside. After the plastic solidifies around it, it will be as strong as the plastic it's placed in, which should be plenty for this application if you use the right materials and density for your 3D prints. I like 30% in this case. Doing this for all components makes

**“ALL YOU NEED IS
A 2D PROJECTION
OF YOUR DESIGN”**

the right-hand side of the screen and the user is asked to confirm their choice. The next step is to input the strength of the cocktail and the amount of ice desired. At 50% strength, a regular drink will be prepared. At 100%, your drink will not contain any mixers. 0% is the same thing but for the alcoholic parts. The ice selection bar works similarly, where at 0%, your drink will not contain any ice. Any higher and the amount of ice will also increase. When both choices are confirmed, the preparation of your cocktail will start. The screen automatically updates to let you know how much liquid is still left to pour. When everything is done, the machine lights up and you can take your finished cocktail!

BUILD PROCESS

During this project, I used two new techniques to put everything together, which turned out to be extremely helpful. The first is the aforementioned laser-cutting of wooden and plastic panels. I had never done this before, but I heard it was possible to order them online, so I decided to give it a try. If you want to follow along with this project, I assume you already have some knowledge of basic CAD design. For laser

assembly surprisingly simple, as you only have to screw them in place. Even allowing you to make a frame to put all electronic components in instead of using a PCB.

THE MECHANISMS

Effectively, the machine consists of three parts: the pumps, the ice mechanism, and an electronics rack hidden in the back. We'll talk about the pumps first.

As explained earlier, they are called peristaltic pumps. Instead of turbine-like mechanisms used in many other types, these pumps sort of squeeze the liquid through a flexible PTFE tube. This ensures the fluids never come into contact with anything but those tubes. For this reason, it is perfect for transporting liquids intended for consumption and is, for example, commonly used in medical devices. Starting

out with this project, I ordered six of these pumps from China. Since it usually takes a while for the parts to arrive, I try to buy everything I will need in one go. Sometimes, unfortunately, this means a part doesn't work exactly as expected. In those cases, I still try to make the best of it. For this project, the pumps I ordered turned out to be a little slow when pumping. A large glass needed almost a minute to completely fill up using two pumps, which was too slow for my needs. As a solution, I decided to order a different type of pump that is much faster. Since most cocktails have one dominant ingredient, this faster pump can be used to speed up the whole process. Often this ingredient is Cola. The pumps are all mounted in a 3D-printed rack in the top of the machine. A gap is left open so the new seventh bottle can be stored inside the machine. →

LOW

Below Left ☒ Liquid is sucked up through these tubes by the pumps

Below ⚡ Laser-cut panels made assembly more straightforward



ACHINE

How I Made: Arduino cocktail machine

FEATURE

This seventh bottle is used by the faster pump and should be for the primary ingredient in most cocktails. It is a bit of a mess, but this inner container can also store some ice to keep the bottle cool and for putting in your drink. To achieve this, a so-called Archimedes screw is used. This mechanism lets you vertically lift up small objects or, in certain designs, even liquids. We are using it to lift up ice cubes through the gap that leads to the glass. The ice cubes do have to be small enough to prevent jamming the screw.

Working with cooling elements was a little out of reach for this project, so I decided instead to just let the ice slowly melt. All cracks in the interior are covered up with a food-safe resin to prevent leaking, except for a tiny hole under the Archimedes screw. From here, the water leaks through another PTFE tube to the back of the machine. Here, you can either let the water leak away or redirect it to a basin/sink with another tube.

All of this is controlled by a single Arduino Nano. The recipes are stored in a struct

data type, with each the name of the drink, references to all ingredients, their volumes, and whether it contains alcohol or not. This last characteristic is necessary to adapt the recipe to the user's alcoholic strength preference. When a drink is selected, the Arduino gets to work. Because we are using peristaltic pumps, we know exactly how much liquid flows every second when they are powered on. By simply dividing the required amount according to the recipe by this number, we know the number of seconds each pump should run for. These pumps, together with the motor for the screw, are controlled through eight relays.

To know which pumps to turn on, we first have to tell the Arduino which ingredients are present and where they are placed. Before we select a recipe, we can input this data by going to a special menu where you first select one of the seven platforms and then one of about 30 pre-installed ingredients. This data will be stored even when the machine is turned off, thanks to EEPROM.

Even when not all ingredients are present, the Arduino will find all corresponding pumps and turn them on for the number of seconds we calculated earlier. All seven tubes are individually led to the console in the front. I

Right →
Here you can see the six peristaltic pumps

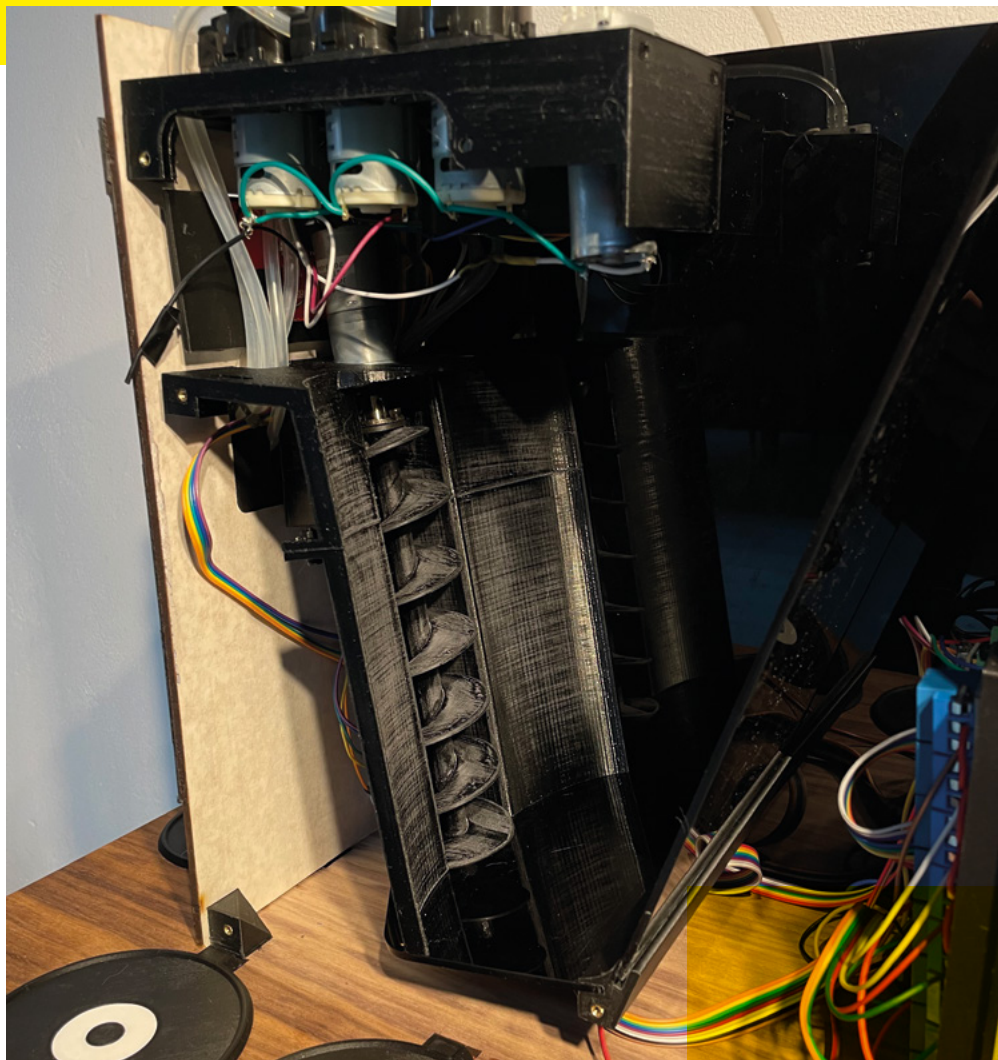


“LONG ISLANDS ARE, OF COURSE, VERY TASTY!”



attempted to connect them all together in one single bigger tube but this caused a lot of back-flow and leaking. There are purpose-built connectors to achieve this, but at this point, I was nearing the end of the project and I did not intend on spending any more money on it. It is also not recommended to 3D-print such components. 3D-printed objects are inherently not food-safe because of the tiny cracks that exist on their surfaces. These spaces allow bacteria to hide away and are very hard to properly clean. Or even impossible when it's hidden inside a machine. So, in this case, I preferred my ad hoc solution.

The remainder of the electronics are very simple. The pumps and screw motor all need 12 volts. With a maximum of 2 amps, this is supplied by a simple power supply. Since the microprocessor needs no more than 5 volts, this is stepped down first. This power is also used to power the NeoPixel rings in the platforms and the display on the front. I am still very much learning about electronics, so in my projects, I like to use screw terminals. These allow you to change and adapt as you go while still giving rigid connections.



CHEERS!

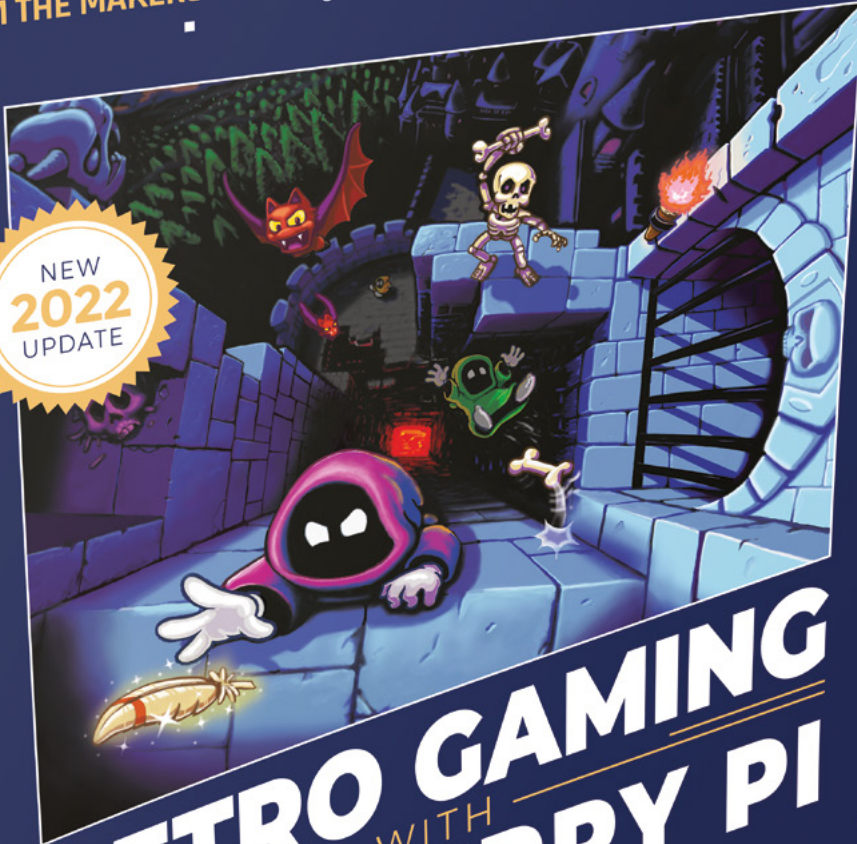
Altogether, making this machine has been a great learning process. Laser cutting seems like a viable tool if your design is large enough since you usually have to pay for the whole sheet. In our case, some leftover space was used to cut out a door sign with mine, my girlfriend's and, of course, the cat's name engraved into the surface. After weeks, the cocktail maker still works beautifully. Soft drinks come out a little flat and ice unfortunately does melt, but I had a ton of fun building it, and Long Islands are, of course, very tasty! □

Above ↑
This screw mechanism lifts the ice up and into the drink

Above Left ↖
A spiny knob is a great way to control a machine

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

NEW
2022
UPDATE

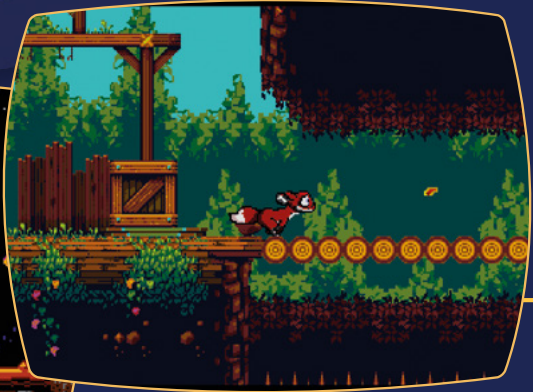


PLAY
& CODE
GAMES!

RETRO GAMING WITH RASPBERRY PI

2ND EDITION

164 PAGES OF
VIDEO GAME PROJECTS



RETRO GAMING

WITH

RASPBERRY PI

2ND EDITION


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


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



BUY ONLINE: magpi.cc/store



Above  The noise-cancelling headphones that sparked Jude's epic quest to change a battery

Right  Various sizes of LiPo battery



HackSpace magazine meets...

Jude Pullen – Fight to Repair

How hard can it be to change a battery?

T echnologist, prototyping expert, engineer, life hacker, product designer – Jude Pullen's CV is pretty jolly impressive. He's worked on medical devices, he's worked at LEGO, and Dyson, and he's sent *Superman* into space for RS Components. So you'd think that he'd have no trouble changing a battery, right? Wrong.

Thanks to a combination of business models and poor design decisions, repairability is in trouble. Where we could once prolong the lifespan of our devices by making simple fixes ourselves, manufacturers are now making things more difficult – at huge cost to the environment. Jude's been exploring this issue, and the efforts he's had to go to just to change a battery have been extraordinary.

As soon as you have an hour or so spare, we'd strongly recommend you go to hsmag.cc/fight-to-repair and follow Jude's fight to repair his own stuff. →

HACKSPACE So, I gather you have had a bit of bother with your headphones, then?

JUDE PULLEN It's true. One night my headphones stopped working. And I was thinking oh, yeah, my headphone battery's dead. I've had six years of use from it, and I didn't have a warranty or anything. So I thought I'd get stuck in and take it apart.

[Batteries are dangerous – as he says, Jude's a trained engineer, so he knows what he's doing. If you don't know what you're doing, don't do it.]

And so I went on to YouTube, as most engineers do, typed in the model and then the word 'repair', and it all looked like it was going to be plain sailing.

The manufacturer didn't have any replacement batteries in stock, but I wasn't shocked by that being an engineer – I know that LiPos are basically a pain in the behind for companies to stock and sell in OEM. And so I took it upon myself, as someone with enough qualification and safety awareness, to sort of DIY it. And it didn't work.

So I got more batteries and tried to figure it out, but it was still being problematic. I've now spent 50 quid on batteries. And I thought, this would be a good starting point for a look at repairability, if I just took this to the extreme conclusion of trying to fix it at all costs. Being 50 quid in the hole already, the sensible thing would be to cut my losses and just buy a new pair of headphones. But it led me to loads of questions: was this a sort of planned obsolescence? I don't like the term conspiracy, but is it just a business tactic, to not make this easy to repair?

Or is it just actually that the LiPo batteries are dodgy, and it's not the hardware's fault at all? Or, is it something I hadn't thought of?

HS So why is it so hard to put a new battery in a bit of hardware? Phones for example – you used to be able to take the old battery out and put a new one in.

JP That is the thing that baked my noodle. Usually, as a technologist, you're talking about advanced technology that hasn't been scaled, or hasn't been invented yet. But we've been making mobile phones with batteries that you can pop out since the 1990s.

Part of the justification for that is the perceived desirability of making phones waterproof. Whereas most of the time, it just needs to stop sweat. Who uses their phone in torrential rain? You don't; you'd move inside a shop entrance or something to take a call. So, truly, unless you're going up a mountain, it doesn't need to be waterproof. Apple claims that [its iPhone model] is rated to IP67 standard waterproof, or whatever it is.

But this is a notion that doesn't really get tested in most people's day-to-day life. And yet, as a consequence of making it waterproof, it's absolutely not repairable

I don't like the term conspiracy, but is it just a business tactic, to not make this easy to repair?

in any sort of user sense of the word. And so, I feel this sort of pushing the waterproof thing is a kind of... I don't want to say Apple is being wilfully duplicitous, but I feel it's leading us all down completely the wrong road.

It feels like an ill-conceived marketing strategy, that we should fixate on something which is incredibly rare, and yet prohibits a lot of other things, which would give us a lot more happiness, a sense of ownership and longevity to the product. And I think the only conclusion you come to as a consequence of that is that Apple still wants you to buy a phone every two years when your contract changes.

If you look around the best of R&D, I genuinely believe you could task the

geniuses in Apple's ranks to make a moderately water-resistant – not waterproof – phone that is still repairable, because I think the iPhone, like the Fairphone 5, which is something like IP55, [could just be] splash-proof. For me, that's absolutely reasonable. That's completely delivering the job. You should sign off on waterproof-ness at a sensible level, because the other benefit, repairability, is so amazingly important right now.

Once upon a time, the idea that a pocket watch could be waterproof would seem preposterous, and yet Timex makes a 100-metre waterproof watch with a battery that's replaceable, and you can twist it off with a coin.

HS You don't want to use the phrase 'planned obsolescence conspiracy', but at least on the face of it, there is some evidence for it existing.

JP Just to sort of ease off Apple for a minute, I was considering getting a refillable inkjet printer. But the reviews are basically saying this is a total scam. It excessively cleans its nozzles, and basically squirts all this ink into the back of the thing into a sort of sanitary towel that absorbs all the ink. So you think you're doing the right thing for the environment, but it's actually wastefully burning through the ink, because for some reason, they haven't developed a nozzle that doesn't clog.

How did you design a nozzle that didn't clog before, and this one does now? I've worked in R&D enough to know that if that printer had gone past James Dyson, he would have sent it back to the drawing board.

The shame is that those printers are going to poison the well. Consumers will go, 'Ah, see, I told you it was greenwashing – you should just go back to the standard, that's the best.' And so, as a result, we have a counter-reaction that sets back the notion of refillable cartridges.

Whereas evidently, what you need is a little syringe that goes into the cartridge, and for it to somehow repressurise and →




! LIPO BATTERIES: DON'T DO IT!

HS: You used an ammunition box to store batteries while you were working on this project. Why is that necessary?

"I was thinking of all the stuff I'd read about batteries exploding: if you've got something that can contain an accidental explosion from a bullet, then it's probably pretty strong. And, of course, when I ordered one for what I think is the bargain price of £15, it's incredibly tough, it's made out of something like 2mm thick steel. And the latch on the top is so powerful that actually it's quite white-knuckle to even clip it. It's not an official ISO standard, but I do think it's a really good thing if you're doing a bunch of experiments on lithium batteries. Like, I think it really does keep you safe."

By experiments, what Jude means is that LiPo batteries can explode into a fiery chemical furnace if they get too hot; if they're punctured (easy to do by accident); if their connectors are short-circuited (EXTREMELY easy to do). Another consideration is that they can swell a little in use, and if there isn't enough room in the battery housing, this can cause damage (which can cause fires). Oh, and they release gases even when they're not on fire. Unless you're an electrical engineer, don't do it!

Above  You shouldn't have to go to these ridiculous lengths just to change a battery



Above
It must be a bittersweet feeling to see a product that you helped develop thrown away for recycling

reseat. And then you don't need these big stupid cables – I can tell you already using a silicone cable is just ridiculous because the ink is going to dehydrate in there. I could have told you that without even starting the project. Just being an experienced engineer, I know this. And I could probably find you five engineers, and at least two of them would know that. It's just really common knowledge, and it's a really dumb design to anyone who has any experience. You would never use 30 cm-long silicone pipes and expect the liquid not to dry out.

It annoys the hell out of me that this has been designed for sort of virtue signalling and smoke and mirrors, and hasn't actually had the best of their R&D on it. I think that's almost the acid test here: is this product truly the best of the best, or are you just doing this as a little bit of marketing fluff?

I've met some of the people who've worked in places like [redacted]; they would not do something as rookie as that. It's not the design that's the problem; it's the business model.

Printers and ink is the business model, because you can't actually sell a decent printer for 50 quid – that is non-viable, economically speaking. You're making a gamble that you will sell subsequent ink at probably 1000% mark-up. And then you recoup that over the lifetime of the product. And I think that phones and other electronics goods, they are banking on you upgrading or replacing in two years' time. And I think that that's why, currently, battery replacement problems can't be solved; it's that the business model needs to change to allow it to be solved.

Planned obsolescence isn't necessarily some grand evil; it's just the natural consequence of business models that people are happy to tolerate sub-prime outcomes.

HS But there is a value in fixing things that's lost when you just throw something away. I feel like I'm winning at life when I can extend the use I get out of a device.

JP I think we, as consumers, have to recognise our relationship with what we call value, and where we derive pleasure from things. Fairphone is terrific, but only for the sort of person who enjoys that level of tinkering.

But let's use an analogy of cars: maybe 1% of people change the oil and repair the carburettor and balance their brakes and all that sort of stuff. But the point is, we have an industry around that sort of thing. And so maybe that's the future: Fairphone is for the people who are like the grease monkeys who want to spend

It annoys the hell out of me that this has been designed for sort of virtue signalling and smoke and mirrors

their Sunday tinkering. But actually, the rest of us can just take a broken phone down to the store. And it will be half an hour, and it'll cost you 100 quid.

HS In the course of your investigations, you came across single-use vapes. What's going on there?

JP First of all, the point isn't being anti-vape. Although I would encourage people to watch the Netflix documentary *Big Vape* if they did want to hear a little bit more about the industry and how it works. But anyway, the disposable vapes which most people have seen littering the streets more and more recently – those basically have a lithium polymer battery in, and the thing about lithium polymer is that the chemistry is inherently rechargeable. So there's no such thing as a non-rechargeable lithium polymer battery. You can buy a non-rechargeable lithium coin cell, but that isn't what's in a vape. So a vape has a battery. And literally, if you take one apart, type the serial number into Google, you will find a

supplier that sells it. And that supplier will give you the recharge cycles in the order of magnitude of 500, which means you can recharge it 500 times.

So I commissioned a series of articles with a guy called Jo Hinchliffe [the very same Jo Hinchliffe who's been guiding HackSpace readers through the intricacies of KiCad – see page 60 for his latest instalment] to build some stuff to prove how rechargeable and how powerful the batteries still are. He's built drones, MP3 players, little torches, all sorts of stuff to just illustrate the point that these batteries are still good.

These sorts of things should not be going into household trash. And they're a huge problem for waste management companies to separate these things from the waste streams, because they don't want the batteries going through the shredders. They need to separate them and send them to a different plant.

HS Aren't we running out of lithium? I mean, unless we smash some other elements together in the Large Hadron Collider to make some more?

JP Well, the lithium is very abundant. It's all the other stuff like cobalt which is problematic, I think. So, yeah, I think there is talk of things like solid-state batteries. But again, these have all sorts of pros and cons. Which means it's sort of, just because NASA has been working at it, doesn't mean it's at scale. For all sorts of things. So first thing, that's what I mean about the trickle-down of technology. And of course, the sort of battery that's good for a set of headphones is maybe different than what's used in a car. Ultimately, we might end up actually changing the chemistry for very good reasons. So, yeah.

HS That sounds grim. Is anyone doing repairability right?

JP There's an air quality monitor from IKEA that stands out. I don't think IKEA →



Above ◆

You don't need to use LiPo batteries in your builds: these hair clippers work perfectly well with AAs.

could legally and responsibly encourage people to hack their hardware. But what's interesting is that the PCB is labelled such that if you are competent, and are qualified, you would smile and say, 'Oh, look, that's very kind of you. You've put all these additional GPIO-type pins and labelled them, which now means I don't need to find a schematic'. So it becomes a self-selecting criterion, that if you don't know what any of those little symbols or things mean, you probably weren't going to hack it anyway. But if you are actually an electronics engineer, and you understand what all those things are, you're probably qualified and you're probably not going to burn your house down. Aside from the fact it's a 5 V USB-C appliance, so chances of you burning your house down are quite slim, you know.

