MAKE | BUILD | HACK | CREATE

HackSpace

TECHNOLOGY IN YOUR HANDS

hsmag.cc | October 2023 | Issue #71

Eben Upton presents Raspberry Pi 5

$25 million / 8 years / 1 power switch

Oct 2023
Issue #71 £6

KICAD BLUETOOTH 3D PRINTING

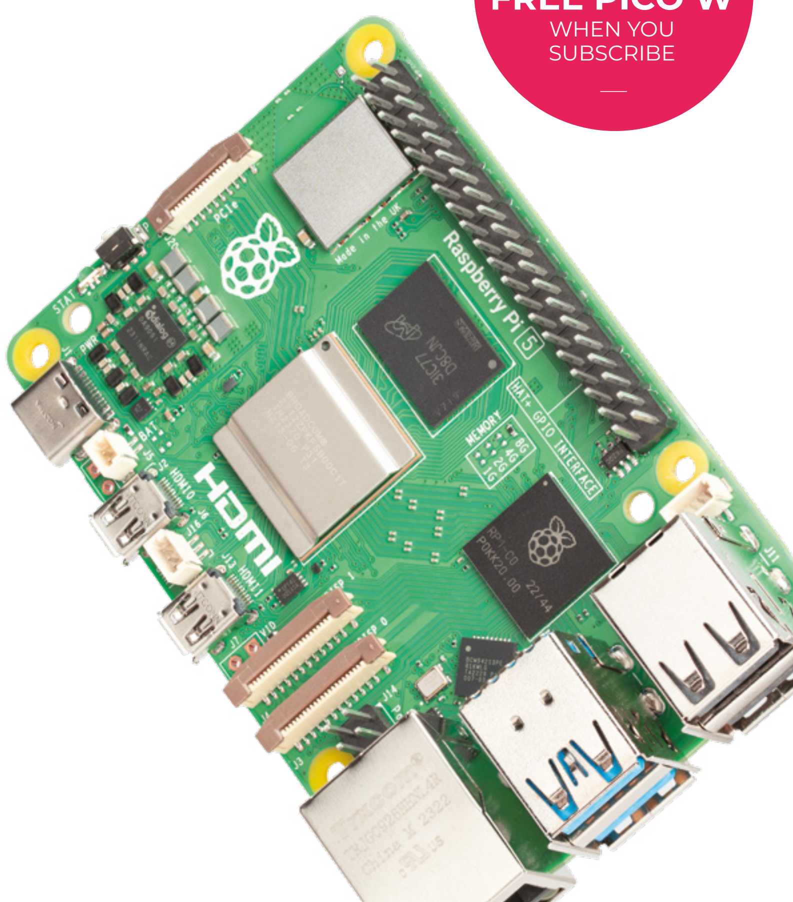
Welcome to HackSpace magazine

It's been a tough couple of years to be a Raspberry Pi enthusiast and having to eke out your supply of boards, or search through online stores trying to find some for sale. Today, we firmly put that part of history behind us. The Raspberry Pi 5 is almost ready and if you're a subscriber, we have reserved stock that you can pre-order (for recommended retail price). If you're not a subscriber, don't worry, there's still time to subscribe and claim your spot at the front of the queue.

To find out about the new release, we headed to Cambridge to get the inside story of the board from Eben Upton and James Adams. We found out about the shiny new SoC, the extra features and, of course, the power button.

BEN EVERARD
Editor ben.everard@raspberrypi.com

PAGE 34
FREE PICO W
WHEN YOU SUBSCRIBE



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

Available on the App Store

GET IT ON Google Play

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, Nicola King, Andrew Lewis, Rob Miles, Christopher Parrott

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, Maurice Wilkes Building, St. John's Innovation Park, Cowley Road, Cambridge, CB4 0DS The publisher, editor, and contributors accept no responsibility in respect of any omissions or errors relating to goods, products or services referred to or advertised. ISSN: 2515-5148.

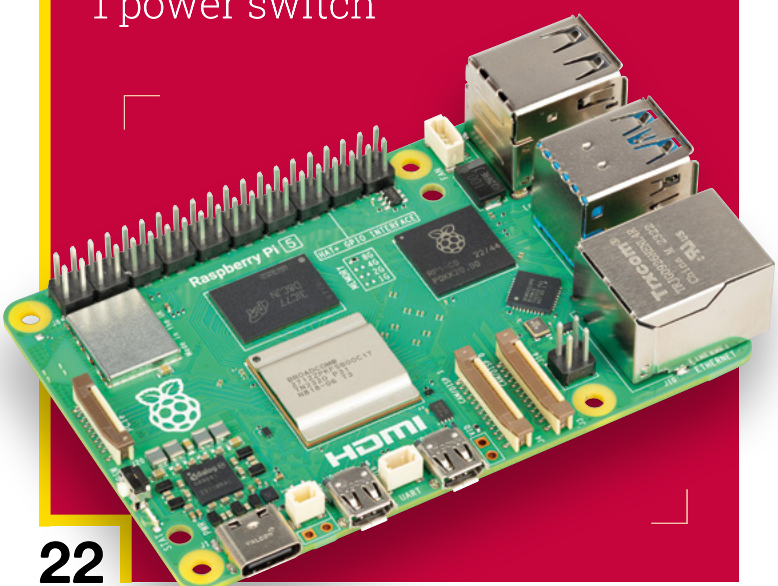
Contents

06

Cover Feature

RASPBERRY PI 5

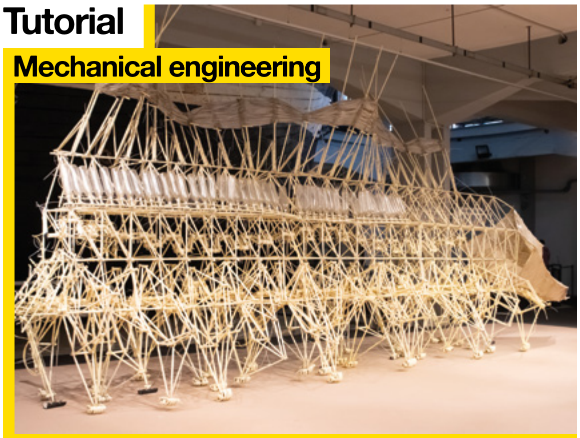
$25 million
8 years
1 power switch



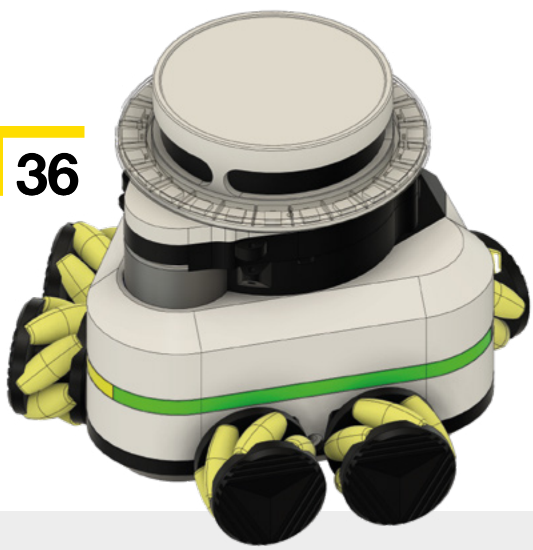
22

Tutorial

Mechanical engineering



64 Print useful things that convert motion into other forms of motion



36

Tutorial
Paper circuits

Crowdfunding
Computer Engineering for Big Babies

47 FORGE

83 FIELD TEST

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, Maurice Wilkes Building, St. John's Innovation Park, Cowley Road, Cambridge, CB4 0DS, 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.

KOAT0 Portable Terminal

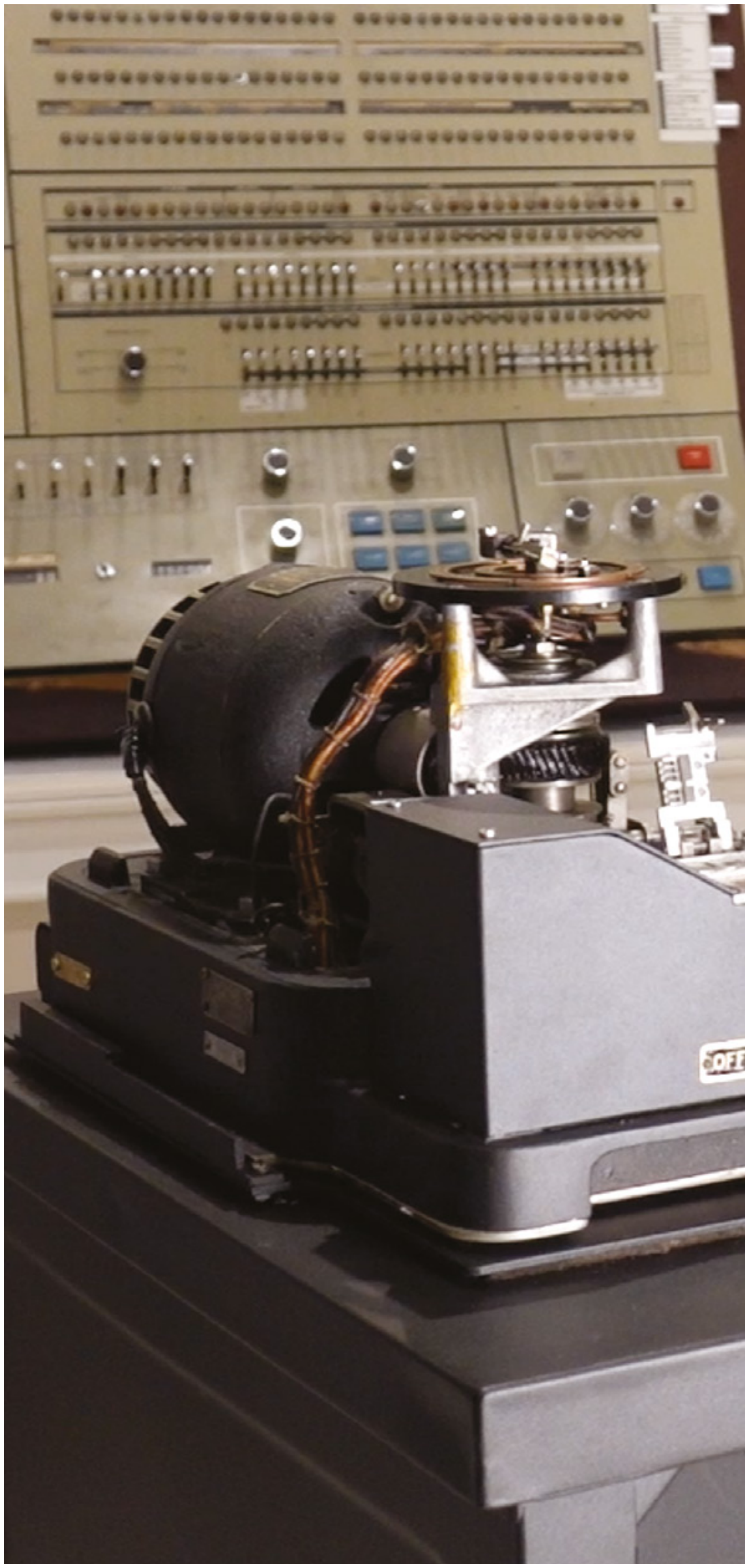
By **Robson Couto** hsmag.cc/KOAT0

We've seen many homemade portable computer builds, but not too many that purposefully eschew the usual OLED screens in favour of a dinky 256×48 VFD display. The maker, Robson Couto, concedes that this isn't really enough screen real estate to be useful but that, in this case, he's aiming to make something beautiful rather than functional.

The build uses a Raspberry Pi 3 (because that's what Robson had available to him at the time), a K68 mechanical keyboard, and a cool 3D-printed handle/case, so in the event of a zombie apocalypse, it'll be handy to use on the move.

Right Robson has made the 3D files available to anyone who wants to build their own KOAT0


KOATO



1930s Linux Teletype

By **CuriousMarc** hsmag.cc/LinuxTeletype

In the dim and distant past, before PCs, organisations would have one big computer (a mainframe) connected to many smaller computers (terminals). You can see the linguistic clues of this past in today's Linux distributions, which all have at least one terminal application where the user interacts with the computer via typed commands. Fans of the terminal say it's often faster, more accurate, and more power-efficient than using a graphical interface. Fans of this Linux terminal, created by CuriousMarc, can also say that it smells nicer and feels better than using a graphical PC.

This beautiful machine takes a Model 15 Teletype machine built in 1930. It's an electromechanical device that was used to send and receive messages typed into a keyboard, and printed out onto paper strips. There's a rich and fascinating history to these devices, but our favourite thing is that when they're running, they smell of hot metal; you can learn more by watching Marc's hypnotic, absorbing video.

Right
Modern Linux machines still have something called a TTY layer, which takes its name from an abbreviation for Teletype


ON
TELETYPE
T/D
REC-30
REPERF MUTE
LINE 1
TEST
LINE 2

Touch Screen Camera

By Mukesh Sankhla | hsmag.cc/TouchCam

You may have seen the Pi Cam by Mukesh Sankhla, an internet-connected Raspberry Pi-based camera capable of streaming video and images. This redesign takes that already nifty design and adds a couple of new features to make it even better.

First and most obviously, there's the touchscreen display, providing a visual interface for camera controls, image previews, and video playback. Mukesh has also added a 5V fan and heatsinks to provide efficient temperature regulation, and the whole thing is housed in an all-new 3D-printed housing designed in Fusion 360.

Right The Touch Cam uses a Raspberry Pi 4 to capture and stream images and video


Raspberry Pi

PikoQube

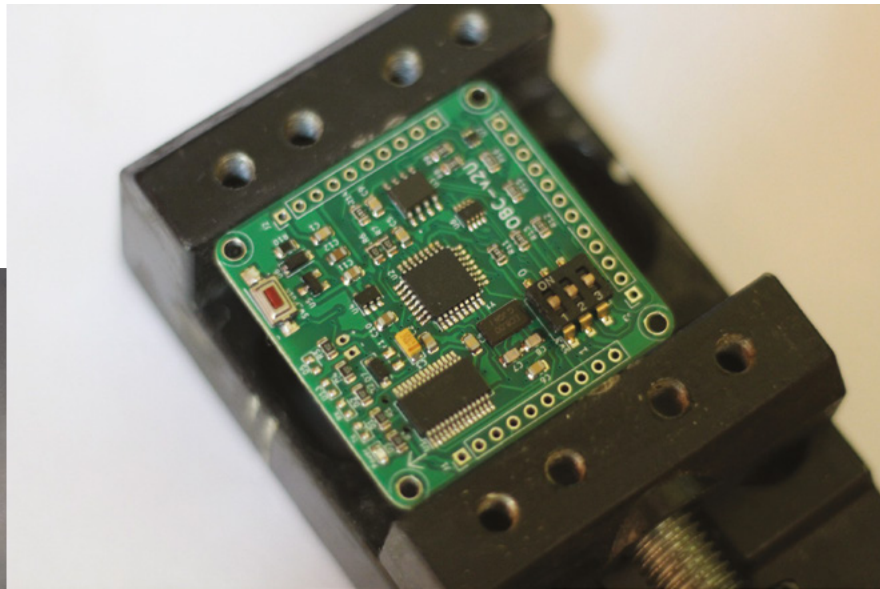
By **Orion Space** hsmag.cc/OrionSpace

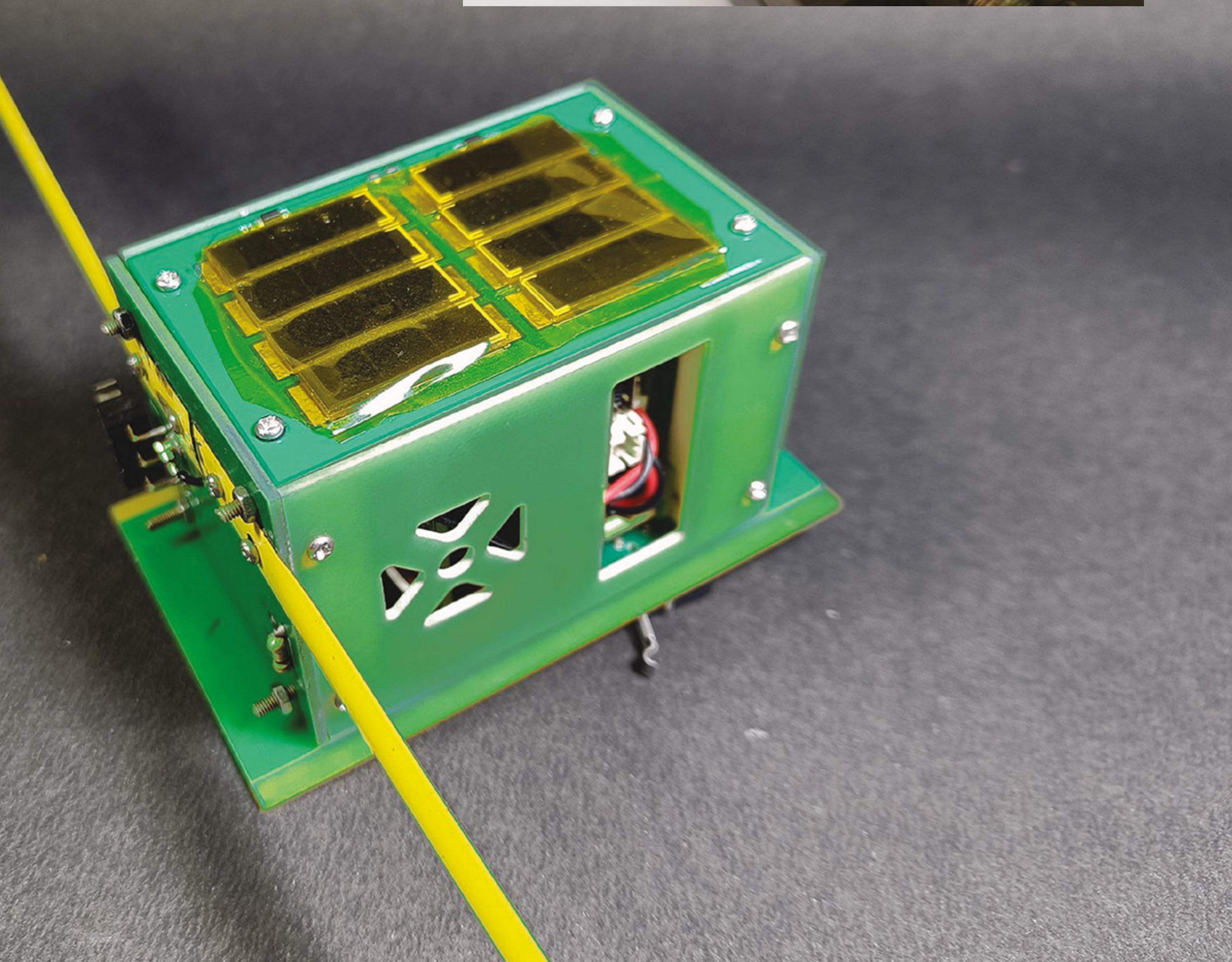
5 × 5 × 75 mm satellite on the form factor of a 1.5P PocketQube. One of the sides is covered by a solar panel. It consists of four (inside) PCBs: EPS, OBC, COM, and Payload. Solar panels, base plate, and other panels make up the structure of the PocketQube.

The PocketQube satellite format is a wonderful example of citizen science. It's made space accessible to schools, colleges, universities, and other enthusiast groups, and as such, it's a natural for use with open-source and homemade electronics.

This training kit from Orion Space gives would-be participants in the PocketQube program the ability to practice on a PocketQube satellite without the price tag. It's 1.5× the size of a standard PocketQube at 50 × 50 × 75 mm, and features the battery casing, solar panels, power management subsystems, an ATMEGA328 microcontroller, and all the rest of the electronics that students will need to master before they can get their own project up into space.

Right Orion Space sent Nepal's first pocket satellite into space with the SanoSat-1, launched in January 2022





Tide Clock

By **Subsystems**

hsmag.cc/Tide_Clock

This clock is a great example of taking a very simple mechanism and making it a lot more interesting using a bit of ingenuity. The Tide Clock uses an ordinary circular clock mechanism, and adds a linkage to make an indicator slide up and down. As the indicator reaches the top or the bottom, it slows down; towards the middle of its travel, it speeds up. This is, as its creator explains, the sine function at work, and it's also exactly how the tides move. □

Right
The clock comes in a kit of laser-cut plywood, with everything you need to monitor the tides, except for a battery


The Tide
HIGH
90
80
70
60
50
30
20
10
LOW
RISING
Trending

Objet 3d'art

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

This portable music player from the Ruiz brothers uses a host of Adafruit goodness, a Raspberry Pi Zero W, and a smart 3D-printed enclosure to transmit your tunes. It plays audio streamed from your iOS device over Apple's AirPlay protocol; and it's battery-powered, so you can use it on the move.

The guts of this build include a Raspberry Pi Zero W, an Adafruit Stereo Speaker Bonnet, and a Pimoroni Pico HAT Hacker to give access to all 40 of the Raspberry Pi Zero W's GPIO pins. There's also a clever bit of power management, with a 3.7V mAh battery and an Adafruit Power Boost to bring the output up to a more useful 5V. All this, plus a 3D-printed enclosure. Nice work. □

hsmac.cc/BoomyPi

Right
The lines etched in the clear acrylic panel are traces from the Speaker Bonnet's PCB design

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**

RAD ROCKETS

I was so excited to open my favourite maker magazine and see my favourite YouTuber – Joel Gomes, aka Integza. I love the sheer determination he has to make a rocket engine with a 3D printer, despite it being clear that a 3D printer is clearly an inappropriate tool for making a rocket engine.

P.S. Tomatoes *are* disgusting.

Becca
Newcastle

Ben says: Joel is such a genuinely great guy. His enthusiasm is infectious. I honestly didn't have much interest in rockets or jet engines before, but I've been binging my way through his back catalogue, and now I can geek out over nozzles and thrust. (And yes, tomatoes *are* disgusting)

EXISTENTIAL AI

Artificial intelligence, machine learning, or whatever the word of the day is for this technology – it makes me feel uneasy. Should we be using it as makers? Is it the future we want, or just some dystopian nightmare fuel? Mentioning robot overlords is cliché, but seriously, should we be worried about this?

Issue 70 has triggered a bit of an existential crisis for me. Can you help me calm down a bit?

Paul
Birmingham

Ben says: Honestly, we don't have any answers for you. It's a tricky moral and philosophical question that I think everyone has to grapple with in their own way. We're still finding our own way through it. As with most technologies, there are certainly ways it can be used to make the world a better place, and there are certainly ways it can be used to impoverish society, not just in an economic sense, but in a cultural sense as well.

Part of the maker movement has always been democratising technology. Getting tools into the hands of more people can be a key step in ensuring that the technology is beneficial to people, not just corporations. Our current strategy is to deal with AI in the same way. By exploring its use for makers, we can better understand it and make informed decisions about how involved we want it in our lives.