What annoys you sometimes as a consumer is when you've bought something that you really, really love. And let's say it's a drill and the brushes are worn out. I don't want your latest model,

because it just feels different – people do fall in love with their tools. Coming back to the business model, you should still charge people a pretty penny for those brushes. Let's say the brushes actually cost 10p, probably less than that. But I haven't got a problem with paying £10 for a new set of brushes for my DEWALT drill, because for me, it's a £150 drill, and I love it. It's a brilliant drill, and it endears me to the DEWALT brand. And I think that's why sales managers need to see it holistically – make me love your brand, rather than try to trap me with your brand's eccentricities.

HS That sounds like a company culture thing rather than a simple case of 'do this, not this'.

JP Much of it is about business. Can the company still operate with its financial targets while doing good for the planet? If the company says, 'You're just asking us to incur a load of cost and a load of risk of litigation that users will tinker and get it wrong and blow themselves up', my counterpoint is still 'Yeah, but cars'. How did we have the Haynes manuals for repair, which so many people really loved when I was growing up?

Toyota, anecdotally, makes extremely repairable cars. You know, if you go for an MOT, it doesn't cost you an arm and a leg. Whereas other models will be cheaper to buy, but the manufacturer will get it all back. I'm excited for different proposals around repairable business models that also give people the chance, like a Haynes manual, to say, if you're really, really sure, you can do this. But if not, there is a good industry around reparability. This isn't about asking companies not to sell a product, but I don't think you should buy an entirely new iPhone when 90% of that thing was fine. That has massive issues with it, ethically.

HS What can people do at home to make their own builds more repairable?

JP Use batteries with JST plugs. It's a standardised plug, so even if you had to

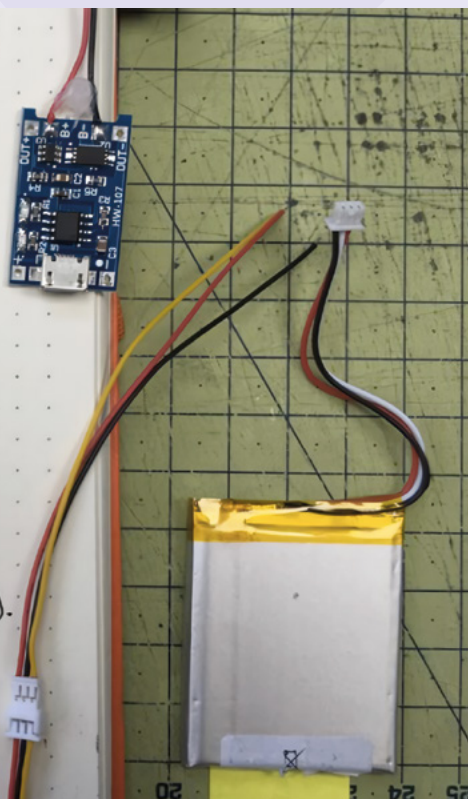
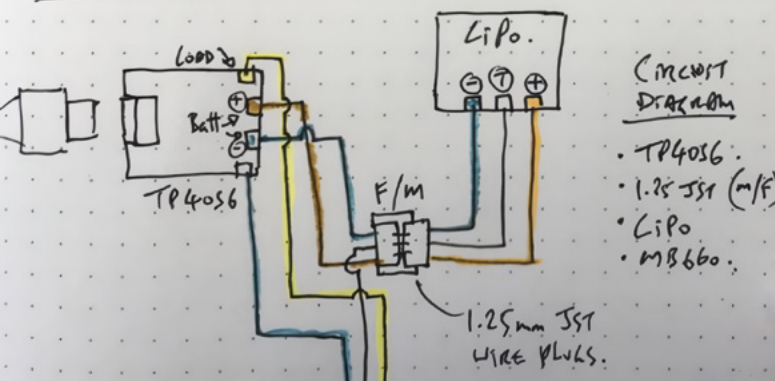
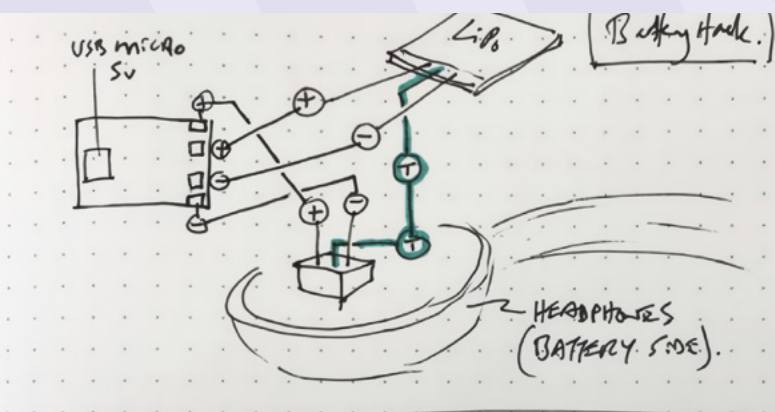
splice on new wires for that battery, the point is you're not soldiering and risking getting it wrong. [Without JST plugs] it's too easy to de-solder from two pads, then forget which way round the terminals are because they aren't labelled '+' and '-', and then you put it the wrong way around, and you could blow it up.

Another one is use screws rather than clips or glue. At Dyson, so much of our testing required you to strip down or tear down the entire machine multiple times, so having products designed with screws rather than glue is actually almost a product of Dyson's innovation process.

Part of the reason I love making things repairable is that people can also change out parts easily. If you just stick it all together with glue, like with the radio globe project I designed and shared, I wouldn't have had people remix it and put speakers in and do all sorts of different bits. And it's because it's got this brass insert set system, that it means that people can locally adapt something to their needs on the device without having to basically redesign the entire thing.

Another good real-world example of this is IKEA hackers. It's slightly torturous in that IKEA tried to sue the person who created it. And then the public backlash was so strong, they ended up buying them instead. But my point is, allowing people to be creative, and taking a little bit of awareness that people will get it wrong, and trying to keep them as safe as possible is a really good thing. That could be as simple as, if you design a product, make it so that the battery housing really does have a massive positive and negative on the silkscreen. You should be silently making it safer and easier to repair and keep it going, even if deep down, you know that a lot of people will still just chuck it in landfill.

We will only win this by a generation growing up that builds confidence on saying, 'Well, I've repaired the brushes on my vacuum cleaner. I repaired the gearbox on my drill. I changed the battery out on my little Bluetooth speaker.' The more brands that do this... it has almost a compound interest of confidence. □



Above Anything we can do to minimise products' environmental impact has got to be worth it

Left Jude eventually managed to fix the headphones, adding a charging board and a magnetic power cord – proving that the ability to repair your own stuff can result in better products

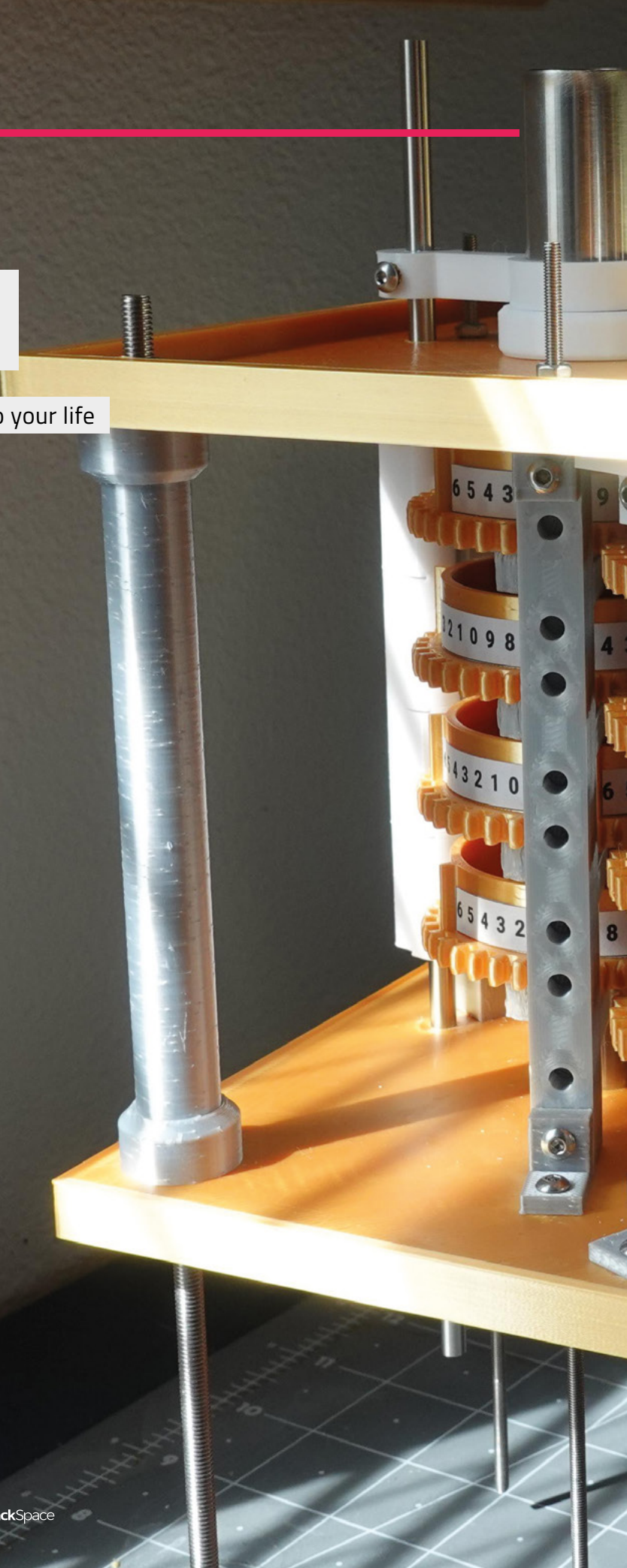
Objet 3d'art

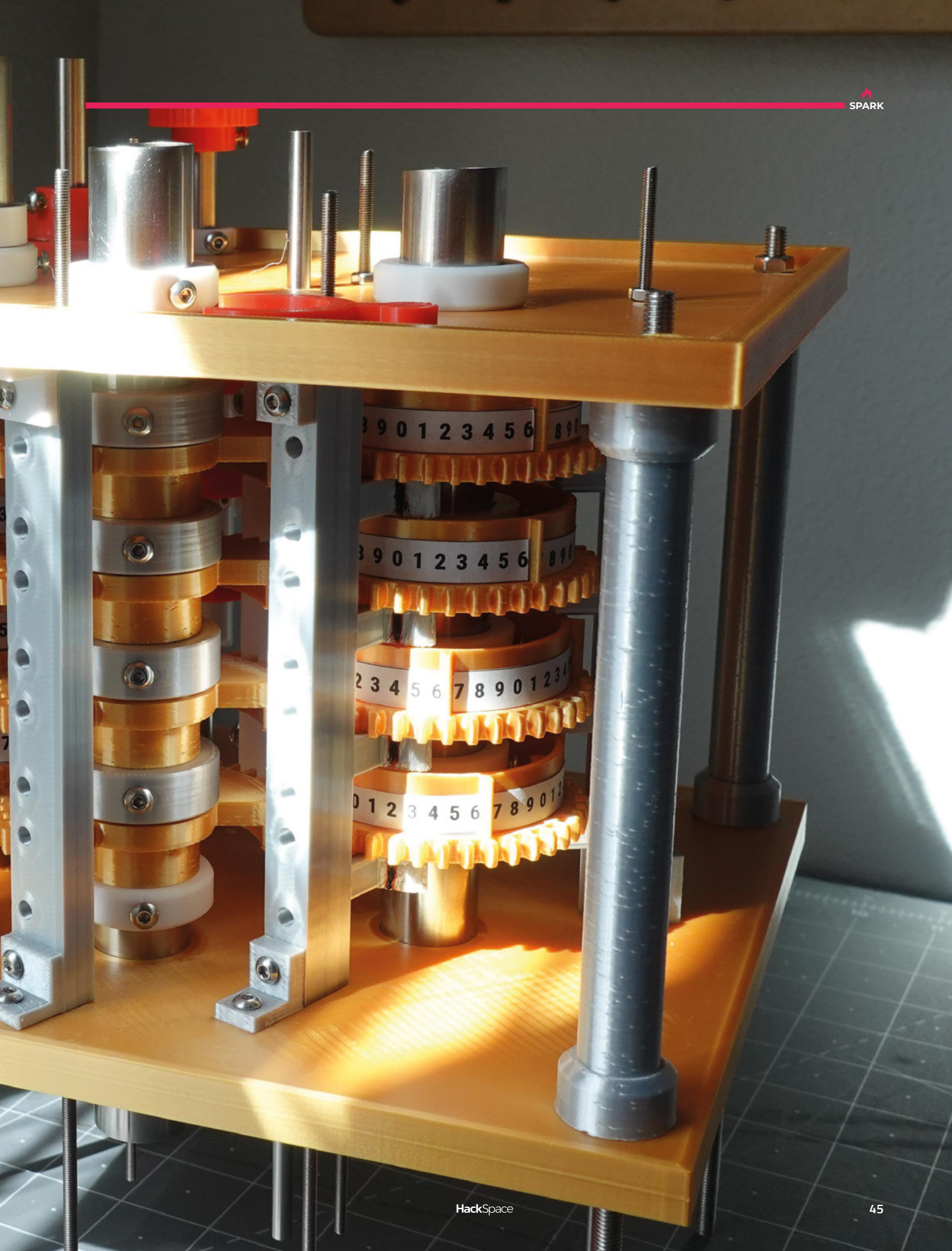
3D-printed artwork to bring more beauty into your life

In the old days (not so old days actually), every affordable FDM printer had a sheet of sandpaper next to it. This essential tool would enable the user to take the rough edges off printed objects that had to fit together. Fail to sand the surfaces you're joining, and you risked cracking, jamming, or otherwise ruining the print you'd just waited hours for. We could not then have imagined how easy it would be, just a few years later, to print mechanically accurate, smooth-rolling gears.

It's even more amazing that those gears are accurate enough to build a computer. Charles Babbage's original Difference Engine was never completed, as the metalworking techniques of the time were too complicated to allow him to realise his vision; brilliantly, the UK government of the time funded his work for a while, but then withdrew funding, presumably believing that there was no future in this computing nonsense. Designs for the Difference Engine No.2 followed, demanding fewer moving parts, but this was never made. Thankfully Hackaday contributor MechRedPanda can print gears more easily than mid-Victorian machinists could make them out of metal, and so the Difference Engine No.2 is reborn, fully functional, in 3D-printed plastic. □

hsmag.cc/DifferenceEngine2





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

RING RING

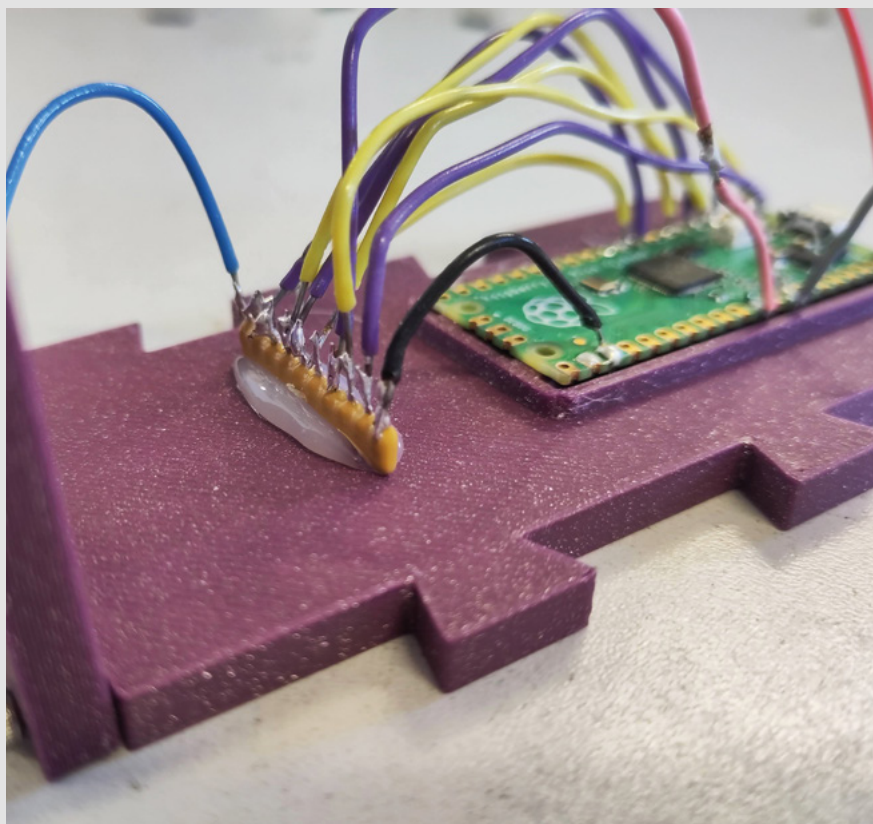
I love the idea of a cheap, hackable modular synth, so thanks for building this. I have just one request – can you make a ring modulator? I want to create my own take on the *Doctor Who* effects. I'll be able to terrify the local youths into thinking that the Cybermen are taking over. Do kids still know who Cybermen are?

Paul

Sunderland

Ben Says: Ring modulators are an absolute staple of modular synths, and you're not the only person to ask for one, so of course we'll have to create one. We've got a few things we need to get through first, not least getting our setup in tune.

I've got no idea if kids these days know who Cybermen are. I just found out that my daughter had a history lesson where they talked about how, in the olden days, people went into shops rather than ordering things online, and now I just feel old and out of touch.





CUSTOM CHIPS

I read your interview with Matt Venn and I can't believe that it's possible for makers to design their own chips. Mind blown. It had never even occurred to me that this was a possibility. I have no idea what I want to design a chip to do, but I want my own chip!

Sara

Chippenham

Ben Says: I know, it's wild isn't it? I've got a design on Tiny Tapeout #2 that should be shipping soon. The chips are made, they've been tested and, as I write this, we're just waiting on the PCBs and assembly. I'm so excited. I can also barely remember what was on the chip. I designed it a year ago. It produces LED effects of some sort. What's also exciting is that there are loads of designs by other makers on the same chip. I'll be able to play with the silicon designed by people who are far cleverer than me.

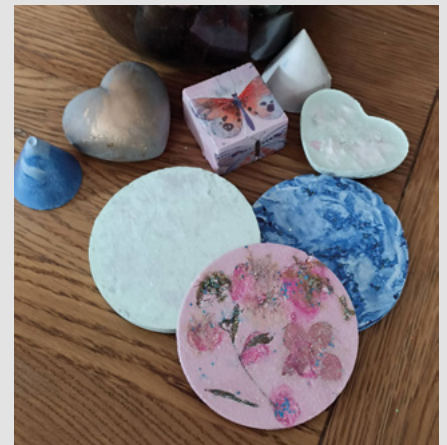
TREE HUGGING

Thanks for your article on eco resin. It sometimes seems like the maker world is obsessed with poisoning themselves and the world with toxic chemicals. Anything we can do to fight back against this is a good thing.

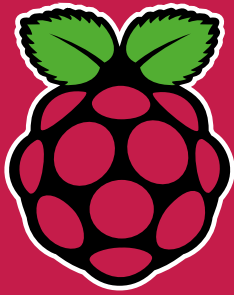
Owen

Rhyl

Ben Says: There are certainly some processes that can potentially be harmful that certain people on the internet perform with a very cavalier attitude to health and safety (cough, resin printing, cough). Here at HackSpace magazine, we're not against working with potentially harmful substances (provided they're done in a safe way for both the user and the environment), but it's nice to not have to worry quite so much about the long-term health effects of your hobbies if you don't have to.



THE *Official*

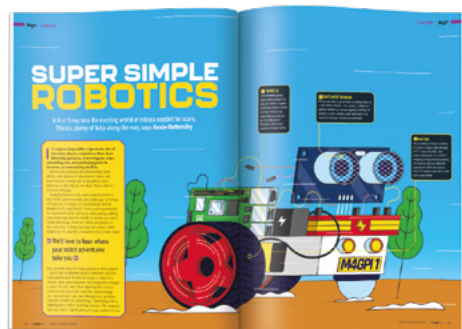
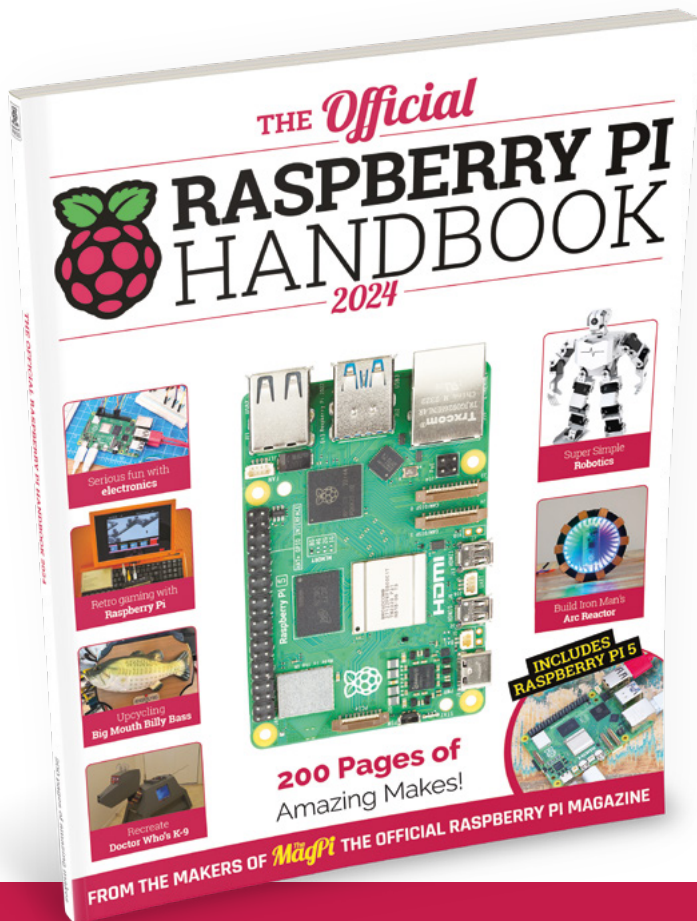


RASPBERRY PI HANDBOOK

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

FORGE

HACK | MAKE | BUILD | CREATE

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

PG
54

FINE PHOTOS

Get the most out of your Raspberry Pi Camera

PG
60

PCB MATERIALS

Find the perfect substrate for your next circuit



PG
66

3D-PRINTED MUSEUM

Fill your house with artefacts from around the world

PG
68

FIXING CAMERAS

Give a vintage camera a new lease of life

PG
50

SCHOOL OF MAKING

Start your journey to craftsmanship with these essential skills

50 Pico Modular

PG
72

HULL PIXELBOT

Create a simple, programmable robot

PG
80

ULTIMATE BIKE LIGHTS

Ensure there is no more 'Sorry mate, I didn't see you!'

Pico Synthesizer: MIDI input

Get some music into your microcontroller



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.

Last month, we introduced our project to make a Pico-based modular synthesizer and built our first module: a voltage-controlled oscillator (VCO). The problem with this is that we need voltages to go into it to make noise. We'll experiment with different inputs in the future, but let's start with the simplest way of getting music into a system: MIDI.