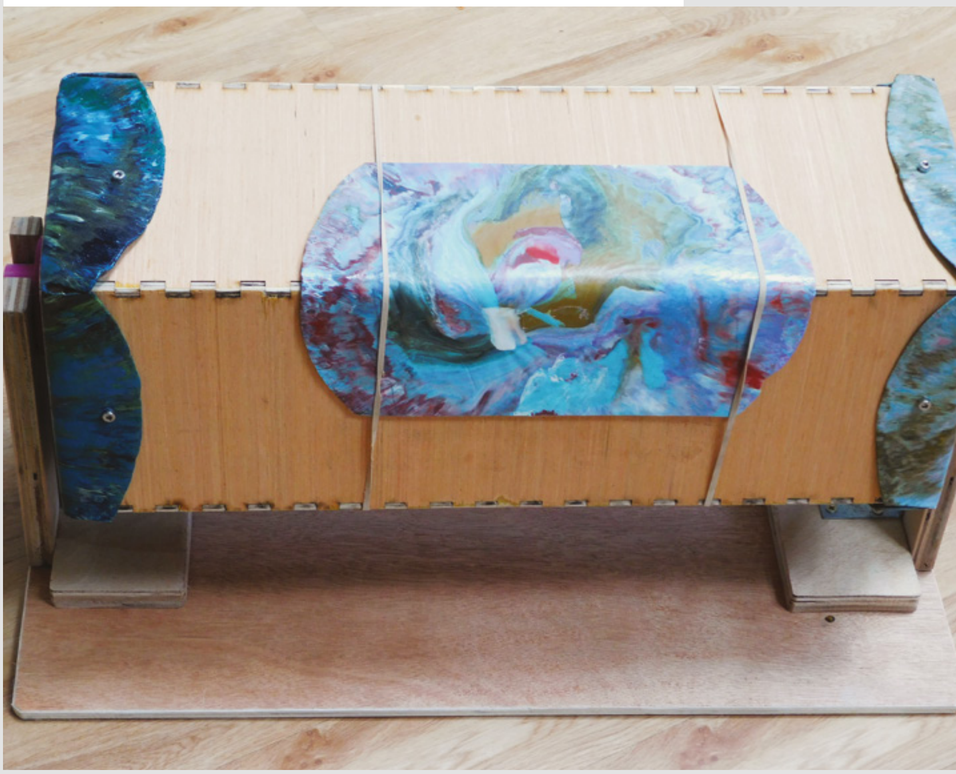
Ultimately, we can't tell you what you should use any technology for; we can only investigate how they can be used and let you decide for yourself.

DO IT PROPERLY

Just FYI, our local summer fair uses a cardboard box as a tombola.

John
Bristol

Ben says: I hold my hands up. I'm an over-engineer-er (and a word-maker-up-er). The chicken coop in my garden can withstand a hurricane, my patio has deeper foundations than most high-rise buildings and, if I'm asked to make a tombola, you can be sure it's going to tombol (I told you I make up words) properly. OK, so the door doesn't have a hinge, and it's unwieldy to store and transport, but everyone knows you have a better chance of winning if you can give it a lucky spin.



HackSpace
TECHNOLOGY IN YOUR HANDS

LENS

HACK | MAKE | BUILD | CREATE

Uncover the technology that's powering the future

PG 36

HOW I MADE:

L-HEX

Inside the development of a lidar-equipped, mecanum-wheeled, custom-made robot

PG 44

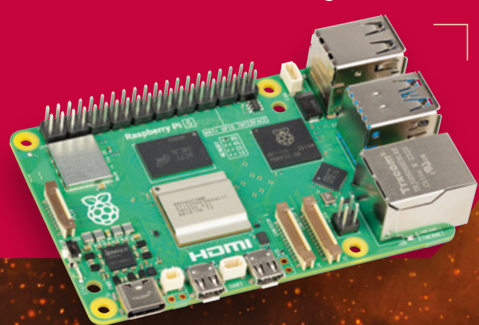
IN THE WORKSHOP

Help us – we've fallen into maker inception, making things to make other things...

PG 22

RASPBERRY PI 5

Introducing the computer that the world has been waiting for



Raspberry

The inside story on the new computer

The next iteration in the world-conquering Raspberry Pi line-up has been announced. Raspberry Pi 5 will go on sale in October. We caught up with Eben Upton (co-founder of Raspberry Pi and CEO of Raspberry Pi Ltd), and James Adams (CTO and principal hardware architect at Raspberry Pi, and designer of six of the seven flagship Raspberry Pi boards).

Here's what they had to say…

HS Raspberry Pi 5. What is it?

Eben It's our new flagship product. It's the seventh iteration of the flagship product: we've had Raspberry Pi 1–4, and the 1+, and the 3+. It's roughly two-and-a-half times as fast as a Raspberry Pi 4, which makes it about 130 times as fast as Raspberry Pi 1. If you measure using the JetStream JavaScript benchmark, it has a little over half the performance of my last-generation Intel MacBook Air.

HS So over 10, 11 years of development you're pretty much keeping up with Moore's law. Nice!

Eben We're about on track. Eleven years is about seven iterations of Moore's law [Gordon Moore's observation that computing power doubles every 18 months], which is 128, and we posted 130, so I think we are more or less clinging by our fingernails to Moore's law.

James It works very nicely as a desktop machine now. I mean, Raspberry Pi 4 did as well.

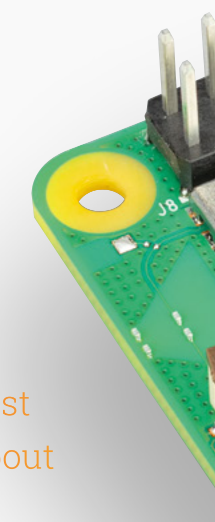
Eben We have this idea that there's a kind of bell curve of demand for computing. There are some people who don't really want very much performance at all [Eben points to the left-hand side]. And then some people who want as much performance as they can get – extremely high-end gaming, or CAD [Computer Aided Design] programs. [Eben points to the right-hand side.]

Some people, say, just want to read a value from a sensor once a minute in a Python script and log it to the network and turn an LED on and off – Raspberry Pi 1 could do this, of course. So, there was a market for Raspberry Pi 1 as a PC, but only for people with relatively modest demands.

People's ambient expectation of how much computing power they need drifts to the right, but it doesn't charge to the right. So, over the generations, we've been catching up.

> "It's roughly two-and-a-half times as fast as a Raspberry Pi 4, which makes it about 130 times as fast as Raspberry Pi 1

The interesting thing about Raspberry Pi 4 was that it got past that median. I could give it to my parents, to use as a generic PC. They can't really tell that it's not a MacBook Pro, because they don't need a MacBook Pro's worth of performance. With Raspberry Pi 5, we're now going back down the other side of the slope: most people will be in the same camp as my parents. →



Pi 5


Raspberry Pi 5
HAT+ GPIO INTERFACE
MEMORY
8G
4G
2G
1G
FAN
RP1-C0
BROADCOM®
HDMI
UART
PCIe
PWR
STAT
Made in the UK
Trxcom

The heart of the board

A custom-designed system on a chip

HS What's new on the Raspberry Pi 5?

Eben Let's talk about BCM2712. It was developed by Broadcom. But it was co-developed to a great degree.

James It's architecturally pretty similar to BCM2711 [the SoC in Raspberry Pi 4]. It's got faster Arm cores, more cache, beefed-up fabric, better display pipeline, faster multimedia blocks, like the GPU and image signal processor.

We always wanted a user PCI Express port, and it's nice to squeeze one on. It's not M.2 or any sort of standard because it just won't fit on the board. But we will sell adapters to take that and turn it into an M.2.

HS There are a lot of things that you can plug into PCIe. What are you expecting to work?

James Almost everything!

BCM2711 had PCI Express root complex that wasn't fully compliant with all the corners of the standard – I mean, PCI Express is a very complicated standard. Devices like PCs just sort of put all the features in and they beat them up to make sure the corner cases work. This isn't always the case with [Arm hardware]. But Broadcom, with our help, have done a much better job of trying to make the new root complex more standards-compliant, so it should work with more devices.

Eben That includes things like Coral [the Google AI Accelerator]. [PCIe on the Raspberry Pi 4] would always work with things like NVMe [solid-state drives] but 64-bit register accesses didn't work due to a bug in BCM2711, and that permanently meant you could not connect to the Coral.

I think honestly, if people at Google could have worked a little bit harder to help us debug what was going on there, we probably could have made it work. [But on Raspberry Pi 5] Coral should work. Whether giant PC graphics cards will work, and Jeff [Geerling, a YouTuber famous for trying to hook PCI graphics cards to Raspberry Pi] will have a good experience: search me, I don't know.

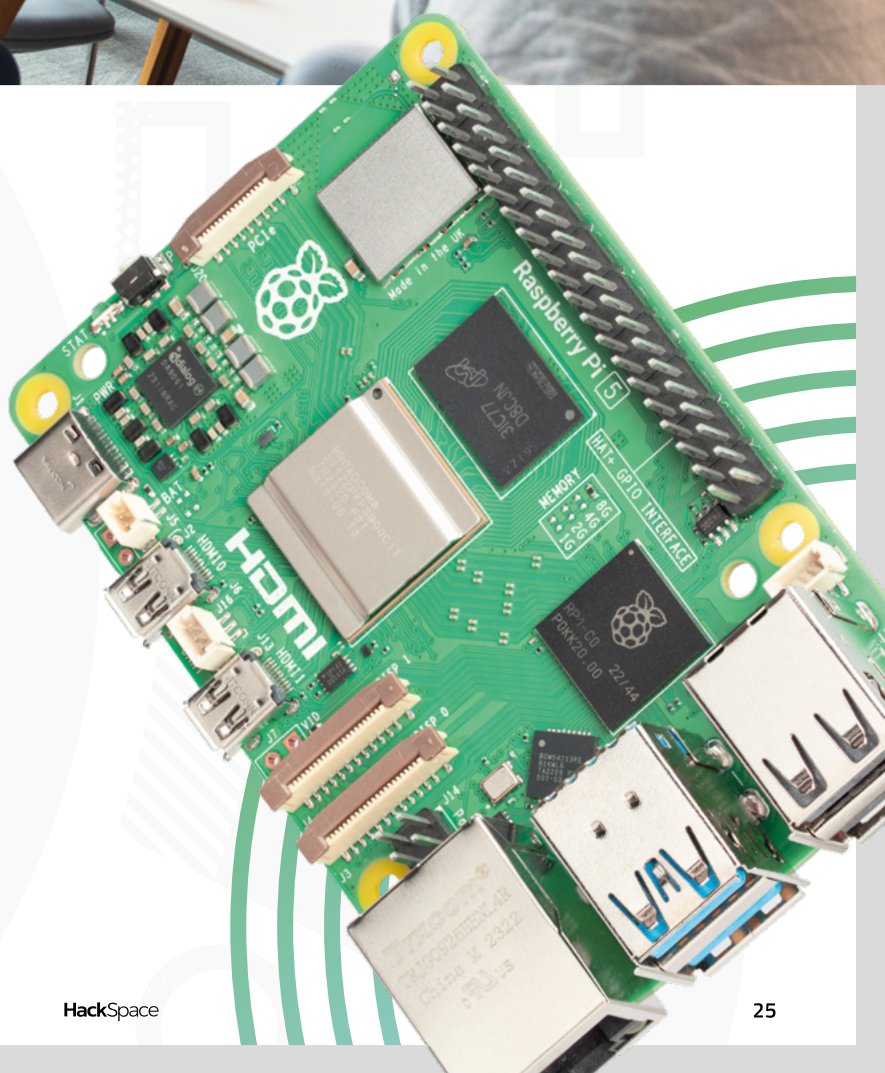
There will be fewer bugs, and there are more supported features. But your guess is as good as mine as to whether that collection of features will be sufficient to resuscitate an AMD graphics card.

HS And what about the processing cores?

Eben We've gone from a quad Arm Cortex-A72 at 1.8GHz [on the Raspberry Pi 4], with a 1MB shared L2 cache, to a quad Cortex-A76 at 2.4GHz. Each core has its own 512kB L2 cache, and they share a 2MB L3 cache, so they've got a lot more cache memory – you've got a total of 4MB of cache in the system.

They're architecturally more performant – so they're about 50% faster clock-for-clock, maybe a little more than that – and they're at a higher clock frequency too. And those effects stack together for about a two-and-a-half times performance [improvement over Raspberry Pi 4]. →

> We always wanted a user PCI Express port, and it's nice to squeeze one on



Power for the peripherals

Keeping electrons in order

HS The power system has had an overhaul. Can you tell us about it?

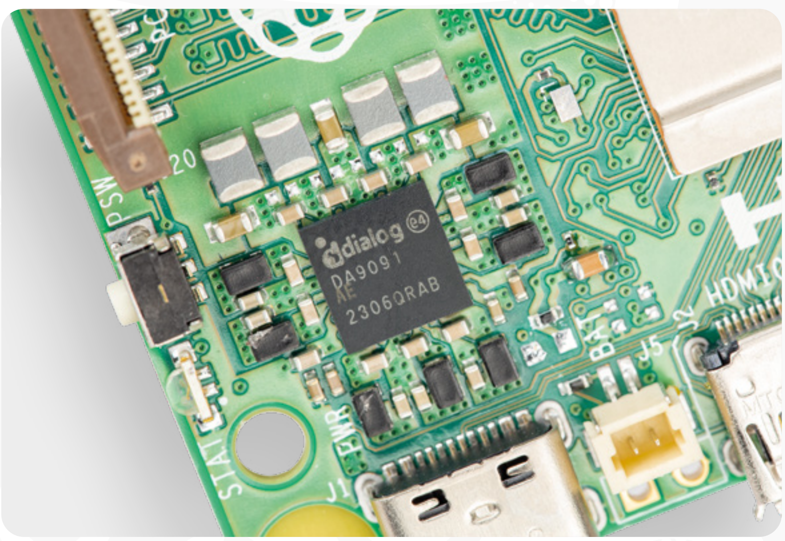
James We've worked with Dialog (now Renesas) to build this chunky power management chip. You can see there are a lot of inductors around it. These little black guys are inductors, and these four grey things at the top are inductors. It's a quad-phase 18–20 amp switcher, which supplies the core of the chip.

Eben You have two-ish megahertz PWM [pulse-width modulation], so every half a microsecond you connect the five volts to the output node for a fraction of half a microsecond. The more load it's under, the greater the duty cycle of the PWM. And that's how most of the switchers work, how the rails work.

But with the quad-phase rail, you have four of these machines all feeding the same node through their own inductors, and those periods are offset by an eighth of a microsecond, so they are at 90-degree phase offsets.

And that means that you can respond faster [to changes in the board's power demand].

At launch, Raspberry Pi 4 had a single-phase core supply. But the newer boards have dual-phase, and this is now quad-phase. Going from single-phase to dual-phase is what enables the 1.8GHz operating point on Raspberry Pi 400 and the more modern Raspberry Pi 4s.



HS And, of course, the big news is that there's now a power switch!

James We've got a power switch, which works much like a PC or laptop power button. It has a soft and hard power-off modes: if you touch the button when it's booted, it will tell Linux to shut down; if you hold the button for long enough, it powers off hard, by cutting the supply rails.

The other thing that this power management chip does is it talks ‘PD’ to USB power supplies [Power Distribution – the standard for negotiating with the type of USB-C power supply used by higher-power devices].

USB PD provides a serial interface to the USB power supply that a chip can talk to and say, ‘Hey, what voltages and currents can you give me?’, and then it can choose one. [Raspberry Pi 5] will always choose 5V, but we now have the ability particularly to talk to the supply to check that it can support 5 amps.

[Ed note: Raspberry Pi is launching a 5V, 5A power supply, but currently most other USB PD power supplies won’t support this mode.]

If you don’t have a five-amp supply [for example, the Raspberry Pi 4 power supply was 3A], it restricts the USB current output quite aggressively, to 600 milliamps maximum, instead of about 1.1 on the Raspberry Pi 4. So, you can run your mouse and keyboard, happily, but it won’t boot from USB mass-storage devices: it will decline to do that by default.

Eben This is one of these really difficult engineering things that we had a lot of discussion about. When we say the board can consume 12 watts, we mean that it can consume 12 watts if you craft a horrible use case for it that’s deliberately designed to do nothing useful, but to consume 12 watts.

So, we have 600 milliamps and a 1.5 amp mode for USB. Raspberry Pi 5 selects between them on the basis of whether it detects a 5-amp power supply or not. Or, you can stick something in config.txt that overrides it.

And the vast majority of people with an existing [3-amp] power supply will plug it in, set the override in config.txt and just forget about it.

> The other thing that this power management chip does is it talks ‘PD’ to USB power supplies

James It’s still five volts, because we don’t want to do power conversion from, say, nine volts, which would be what most people use to get more power into the board. They get more voltage in, and then they convert it into five volts. We don’t do that because it’s costly in silicon, and it’s costly in wasted energy, which just ends up heating the board up. We’ve done the more Raspberry Pi thing, which is make a supply that can drive a five-amp load at five volts, which isn’t a standard PD mode, but you can negotiate it.

Eben We encourage people to buy the new supply! →

Manufacturing improvements

Making robots redundant

HS The bottom of the board is a bit less spiky than before – what's changed there?

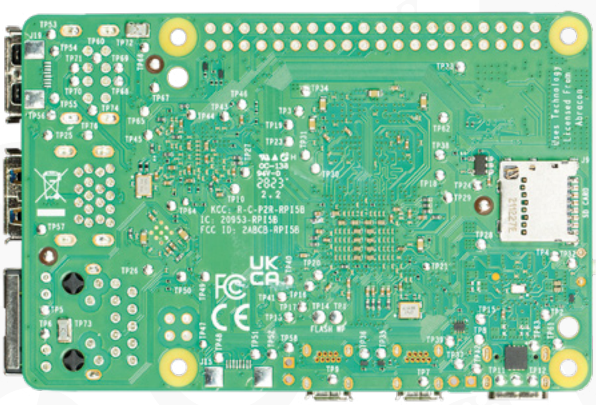
James The soldering technology is different. We've taken out one step – one big step.

Instead of having robot arms put these [through-hole components in] and splashing molten solder underneath, we use surface-mount.

HS You say surface-mount, but it does still go into the holes?

James That's right. The SMT – surface-mount technology – process is that you take a screen with little holes and you squeegee your [solder] paste over the top. The pick-and-place machine puts the components down. Then you run it through an oven which melts the solder and everything sticks.

You do that for the back, and then traditionally you do that for the top without any of the things that go through the board. And then you do those through-hole parts as a final step, with wave or selective soldering.



But we're no longer doing that last step. Part of the reason is that now the pick-and-place machines can pick up bigger things, because we've got these new heads, but also we've worked with the connector manufacturers to trim the pins and put higher temperature plastics in various places, and iterated on the footprints and the hole sizes and the paste, so you now paste everything on the top layer, the pick-and-place machine does every single component, and then you just put it in the oven. With through-hole things, the paste on the top melts and it goes into the holes. Job done!

Eben Except … we discovered a fringe benefit of having these pins sticking through the board, which is that, along with the SD card connector, they protect the bottom of the board against mechanical damage.

HS Is that what the little loops are for?

James Yes, these things are to stop the capacitors smashing on the desk.

HS So, there's a surface-mounted through-hole pin?

Eben Ha! Yes! Two of them!

HS I'd assumed they were ground test points.

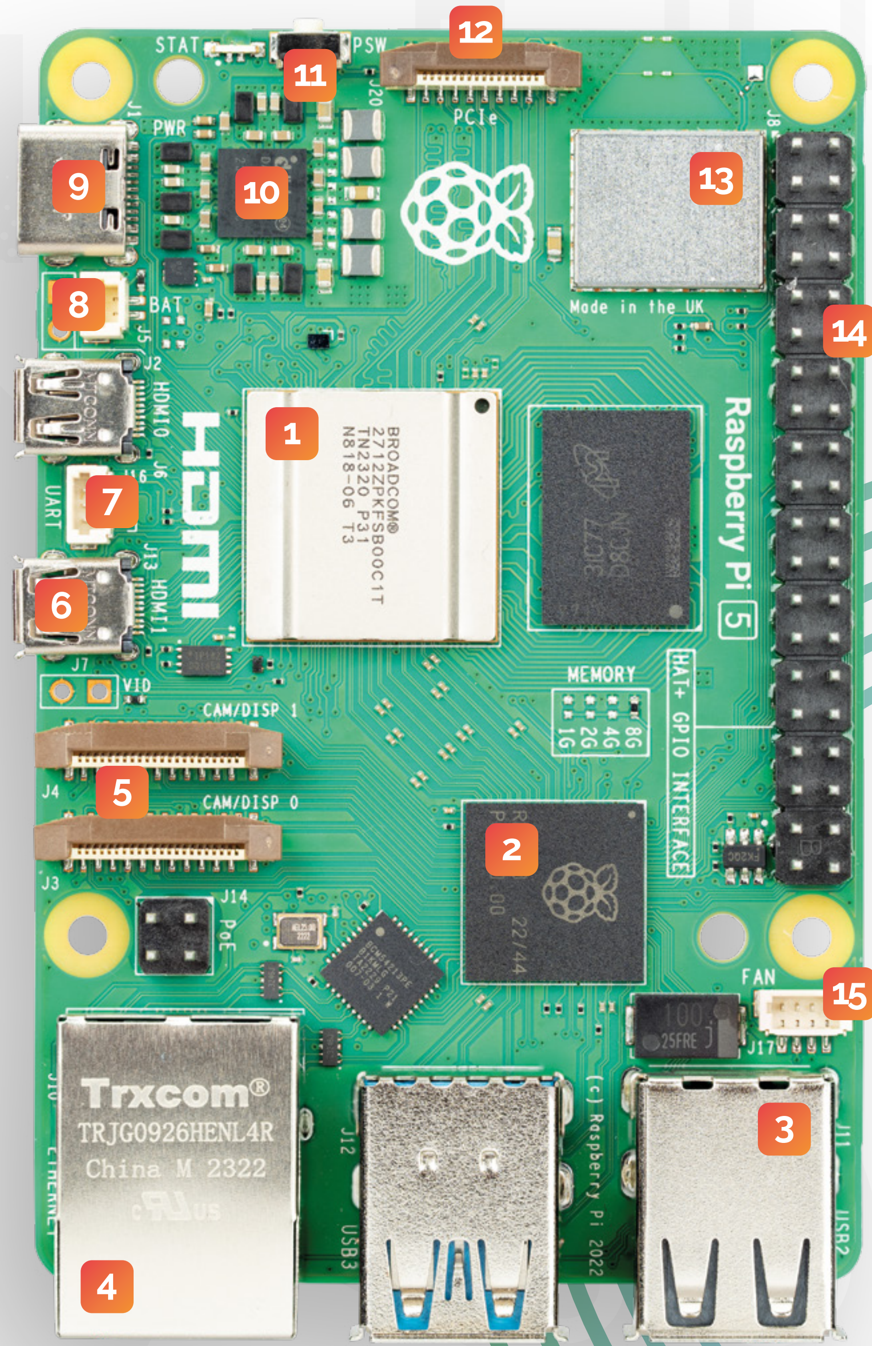
James They are sort of useful if you're prodding things. They are grounded.