MIDI is a very simple protocol for controlling musical instruments. It does have its own style of cables and connections, but to get started, we're going to use MIDI over USB, which will let us control our synthesizer from software on a computer.

Fortunately, MIDI is well supported on CircuitPython, which runs on Pico, so getting the messages in is fairly straightforward.

MIDI has a series of channels, each operating entirely separately. This way, a single MIDI connection could control many different instruments. Notes are represented by numbers from 0 (C2 at 8Hz) to 127 (G9 at 12543Hz). However, our hardware isn't going to have the capacity to play all these notes and channels, so we'll have to trim things down a bit.

Last month, we used an R-2R ladder to create an 8-bit DAC. This was fine for the purposes we needed

it for, but it won't cut it for our control voltage output. If we stick with the 1V per octave standard, this only gives us seven separate values per note, which isn't really enough to tune it in.

Since the ADC that will be reading the analogue voltage is 12 bits, there's not much point in going above this, so we'll match it with a 12-bit DAC. This gives us a whopping 100 values per semitone, which should be enough to make sure we're hitting the right notes.

The number of channels we can play depends on the number of DACs we have to create the control voltages.

Taking a quick look at the available hardware, there are two DACs that we might want to consider: the MCP4725 and MCP4728. The first of these is a single-channel 12-bit DAC, and the latter is the same but with four channels. They're a bit fiddly to work with on their own, but Adafruit has modules with them broken out for \$4.95 and \$7.50, respectively. We've tried with both of these and have code that works.

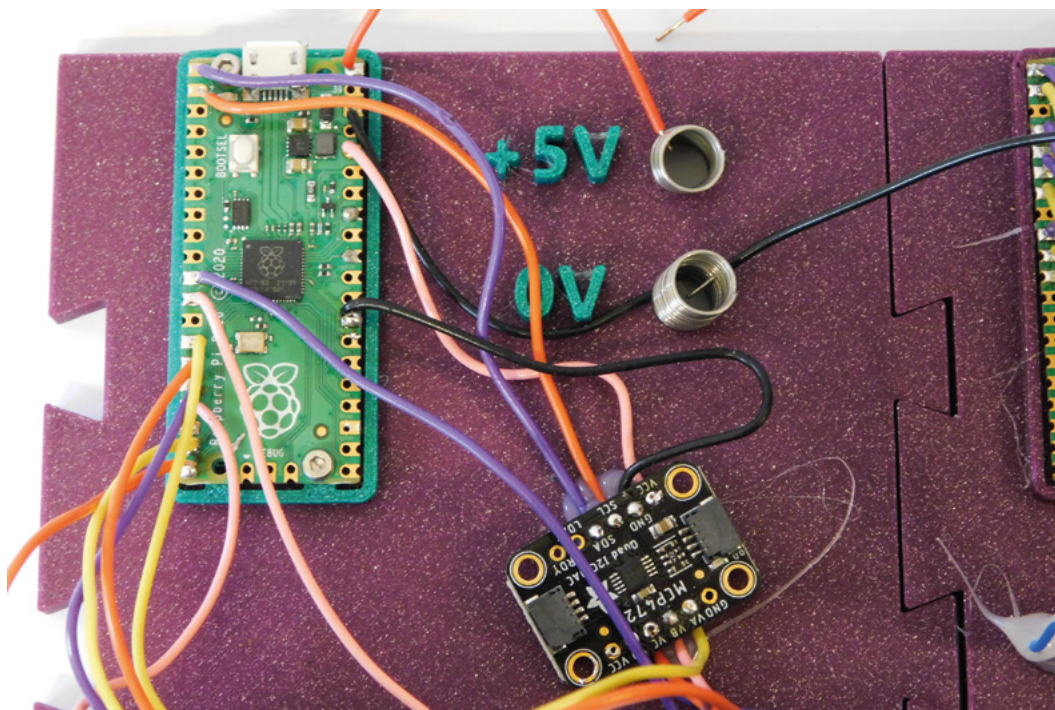
Both the MCP4725 and the MCP4728 are controlled over I2C, which requires two GPIO pins. These are known as SDA and SCL (Serial Data and Serial Clock). We've used GPIO 0 and 1, respectively. You can use others, but not all GPIOs are capable of being SDA and SCL. Check the documentation if you want to wire it up differently.

Control voltage (CV) gives us a pitch, but we also want the ability to start and stop notes. This is known as a 'gate'. Turning the gate output on means that the note should play; turning it off means it should stop.

Of course, there are some things that don't need a control voltage. If we had a module that created percussion sounds, then a simple gate might be enough, so we're going to have an additional four channels of just gate outputs.



Right We're comparing spring connectors and banana plugs for connecting up the synth



Left ◆
Using different
coloured wires
makes it easy to
follow where
each one goes

Let's take a look at the code for this:

```
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))
```

// We're going to use MIDI over
USB, which will let us
control our synthesizer from
software on a computer **//**

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

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

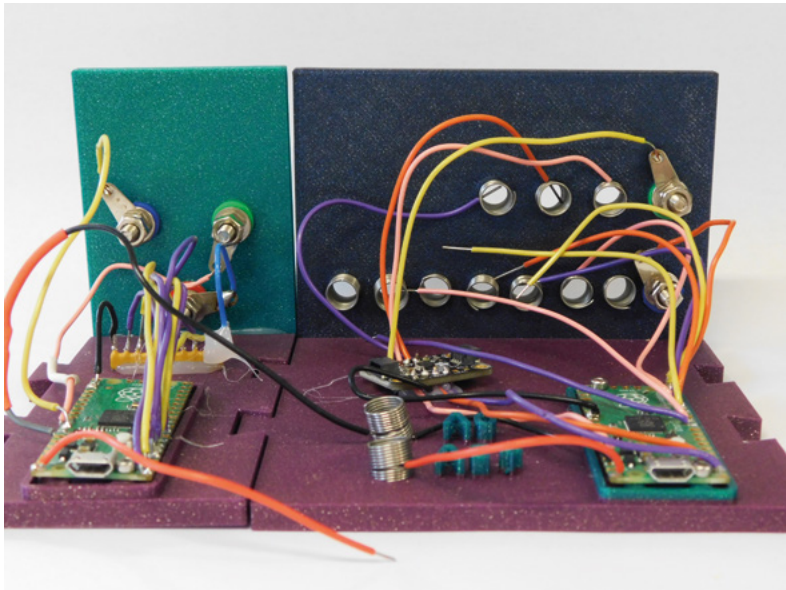
octave_size = int(65535/3.3)
notes = []

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

def note_to_dac_val(note):
    if note < basenote: return 0
    if note >= basenote+int(3.3*12): return max_ ->
```

Below ◆
You can add text in
multiple different
colours by making it
different heights





Above ♦
Banana sockets are more robust and less likely to lose their connections

```

val
    return notes[note-basenote]

last_note = [0,0,0,0]

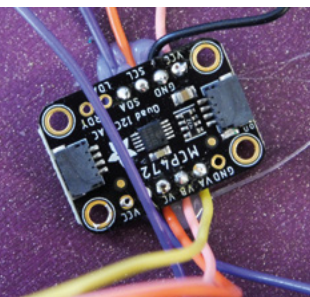
while True:
    msg = midi.receive()
    if isinstance(msg, NoteOn) and msg.velocity != 0:
        print(" channel: ", msg.channel, " on: ",
msg.note)
        print("value: ", note_to_dac_val(msg.note))
        if (msg.channel < 4):
            dacs[msg.channel].value = note_to_dac_val(msg.note)
            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
                print(" channel: ", msg.channel,"off: ",
msg.note)
            else:
                print("not last note off")
    
```

Below ♦
We didn't add mounting holes for the DAC. Instead, we're relying on a drop of hot glue. Will this prove to be a mistake?



At the moment, we have a very rudimentary connection between MIDI note and voltage (and then voltage and note in the VCO). We're going to take a more detailed look at tuning everything and getting the right note out in a future article. For now, we'll

content ourselves with getting a note out. The notes should be in order, so a higher note number produces a higher pitch, but that's about as technical as we'll get for now. As we build out our machine, we'll tune it so we're creating exactly the notes we want.

This loops through, sees if there's been a MIDI message, and, if there has, acts appropriately. There is a bit of confusion because you can create a MIDI score that the synth can't play. Our VCO is monophonic. That means that it can only play one note at a time, but there's nothing about MIDI that requires this to be the case. A MIDI producer might send out multiple notes on events without a note-off. What should our system do in this case?

It would be perfectly reasonable to always play the most recent note and stop the previous note; however, we've gone for a slightly simpler solution and only playing a note if no other note is currently playing. The real solution here is to make sure that your MIDI input is suitable, though.

This produces a note (as control voltage) and a gate (as an on/off). Last month, we created a module that takes an input and converts it into an audio-frequency wave, with the frequency based on that output. For our use here, we've tweaked this so that it also takes a gate input and the volume of the wave is set by the voltage on the gate input. This voltage can be anywhere between 0 and 3.3V. Our current MIDI module can only output either 0 or 3.3V on the gate, but in the future, we'll look at other modules that can do more complex things with the gate input.

We won't go through the full code again. Most of it is still setting up the wave table and the PIO state machine. The main loop is now:

```

while(true) {
    adc_select_input(0);
    uint16_t result = adc_read();
    freq = LOWEST_FREQ * pow(2,
result*conversion_factor);
    divider = calc_clkdiv(freq,
WAVESIZE);
    pio_sm_set_clkdiv(pio, sm,
divider);

    adc_select_input(1);
    uint16_t gate_adc = adc_read();

    if ((gate_adc > (last_gate_val +
GATE_THRESHOLD)) || (gate_adc < (last_gate_val -
GATE_THRESHOLD))) {
        printf("voltage change,
%d\n", gate_adc);
    }
}
    
```



```

        scale_wave(gate_adc);
        flip = true;
        last_gate_val = gate_
adc;
    }
}

```

This monitors the voltage on the gate ADC and if it changes by more than the threshold it calls the `scale_wave` function:

```

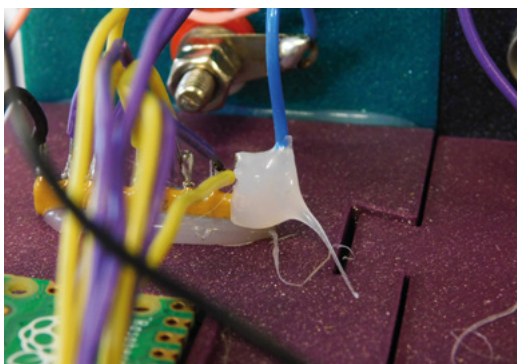
void scale_wave(int scale_adc) {
    int back_buffer = (playing_buffer + 1) % 2;

    float scale_factor = (float)scale_adc /
(float)MAX_ADC;
    for(int i=0; i<WAVESIZE; i++) {
        buffers[back_buffer][i] =
original_wave[i]*scale_factor;
    }
}

```

This simply adjusts the size of the values in the wave. This does cause us a bit of a problem in that we only have 8 bits to play with at full volume, and as we go into smaller and smaller volumes, we're going to have fewer bits to use to generate our wave. At low volumes, this could introduce quite a bit of quantisation error. This is a trade-off with using the cheaper R-2R DAC. Whether or not this is a good trade-off depends on what you want to play. Later on in the series, we'll investigate what the difference actually sounds like.

The final change is in the loop that runs on the other core. We don't want to modify the waveform while it's playing, so we store it in two places. One where it's



Above

The R-2R array proved a bit weak, so we've strengthened it with a blob of hot-glue

currently playing and another where we can modify it. Once we've modified it, we set a flag to let the playing loop know to play from the new modified buffer, and we set the other buffer as the one to be modified.

```

void core1_loop() {
    while(true) {
        if (flip) {
            playing_buffer =
(playing_buffer + 1) % 2;
            flip = false;
        }

        for(int i=0; i<WAVESIZE;i++){
            pio_sm_put_
blocking(pio, sm, buffers[playing_buffer][i]);
        }
    }
}

```

We now have two simple modules that, when we combine them, we can play a tune. You can design your music using any software that can output to MIDI. For example, there's an online tool at signal.vercel.app/edit – this can create multiple tracks that go to different MIDI channels.

We now have a playable but very basic synthesizer. In the coming months, we'll tune it up, add some new ways to make more complex sounds, look at the various ideas behind making sounds, and hopefully build an instrument that can make some fun and unique music. □

PHYSICAL SETUP

You'll notice that we've said very little about the physical setup so far, and that's because we're still experimenting and we want to have something that's relatively stable before we make it public. However, let's take a look at where we are now, and what we're working towards.

We want a system that's 3D-printable (though if it needs a few screws, etc., that's fine). We want it to look great, and we want it to be both expandable and mix-matchable. The process we're hoping to get to work is modules that connect together using a jigsaw-like arrangement. This is expandable almost infinitely in the horizontal axis, and we're working on a way of making it scale vertically as well.

We initially tried making each module 70 cm high, but this is proving a bit limited, so we'll probably expand this to give more space – this is needed both because it's good to be able to fit in lots of bits, and also because 3D-printed text needs to be fairly large in order to read it.

We're still looking through the options for electrically connecting everything. In the first iteration, we've tried springs like you got in electronics sets in the 1980s and 1990s. While these are very satisfying to use, they're not very reliable.

We're currently experimenting with banana plugs, which seem to work well, but we haven't yet experimented with them enough to see if they'll do what we're after.

Camera Module: Understanding camera modes



David Plowman

David is an engineer at Raspberry Pi with a special interest in camera software and algorithms, and image processing hardware.

raspberrypi.com

Take control of your Raspberry Pi Camera Module and how it captures images

Cameras deliver sequences of images and we can control certain aspects of how they are delivered, like the exposure time or frame rate. However, the limits of these parameters are all governed by the mode in which the camera is operating. For example, the current camera mode may have a maximum frame rate that it will allow, and therefore any attempt to ask for a higher frame rate will fail unless a new camera mode is selected beforehand.

Selecting a new camera mode is a relatively expensive operation. It requires reprogramming much of the sensor's internal configuration. Therefore, it cannot happen while the camera is running and delivering images. Instead, the camera must be stopped completely, reconfigured into the chosen mode, and finally restarted. This whole process can easily take several hundreds of milliseconds.

Camera mode range

The range of available camera modes is determined by the driver for the image sensor. To all intents and purposes, this list of modes is fixed. Adding new modes is a quite technical procedure that will

require detailed knowledge of the internals of the image sensor (probably from the manufacturer) and also of Linux kernel programming.

Image resolution

A sensor has a 'native' resolution, which corresponds to the maximum image resolution that the sensor can provide. However, we can usually configure the sensor to output images of different sizes, as follows.

Cropping

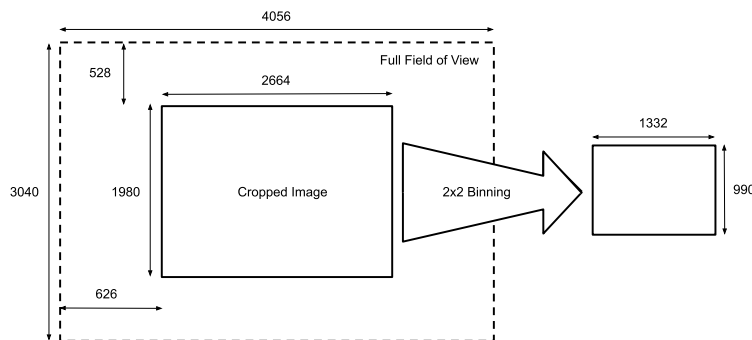
Instead of reading out the sensor's full array of pixels, giving us the 'native' resolution, we can normally configure a sensor to read out only a window, or 'crop', from within the entire array of pixels. This limits the field of view - you don't get the full image that the sensor could have delivered - but in return you may be able to read the image out more quickly, giving a higher maximum frame rate. When we describe the 'crop' associated with a camera mode, we measure it in units of the full native sensor resolution, which we'll demonstrate in one of the diagrams.

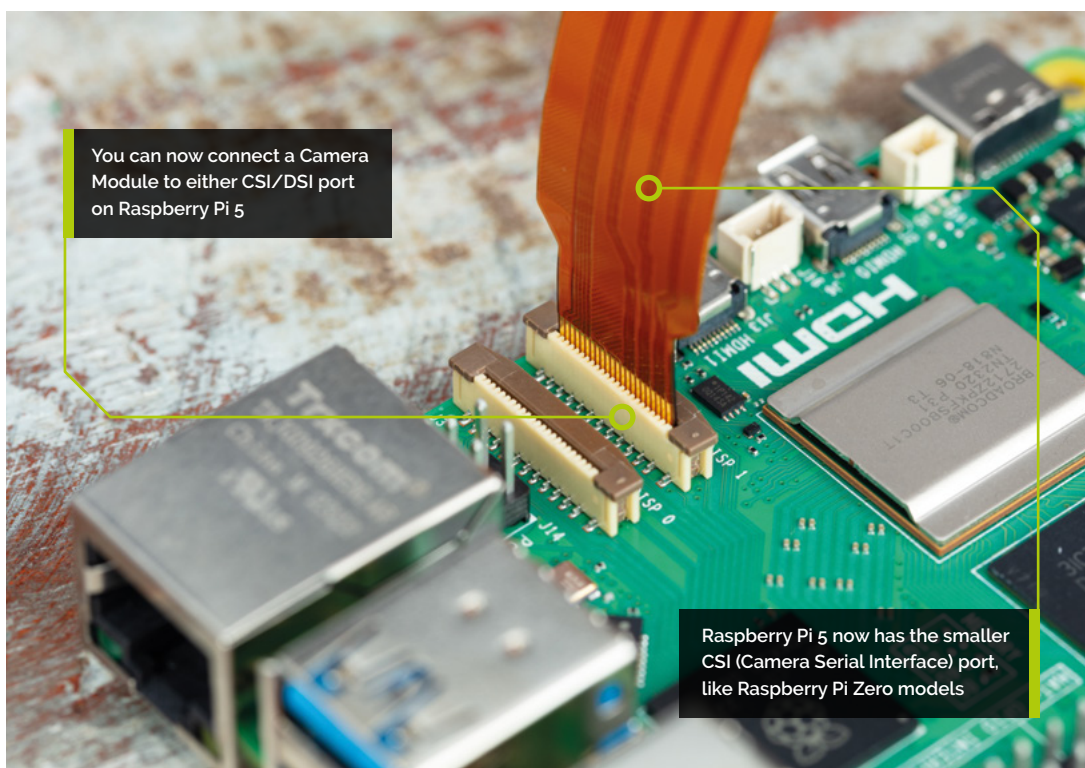
Binning

Many sensors have a feature where every little 2x2 block of pixels is averaged to create a single output pixel, and this process is known as 'binning'. The end result is an image with half the height and half the width of the full sensor resolution (see **Figure 1**).

Binned modes produce images with lower noise (because of the averaging). As with cropping, they are able to run at significantly higher frame rates. Binned modes also give applications the possibility to deal with fewer pixels in every image, which is often helpful in preview or video use cases.

▼ **Figure 1** Cropping and binning to create a smaller output resolution





Very similar to binning is 'skipping'. This too turns blocks of 2×2 pixels into a single output pixel, though simply by dropping three of them. Therefore it doesn't benefit from the improved noise levels.

The process of turning 2×2 pixel blocks into a single output pixel is referred to as ' 2×2 binning'. Although less common, some sensors may even support 4×4 binning modes that involve both 2×2 binning and 2×2 skipping.

Sensors normally also allow both cropping and binning, and we'll look at an example of this later.

Bit depth

The camera mode also determines the number of bits in every pixel that it reads out. Although the output images from the camera system normally use 8-bit pixels, the values that we read from the image sensor are normally larger, with typically 10 or 12 bits. Again, the sensor will normally have a native bit depth, although a camera mode may allow you to read out fewer. Here too the benefit would be in achieving slightly higher frame rates.

Other special modes

Beyond this, certain cameras may have special modes that support less common or particularly advanced features. One such example is the Camera Module 3, which has a special High Dynamic Range (HDR) mode that merges multiple pixels together to create an HDR image.

Finding out the available modes

The easiest way to find out what camera modes are available is to open a Terminal window and type:

```
libcamera-hello --list-cameras
```

This will list the cameras attached to the system (usually just one) and, for each, it will list the available camera modes. For the HQ Camera, for example, it will return:

```
Available cameras
-----
0 : imx477 [4056x3040 12-bit RGB] (/base/
soc/i2c0mux/i2c@1/imx477@1a)
  Modes: 'SRGGB10_CSI2P' : 1332x990 [120.05
fps - (696, 528)/2664x1980 crop]
        'SRGGB12_CSI2P' : 2028x1080 [50.03
fps - (0, 440)/4056x2160 crop]
        2028x1520 [40.01
fps - (0, 0)/4056x3040 crop]
        4056x3040 [10.00
fps - (0, 0)/4056x3040 crop]
```

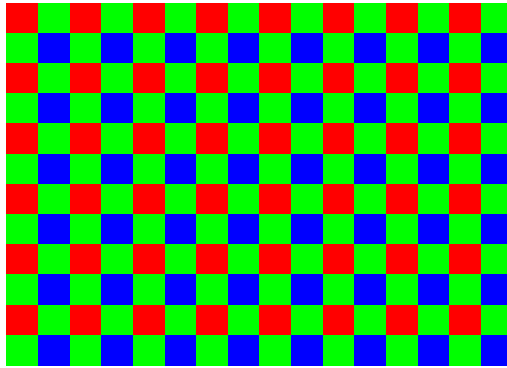
We can interpret this as follows:

1. There is just a single camera, numbered as camera 0. This line also identifies it as an imx477 (the sensor in the HQ Camera), with a native resolution of 4056×3040 pixels, and a native pixel depth of 12 bits.

THE MAGPI



This tutorial is from in The MagPi, the official Raspberry Pi magazine. Each issue includes a huge variety of projects, tutorials, tips and tricks to help you get the most out of your Raspberry Pi. Find out more at magpi.cc



► **Figure 2** A Bayer pattern – every pixel is only red, green or blue

2. Next, it tells us that there is just one ‘SRGGB10_CSI2P’ mode. This is the ‘pixel format’. Most of it isn’t too important, but the number 10 here tells us that this is a 10-bit mode, rather than the more usual 12-bit modes below. It’s clear this has been chosen to give us a maximum of just over 120 frames per second.
3. After the pixel format, every mode lists the output resolution for that camera mode. If you ask your camera application for images larger than this, while using this mode, then it will have to upscale them.
4. Next we have three ‘SRGGB12_CSI2P’ or 12-bit modes, with frame rates ranging from about 50 fps down to about 10 fps for the full-resolution version.
5. At the end of each line we have the crop for that mode. This is given in the form (x_offset, y_offset)/width × height. So the first mode (the 10-bit one) reads a 2664×1980 rectangle out of the native 4056×3040 image but offset with its top left corner at (696, 528) in the native image. A quick calculation that $4056 = 2 \times 696 + 2664$ should confirm that we’re reading out the central portion of the image (and the same again in the vertical axis). Note how this camera mode involves both cropping and binning.
6. Finally, we note, again for the 10-bit mode, that the crop window of 2664×1980 is exactly twice the output image size, confirming that this is a 2×2 binning mode.
7. Note also how the 2028×1080 mode is very similar to the 2028×1520 mode – same width, both using 2×2 binning – but the slightly shorter image height allows a higher frame rate. It’s clearly intended to allow applications to record 1080p50 videos.

Top Tip



Full field of view

When the two crop offsets (x_offset and y_offset above) are both zero, then this is a full field of view mode.

Which mode to use?

This is a surprisingly difficult question. Different camera modes have different features and advantages, and there’s no one-size-fits-all method to decide which mode to use based on, for example, the final output image size that an application has requested.

For instance, is the application happy to accept some upscaling? Or would it rather use the highest resolution camera modes, and downscale the image, to get better image quality?

Does an application prioritise getting the widest possible field of view from the image sensor, or would it be happy to sacrifice the field of view in return for faster frame rates?

Whilst libcamera-apps will attempt to make a reasonable choice for you based on the arguments provided, it won’t always be right. For this reason, it’s possible for the user to override this selection process and choose exactly which camera mode to use.

“ It’s possible for the user to override the selection process and choose exactly which camera mode to use ”

Selecting a camera mode explicitly

The libcamera-apps have a number of mode-related command-line parameters that allow you to select exactly which camera mode you want to use. They all take the form ‘width:height:bitdepth:packing’, where:

- **width** is the output width of the camera mode
- **height** is the output height of the camera mode
- **bitdepth** is the bit depth of the camera mode
- **packing** is the letter P (‘packed’) or U (‘unpacked’). Packed formats will be preferable for most people as they use less memory bandwidth and are therefore more efficient, though users who want access to the raw pixel data from the sensor (beyond the scope of this guide) may prefer unpacked formats.

The command-line parameters are named as follows:

- **libcamera-hello** accepts the parameter `--viewfinder-mode mode` where mode is as defined above.

- **libcamera-vid** accepts the parameter `--mode mode` with mode as defined above.
- **libcamera-still** accepts both the parameters `--viewfinder-mode mode` which defines the camera mode for the preview phase, and `--mode mode` which defines the camera mode for the actual still image capture.

For example, using an HQ Camera:

```
libcamera-vid -o 1080p50.h264 --mode
2028:1080:12:P --framerate 50
```

...will record a 1080p resolution at 50 fps. We know it can reach this frame rate because we have explicitly chosen a camera mode that is capable of this. The camera mode, outputting a larger image than 1920×1080, also guarantees that no upscaling is happening.

Raw images and camera tuning

We've mentioned how camera modes define the resolution and format of the 'raw pixels' that are read from the image sensor. But what exactly are these 'raw pixels'?

The pixels that we obtain from the sensor require a large amount of processing to turn them into viewable images. For a start, each pixel records only one colour – red, green, or blue – instead of all three. This kind of image is often referred to as a Bayer image, named after Bryce Bayer who pioneered their use at Kodak. Although other forms of image sensor exist, so-called Bayer sensors remain very popular because they work well and are cost-effective (see **Figure 2**).

You're also now in a position to understand why the IMX477 was described by `libcamera-hello --list-cameras` as an RGGB sensor. It's because it's made up of pairs of rows where every Red-Green pixel row is followed by a Green-Blue row.

Not only do the 'missing' colours have to be calculated (or interpolated), but a whole host of other corrections have to be applied. This includes fixing any stuck pixels, reducing noise levels, and making the brightness levels and colours look correct. Finally, the image needs to be delivered in the correct resolution and format to the end-user or application.


Most of the time, all we want is the fully processed final image that comes out of the camera system, but there are certainly applications – sometimes scientific in nature, or for professional photographers – where access to the raw sensor data can be useful. We can even

capture the raw sensor data in a DNG file (Digital NeGative). For example:

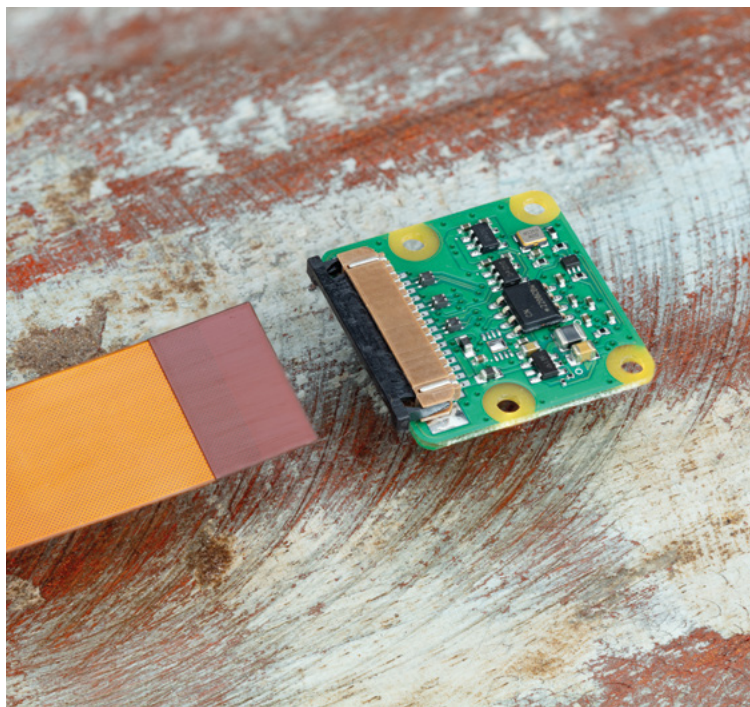
```
libcamera-still --raw -o image.jpg
```

...will save the raw file **image.dng** alongside the requested JPEG file **image.jpg**. You would need special software tools to use or convert these DNG files, which lie beyond the scope of this guide.

Tuning files

Finally, it's worth noting that we actually have many choices in how we turn the raw image into a final picture. Raspberry Pi provides a default 'tuning file' which supplies the various parameters that control this process, and you can look at them in the file **imx477.json**, which should be available on Raspberry Pi OS's file system (usually in `/usr/share/libcamera/ipa/rpi/vc4`). This is a JSON-formatted text file for the IMX477 sensor (in the HQ Camera) with each named block of parameters controlling one specific aspect of the process. Where the default camera tuning does not suit a user's applications, we would encourage them to develop their own camera tuning files, and to consider contributing them back to the wider Raspberry Pi user community. Help and documentation are available for those who would like to get involved: magpi.cc/cameradatasheet. 

▼ Connecting Raspberry Pi cable to Camera Module



RASPBERRY PI 5

Priority Boarding

We've reserved Raspberry Pi 5 boards
for HackSpace magazine subscribers

GET YOUR RASPBERRY PI 5 NOW!

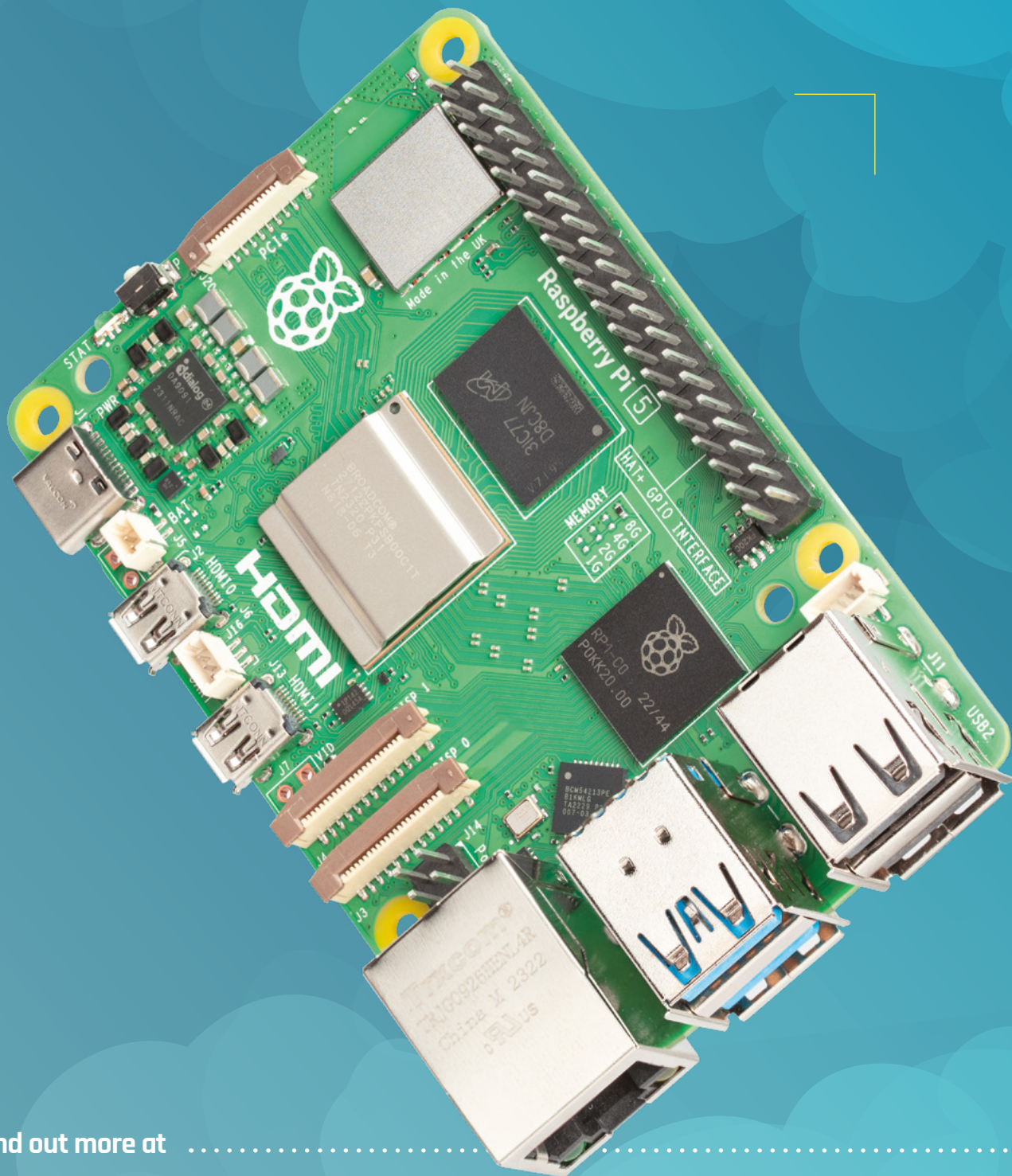
Want to get Raspberry Pi 5 sent to you right now, without waiting for stock? With Priority Boarding, you can order your Raspberry Pi 5 (4GB or 8GB) and it'll be sent out right away. Raspberry Pi has set aside thousands of Raspberry Pi 5 computers for The MagPi and HackSpace magazine print subscribers to buy. Enough to guarantee every subscriber can get one without going out of stock.

How it works!

If you take out a print subscription to HackSpace magazine or The MagPi, we'll send you a unique code in the next few days. Along with the code, you'll get details of how to use it through our partners (Pi Shop in USA and Canada and the Pi Hut everywhere else).

Get started

Get your subscription at hsmag.cc/subscribe. You'll be able to buy your Raspberry Pi 5 now and get a magazine packed full of incredible maker tutorials, projects, and reviews. As an added bonus, we'll also send you a Raspberry Pi Pico W as a free gift.



Find out more at

hsmag.cc/priorityboarding

Terms & Conditions Priority Boarding codes will be emailed to everybody with a print subscription to The MagPi or HackSpace magazine. People who subscribe to both magazines (print only) will receive two codes. Priority boarding does not apply to people with App Store, Google Play, ZINIO, PDF contributions, or other paid-for subscriptions. Each code will entitle you to purchase 1 x Raspberry Pi 5 model (4GB or 8GB) for the standard retail price and delivery. Multiple codes need to be used individually. This is a limited offer and is subject to change or withdrawal at any time.

KiCad, different PCB substrates

In this part of the ongoing KiCad series, let's look at some of the different materials PCBs can be fabricated from



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!

So far in this KiCad series, we have designed our PCBs and had them manufactured using the common PCB material known as FR4. On FR4, our copper traces sit on top of fibreglass, and this is what most people think of when they hear the phrase 'PCB'. However, it's not the only option, and most of the PCB fabrication houses have a variety of different materials that they can make a PCB from. The circuit is typically still copper, but these traces sit on top of other materials. Let's take a look at the alternatives.

FR4 is a standard of material for the dielectric, the insulating non-conducting part of a PCB. FR4 is a type of composite fibreglass material, made up from fine glass reinforcing fibres and epoxy resin,

and is very non-conductive. The 'F' and 'R' are an abbreviation of 'fire-retardant', which is one of the many benefits and safety features of the material. Whilst fibreglass composite materials have inherent fire-resistant qualities, FR4 has bromine added which characteristically will reduce the spread of fire.

As well as being essentially fireproof, FR4 has a low thermal expansion coefficient – this means that it won't expand or contract very much in hot or cold environments. This is important because if a PCB expands or contracts a large amount, then it is likely that traces and components on the board will be damaged, crack, or disconnect. Fibreglass composites are strong and most PCB houses will offer a choice of thickness of FR4 boards, allowing you to choose a thickness and strength suitable for your application.



Right ♦ Some small polyamide flexible PCB antennas fabricated by OSH Park

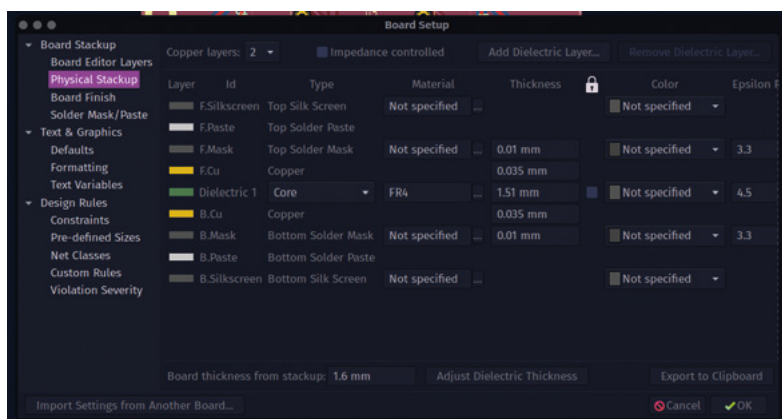
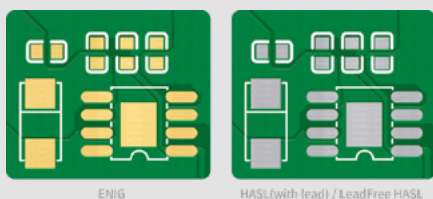
GOLD OR SILVER

There are lots of other choices that can be made when getting a PCB fabricated. One choice is the type of surface finish applied to any exposed copper pads. These take the form of differing types of covering or plating, which act to both allow the easy fitting of components, solder, or solder paste, whilst stopping bare copper pads from oxidising if left uncovered.

'Hot Air Solder Levelling' (HASL) is a common option where the board is dipped fully into molten solder, removed, and excess solder driven off. This results in all copper areas being covered in a flat, thin layer of solder. It's an older technology and has been used in PCB fabrication for multiple decades and, as such, it's usually an affordable option. It is easy to solder onto, offers good protection against oxidisation, and is stable and long-lasting. It has a long shelf life, so if you don't get around to populating your PCBs for a long time, they will still be in good condition. Finally, HASL is often offered with options that contain lead or options that are lead-free.

'Electroless Nickel Immersion Gold' (ENIG) is a newer technology that applies a two-layer coating to the exposed copper parts of a PCB. The layer directly on top of the copper is a thin layer of hard nickel which creates a barrier stopping the copper from oxidising. To stop the nickel plating from oxidising, a fine layer of gold is immersion-plated over the top. When components are soldered onto ENIG surfaces, they create a strong bond with the nickel layer, and therefore the copper underneath. ENIG is also suitable for covering larger planes on the PCB, so has become a popular choice.

There are other surface finishes, such as 'Immersion Tin', 'Organic Coating' (OSP), and 'Immersion Silver', but HASL and ENIG are most common. Most PCB manufacturers use either HASL, lead-free HASL, or ENIG, or some PCB manufacturers will offer them as choices. For example, JLCPCB has HASL selected as default, but you can switch to lead-free HASL or ENIG. OSH Park PCBs are all created with an ENIG surface finish. Over on PCBWay, you can select from a larger range that includes HASL, ENIG, OSP and more – you can even specify to have no coating whatsoever.



As FR4 is the most common PCB material, there aren't many special considerations or changes needed when designing a PCB for FR4. If you want to set up KiCad to display your target board thickness when using the 3D viewer, you can change this by adjusting the dielectric thickness value via the Board Setup. This dialog is found at File > Board Setup in the PCB Editor, then you need to select the 'Physical Stackup' option from the list, and then adjust the 'Dielectric 1' thickness variable (Figure 1).

Flexible PCB designs are commonplace in modern electronics, with many use cases. Probably the most

Figure 1 Changing the thickness of the 'Dielectric 1' layer to change the overall thickness of our PCB design

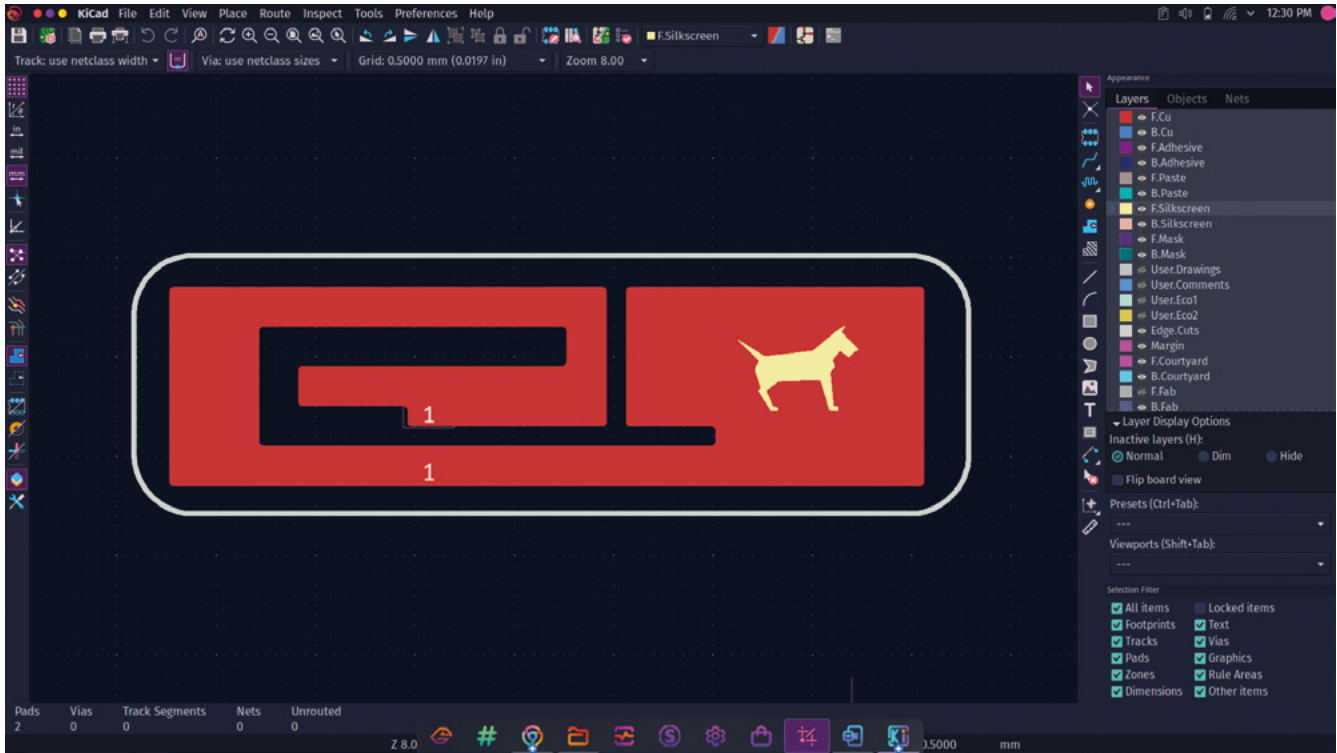
There are numerous types of flexible PCB substrate, with the most common being polyamide film

common to come across are simple flex connectors. These have obvious advantages, in that they can connect components or PCB modules in different locations in a system. Flex connectors can be inserted into specific clamping sockets or, indeed, can be designed to directly solder onto copper pads. They have the benefit of being able to fold and bend around, allowing complex layouts of PCBs which do not need larger header pins or sockets. There are numerous types of flexible PCB substrate, with the most common being polyamide film. Traces and pads work in a very similar manner to any PCB, →

QUICK TIP

Most PCB fabrication services have detailed information on the thickness of each layer of their PCBs. You can use this to emulate the PCB accurately in KiCad.

TUTORIAL

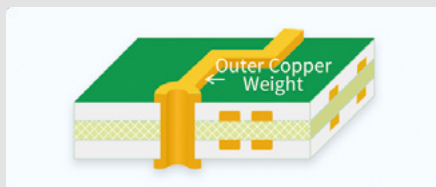


Above ♦
Our flexible antenna design in KiCad

COPPER CONUNDRUM

Copper weight describes the thickness of copper at any given point on a PCB. It's often expressed as ounces per square foot, so a 1 oz/ft² copper weight will be thinner than 2 oz/ft². Copper weight can be important in terms of designing PCB traces, as the weight affects the depth of the trace. Different trace widths and weights may need to be used when considering the amount of current particular parts of a circuit may be passing. Similarly, track impedance and RF qualities of tracks may well come into play, as well as track sizes, when trying to match lengths of track in more complex PCB designs.

Most common copper weightings are 1 oz or 2 oz, and many PCB houses will offer these as a choice. If you require a precise copper weighting, it's possible, at a price, for some PCB fabricators to offer more bespoke weight of copper by plating or etching extra material to or from the board.



It's good advice to read any guidance your PCB fabrication house has to offer and to speak to them directly

in that they are a thin layer of copper. Flexible PCBs can become quite complex to design for, and many PCB fabrication houses will have options for custom layer materials in flex PCBs. These can also include rigid sections, which means you have to supply a design that can distinguish different substrates within it. If you plan to use this technology, it's good advice to read any guidance your PCB fabrication house has to offer and to speak to them directly to explain your concept. At the simpler end of flex PCBs, it can be as straightforward as laying out a regular PCB design, exporting some Gerbers, or sending a project file to your PCB fabricator.

A good use case for flex PCBs, and a good simple example, is a flexible antenna. We'd found a paper online with an image and dimensions for a dual-band 2.4GHz and 5GHz patch antenna, which piqued our curiosity, so we set about laying it out in KiCad. We actually began in Inkscape: the PDF source we

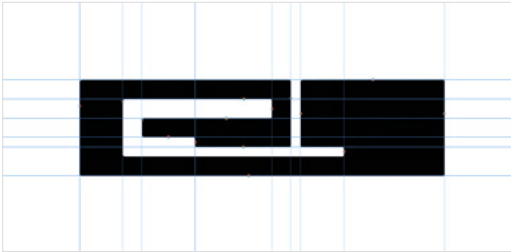
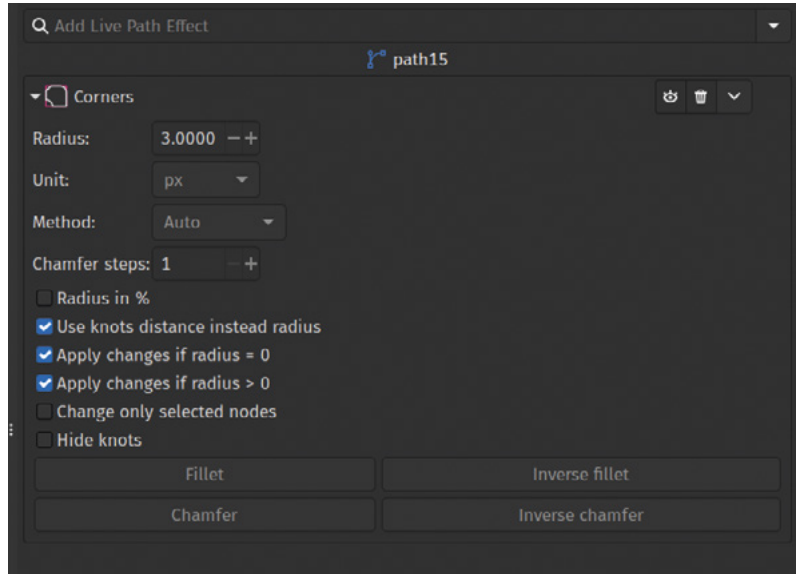


Figure 2 ♦ Using Guidelines in Inkscape to manually trace the antenna design image

had for the design was imported into Inkscape, but it had a gradient fill and therefore wouldn't export correctly as the whole object was full of nodes. If this was a bitmap image, it would have been a good candidate for Inkscape's 'Trace Bitmap' function, but as it was a vector file, this wouldn't work. However, as a vector image, it was straightforward to drag in vertical and horizontal guide lines and snap these to the edges of the PDF antenna drawing (**Figure 2**).

Once we had a guideline for every part of the antenna in place, it was a simple job to use the pen tool to draw a continuous line around the part, closing it to form a solid object. There is some discussion online around not using sharp 90-degree angles in traces when aiming for a flex PCB as, if the PCB is flexed, the sharp corners can be origin points for tears and failures. Whilst not a massive concern for this design, it was easy to add internal and external radial chamfers to the antenna object



by selecting the object, then using the 'Path effects' dialog to apply the 'Corners' path effect chamfer option (**Figure 3**). Finally, we deleted the original imported PDF, resized the document to fit our antenna design, and then saved it as an SVG.

In a new KiCad project, we actually ignored our usual workflow of creating a schematic and associating parts to schematic symbols and went straight to the PCB Editor. You can directly add components and create traces in the PCB Editor with no schematic in place. For a more complex project, we wouldn't recommend this approach, as you →

Figure 3 ♦ Using the 'Corners' path effect in Inkscape to add internal and external chamfers

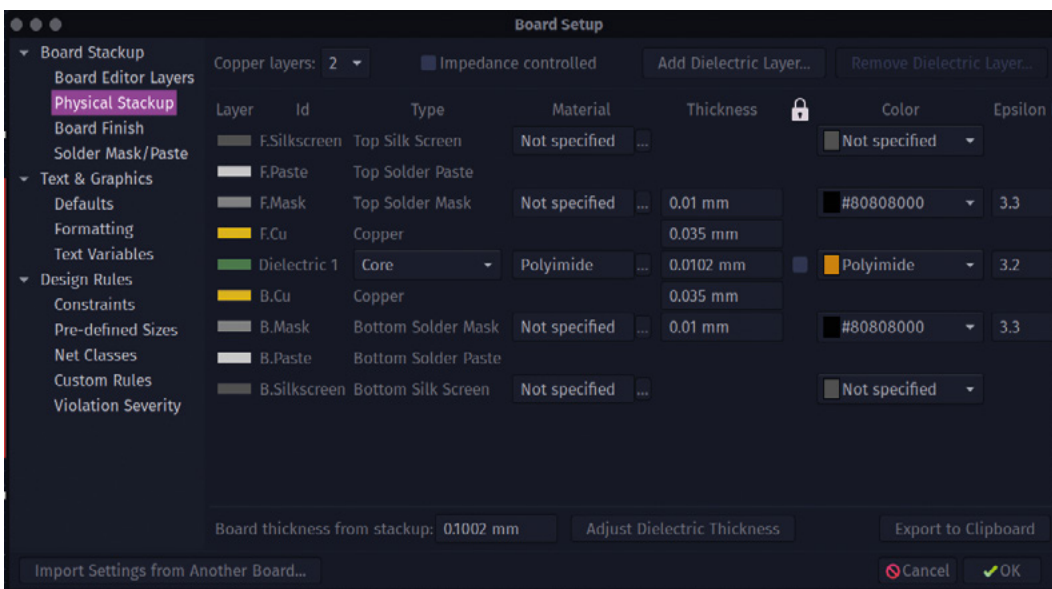


Figure 4 ♦ By setting the board material, dielectric thickness, and solder mask colours, we can emulate a flex PCB in the KiCad 3D viewer

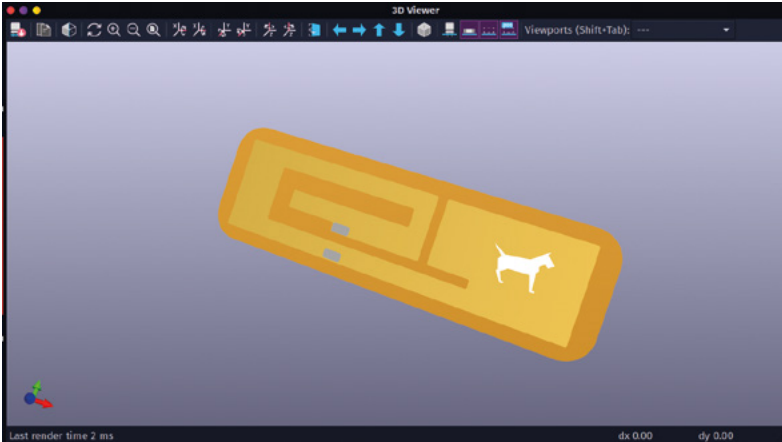


Figure 5 ♦
A 3D render of our flex circuit

have no means of checking connectivity or creating net connections, but for small simple projects like this, it's pretty easy.

We imported our antenna SVG, making sure to import it onto the Front Copper layer (F.Cu). We then used the 'Add a Footprint' tool to add two small SMD pads. We positioned these on the points in the antenna design that the original design had indicated. With no net connections due to no schematic, these pads will be directly connected to the large copper antenna design that we just placed, but of course, the pads will have no solder mask

|| All that remains is to then quickly draw another SVG for the outline of the antenna

over them, allowing us to solder on a connecting coaxial cable. All that remains is to then quickly draw another SVG for the outline of the antenna, which we did in Inkscape, but you could just draw in KiCad. With the outline imported to the edge cuts layer, we have a completed design. To get the design fabricated, we used OSH Park which has a flex PCB offering. One of the nice features of OSH Park is that you don't have to produce Gerber files to upload – you can upload your KiCad PCB file directly to the website for previewing and ordering. Conveniently, either project files or Gerber files don't particularly specify the board substrate or thickness, so we don't need to specify a thin flex board design in KiCad. However, we might want to model the board

FILLING HOLES

Vias, the small plated through-holes that connect different layers of the PCB, have different ways of being finished. For many projects, the PCB fabrication house default will be fine, but it's worth looking at the common options offered.

'Tented vias' are covered with the solder mask, so no solder would stick to them. The hole may or may not be filled, depending on the size of the via. Other benefits of tenting are that you reduce the risk of unintended shorts when boards are being assembled into products or being handled etc.

'Un-tented vias' have no covering, so are finished in the selected surface finish in the same way as exposed pads and other copper features. Whilst this may well be fine, there is a risk of accidentally soldering to vias or for short circuits.

'Plugged vias' are filled in a couple of different ways, or you may have a choice. One way is to fill the via with solder mask; another is to fill the via with epoxy resin. Some manufacturers may only be able to fill vias up to a certain diameter or, indeed, some PCB houses can offer custom approaches where you can ask for all vias of a certain diameter to be filled. The benefit of plugging vias is that the via can't accidentally become filled with solder or other conductive material.

'Conductive plugged vias' are not the most common choice, but some PCB houses can fill vias with conductive material. This can increase the amount of current the via is capable of passing. There are trade-offs in that the conductive filler may thermally expand at different rates than the other board materials, causing small flexes leading to potential failures. As an example, JLCPCB offers the option to fill vias with conductive copper-filled epoxy.

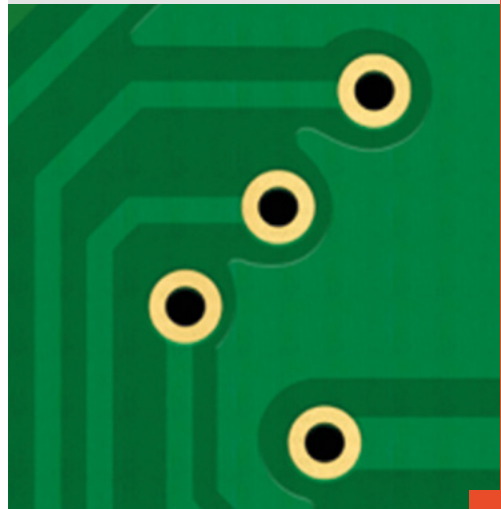




Figure 6 ♦
Our small aluminium
LED module

accurately in KiCad, especially if we are using either renders of the KiCad design in promotion or if we are exporting the board 3D model for use in other CAD programs.