Eben It really is quite flat. [To James] Did you pick ones that are the same height?

James Yes, they're the same height as the SD card holder. →

A tour of Raspberry Pi 5

1. 2.4GHz quad-core 64-bit Arm Cortex-A76 CPU
2. RP1 providing GPIO, USB, and the majority of connectivity
3. USB 3.0 ports providing 5Gbps of bandwidth
4. Gigabit Ethernet
5. Two Camera / Display connectors
6. Two 4Kp60 HDMI ports
7. UART port for debugging
8. RTC battery connector
9. USB-C power that can accept up to 5A at 5V
10. Custom PMIC
11. Power switch
12. User-accessible PCIe port
13. Wi-Fi and Bluetooth
14. Raspberry Pi GPIO connector
15. Fan connector



Why a new chip?

The first Raspberry Pi chip makes an appearance

HS Can you tell us about the new chip on the board? Is it called Project Y or RP1?

James Project Y is our internal name for RP1. It was roughly a $15m programme. It's what would be called a southbridge in a PC – so it's the I/O control. It talks to the Broadcom chip over four-lane PCI. That's the big slug of wires that go from one chip to the other.

HS So obviously on Raspberry Pi 4 there wasn't a Project Y, there were a bunch of other chips that did an equivalent job.

Eben There were actually two places where I/O functionality was provided on Raspberry Pi 4: some of it was provided by the I/O expander chip, the VL805 USB 3.0 controller and hub, and quite a lot of it was on the core silicon.

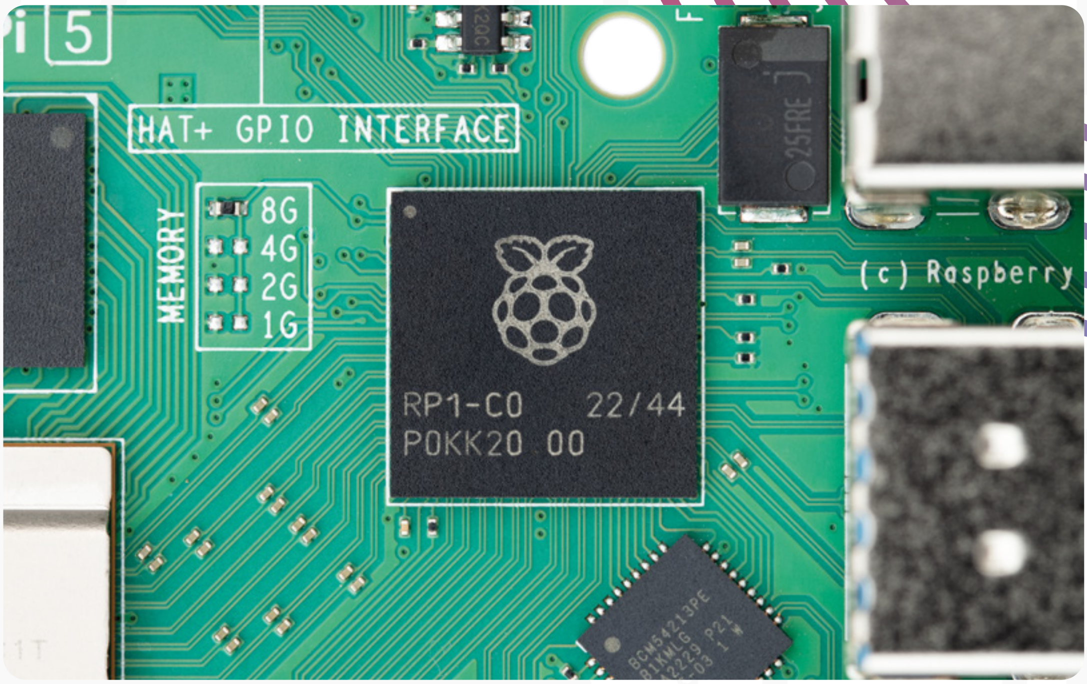
In this generation, all of that stuff has been schlurped out of the core silicon and integrated into this I/O controller. MIPI is no longer provided by the Broadcom silicon. Analogue television is no longer provided by the Broadcom silicon. GPIO is no longer provided by the Broadcom silicon. USB, Ethernet, those are all provided in our I/O controller. So all that's left in the Broadcom silicon is an enormous amount of high-speed digital logic – so CPUs, GPUs, the DRAM interface, the HDMI (it's very high bandwidth so you can't push that off to an I/O device), and the PCI Express that talks to the I/O control.

From an analogue perspective, the core silicon is now much simpler. Why is that good? Because it means you can take that core silicon down to smaller process nodes without worrying about 'oh, how do I provide 3V3-tolerant ESD-hardened GPIO pads on a three-nanometre device?' You don't need to worry about that any more.

That's what we call a chiplet architecture.

James That's right.

We certainly appear to have reinvented (or pre-invented?) the chiplet architecture. And also, the analogue doesn't shrink very well into the small geometries, so it costs you more and more as you shrink the digital stuff in the analogue stuff. And it costs you time and engineering to actually shoehorn it on, test it, and make it work every time.



HS So, the point of that chip isn't the I/O itself, it's to allow the CPU to be faster and better?

Eben It's to simplify the core device. It's better in some respects than before. So a simple example: your VL805 chip is a single USB 3.0 controller with an integrated hub. Yeah, so you have five gigabits of downstream bandwidth.

But you only actually have four gigabits of upstream bandwidth. So, you can't even get all of those five gigabits: it's shared between the two USB 3.0 ports. But you can't even get all five gigs because you only have a single PCI Express, so you have four shared between both ports.

[Raspberry Pi 5] has four-lane PCI Express, so you have 16 gigabits of upstream bandwidth, and each USB 3.0 gets a full five gigabits of bandwidth.

HS And this is the second chip that Raspberry Pi has released. How does it compare with RP2040?

James It's a lot more complicated. It's the first chip that we built the chip team to build back in 2015. It was the start of the programme. It's why the chip team exists. Which is also why it's called RP1.

> It's the first chip that we built the chip team to build back in 2015. It was the start of the programme

Eben I'm amused that nobody ever asked us about this. RP2040 has RP2 written on it, and people think that's short for RP2040. And the chip that was on the Zero has RP3 written on it. People think that that's somehow a reference to Raspberry Pi 3, because it's the same die, but it's not: RP3 is our third chip; RP2 is our second chip; RP1 is our first chip. →

Going against inflation

They fought the law, but the law won

Eben The Raspberry Pi 5 cost $25 million to make. And it's composed out of three enormous engineering programmes (BCM2712, RP1, and the DA9091 PMIC), and it itself is an enormous engineering programme. That power supply is itself a vastly larger engineering programme than pretty much anything other than we've ever done, apart from Raspberry Pi 4.

Raspberry Pi 4 was a few million dollars. It wasn't cheap. But the Raspberry Pi 4 was probably a $5 million programme for us.

HS And in terms of cost benefit, I guess, does that pay-off come from selling more units or making more profit per unit?

Eben It comes from continuing to sell units. It's a Red Queen's race, against our competitors, and against the ever more demanding expectations of the public.

That's the thing: you're running to stand still. It's to make the same number of units and make the same amount of money per unit, but to keep doing it for another few years. Now, in practice, of course, we'll make less money per unit on this than we do with the Pi 4 – this is going to be $5 more expensive than the Pi 4, but it costs more than $5 more than a Pi 4 to make. So, we'll make less money. But I think we will sell more units, because the market's growing, and we've got extra features and performance.

HS This is the first Raspberry Pi to cost more than $35.

James Silicon has got more expensive. People are charging us more for silicon.

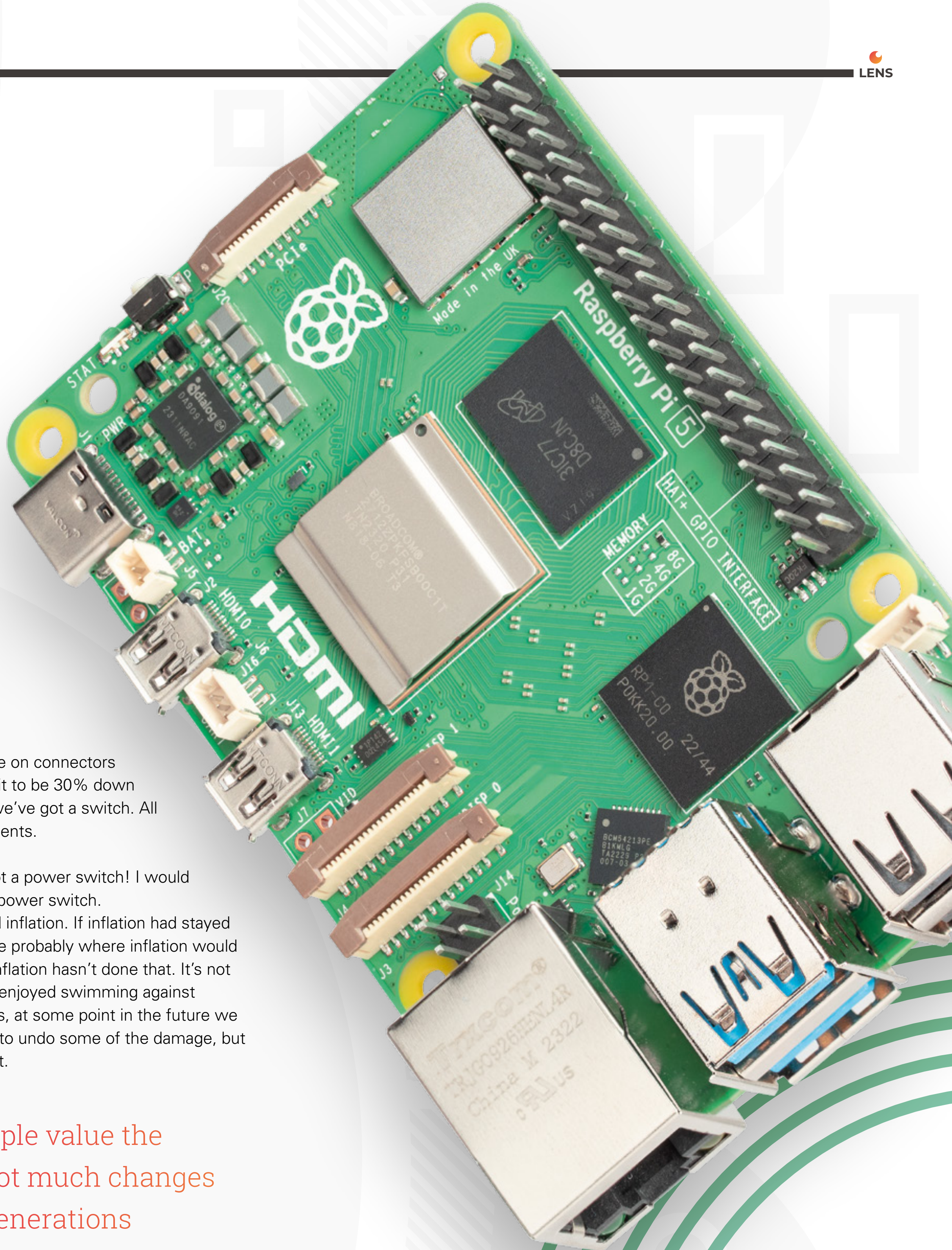
Eben And, it's silicon that is more complex. It's the first time our cost structure has materially degraded between generations. [Pi] 3+ and the 4 are probably naturally five bucks more than their predecessor [but Raspberry Pi didn't pass this cost along to the consumer] and the [Pi] 5 is five bucks plus that.

HS So it's inflation catching up?

James It's the silicon pricing that has really driven it because we expected it to go down, but it's gone up.

Eben You [to James] were modelling it 30% down, and it's actually 30% up.

HS So, you didn't start the design cycle saying, 'Let's spend an extra ten bucks'?



James I spent more on connectors because I expected it to be 30% down on the silicon. And we've got a switch. All of those add a few cents.

Eben And we've got a power switch! I would pay five bucks for a power switch.

We're way behind inflation. If inflation had stayed at its low level, we're probably where inflation would have been, except inflation hasn't done that. It's not desirable. We really enjoyed swimming against inflation. Who knows, at some point in the future we may find some way to undo some of the damage, but for now we are adrift.

> I think people value the fact that not much changes between generations

James I think people value the fact that not much changes between generations. You get more of stuff and it gets faster for the same price – although the price is going up now. I'm pleased with the innovation that's gone into stuff like the soldering process. This is the most Raspberry Pi [model] ever, in terms of all the stuff that we've touched on it. We've had a hand in almost everything. It's quite a triumph, I think. Let's hope there are no bugs.

GUARANTEED

RASPBERRY PI 5 AVAILABLE AT LAUNCH

for all subscribers

SUBSCRIBER BENEFITS:

> **Get every issue of HackSpace magazine delivered to your door**

> Beat the crowds with guaranteed Raspberry Pi 5 stock for subscribers

> Early access to the PDF edition

> **Get a free Raspberry Pi Pico W**

hsmag.cc/subscribe

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

FEATURE

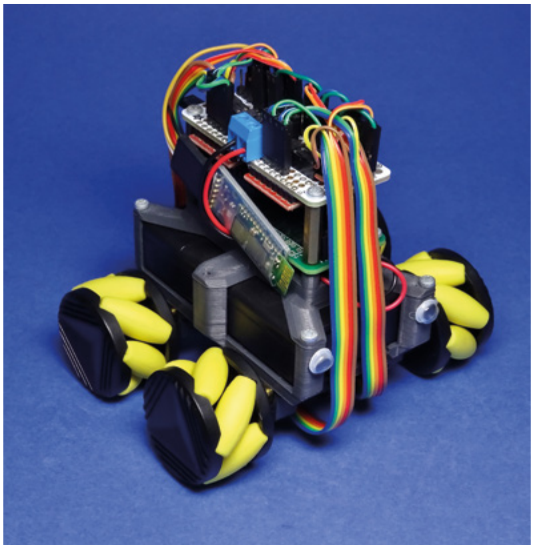
HOW I MADE

L-HEX: A LIDAR-EQUIPPED OMNIDIRECTIONAL ROBOT

By Dr **Christopher Parrott**

There is a special kind of enjoyment that comes from making robots. Creating something that can interact with the real world, using parts you have built and code you have written, feels almost magical. Every robot offers its own challenges in mechanics, electronics, and computing that, when overcome, make the end result all that more rewarding. This is the reason for my ever-growing robot collection!

L-Hex, a lidar-equipped hex (as in six) mecanum-wheeled robot, is the latest addition to my collection. It is triangular in shape, measuring 162 mm in diameter and 120 mm tall. On each side are mirrored pairs of mecanum wheels for movement, driven by DC motors with encoders. Taking up a good portion of the robot's height is a low-cost, 360-degree light detection and ranging



Above
The Mini Mecanum robot showing its stack of components

Above
L-Hex in all its RGB, Mecanum-wheeled, lidar-using glory

(lidar) scanner, which uses a spinning infrared laser to measure distances to objects. At L-Hex's heart are two Raspberry Pi RP2040 microcontrollers: one operating the lidar and the other driving the six motors.

The combination of lidar scanner and mecanum wheels is interesting as they let L-Hex sense its environment with great detail and use that information to move in any direction to follow or avoid obstacles without the need to turn to change direction.

HOW IT STARTED

At the tail end of 2019, I was exploring the use of mecanum wheels on small robots. For this, I created a car-like robot dubbed Mini Mecanum. It used four 50:1 N20 motors with magnetic encoders for feedback, two TB6612 dual motor drivers on a protoboard, a Teensy microcontroller, a Bluetooth serial receiver, and a 3 × AA battery pack for power. All of this was mounted on a bare-bones 3D-printed chassis designed in Fusion 360. 3D printing has been my go-to manufacturing method ever since I was introduced to it at university a decade ago.

To drive Mini Mecanum, I programmed the Teensy to take serial commands received via Bluetooth and translate them to motor speeds, which, along with measured speeds from the encoders, were fed into proportional-integral-derivative (PID) control loops to produce accurate motor outputs.

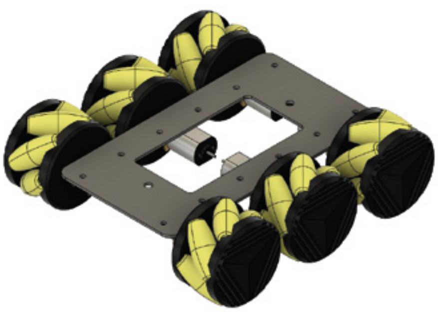
For the robot's mecanum wheels, I attempted to find ready-made ones online. It took some time, but I eventually found a Taobao listing for N20-compatible mecanum wheels – offered in a range of colours, too! Being unfamiliar with ordering from Taobao though, I took advantage of my Sheffield connection and reached out to Pimoroni to see if it could source them, which it could! This was a whole year before getting employed by the company →

Above The three colours of wheels I acquired

– back when I had only met its crew a few times at Raspberry Pi events.

In early 2020, the mecanum wheels were listed on Pimoroni's website in yellow, green, and black. Surprised by having colour choices, I got a set of each and fitted the yellow ones onto Mini Mecanum. They worked better than I had hoped! Not only could the robot move in all directions with ease, but I was able to enhance my code to have it drive autonomously in a square, and even rotate as it translated along each edge!

Below Should I go with three wheels on a side, or six?

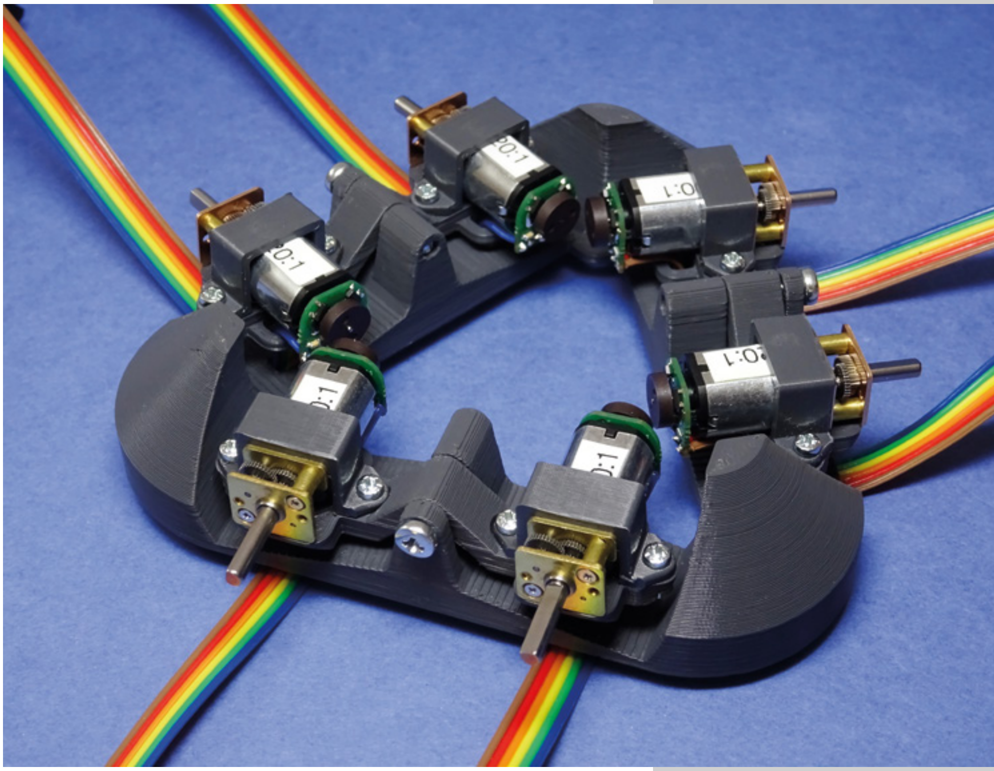


SIX WHEELS

Having a working mecanum robot, I started thinking up ideas for other robots to create. I was particularly interested in using more than four wheels as I had seen videos of industrial robots with twelve or more used to move trains and aircraft around factories. Also, my code enhancements had theoretically removed any limits on the number of mecanum wheels that could be used together, so I wanted to test this with hardware.

Seeing as six wheels was the natural step up from four, I sketched up two ideas in Fusion 360. The first idea was a truck-like arrangement with three wheels on each side. The second idea was a triangle arrangement with two wheels on each of the three sides. I shared both concepts on Twitter, and the triangular shape seemed to gather more interest. This shape also appealed to me as I could distinguish each side with its own wheel colour!

Above The first assembly of Hex Mecanum



Above The underside of the frame prior to wheels, showing the suspension pivot

WHAT ARE MECANUM WHEELS?

Driving in any direction on a surface, known as omnidirectional motion, is a beneficial trait for robots needing to navigate complex environments or accurately position themselves to manipulate objects. One way to achieve this is to use a family of wheels known as omni wheels.

Omni wheels are specially constructed wheels with many free-spinning rollers around their perimeters. These rollers let the wheels slide in their rolling direction, creating the effect of having practically zero surface friction in that direction. By arranging multiple wheels so that the driving directions of each overlap the rolling directions of the others, omnidirectional movement can be achieved.

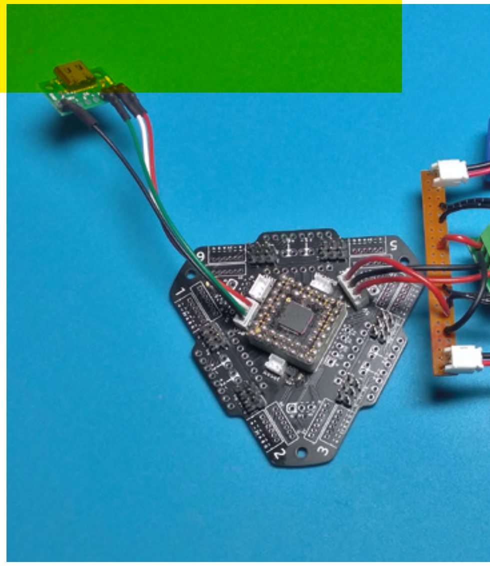
A typical omni wheel has its rollers angled at 90 degrees to the rotation axis of the wheel, making them slide sideways. This gives rise to two common chassis arrangements, triangle and square, both with one wheel on each side. Moving such a chassis is a matter of driving each wheel with a proportion of the motion in that direction.