Again, we can use the board setup dialog and the 'Physical Stackup' tab in the PCB Editor to emulate a flex circuit. Your PCB fabrication house will have data about all the thicknesses of each layer of their flexible PCB offerings and you can use this to set thicknesses in the board setup. If you just need a close enough PCB view that looks like your flex design, you can simply adapt the major thickness of the board by changing the material to 'Polyamide' and setting the thickness of that layer to 0.0102 mm (this is the OSH Park flex polyamide layer thickness), as in **Figure 4**. You can then set the colours of the top and bottom solder mask layers to transparent by reducing the opacity to zero using the 'Custom colour' option. This, in the 3D viewer, will then give you a reasonable approximation of a flex PCB (**Figure 5**). Often, PCB manufacturers offer metal substrates for PCBs, commonly copper or aluminium. These substrates can be useful when you need to dissipate heat quickly through a system. Often, aluminium substrates can make sense for

temperature sensor modules where you want the sensor to be accurate but the board not to soak up heat. Another application for metal substrate PCBs is where you want the PCB to act as a heatsink. For example, we had JLCPCB make and assemble some 1-watt LED modules (**Figure 6**). Running at 1 watt, these LEDs generate a reasonable amount of heat and, to promote their long life, it's useful to have some kind of heatsink attached. The reverse side of the LED has a large thermal pad which connects to the board, and the reverse of our aluminium board is bare metal. This acts as a heatsink for the LED module and the temperature is reduced. One thing of note is that aluminium and copper substrate PCBs can be bent if they are put under pressure or load. In fact, when your aluminium PCBs arrive in panels, it can be quite hard to remove them without bending the PCB!

FR4, flexible, and aluminium are not the only options often when we move to high-speed designs, designs running at microwave frequencies, or when we have very special applications like medical devices – there are other substrates available. Rogers, PTFE, Teflon, Copper Core – it's fun to read around some of these more esoteric materials. □

3D-printed museum

Artefacts you can handle without being shouted at



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.

There are a few websites that hold 3D models designed for 3D printing. The quality of the models on them varies from the excellent to the, well, less excellent. Despite the varying quality, they are typically all intended for 3D printing. However, this month, we've eschewed models designed for fabrication and taken a look at models created to be self-replicating. In other words, biological models.

Many museums around the world scan their artefacts and upload them to the web. We scoured the web for scans to print and examine. It's a great way of getting hands-on with objects that are usually locked away behind glass screens.

What really struck us was the complexity of the objects. It's quite common to come across re-creations of skeletons (especially around the end of October), but these tend to be vastly simplified models, constrained by their manufacturing techniques. We set out to see if we could make faithful re-creations of the actual objects, and thereby gain much more of an insight into the reality.

SUPPORTIVE ENVIRONMENT

Support material has a bad name in 3D printing. If you're designing an object, most people will try to reduce or eliminate support material. However, we're not designing it. We can try to minimise the necessary support material by changing the orientation, but we're never going to eliminate it – we're not even going to reduce it that much.



Right ♦ The Natural History Museum's famous Diplodocus skeleton, affectionately known as 'Dippy'. Though it has fierce-looking teeth, it was a herbivore – these teeth were for ripping leaves off branches



Above Older readers may know this as a sabre-toothed tiger, but the Smilodon wasn't a feline, and isn't closely related to modern cats

However, slicing software has gotten a lot better in recent years. We're particularly fans of PrusaSlicers' automatic support painting. This process adds minimal supports to the print, allowing it to be printed well, but also not adding as many as the more traditional 'supports everywhere' option. This not only means you're not wasting filament on unnecessary supports, but the lower density of supports makes them much easier to remove at the end.

It's a great way of getting hands-on with objects that are usually locked away behind glass screens

There are a few support types available. Organic supports look great and have gotten a lot of attention recently; however, we've not found them to be the most reliable. We find we get the best balance between speed of printing and actual support with 'snug' supports.

As well as supports, here are our other top tips for producing great-looking organic prints:

ORIENTATION. It can be tempting to pick the orientation that you think will minimise supports; however, top surface finish is an important factor. You don't get a great finish on curved surfaces that are supported, so try and hide these. This often means printing in a similar orientation to how

you will view the object. Similarly, surfaces that are almost, but not quite, flat can get a lot of false contours if they're horizontal.

LAYER HEIGHT. Adaptive layer height may work on a few models, but in general, we've found organic models to be so complex that there aren't many low-detail areas that can benefit from it. This is really a trade-off between your feeling on layer lines and your feeling on print times.

FILAMENT. OK, we'll be honest and say that we're absolute suckers for sparkly filament. Several manufacturers also make specific bone-coloured filament, and marble filament would also look great. These filaments with a bit of colour variation in them help minimise how visible layer lines are. All our prints are in PLA, as this prints difficult features well.

SIZE. Size and print time are intricately related, and shrinking or growing an object depending on how long you want to spend on it is a perfectly reasonable thing to do. Just beware that 3D scans aren't designed to be printed and, in some cases, they'll become unstable or too thin if you scale them down.

There are loads of models out there, and while we love going to museums, it's not always possible, and they often frown on people handling the exhibits. With a 3D printer and a bit of filament, you can bring items from collections around the world to your home.



Above We printed this wolf skull upside down so that the teeth would print without supports. Unfortunately, this has left it with a messy area on the top of the skull



Left The chimpanzee is one of our closest relations, and this skull looks very human until you notice the teeth



Twin-lens reflex: Old-school cool

Get started with vintage Yashica cameras



Dr Andrew Lewis

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

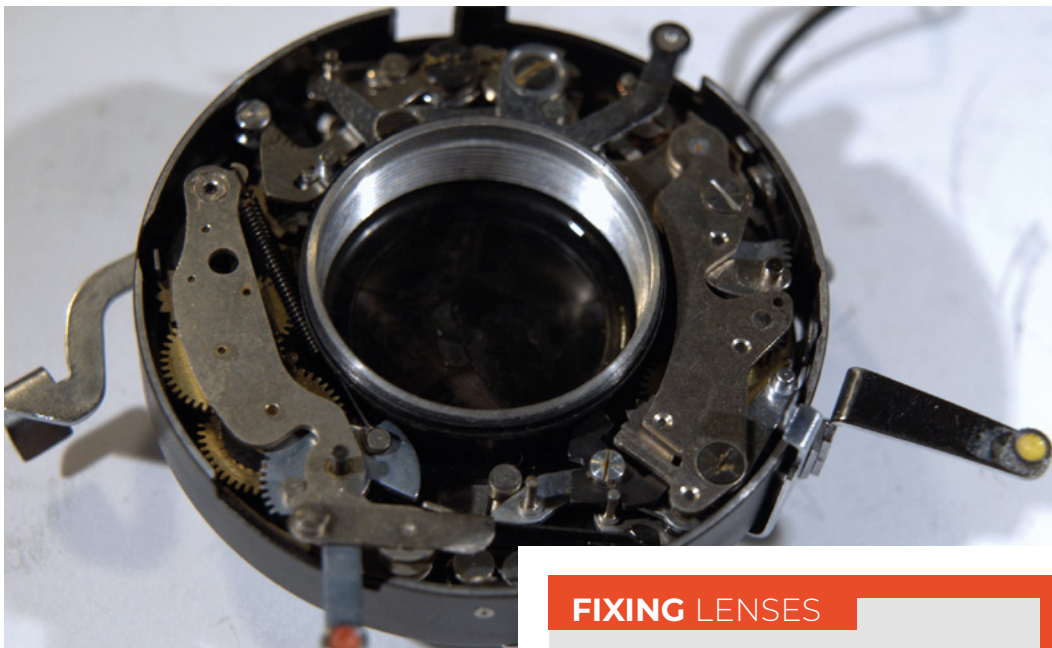
Vintage cameras are a bit like vintage cars. The older they are, the fewer things there are to go wrong. When they do go wrong, the easier they are to fix. In this article, you'll take a look at a classic twin-lens reflex (TLR) camera, see the common problems they acquire over time, and hopefully see why it's worth the effort of getting them back into good working order. You'll also see how modern technology, like Raspberry Pi Pico, can be used to help with the repair process, and learn a few old-school tips that will help you take good photographs.

GRAIN NOT PIXELS

The TLR style of camera is a design classic. It's simple, easy to maintain and, in the case of cameras

like the Yashica TLR, practically modular. Accepting a 120 (or 127) film to take a total of twelve 6cm square exposures, the Yashica TLR is capable of taking absolutely stunning photographs.

At their simplest level, film cameras all have the same very basic design. A lens of some type focuses light through a shuttered aperture for a certain amount of time, projecting an image onto a sheet of light-sensitive chemicals. Everything else about a camera is a design feature – lenses or pinholes, winding mechanisms, mechanical shutters, timers, film size, and viewfinder types are all essentially design choices made by the manufacturer or the end user. In most cameras, these things get wound together into a package and sold to the end user as a complete item that can't really be changed. A few very expensive (think 'did I just pay more for a



Left ♦ Camera mechanisms are a bit like watches or clocks to work with. Have a look at some watch repair videos to get an idea of how to keep parts ordered and what tools to use for the job. Be aware that there are springs inside the shutter, and they are liable to fly off into the wilderness at the slightest flick of a screwdriver. Be very careful, and work somewhere clean

camera than a car?’ type of expensive) cameras have an impressive range of interchangeable parts, but it’s rare for a moderately priced camera to be modular or even moderately hackable to the extent that the Yashica TLR is.

If you’re lucky, you can find a Yashica TLR for sale at around £60, although it’s likely that it’ll be faulty and will need some work to get going. With the flexibility of the Yashica TLR, it’s probably going to be much easier to fix than you might think. The main

“ **If you’re lucky, you can find a Yashica TLR for sale at around £60, although it’s likely that it’ll be faulty** ”

areas you’ll be looking at when you’re assessing a TLR camera are the lenses and focusing systems, the film winder, and the shutter. The cosmetics of the camera don’t really affect its function, but the condition of the light seals does need to be checked and they should be replaced if they’re worn out.

The camera’s lenses screw into the lens assembly and into the shutter. One problem to watch out for is that the top viewfinder lens and the bottom shutter lens both need to be screwed into place by →

FIXING LENSES

The lenses of the Yashica TLR camera are glued together using a blend of tree sap and turpentine called Canada balsam. Over time, variations in temperature and pressure can cause these lenses to separate and leave flecks or mist between the lens elements that will affect the quality of the photographs you take. Fixing this problem isn’t as difficult as you might imagine, and there are a few different ways you can do it depending on the tools you have access to and your confidence level.

CLEAN. LUBE. ADJUST

One method is to use heat to remelt the balsam and then apply pressure to the lens while it cools. This can be done by hand with gloves, a heat-resistant foam pad, and a wooden stick with a felt end. Separate the lens from the camera, and heat it using a temperature-controlled heat gun until you see the balsam melt, then just push the two lenses together. Get comfortable before you start because this is a slow process, and you need to heat the elements gradually and evenly, then apply steady pressure for several minutes while the elements cool slowly. Allowing the elements to cool too quickly can cause them to shatter. If this process doesn’t work for you, you can instead remove the existing balsam using a solvent like Xylene, and glue the lenses back together using a modern lens glue. If the lenses are not fully separated but are clouded or spotted internally, applying some heat will help get them apart. How the balsam reacts to the heat seems to be partly down to the method of balsam application originally used, so mileage may vary with different lenses and you’ll need to experiment to find the best method for your particular lens. A quick internet search on the subject will provide some additional methods and specific glues to consider.

Below ♦ To remove the front assembly of a Yashica TLR, you need to remove the leather from the front and undo the four screws that hold it in place. Removing the rest of the screws will allow you to get to the lenses and shutter mechanism





QUICK TIP

Beware of Yashica cameras that use 127 film – it hasn't been manufactured commonly since the mid-1990s, and while it's possible to re-wrap 120 film onto a 127 spool, it's a fiddly job.

Left ♦

The Yashica MAT LM has a built-in light meter, although it isn't actually very good. If you aren't bothered about maintaining a pristine factory-standard camera, you can remove the light meter by removing a few screws

some well-placed heavy objects, turn the focus knob to its lowest point, and place a distinctive object exactly one metre away from the camera lens. With no macro lenses fitted, the camera should have a minimum focus of one metre. With the greaseproof 'film' loaded into the camera, set the shutter to bulb mode and hold it open so that you can see the scene in front of the camera projected onto the greaseproof paper. Look at the projected image and check whether it's in focus. If it isn't, you can adjust the focus by screwing the bottom lens in or out. Now look at the viewfinder screen, and check that the viewfinder image is also in focus. If it isn't, screw or unscrew the top lens of the camera until the image resolves. Hold the threads in place with a dab of lacquer or glue. Your camera should now be focused well enough for general use.



If you really want the camera to stand out, you can replace the leather with something different



When it comes to cosmetics, it's amazing how much a bit of black Sharpie, a chrome touch-up pen, and some wax polish can accomplish. A couple of hours of cleaning and applying subdued paint effects can make the camera look almost new, and if you really want the camera to stand out, you can replace the leather with something different. □

QUICK TIP

There are specific low viscosity oils and specialist greases produced for optical devices, and these exist for a reason. WD-40 and sewing machine oil are not an acceptable substitute.

Pico-powered Hull Pixelbot

Make a Pico-powered pixel packing robot you can program from your browser in 'Python-ish'



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.

Find out how to create a little robot you can program from your PC browser. Along the way, we'll dig into the deepest recesses of computer interrupts, build a programming language that has angry and happy commands, and use a web-based robot development environment.

The Hull Pixelbot is a little robot with a pixel on the top. **Figure 1** shows the very first one. The author wanted to find out just how much you could do with a robot powered by an Arduino Uno, a very popular (if somewhat limited) microcontroller. Version 1 just moved around the desk. Version 2 had a pixel and distance sensor and became the Hull Pixelbot. At first the robot was controlled by C++ compiled on a PC and loaded into the robot. But then a tiny programming language, 'Python-ish', was developed that runs directly on the robot. The robot can obey Python-ish commands directly at any time – even when running a program. Compiled code can be stored on the robot and executed automatically when the robot is switched on. You can add a network

connection (coming next month) so that you can update the program from anywhere.

The Pixelbot is a great way to upcycle old Arduinos and use them to power fun robots. There are also versions powered by the Raspberry Pi Pico or the Espressif ESP32. You can download the software and find out all about the project at hullpixelbot.com.

ROBOT CIRCUITRY

Figure 2 shows the circuit diagram for the robot. The devices fit into a 400-point breadboard which sits on top of the robot. There are plenty of spare connections for other interfaces you might want to add. Let's start by looking at the various devices on the robot, starting with the one that tells the robot if there is something in front of it.

MEASURING DISTANCE

Robots that crash into things look silly. So, the robot has a distance sensor on the front to detect any wayward brick walls. The HC-SR04 sensor sends out a burst of high-frequency sound and generates a signal pulse when it gets a response. Software in the robot measures the length of the pulse and because we know the speed of sound, it can calculate the distance the robot is from an obstacle. The distance sensor uses two signals. The trigger signal asks the sensor to start a measurement and send a burst of sound. The sensor lifts the echo signal to 5 volts when the sound reflection is detected. We could write a function to trigger the sensor and spin in a loop counting and waiting for the echo signal to go up. This would work, but it would mean that the program would have to stop while distance reading is taken. Sound travels at around 330 metres a second, so if an obstacle is a metre away, it would take around 150th of a second for the reading to be taken. This doesn't sound like much, but the author hates writing code that waits around for a response, so hardware interrupts are used to process the echo signal.

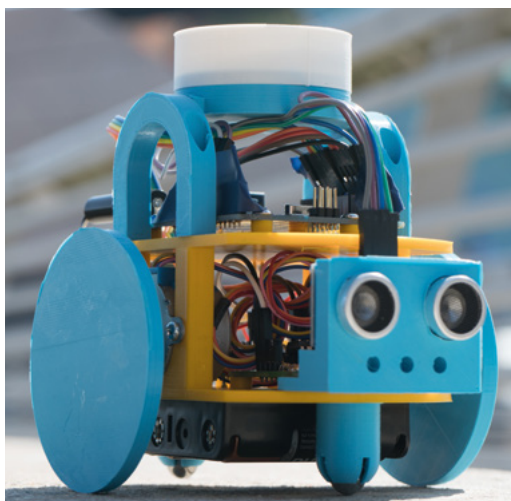


Figure 1 ♦ It's called the Hull Pixelbot because it was created in the city of Hull in the UK

MAKING A DEVICE COME ALIVE

The author was very keen to make the robot seem a little alive. Most robots tend to do only one thing at once – a Hull Pixelbot can animate its pixel, move, and measure distance, all at the same time. Programs can even express emotion. The ‘angry’ command will make the lights in the pixel flash on and off aggressively; the ‘happy’ command makes them fade on and off in a much more relaxed fashion.

IF I MAY INTERRUPT YOU...

The first computers didn’t have interrupts. Programs had to constantly check for changes to their inputs. The interrupt was added in the 1950s. Hardware in the computer processor detects a change in state of an input and diverts program execution to a piece of code called an ‘interrupt handler’ to process the incoming event. When the event has been dealt with, the original program continues. Interrupts are part of how modern computers work. Whenever you press a key, move the mouse, or your computer receives a network message, an interrupt handler takes control to deal with the event. We can use interrupts to allow the robot control program to run while a distance reading is being taken.

“ The ‘angry’ command will make the lights in the pixel flash on and off aggressively ”