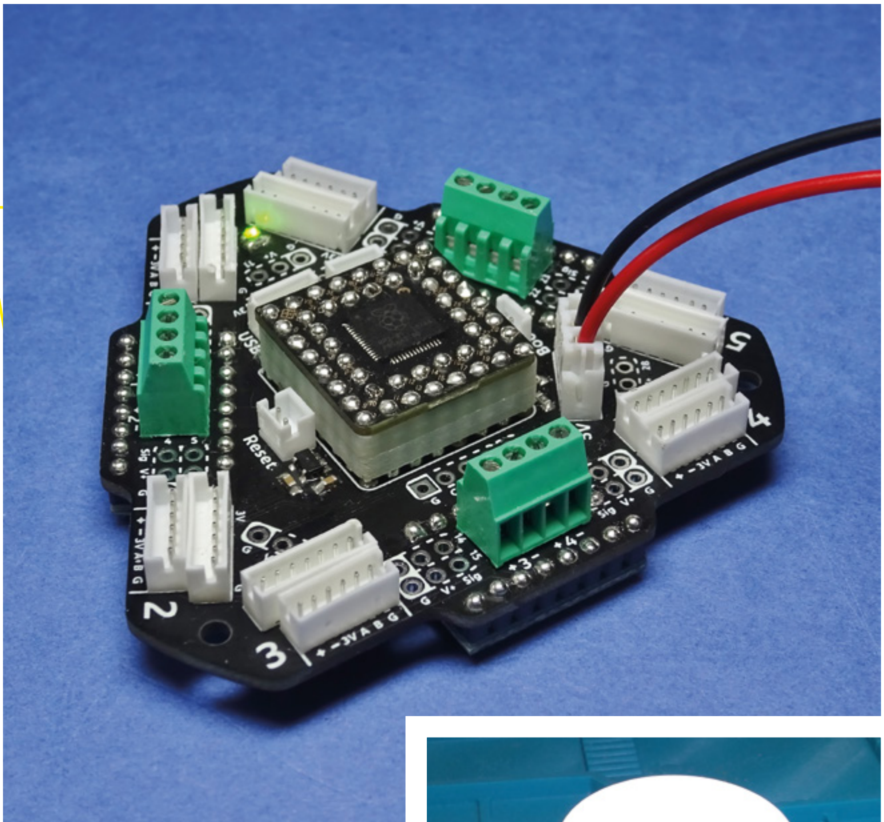
Mecanum wheels differ from typical omni wheels in that they have their rollers at 45 degrees, giving them the tendency to slide diagonally. Placing four mecanum wheels, as mirrored pairs, on a car-like chassis makes their driving directions and rolling directions naturally overlap. As a result, movements of this chassis behave the same as they do for a regular fixed axle car, with the bonus of being able to strafe sideways by having each side's wheels drive towards or away from each other.

I began the new robot, dubbed Hex Mecanum, by making a frame to mount the motors to. Since 50:1 N20s worked well on Mini Mecanum, I opted to use them on this robot too. One valuable piece of feedback I received when sharing the robot idea was to include suspension on the wheels to ensure they all stayed in contact with the ground to work properly. For this, I designed a simple see-saw mechanism for each triangle side that had a motor on each end and a pivot in the middle.

“I WAS INTERESTED IN USING MORE THAN FOUR WHEELS”

For driving the motors, I initially soldered up a protoboard with three TB6612 dual-motor drivers. Unfortunately, its rectangular shape did not fit well on the triangular frame. Wire routing was also →





Above ↑
The Hex Driver board assembled and externally powered

Right →
The moment that gave the inspiration to add a lidar to the robot

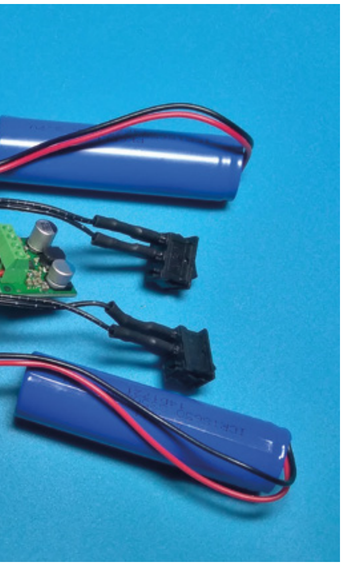
an issue, with cables from motors crossing over others to connect to the correct spots. Not seeing an immediate way to overcome these issues, I put the robot to one side for what ended up being a whole year!

THE ELECTRONICS

I picked Hex Mecanum back up in 2021. I had recently become a product engineer at Pimoroni, helping design and launch products for the newly released Raspberry Pi Pico and RP2040 microcontroller. As part of the role, I was learning PCB design with EAGLE. This reminded me of Hex Mecanum's problems and how a custom triangular PCB with conveniently positioned outputs could solve them.

Around this time Pimoroni was developing PGA2040, a breakout board for the RP2040 that included the essential components to make the chip run, and exposed all 30 GPIO pins. This was an ideal fit for the project as it was small and gave enough GPIO to cover the 24 needed for the six motors and encoders, with some spare for extras. Plus, I was already becoming quite familiar with programming the RP2040.

I started work on the PCB, called Hex Driver 2040, creating the schematic with all



Above
The power electronics, showing the batteries and regulator

the components needed. To keep the size down, I swapped the motor drivers out for DRV8833 breakouts from Pololu. Then came the board design stage, where I quickly found that using a square alignment grid on a triangular board left me with no space to route all the traces I needed, even though I could see that there was space. My solution: 'route' the board in Fusion 360! I imported all the component positions and then drew lines for each trace manually, applying angle and distance constraints to them to get the layout I wanted. I could even apply symmetry operations to duplicate similar sections.

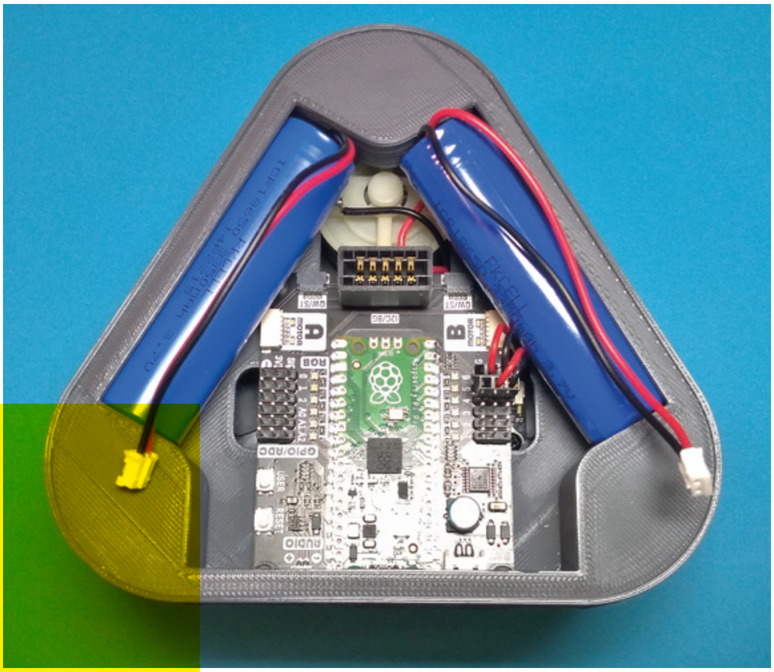
Once finished, I hoped I could just export the lines from Fusion and import them into EAGLE as traces, but I could not find a way to do it. My workaround was the laborious process of manually copying each line's coordinates from Fusion and pasting them into EAGLE's trace properties. This was time-consuming to say the least, but the end result was worth it. To avoid having to repeat this in the near future, I included alternate connector landings to support other projects that could benefit from a triangular board. This meant the final hex driver was capable of driving 6 × DC motors with encoders, or 3 × stepper motors, or 12 × servos, with up to 7 × I2C sensors or a single SPI sensor.

I ordered the completed hex drivers from JLCPCB, which arrived just in time for PGA2040's launch. I soldered one up straight away and did some simple motor movements to test it worked as intended, which it did! I then started converting

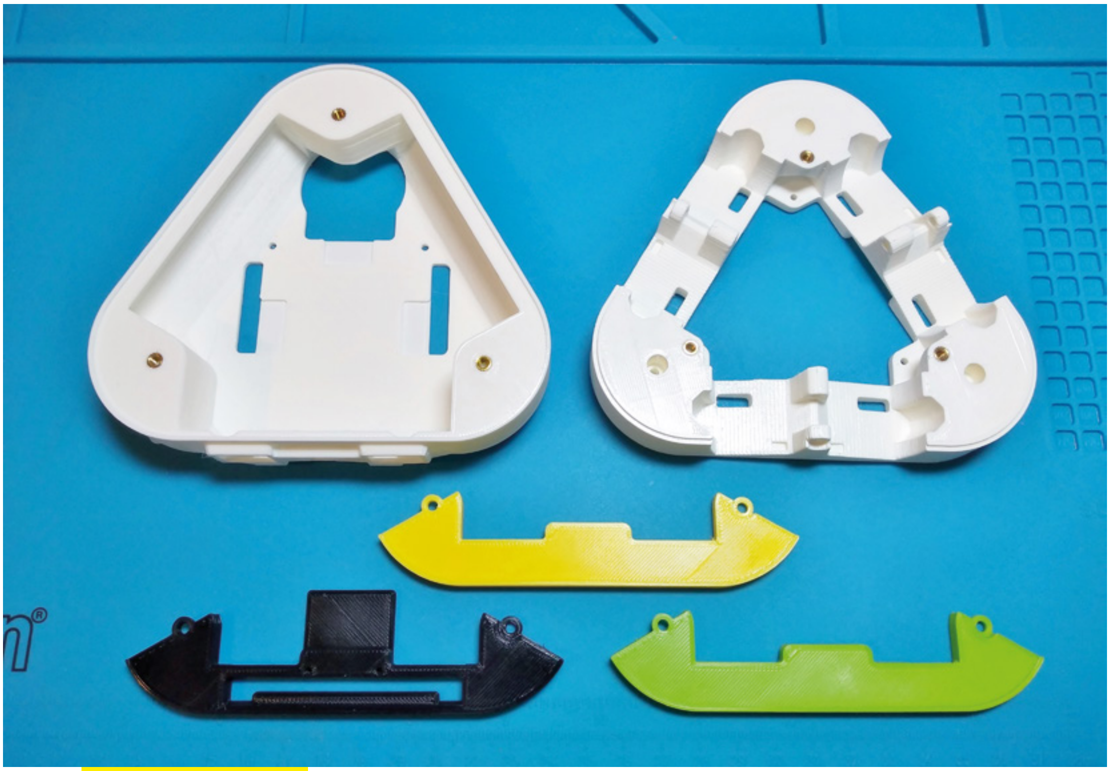
"I PUSHED MY ROBOTICS AGENDA ONTO PIMORONI"

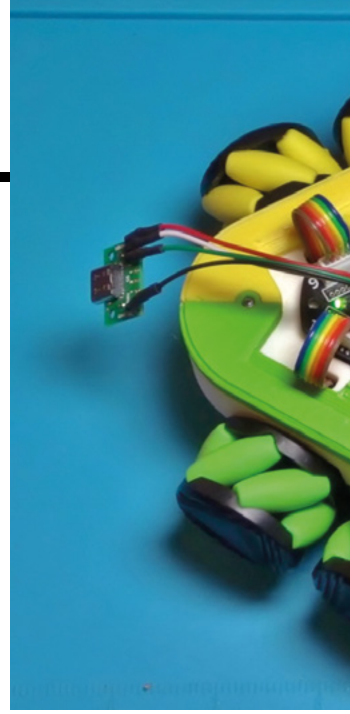
my Mini Mecanum software over to it but hit a snag when it came to motors and encoders with RP2040. Realising I would need to write my own support to get the functionality I wanted, I pushed my robotics agenda onto Pimoroni so that I could implement it on work time. This worked, resulting in the Pimoroni flavour of MicroPython receiving motor and encoder support, and a bunch of robot products getting launched too! This wasn't a quick process though, delaying Hex Mecanum for another year.