```
#define DISTANCE_TRIG_PIN 17
#define DISTANCE_ECHO_PIN 16
```

The code above defines the two symbols which will represent the trigger and echo pin numbers to be used on the Pico. Using symbols makes it easy for us to change the pin numbers if the hardware design changes. We are going to use pins 16 and 17.

```
pinMode(DISTANCE_TRIG_PIN, OUTPUT);
pinMode(DISTANCE_ECHO_PIN, INPUT);
```

The two statements above set up the pins. The trigger will be an output, and the echo an input.

```
attachInterrupt(digitalPinToInterrupt(DISTANCE_ECHO_PIN),
```

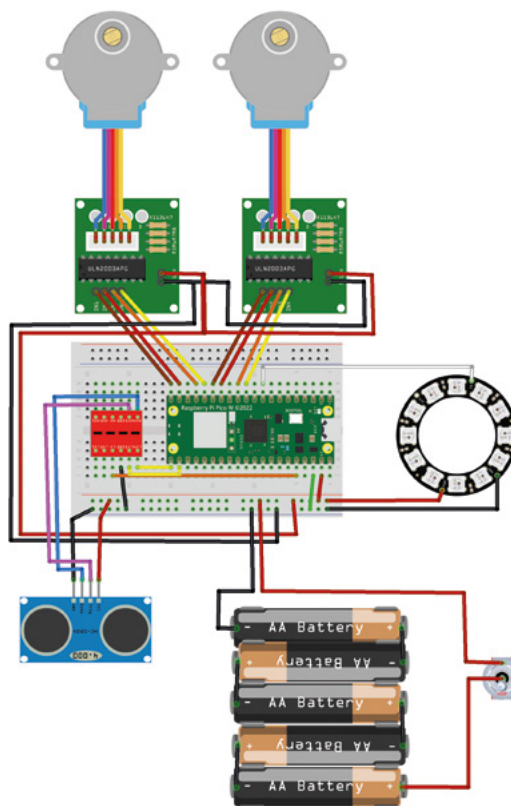


Figure 2 The red component is a level converter which is needed because the distance sensor uses 5-volt signals rather than the 3.3 volts used by the Pico

```
pulseEvent,
CHANGE);
```

This statement attaches an interrupt handler to the echo pin. The `digitalPinToInterrupt` function takes the number of the interrupt pin and converts it into a corresponding number for the specific hardware in use. The handler function is called `pulseEvent` and the interrupt will fire when the pin changes state (either goes from high to low or low to high).

```
void pulseEvent()
{
  if(digitalRead(DISTANCE_ECHO_PIN)) {
    // pulse gone high - record start
    pulseStartTime = micros();
  }
  else
  {
    pulseWidth = micros() - pulseStartTime;
    distanceSensorState = DISTANCE_SENSOR_READING_READY;
  }
}
```

YOU’LL NEED

- ◆ A Raspberry Pi Pico or Pico W. If you want to control the robot over WiFi, use the Pico W version. If you have an old Arduino Uno, you can use that instead, although you’ll have to add an ESP8266 to connect your robot to WiFi.
- ◆ A 12-pixel NeoPixel Ring
- ◆ A 4 * AA battery holder with switch
- ◆ A couple of 28BYJ-48 stepper motors with ULN2003 Driver Board Search for ‘Arduino stepper’
- ◆ An HC-SR04 distance sensor
- ◆ 20 or so plug-to-socket 20cm DuPont cables
- ◆ A 400-point solderless breadboard
- ◆ A level converter chip (search for ‘arduino level shifter’ on your favourite e-commerce site)
- ◆ A USB cable
- ◆ A chassis and some wheels You can make your own or there are 3D-printable or laser-cuttable designs you can download from hullpixelbot.com

QUICK TIP

Not all microcontrollers provide byte-wide input and output. When using the Pico, we have to set each output bit in turn.

The `pulseEvent` function runs when the echo signal changes state. The echo signal goes high at the start of a reading and low when the response has been received. The code above checks to see if the pulse has gone high. If it has, it records the current time in microseconds. If the pulse has gone low, the code determines the width of the pulse by subtracting the start time from the current time in milliseconds. It then sets a flag to indicate that a new distance sensor reading has been obtained. The robot code can check this flag every now and then to see if a new reading has arrived and update the `@distance` value in the Python-ish program running on the robot. This means you can use constructions like the code below which makes the robot pixel turn red and flash in an angry way if you get too close.

```
if @distance < 100:  
    angry  
    red
```

SPINNING THE WHEELS

Stepper motors are used to turn the robot wheels. These are slower than servos or simple electric motors, but they have much greater precision. You can instruct the robot to move 100 millimetres and it will move exactly that distance. The author is happy to trade speed for accuracy. He's found that fast-moving robots tend to crash into things or fall off the table.

Figure 3 represents the inside of the robot stepper motor. The round thing in the middle is the motor shaft, and the block on the shaft is a little magnet. If we turn on one of the four coils, the magnet is attracted to that coil. In **Figure 3**, coil 1 is turned on and the magnet has been pulled towards

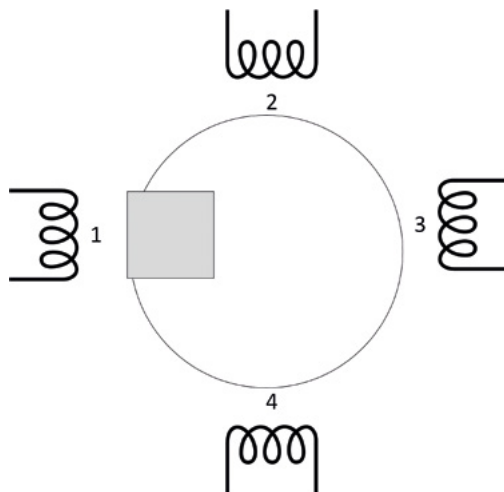


Figure 3 ♦ The coils and magnets are arranged slightly differently in a real motor, but the principle is the same

it. We can make the shaft turn by performing the following:

- Turn on coil 2 (magnet is pulled between coils 1 and 2)
- Turn off coil 1 (magnet is pulled to coil 2)
- Turn on coil 3 (magnet is pulled between coils 2 and 3)
- Turn off coil 2 (magnet is pulled to coil 3)
- Turn on coil 4 (magnet is pulled between coils 3 and 4)
- Turn off coil 3 (magnet is pulled to coil 4)
- Turn on coil 1 (magnet is pulled between coils 4 and 1)
- Turn off coil 4 (magnet is pulled to coil 1)

A program that repeated these changes would make the shaft turn continuously. We can represent the motor coil settings using an array of eight bytes:

// You can instruct the robot to move 100 millimetres and it will move exactly that distance //

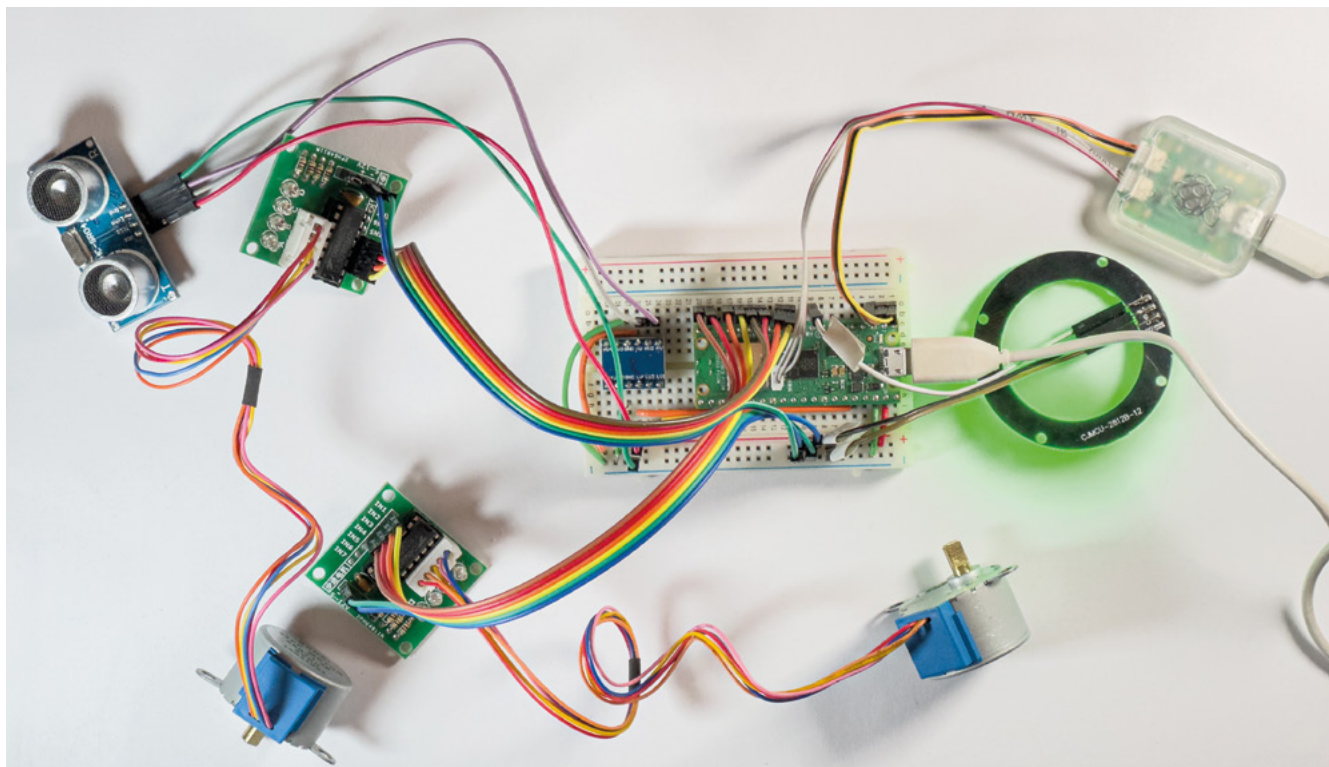
```
const byte rightMotorWaveformLookup[8] = { 0b01000,  
0b01100, 0b00100, 0bB00110, 0b00010, 0b00011,  
0b00001, 0b01001 };
```

If you look at the bit patterns in the binary constants in the code above, you can see that coil 1 is represented by bit `0b01000`, coil 2 by bit `0b00100`, and so on. In the Arduino Uno, these bit patterns can be loaded directly into an output register to turn on the specified outputs.

```
PORTB = (PORTB & 0xF0) +  
rightMotorWaveformLookup [rightMotorWaveformPos];
```

The statement above takes a value from the lookup table and drops it onto the bottom four bits of the `PORTB` output register, setting all four signal bits at the same time. This is much more efficient than setting each output bit individually.

A program can make the right-hand motor turn just by stepping `rightMotorWaveformPos` from 0 to 7, which would access each of the elements in the `rightMotorWaveformLookup` array. The program would need to pause between updates to allow the magnet to move into position. We can control the speed of the motor by changing the delay between each update.



The robot contains two motors, so the above code must be repeated to drive them both. However, we now have a problem. We want the robot to be able to do things while the wheels are turning. This is easy if we are using a DC servo motor that turns when power is applied, but stepper motors require continuously changing signals from the software. If the processor is updating the motor position, it can't do anything else. What we need is a way of generating the above signals without the processor having to wait around. It turns out that there is a way, and it involves using interrupts again.

HARDWARE TIMER TO THE RESCUE


A hardware timer can generate interrupts at a specified interval. Code in a timer interrupt handler can move the motors 'in the background' while the foreground process animates the pixel and runs the program in the robot. The motor driver code calculates the next time that each motor should move to its next step. The robot can move in curves (or the wheels might be of different sizes), so the wheels may be rotating at different speeds. The driver decides which motor needs to be driven first and sets a timer to generate an interrupt at that time. Then, after the interrupt for that step has been performed, it does the same calculation for the next one. The code to make this work reliably was a bit fiddly to write, but the result is motors that move with perfect accuracy without the foreground program doing anything.

```
currentMicros = micros();

if (rightMotorWaveformDelta != 0)
{
    // right is moving - see if it is time to do a step
    rightTimeSinceLastStep =
        ulongDiff(currentMicros, rightTimeOfLastStep);
}
```

The code above shows how it works. It gets the current time and then decides if the right-hand motor needs to move. There is similar code for the left-hand motor. The variable `rightMotorWaveformDelta` is set to 1 if the motor is moving forwards or -1 if it is moving backwards. So, if `rightMotorWaveformDelta` is non-zero, it might be time to move the right motor. The code reads the current time in microseconds and stores it in the `currentMicros` variable. It stores the difference between the current time and the time when the motor was last moved in the variable `rightTimeSinceLastStep`.

The `ulongDiff` function is used to get this value. It compares two integers and allows for them 'wrapping round'. Values wrapping round is a problem in computer programs. Data values are allocated a certain amount of memory space and if they are given a value which won't fit in this space, they will wrap around. The microsecond counter in the Arduino is held in a 32-bit variable which will wrap around back to zero when it reaches 4,294,967,295. This will happen every 71 minutes or so. The `ulongDiff` function calculates the time value allowing for this wrap round. →

Figure 4  The C++ program that implements Python-ish is being debugged using a Pico Debug Probe. This lets us look inside the program and see what it is doing, even when it is running an interrupt handler

QUICK TIP

We can make the stepper motor run backwards by going through the array in reverse order. →

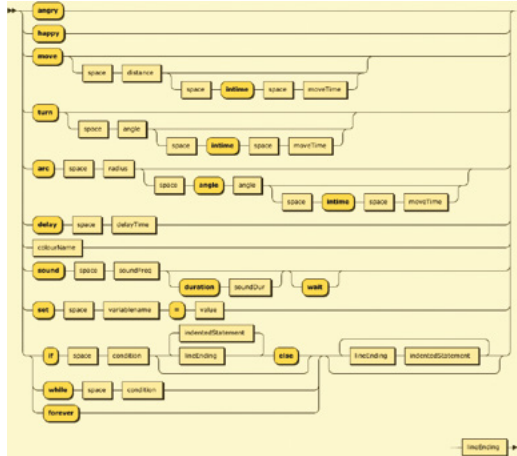


Figure 5 ♦ The diagram is created from a grammar which describes the language structure. All the details are in the language description at hullpixelbot.com

```
void setFlickerUpdateSpeed(byte speed)
{
    // clamp the value
    if (speed > 20)
        speed = 20;
    if (speed < 1)
        speed = 1;

    // set the new flicker update speed
    flickerUpdateSpeed = 21 - speed;
}
```

```
if (rightTimeSinceLastStep >=
rightIntervalBetweenSteps)
{
    rightStep(); // move the motor
    rightTimeOfLastStep = currentMicros -
(rightTimeSinceLastStep -
rightIntervalBetweenSteps);
    rightTimeOfNextStep = currentMicros +
rightIntervalBetweenSteps;
}
}
```

Once we have the `rightTimeSinceLastStep` value, we can use this to decide if it is time to move the motor. If it is time to move, the `rightStep` function is called to move the motor and the time of the next step is set. The same process is performed for the left motor, and then the time of the next interrupt is set depending on whether left or right need to move next. The statement below would run if the right motor must move next:

```
Timer1.setInterval(timeToRight, motorUpdate);
```

The `motorUpdate` function checks to see which motor is to be moved and sends motor pulses as required.

PIXEL POWER

The pixel is the simplest part of the robot. The software contains a set of counters which are used to manage the brightness of the colours that are displayed. At regular intervals the colour values are updated. The speed at which they change is controlled by a variable called `flickerUpdateSpeed`. The smaller the value, the faster the colours change. The function below sets a new value of the flicker speed, making sure that it can't exceed the limits.

```
int flickerUpdateSpeed = 8;
```

BUILDING THE ROBOT

Figure 4, overleaf, shows the control circuit under construction. It was tested on the desktop before being fitted onto the robot chassis.

PYTHON-ISH

You use a programming language to tell a computer what to do. Popular ones include C++, Python, and JavaScript. The robot runs a custom language which was developed to fit into the tiny 24K program, 2K memory, and 1K of permanent storage available on the Arduino Uno. The language looks a bit like Python. It contains special commands that can interact with the robot hardware.

```
# Don't push me!
forever
    d = @distance
    if d < 100:
        angry
        red
    else
        happy
        green
```

The program above can be loaded into the robot and executed. If you are more than 100mm away from the robot, it shows a happy, green persona. However, get too close and it turns red and angry. The `forever` statement repeats a sequence of statements indefinitely. The `@distance` special variable returns the latest reading of the distance sensor and the words `angry`, `happy`, `green`, and `red` do exactly what you would expect. Like Python, the language uses indenting to indicate nesting. All the statements indented under the `if` condition are controlled by it.

The Python-ish compiler converts the text above into a sequence of two-character commands. This

QUICK TIP

There are commands to set the sizes and spacing of the two driving wheels, so you can use whatever sizes you like. You can even make a robot with differently-sized wheels which will still move in a straight line. The size values are retained in the robot. The command to set the sizes is:

```
*MWll,rr,www
```

'l' is the diameter of the left-hand wheel in mm, 'rr' is the diameter of the right-hand wheel, and 'www' is the distance between the wheels.

is achieved by extracting individual words from the program source and then looking the words up in a long string:

```
// command numbers:      0   1   2   3...
const byte commandNames[] = "angry#happy#move#turn..."
```

The `commandNames` string contains all the 'keywords' for the Python-ish language. Each keyword is separated from the next by a # character. There are 111 keywords, including lots of colour names. The program searches through this string for a command and notes the number of the command it finds. These numbers are defined in the program.

```
#define COMMAND_ANGRY 0
#define COMMAND_HAPPY 1
#define COMMAND_MOVE 2
#define COMMAND_TURN 3
```

Now that the compiler knows which command it is dealing with, it can perform a C++ switch to select the code to deal with that command:

```
int processCommand(byte commandNo)
{
    switch (commandNo)
    {
        case COMMAND_ANGRY:// angry
            return compileAngry();

        case COMMAND_HAPPY:// happy
            return compileHappy();
    }
}
```

The compile function for a given language element

processes the language element and generates a piece of program code which will perform the command that was recognised.

```
const byte angryCommand[] PROGMEM = "PF20";

int compileAngry()
{
    #ifdef SCRIPT_DEBUG
        Serial.println(F("Compiling angry: "));
    #endif // SCRIPT_DEBUG

    sendCommand(angryCommand);
    previousStatementStartedBlock = false;
    return ERROR_OK;
}
```

The `compileAngry` function uses the `sendCommand` function to send "PF20" to the sequence of compiled commands. The letter P means 'pixels' and the letter F means 'flickerspeed'. The value that follows sets the flicker speed to 20 so that the lights on the robot will flicker angrily. The `compileHappy` function generates the sequence "PF1" which makes the lights update in a much more chilled manner. Some of the compiled commands generate more complicated command sequences, but the fundamental principle applies to all the code.

Each 'compiled' statement starts with two characters. The first specifies the 'family' of the command (P for pixel, V for variable, C for control →

QUICK TIP

Once the circuit had been tested, the author spread hot glue around all the pins going into the breadboard. This made the robot much more resilient.

Below ♦ There is a link to the online version of the editor at hullpixelbot.com

A BIT SLIPPERY

Python-ish was created to fit inside the tiny processor in the Arduino Uno. You may be wondering if we need to retain it when using the much more powerful and spacious Pico or ESP32 devices. Why not switch to CircuitPython or MicroPython and take advantage of all the features in those languages to tell our robots what to do? If you wish, you can do this, but the Python-ish language does have advantages. The robot can obey lines of code even when it is running another program. This is particularly useful when the robot is controlled remotely. And the programs themselves can execute multiple concurrent processes and bind behaviours directly to language elements. And because the language does less, it is much easier to learn.



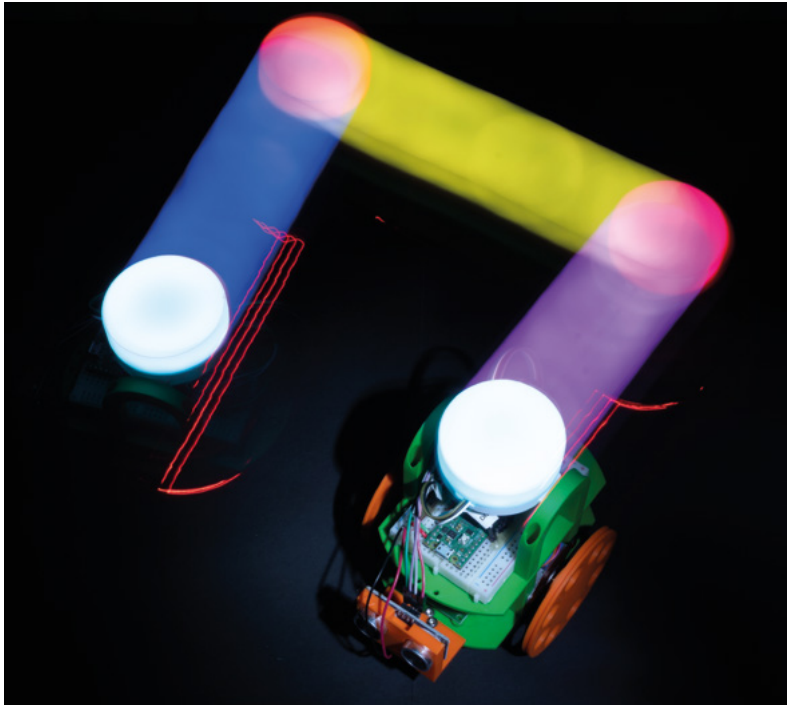


Figure 6 ♦
The red trails are from the stepper motor LEDs

etc.) and the second character gives the command itself (F for flickerspeed in PF or S for variable set in VS). The author could have used numbers for each compiled command, but he wanted to be able to understand the compiled code. The code below is the compiled output for the Python-ish code above. The command CL creates a label, the command CJ performs a jump to a label, and CF performs a jump if the following condition is false.

```
CL11
VSd=@distance
CFd<100,13
PF20
PNr
CJ14
CL13
PF1
PNg
CL14
CJ11
CL12
```

If you look carefully at the statements (and draw some lines connecting jumps to their destinations), you can start to see how the high-level language of Python-ish has been converted into a sequence of low-level steps. When the program runs, it uses more

switch statements to pick the code to perform each command. Note that at no point does the program actually create any Arduino Uno machine code – the run time system is an ‘interpreter’ which just takes a set of instructions and does what they say.

RAILROADS AND LANGUAGE DESIGN

The final part of the language design is the description of the language syntax. The author could have written a bunch of text describing how you can write the programs, but instead he has created some ‘railroad diagrams’. **Figure 5** shows the diagram describing what you can use in a statement. You can use this to work out what you can and can’t do in the language. For example, it shows that a **move** statement can optionally be followed by a space (one or more spaces) and a **distance** value. However, a **delay** statement must be followed by a space and a **delaytime** value. Making these diagrams is quite fun (at least for the author). You can find the Railroad Diagram generator at bottlecaps.de/rr/ui.



The robot is fun to play with, but it is inconvenient to have to keep connecting it to a computer



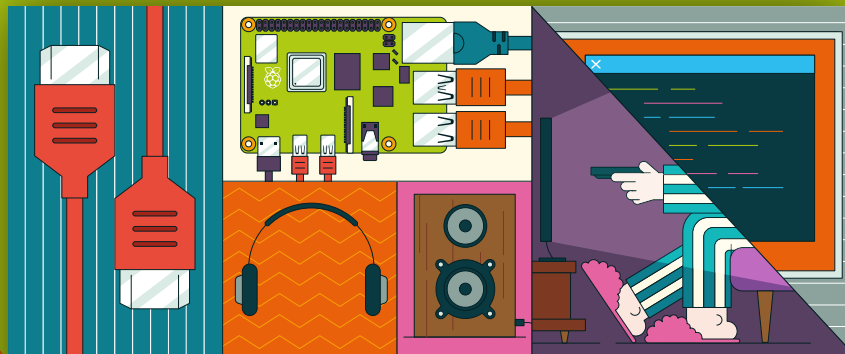
EDITING PYTHON-ISH PROGRAMS

You can edit programs in your robot by plugging it into a serial port on your computer and then opening the program edit page in your browser. You can send your programs into the robot and run and stop them. The editor also has some built in sample programs. The Clear button removes the program from the robot and the View button displays the low-level code in the robot. You can find the editor online at hullpixelbot.com/Python-ish.

FURTHER DEVELOPMENT

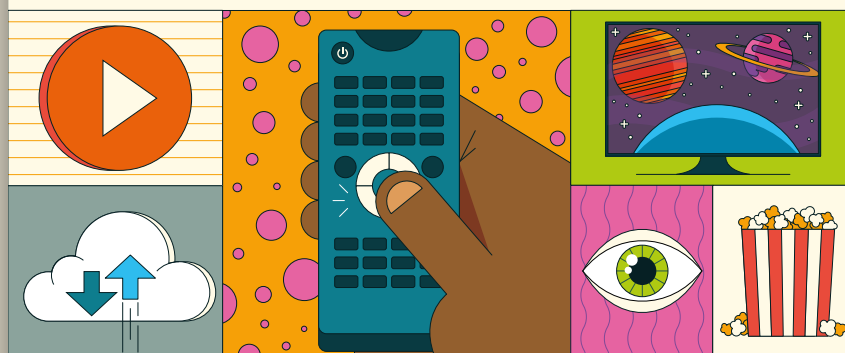
Figure 6 shows a robot on the move. The robot is fun to play with, but it is inconvenient to have to keep connecting it to a computer to program it. In the next article in this series, we’ll discover how you can add a network connection to your robot and enter programs remotely. This makes all kinds of fun things possible, including ‘robot rugby’ where players program their Pixelbots to take on their opponents. □

Your FREE guide to making a smart TV



BUILD A RASPBERRY PI MEDIA PLAYER

Power up your TV and music system



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

magpi.cc/mediaplayer

More blinkies for your rucksack with NeoPXL8

Try to stay alive over winter



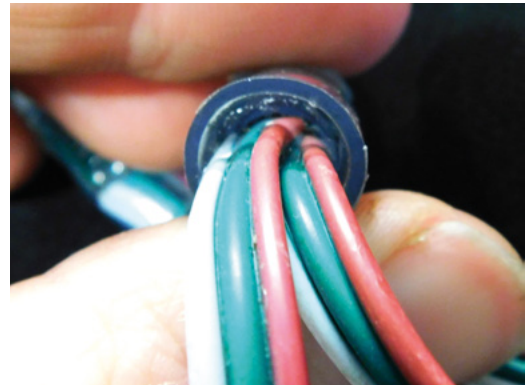
Ben Everard

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.

This writer cycles to the workshop every day. At around the start of November every year it takes a dramatic turn for the dark. Gone are the glorious summer days of cycling home while the sun is still high in the sky. Gone, too, are the glorious early autumn days of the sun gently setting and lighting up the sky in hues of orange and pink. Now it's time for the dark winter days, and if there are going to be some glorious lights, we have to provide them ourselves. And provide them we will.

Regular readers will be familiar with our yearly quest to bring some blinkenlights to the Bristol to Bath cycle path. Here's the basic problem we're trying to solve. Bike lights are great. They let cyclists be seen in the dark. However, a single point of light isn't a great indicator of position or speed, and it requires drivers to use a degree of awareness that seems beyond some of the denizens of the road.

Bike light manufacturers have attempted to solve this problem in a few ways. Brighter lights help up to a point – beyond that just become dazzling (a particular problem on a dark bike path). Flashing lights help a little. Some lights are a little larger, and these can help significantly. However, there's



Above ♦

The electronics for each LED come encapsulated in resin. These ones have already survived being on the rucksack for a year of commuting and are still going strong

a limit to how big you can make a bike light. So, rather than attempting to enlarge a light, we've attempted to illuminate our largest item, this author's cycling rucksack.

For the last couple of years, we've had lights on the back of the rucksack. Originally, these were a simple string of battery-powered fairy lights from the sadly defunct Wilkos. Then we upgraded these with a strip of WS2811 lights (we preferred the form factor of these waterproof string lights to anything available using the smaller WS2812B LEDs). This helps us remain visible from behind, but we've never been too happy with the visibility from the front. We've tried LEDs attached to the bike, but this has always been unsatisfactory. They get knocked about more than LEDs on the rucksack, and they've never lasted long.

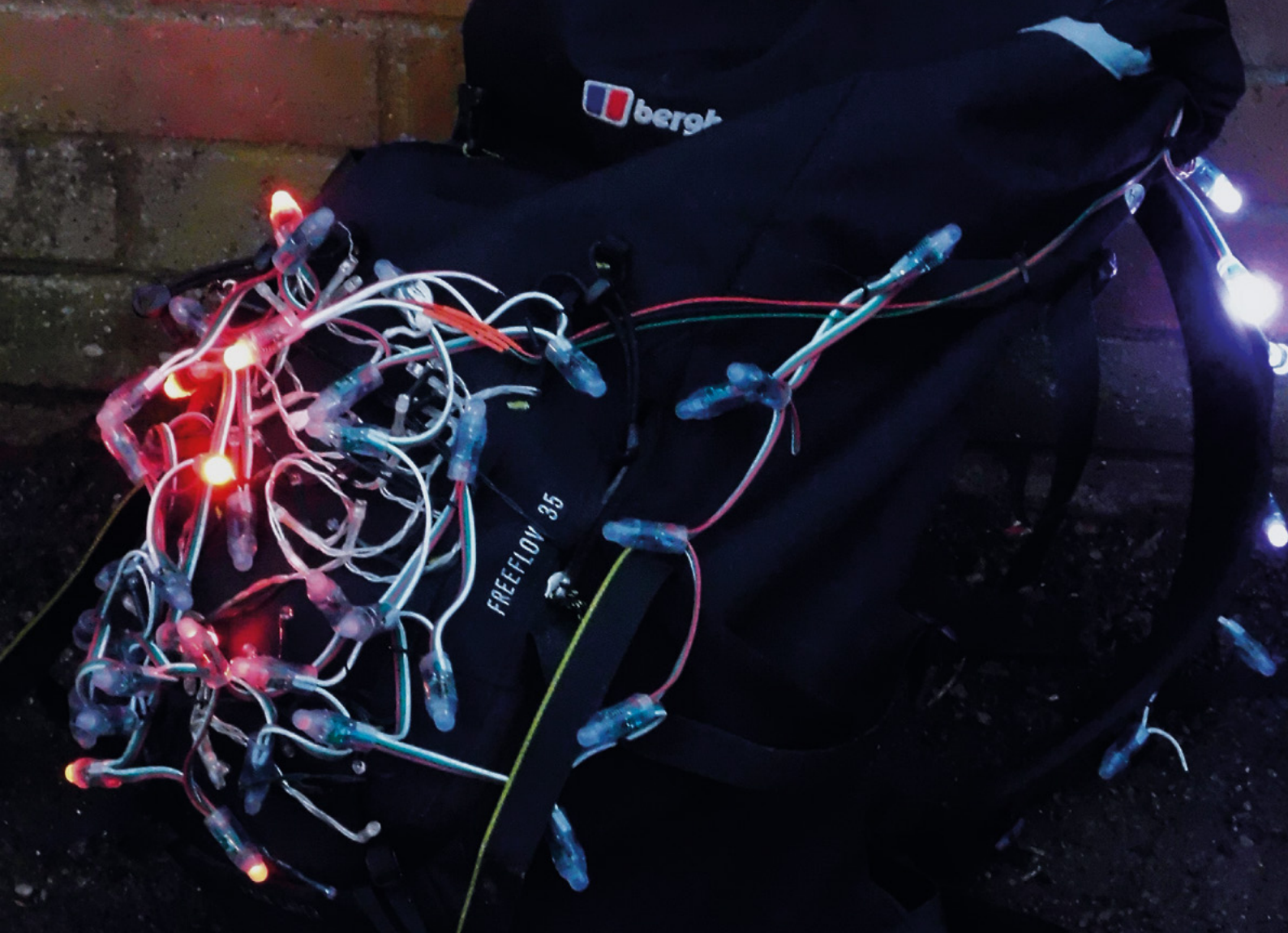
This year, we wanted to up our visibility by attaching LEDs to the front as well, and the only front-facing part of the rucksack – the shoulder straps.

Attaching the LEDs is only half the battle, though. We also need to control them, and fortunately, there



Right ♦

These LEDs include an external WS2811 chip and an RGB LED. WS2812B LEDs have the chip and LED in the same package. Both packages use the same communication protocol



Above ♦ Unfortunately, we can't capture the motion of the lights in a still image, but we're really pleased with how this looks in use

are a couple of things that have made it easier to run multiple animations from a single microcontroller.

The first is NeoPXL8, a library that outputs data to up to eight strings of LEDs simultaneously, and LED animations, which is a module that does exactly what it says. Both of these have been developed by Adafruit and work with CircuitPython on Raspberry Pi Pico.

The first step in getting these modules to work is to download both CircuitPython for Pico and the Library Bundle from circuitpython.org. You can flash CircuitPython to Pico by holding down the BOOTSEL button and plugging it into your computer via USB. You can release the BOOTSEL button once it's plugged in. There should be a new USB drive that appears – drag and drop the UF2 file for CircuitPython that you've just downloaded. Once this has been copied over, the USB device should disappear and a new one called CircuitPython will appear.

Unzip the CircuitPython Library Bundle, and you need to copy across `adafruit_neopxl8` and `adafruit_led_animations` into the `lib` folder on the CircuitPython USB device. The Pico device should now be ready for our code.

There is a selection of animations available (and you can code your own). We selected Comet, which is pretty similar to a Larson scanner, and used the

GETTING STARTED WITH CIRCUITPYTHON

If you've not used CircuitPython before, don't worry, it's all quite straightforward. We would, however, recommend that you first take a look at the getting started guide: hsmag.cc/AdafruitCP.

If you're an experienced programmer, the main thing you need to know is that CircuitPython is sensitive to which text editor you use. This might sound a little odd, but it has to do with the way it saves files to the device.

Mu is a great choice, and you can download it from codewith.mu.

traditional red for the back and white for the front. It gives movement and light that we think is quite eye-catching, but not dazzling or distracting. We dislike lights that turn on and off sharply – we much prefer the softer introduction on this setup.

The code for all this is:

```
import board
import rainbowio
import adafruit_ticks
```

TUTORIAL



Above ♦ There wasn't a good mounting point for Pico, so it's strapped on with Velcro and a cable tie

```
from adafruit_led_animation.animation.comet import Comet
from adafruit_led_animation.color import GREEN, WHITE
from adafruit_led_animation.group import AnimationGroup
from adafruit_led_animation.helper import PixelMap
from adafruit_neopxl8 import NeoPxl8

# Customize for your strands here
num_strands = 3
strand_length = 38
front_len = 6
first_led_pin = board.GP2

num_pixels = num_strands * strand_length

# Make the object to control the pixels
pixels = NeoPxl8(
    first_led_pin,
    num_pixels,
    num_strands=num_strands,
    auto_write=False,
    brightness=0.50,
)
```

OTHER HARDWARE CHOICES

Pico is a great bit of hardware for this, but it's not the only option. Any board with an RP2040 processor and CircuitPython support should work. Adafruit has a board that's purpose-built for controlling lots of LEDs: the Scorpio. This simplifies the wiring a bit and includes a LiPo battery charger if you don't want to use a USB battery pack.

```
def strand(n):
    return PixelMap(
        pixels,
        range(n * strand_length, (n + 1) * strand_
length),
        individual_pixels=True,
    )

strands = [PixelMap(
    pixels,
    range(0,37),
    individual_pixels=True),
    PixelMap(
    pixels,
    range(38,44),
    individual_pixels=True),
    PixelMap(
    pixels,
    range(76,83),
    individual_pixels=True)
]

# For each strand, create a comet animation of a
different color
animations = [
    Comet(strands[0], speed=0.02, color=RED, tail_
length=10, bounce=True),
    Comet(strands[1], speed=0.05, color=WHITE, tail_
length=3, bounce=True),
    Comet(strands[2], speed=0.05, color=WHITE, tail_
length=3, bounce=True),
]

# Advance the animations by varying amounts so that
they become staggered
for i, animation in enumerate(animations):
    animation._tail_start = 30 * 5 * i // 8 #
pylint: disable=protected-access

# Group them so we can run them all at once
animations = AnimationGroup(*animations)

# Run the animations and report on the speed in frame
per second
t0 = adafruit_ticks.ticks_ms()
frame_count = 0
while True:
    animations.animate()
    frame_count += 1
    t1 = adafruit_ticks.ticks_ms()
    dt = adafruit_ticks.ticks_diff(t1, t0)
    if dt > 1000:
        print(f"{frame_count * 1000/dt:.1f}fps")
```