I flashed the hex driver with pirate-brand MicroPython and used my own motor feedback example to control the motors. Then, once I finished porting Mini Mecanum's code and assigned each wheel to its correct location, I connected the chassis up to a bench →



Left
The Inventor 2040 used to drive the lidar. This is actually a pre-production, non-Pico W version, but final robot has a Pico W version





Above ↑
The lower half of the robot assembled, with power system installed

Left ←
Some of the final prints that were done

power supply, and the motions 'just worked'. The chassis was able to translate in various directions, turn on the spot, and move and turn simultaneously, all as I had hoped.

Right ↗
L-Hex render from Fusion 360

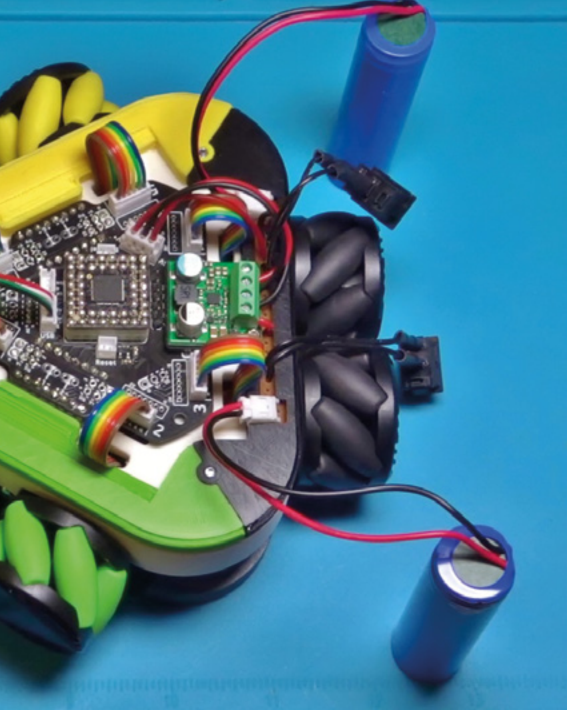
THE ROAD TO CAMJAM

Not long after that success, the Cambridge Raspberry Jam was announced. Being the first event in a long while and the 'première' Jam, I immediately registered for the Show & Tell with the plan to display some robots, including Hex Mecanum. The robot was far from complete though, needing a power source, a chassis that enclosed its internals, and, ideally, a feature that would benefit from its omnidirectional motion.

For power, I went with two 18650-style rechargeable battery packs connected in series. These fit neatly along the sides of the chassis, and gave 7.4V nominal, which was plenty to power a 5V step-down regulator for the hex driver, and could also be used to power the motors directly.

Around this time, Twitter user @NotBlackMagic had found and documented their efforts with a cheap lidar scanner from China. Pimoroni got some in for me to test, and following @NotBlackMagic's information, I quickly got one running from a spare Inventor 2040 W. It needed a motor driver output, and a logic level converter to shift between 3.3V and 5V UART.

While working on the lidar one evening, I casually rested it on top of Hex Mecanum,



and it fit almost perfectly. This was the feature that turned the project from Hex Mecanum into L-Hex!

To complete L-Hex's chassis, I updated the Fusion model with the latest components, tweaking their positions until everything fit well. I then drew a solid shape that enclosed them, which I sliced into sections to work on in stages. These were slowly chipped away at, adding and subtracting material, similar to carving a piece of clay. To check the tolerances of various pieces, I would often slice and 3D-print just the relevant geometry, saving valuable time on what would have otherwise been multi-hour prints. I printed all of L-Hex's parts, going for a white and black colour scheme to match the lidar, with coloured highlights on each side to match the wheels.

With CamJam fast-approaching and software still needing to be written, I opted to simplify matters by treating L-Hex as two independent systems, without any communication between them. The drive system was programmed to perform random on-the-spot rotations, as this seemed safe for displaying on a table.

The lidar system was programmed to show audiences what the robot 'sees' on an RGB LED ring added at the last minute. With these systems working, I performed the final assembly and packed L-Hex up for transport to Cambridge.

CamJam was a great experience, with L-Hex and the rest of my table receiving a good amount of attention. The RGB LED ring proved to be a great addition to the robot, with lots of attendees having fun moving their hands around the lidar and seeing how it reacted. I even had a brief visit from Eben Upton, who said he had been following the project!

After CamJam, I had plans to complete L-Hex's software, but the push to get it ready burnt me out somewhat, so I am giving it a break. If this four-year-long story has highlighted anything though, it's that extended breaks are not unexpected, and often lead to new opportunities that would not exist had the break not happened. For now L-Hex is having a well-deserved rest on a display shelf alongside the rest of my robot collection!

Below
It took some work to get there, but L-Hex was ready in time for the Cambridge Raspberry Jam

"I WOULD OFTEN SLICE AND 3D PRINT JUST THE RELEVANT GEOMETRY"

IN THE WORKSHOP: Truss-rod carving jig

By Andrew Gregory

Keep your neck straight

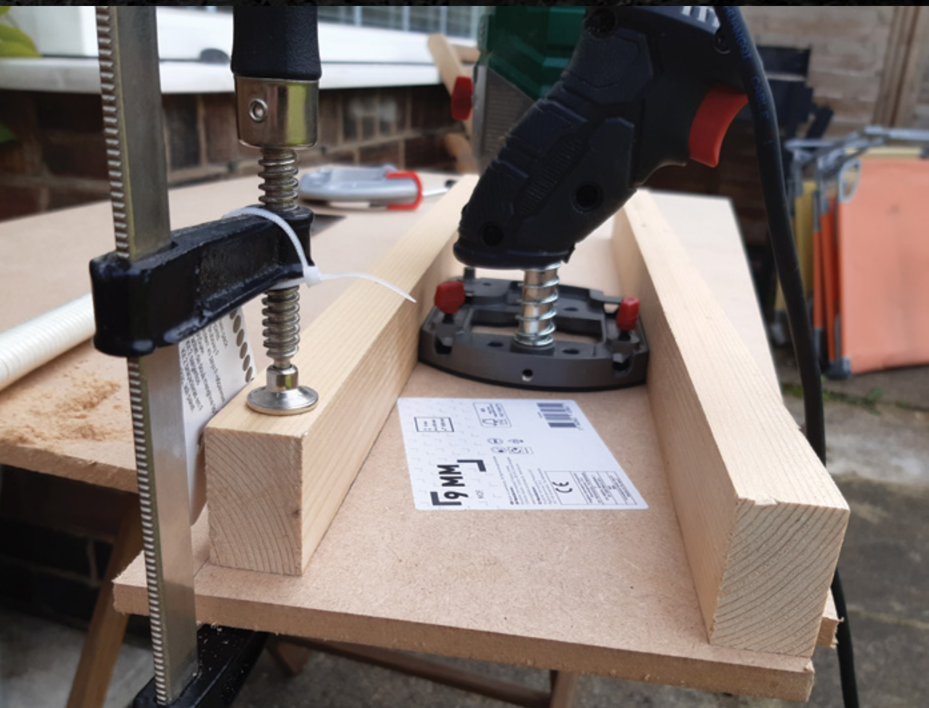
Above The jig will stick to the intended wood with double-sided sticky tape; if I need any more adhesion than that, I've left the base section a little long so that I can add a clamp at each end

A couple of months ago, I decided that I wanted to build an electric guitar. Since then, I've discovered that before you build an electric guitar, you first have to build the tools that you'll use to build the guitar. I'm also finding that the one single most important measurement is the centreline. You can, and people have, installed guitar hardware onto toilet seats, machine guns, shovels, and loads of other objects, but without an accurate measurement of the centreline, it's impossible to get a playable instrument.

Electric guitars have steel strings which, when they're tuned, exert a much stronger pull than the nylon strings found on classical guitars, so anyone making a guitar has to reinforce the guitar neck with a truss-rod.

A truss-rod is an adjustable metal rod that counteracts the natural tendency of a piece of wood to bend when you put strings on it. It's usually in two parts, welded or otherwise joined at one end, and joined by a simple thread and bolt at the other so that you can adjust it to make the truss-rod bend to a greater or lesser degree.

To insert the truss-rod into the neck, you need to carve out a channel in the wood. This channel needs to be as snug as possible, as straight as possible, and it needs to be along the centreline of the neck so that it won't twist when the strings pull on the completed neck. The adjustable end of the truss-rod is going to



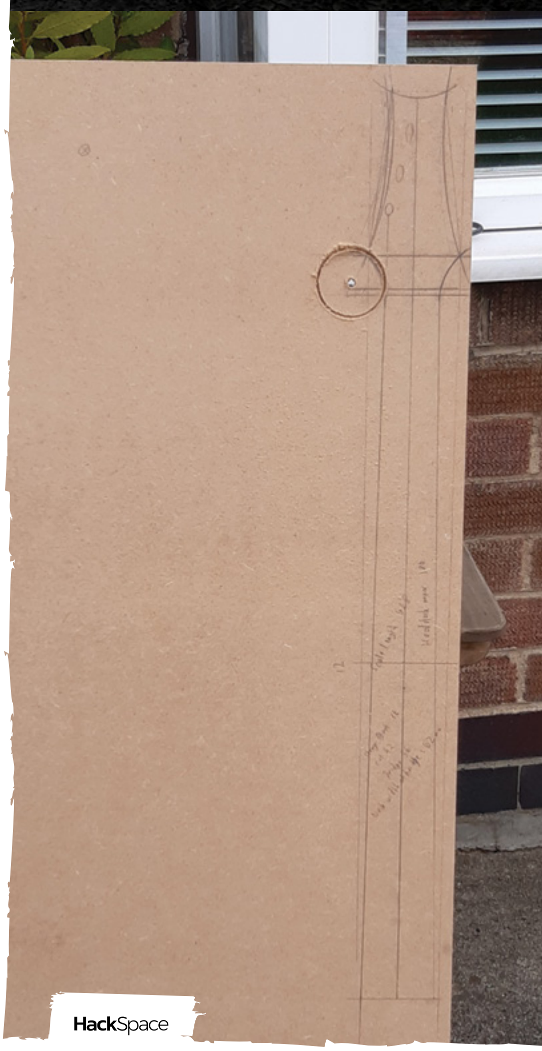
Left MDF isn't the most pleasant material to work with; it blunts tools, it looks dull, and it kicks up a load of toxic dust when you cut it with power tools. The advantage it has is that it doesn't warp, and it's relatively cheap

lie in the centre of the neck, and not quite as far as the nut (later on, I'll drill a hole so that I can access it with an Allen key).

The most sensible tool to use here is a hand router, using a 6 mm bit to carve a channel the exact width of the truss-rod I've acquired from an online auction site. To guarantee a straight carve, I'll need to build a router jig.

You need to carve out a channel in the wood

The jig comprises a base plate, which needs to be long enough to complete a cut the length of the truss-rod, and wide enough to fit the router. It also needs rails along the length of the jig that enable the router to run in a straight line, and only in a straight line, and it needs end stops to prevent the user from accidentally cutting a channel that's too long.
There are plenty of more complicated jigs that will do the same thing, but I've chosen this way because it's simple and it can be made with the minimal tools that are available to me. □



Above By trial and error, I've managed to get the cutting slot exactly the right length for my truss-rod, which should mean fewer trials and less error in future

Left The neck (the long, straight bit) needs to be long and straight; the headstock, where the tuners will go) can be pretty much any shape. Whatever I decide, it's essential that the centreline is marked on everything

FORGE

HACK | MAKE | BUILD | CREATE

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

SCHOOL OF MAKING

KiCad: schematic organisation and hierarchical sheets

How to get your sheets in order before starting a larger project

Jo Hinchliffe

@concreted0g

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!