```
t0 = t1
frame_count = 0
```

This is almost the example code for NeoPXL8 (Thanks, Jeff Epler), but with a few changes to make it work how we want.

The `pixels` object is very similar to one created with the usual NeoPixel library. It's a one-dimensional list with the first strip taking up places 0–37 (in our case), then the second 38–75, etc. The slight confusion here is that our strips aren't all the same length. We have a single strip of 38 pixels, then two strips of six pixels. The NeoPXL8 library can't handle this, so it sets every strip up as 38 pixels and we have 32 'phantom' pixels in the second and third strips. This doesn't cause us any problems – we just have to ignore them.

We don't want to address all our pixels as one list, and we want to interact with the three strips individually. Fortunately, there's the `PixelMap` object for this. We can use this to divide up our `pixels` object into the three strips (and we can also discard the phantom pixels). This is what the `strands` list is – a list of three objects, each of which is an LED strip.

That's the setup done; let's now start with the animations. The LED Animations module contains a series of different animations (as well as a framework you can use for building your own animations). In our case, we've used three Comet animations.

The three animations can be bundled together into an animations group. This makes it a bit easier to update and run them. Finally, we just need to repeatedly call the `animate` method of the group repeatedly, and this will run our animations.

LET'S BUILD

That's the code. Now let's take a look at the physical setup.

Like many rucksacks aimed at people who do outdoor activities, this one comes with a wide range of superfluous bits sewn into it. We're not sure exactly how the good people of Berghaus intend people to use the ice axe loops that are too small for any ice axe we've ever seen, but they are handy for attaching LEDs to. We cable-tied the LED chain to every extraneous bit we could see. It's in a bit of a jumble – this is intended as it gives more of a 'cluster of light' effect than a more mathematical layout might.

As mentioned, there are three separate strips, each of which is wired to a different pin. We have to solder them all up. You can use whatever GPIO pins you like, but they must be consecutive (we used 2, 3, and 4). Pico has enough ground pins, so we don't have to do any creative wiring there, but we do have to connect

LED STRIPS

We really like the fairy light effect you get with this particular type of WS2811 LED; however, if you want to create a different effect, there are other form factors of LED you can use. WS2812B LEDs typically come on strips, and these are sometimes covered in waterproof silicone. You could use these, but you will need to seal the end of the strip if you intend to use it in bad weather.

There are a few types of LED designed to be sewn onto fabric. These are typically single-colour LEDs, and they can work with conductive thread. You could use these to create a similar effect, though there could be quite a bit of sewing.

GlowStitch (hsmag.cc/glowstitch) has machine-sewable LEDs that may simplify things if you want to go down the route of using individual LEDs, though we've not tested these.

the VIN pins of each strip together and then to the single VBUS pin. We joined all three wires together with a big drop of solder and a lot of electrical tape (we don't have heat-shrink the right size for this).

The main thing you need to think about with construction is strain relief and minimising the likelihood of tangles. We want this to last a long time, and it's going to get knocked about as bags tend to do. The main thing we've used is cable ties. A small cable tie through the mounting hole on Pico can hold wires in place and stop them from flexing at the soldered joint (which is the most likely place to fail). Cable-tying wires together and in place as much as possible will help keep things tidy (and therefore safe and working).

It remains to be seen if your setup will prove robust enough. There's definitely more bending on the new front wires than there has been on the back. That's just an unavoidable part of where they're placed. If it breaks, we'll fix it and introduce a bit more robustness then.

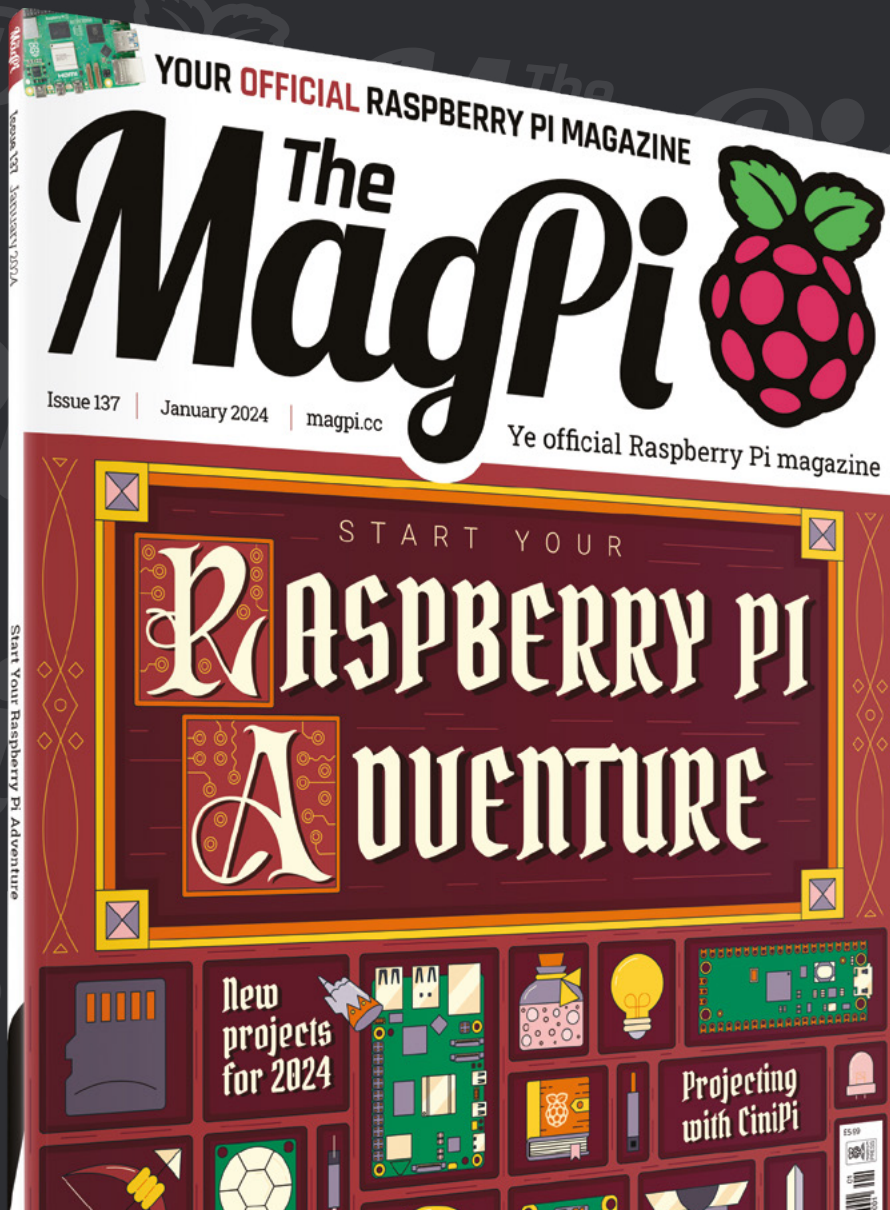
Finally, we need a power supply, and for this, we just use a USB power bank in the rucksack.

We're happy with the new setup, and it seems the good people of the West of England are, too. We get regular comments from fellow cyclists on our look. □



Left ♦
A cable tie through a mounting hole isn't the best form of strain relief, but it's much better than no strain relief

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.



*While stocks last

magpi.cc/subscribe

FIELD TEST

HACK | MAKE | BUILD | CREATE

Hacker gear poked, prodded, taken apart, and investigated



PG
92

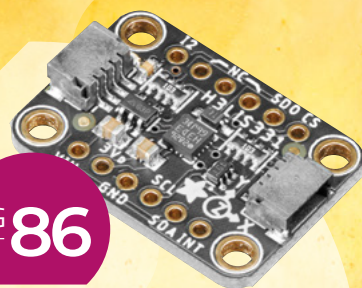
FISHY FILAMENTS

Can you print with fishing nets?

PG
96

CROWDFUNDING

Cast your eye over a Raspberry Pi robot and a screen printer



PG
86

BEST OF BREED

We take to the skies with this rocketry hardware

ONLY THE
BEST

Blast-off! A collection of rocketry-related electronics

Cool kits for aspiring astronauts

By Marc de Vinck

 @devinck

I must admit, I have not been an active member of the model rocketry scene for a very long time. Not to say that I haven't launched my fair share of hobby store rockets in my life, because I have. But it's been a very long time since I've counted down and screamed LIFT-OFF! Maybe too long?

Recently, with all the advances in full-size rocket research, it has sparked my interest in model rocketry once again, and I'm sure a lot of other people, too. I've seen a lot of amazing projects from university engineering programs and even a lot of backyard builders. It's incredible to see some of these amateur-built rockets take off. And even more incredible are the ones that can land just like the full-size rockets on their folding tripods.

The days of simply hooking up a couple of D cell batteries to a plastic launchpad are over for me. It's been replaced by the dream of being able to build a rocket that could potentially land on its own, in a predetermined location, that is really intriguing. So, down the rabbit hole I went, looking for the components that would be required to build a smart rocket. How smart is still to be determined, but I want a little more than just pressing the button and yelling LIFT-OFF!



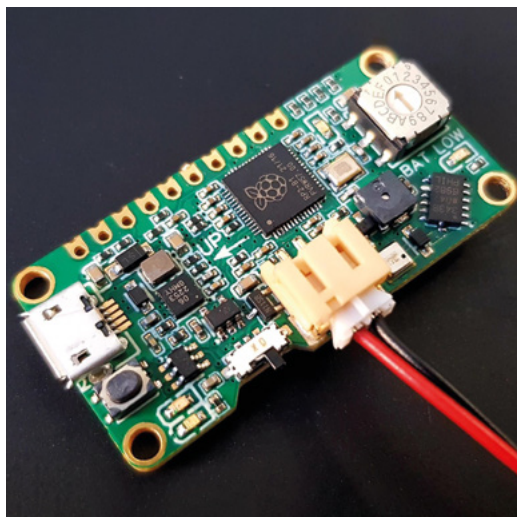
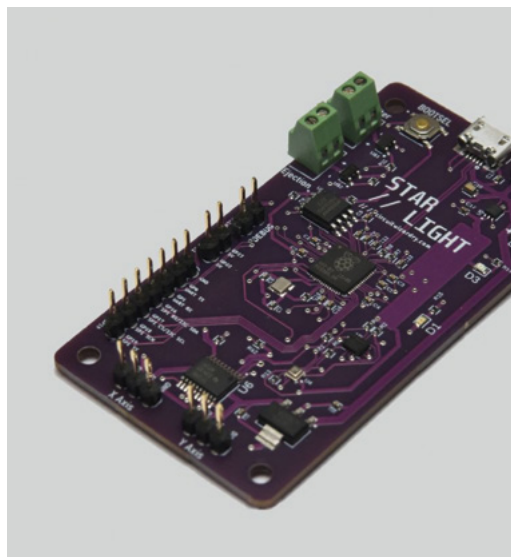
STARLIGHT Model Rocket Flight Computer vs Rokit flight computer

TINDIE ◆ \$49.99 | tindie.com

TINDIE ◆ \$48.99 | tindie.com

The Starlight all-in-one rocket flight control is a great choice for anyone launching 75 mm rockets. The board is packed with sensors and functionality, so you can launch more rockets with higher success rates and gather lots of critical data. And best of all, the board is controlled by our favourite microcontroller, the RP2040.

The Starlight also features a six-axis IMU with gyroscope and accelerometer, a level shifter so you can control 5V servos, dual temperature sensing, a pressure sensor for altitude determination, and six GPIO pins. The board includes 16MB of flash storage not only for the firmware, but also for capturing flight data. There are indicator LEDs, a micro USB port, and dual voltage regulators. To really understand how this board works, check out the tutorials and the GitHub repo. It explains in detail specifics like the igniter pin, ejection pin, battery, and servos usage.



The Rokit (rocket operation computing kit) is an affordable flight computer designed for small model rockets. You

can even use the board with water rockets since it's so small and lightweight!

The board features an RP2040 dual-core microcontroller, altimeter, accelerometer, and data logging via a microSD card. You also have power management for controlling two servo motors. And all that weighs in at a mere 5 grams.

The board measures 44 × 22 × 9 mm. I typically don't give dimensions of products in these reviews, but when you're talking rockets, size matters! The board comes fully assembled and tested and includes header pins if you want to solder them on for added functionality, but that's optional. Just add a microSD card and AAA battery pack with a JST connector and get to coding.

VERDICT

STARLIGHT Model Rocket Flight Computer

A great all-round solution.

10/10

Rokit flight computer

Small, yet powerful.

9/10

BEST OF BREED



Burn Baby Burn, A Nichrome Cutter

TINDIE ♦ \$9.50 | tindie.com

Then I was looking for a topic for this month's Best of Breed, this was the first product I came across. It's what led me down the path of DIY rocketry. The Burn Baby Burn is a specialised little bit of electronics that allows you to burn through rubber bands or fishing line, releasing the parachute of your rocket after a successful take-off. The board is designed to run off a

single-cell LiPo battery. It draws about 1.2 amps, which is all you'll need to cut a typical rubber band or fishing line. And you'll only need that power for a few milliseconds to generate enough heat. Although this seems like an easy piece of electronics to figure out, the creator provides some example Python code for a time-released parachute deployment. And since it's open source, you can check out the website for the source code and design files, too!

VERDICT

Burn Baby Burn, A Nichrome Cutter

A simple solution for your rocket.

9/10


POCKET REF – 4TH EDITION BY THOMAS J. GLOVER

ADAFRUIT ♦ \$12.95 | adafruit.com

If you are building rockets, or building anything really, and you haven't heard of the pocket reference by Thomas J. Glover, then you're in for a real treat. This book, featuring 864 pages of tables, formulas, conversions, and just about everything you've ever wanted to know about building things, is the perfect addition to anyone's lab. And even though many of your questions could be answered by visiting the internet, opening this book to a random page and learning something new is always a lot of fun!




ILLUMINATED TOGGLE SWITCH WITH COVER – RED

ADAFRUIT  \$3.95 | adafruit.com

How can I have a rocket round-up without including at least one illuminated toggle switch? It's the classic: flip the cover, then flip the switch, while screaming 'blast off'! And although I typically associate these with missiles being launched, they can also be used for so many other projects. You don't have to be launching rockets to add this to your next build. It's a great safety device, but I also like them simply for the aesthetics.



Adafruit H3LIS331 Ultra High Range Triple-Axis Accelerometer

ADAFRUIT  \$24.95 | adafruit.com

It's not too difficult to find an accelerometer that can measure 5 or 10 Gs, but what if you needed more Gs... a lot more! Then the Adafruit H3LIS331 Ultra

High Range Triple-Axis Accelerometer has you covered. This board is capable of measuring +/- 400Gs! That's a lot more than a human could handle, but in model rocketry, it's just

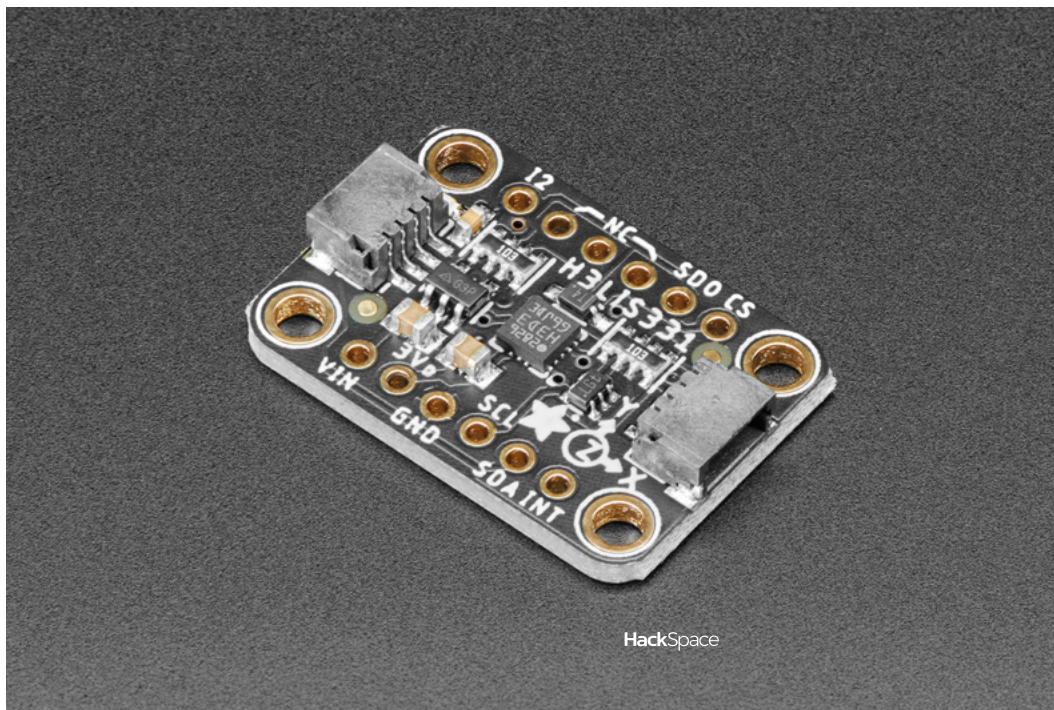
the range you'll need. OK, it's still a lot more than you may need, but you typically can't use a standard G meter because they're usually calibrated for ranges that are for humans. If you need high-range G force measurements, this STEMMA QT format board is for you! Head on over to the website to learn more about its configurable high-pass and low-pass filters, adjustable data rates, and more.

VERDICT

Adafruit
H3LIS331 Ultra
High Range
Triple-Axis
Accelerometer

Perfect for anyone using STEMMA QT-compatible boards.

9/10

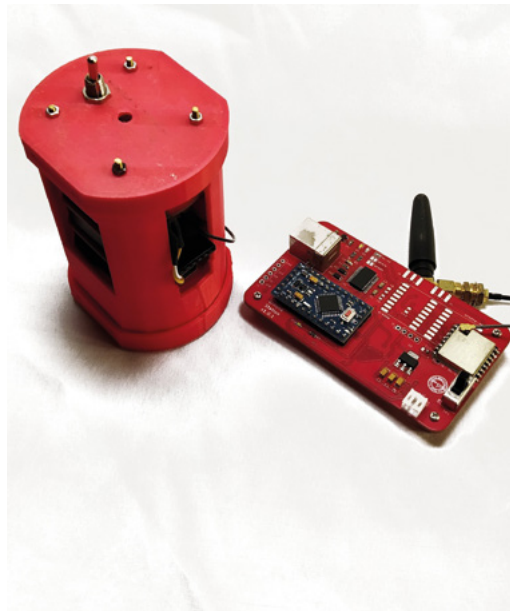


EDUCAN Educational Cansat Kit DIY

TINDIE [\\$150](#) | [tindie.com](#)

The EDUCAN Educational Cansat Kit is an interesting kit for anyone wanting to learn more about rocketry and satellites. Sadly, you can't launch this kit into space, but you can learn a lot about how launching a small soda-can-sized satellite into orbit is possible. The kit is designed for high-school students to help them learn how satellites are built, and how they communicate to ground stations.

Even though you won't be launching this into outer space, it doesn't mean you can't launch it into the sky. The creators suggest using a balloon to do some high-altitude drop tests, or launch it from a small rocket and learn how satellites communicate to ground stations, deploy parachutes, and more. It's an interesting concept, and I can see a lot of value in learning about how to program an Arduino, use LoRa for communications, and generally have some fun. Check out the website for more information, including documentation and source code.



VERDICT

EDUCAN Educational Cansat Kit DIY

Good for learning concepts.

7/10

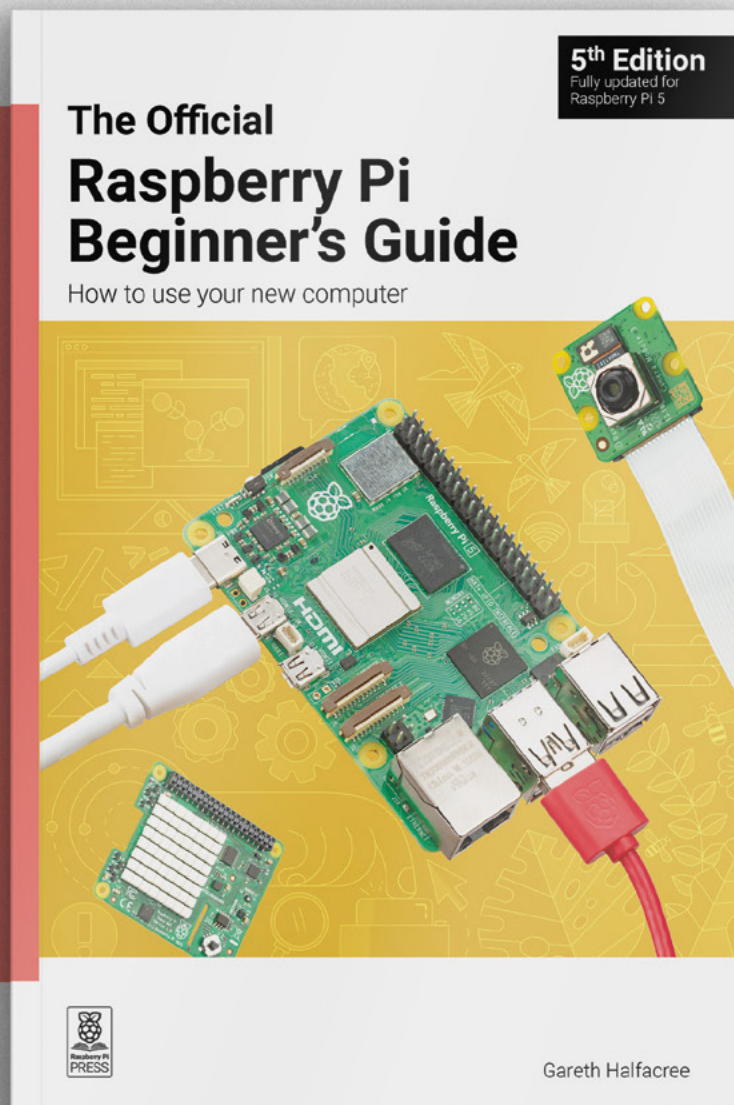
SHORT KIT, ROCKET FIN JIG

TINDIE [\\$10](#) | [tindie.com](#)

I have built a few rockets as a kid, and I know the pain of perfectly aligning the fins to the body of your rocket. And if the fins are off by just a small amount, the trajectory of your rocket can be way off, or even catastrophically off. This handy little jig allows you to line up the body of your rocket with the fins that you are attaching with precision and reliability. You only need to build a few rockets to realise what a handy little addition this rocket fin jig is to your workbench.



- Learn coding ■
- Discover how computers work ■
- Build amazing things! ■



magpi.cc/beginnersguide

Porthcurno recycled nylon printer filament

Do you worry that 3D printing is too easy?

FISHY FILAMENTS ♦ €54.98 / 750G | hsmag.cc/fishyfil

By Ben Everard

Nylon is one of those filaments that usually get lumped under the term 'engineering'. We're not completely sure what makes a filament 'engineering', but we suspect it's a euphemism for expensive and bloody

hard to print.

This particular nylon filament, though, is explicitly not for engineering (Fishy Filaments does an engineering version called Orca which contains carbon fibre). Porthcurno is, according to the website, designed for homewares, wearables, fashion, and other aesthetic products.

Before we dive into this filament in depth, let's take a step back through the life of the plastic and see where it comes from.

Fishy Filaments works with the Cornish Fish Producers' Organisation (CFPO), and takes old and damaged nets that have mostly been used for

catching hake. These nets are made of a type of nylon called PA6. They're made to a high standard to withstand the rigours of working at sea.

These old nets are taken, processed, and turned into 3D printer filament. Porthcurno retains the aquamarine colour of the original nets.

ATTEMPTING TO PRINT

Nylon isn't an easy polymer to print at the best of times, and most nylon filaments have an additive that makes them a little more cooperative. Usually this is carbon fibre, but other options are available. Porthcurno has none of this because it would detract from the colour and texture of the pure plastic. Nylon does have to be printed quite hot – around 285 for the nozzle and 100 for the bed – this is within the reach of many printers these days.

The first problem is getting it to stick to a print bed. We were able to get some prints to stick to glue-covered PEI, but only if they had a large, even base. Any protrusions on the base were prone to peeling up, causing the whole lot to come off. Specialist nylon (also known as PA) print beds are available and may stop this.

The second problem is that nylon is extremely hygroscopic. This means that it sucks up moisture from the atmosphere. You can minimise this by keeping it in a sealed container with a desiccant (such as silica gel), but this won't stop it. If there's moisture in the filament, this will boil as the filament is heated, causing bubbles. These bubbles cause a couple of problems. They build up pressure in the print head, causing plastic to squirt out even when it shouldn't, and they cause little voids in your prints.

Below ♦
Before drying (left) you can see bubbles in the print, while afterwards (right) the layers come out smooth





Above ◆ Blobs of nylon can stick to the nozzle, burn, and then detach onto the print leaving scorch marks

Another concern is the particulate matter given off by this (and other nylons) as they print. It's a bit hard to know how much there is as it's invisible and odourless, but it's recommended that you don't print this without an enclosure or extraction system. This is something that we'd recommend you take seriously, especially if you plan on printing nylon more than occasionally.

That's quite a long list of problems you can have with this filament. You could only justify such a tricky filament if there were significant benefits.

“ You could only justify such a tricky filament if there were significant benefits ”

We'll be honest, the only things we were able to print reliably on our setup (a Prusa MK4 with a filament dryer) were spiral vases. The lack of retractions meant that we didn't get hit by the problems with blobbing that non-spiral prints gave us, and the roughly circular bases stuck well to glue stick-covered PEI. The only problem we had with these prints was occasional blobs of burned filament coming off the nozzle and damaging the print. It is recommended to use a non-stick nozzle for this, and we suspect that this would cure the problem. This also seems to be related to how wet the filament is, and when we got the filament completely dry, this didn't happen much.

It is a bit of a problem that we were only able to reliably print spiral vases, but it does produce excellent spiral vases. The translucent blue-green shimmers in the light, and the semi-flexible nylon



Above ◆ If you get everything working, you can create great-looking prints with Porthurno

creates a durable model that feels good in your hands. To print more than this, you'd need a better filament drying setup (which probably means a dryer that can get hotter), and a print surface designed specifically for nylon (there is one for the Prusa printers, but it's out of stock and we've been unable to test it with this filament). We can't say how well it would print under these scenarios.

We love this filament. It looks great, it's genuinely good for the planet, and it can make parts that are really durable. It's a filament that tells a story – it's not just recycled plastic; it's recycled trawler nets, and its translucent green-blue colour looks like recycled trawler nets. This vivid image helps us understand our place in the circular economy and think about what it means to be responsible users of plastic.

We also hate this filament. It's an absolute sod to print. You have to really want to print with Porthurno for it to make the effort worthwhile. To be honest, we do really want to print with it. While it does look and feel great, there's something a bit deeper to the story that made us keep persevering when we might otherwise have given up.

There is something prophetic about the difficulty in printing this filament. It's almost like it was purpose-designed by an artist to remind us to be mindful of our plastic use because once it's one thing, it's not always easy to make it into something else. Maybe we're just suckers for a material that tells a story. □

VERDICT

If you can get this to print, it looks great and makes durable objects.

9/10

CODE
THE
CLASSICS
VOLUME 1

CODE THE CLASSICS VOLUME 1



Brimble
Crookes
Gillett
Malone
Tracey
Upton?



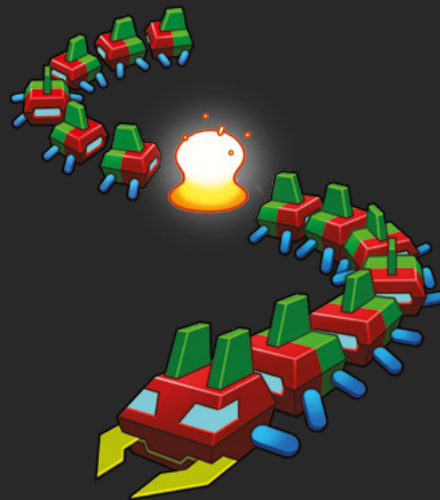


CODE THE CLASSICS VOLUME 1

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



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



Available now hsmag.cc/store

CROWDFUNDING NOW

SaraKIT

Expand a Raspberry Pi Compute Module

From \$99 | hsmag.cc/sarakit | Delivery: Feb 2024

The SaraKIT is an expansion for the Raspberry Pi Compute Module that breaks out some of the features into a circular PCB, 9 cm in diameter. For \$99 (not including the Compute Module), you get three brushless motor drivers, three microphones with direction sensing, an amplifier, a temperature sensor, an accelerometer, and a gyroscope.

The crowdfunding campaign also promises ‘tons of tutorials’, but the link to the documentation returns a ‘404 error’ at the time of writing.

While this looks like an interesting product, we’re having a bit of difficulty working out what it’s for. The two headline features are the three powerful motor drivers and the directional microphone array. There are probably projects out there that need both of these, and if you have one of them in mind, then this could be a good fit. If you only need one of the two, you’ll have more options, and more flexibility by looking for a product that does what you need rather than this more general-purpose device.

Many of the other features are simply the features of the Raspberry Pi 4 Compute Module on which this is based. If the promised tutorials turn up, this could be a good base for learning, but there’s lots of material available for the Raspberry Pi. It would be something special if it manages to eclipse this. □



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.

xTool Screen Printer

Create T-Shirts with your laser cutter

From \$199 | hsmag.cc/xtoolsp | Delivery: Dec 2023

Screen printing is a technique with a long pedigree for small-scale artisans. With a relatively small amount of equipment, you can replicate a design quickly on a variety of surfaces. It's probably best known for making T-shirts, but works on quite a wide range of surfaces.

The biggest drawback is the time it takes to make the screens. These are commonly made by covering a silkscreen with a UV curing paint, exposing part of it, then washing the rest away, leaving gaps where you want paint applied. You can then squeegee paint through this screen and onto your surface of choice.

With this setup by xTool, you can use a laser cutter to create the silkscreen, which is much quicker and easier than other methods.

The crowdfunding campaign makes wild claims about the artistic merits of screen-printing. Yes, Andy Warhol's screen prints sold for an awful lot of money, but it's unlikely that yours will. And we know a few printmakers who would argue with the claim that screen printing is the most artistic printing method.

However, even without such wild claims, screen printing is an interesting technique that's fun to play with, and – if this setup works as intended – would be a great addition to a space that has a laser cutter. ▣



next month

issue

#75

ON SALE
25 JANUARY

MAKE YOUR OWN

GADGETS

ALSO

- RASPBERRY PI 5
- 3D PRINTING
- PYTHON
- LEDS

AND MUCH MORE

DON'T MISS OUT

hsmag.cc/subscribe



3D Scans

Pan troglodytes, also known as the chimpanzee, is a species of great ape native to Africa. Chimpanzees prefer fruit as a source of food, but will also eat leaves, seeds, and bark, as well as non-plant foods such as insects, birds, eggs, and small-to-medium-sized mammals.

The chimpanzee is on the IUCN Red List as an endangered species, and around 200,000 individuals exist in the wild. No chimpanzees were harmed in the making of this magazine.

They don't traditionally wear glasses, but this 3D-printed skull does make an excellent glasses holder. This particular model comes from the collection at the Natural History Museum in London.

PiKVM

Remote control **redefined**

Manage your servers or PCs remotely!



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



HiPi.io

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

List of official resellers by country:

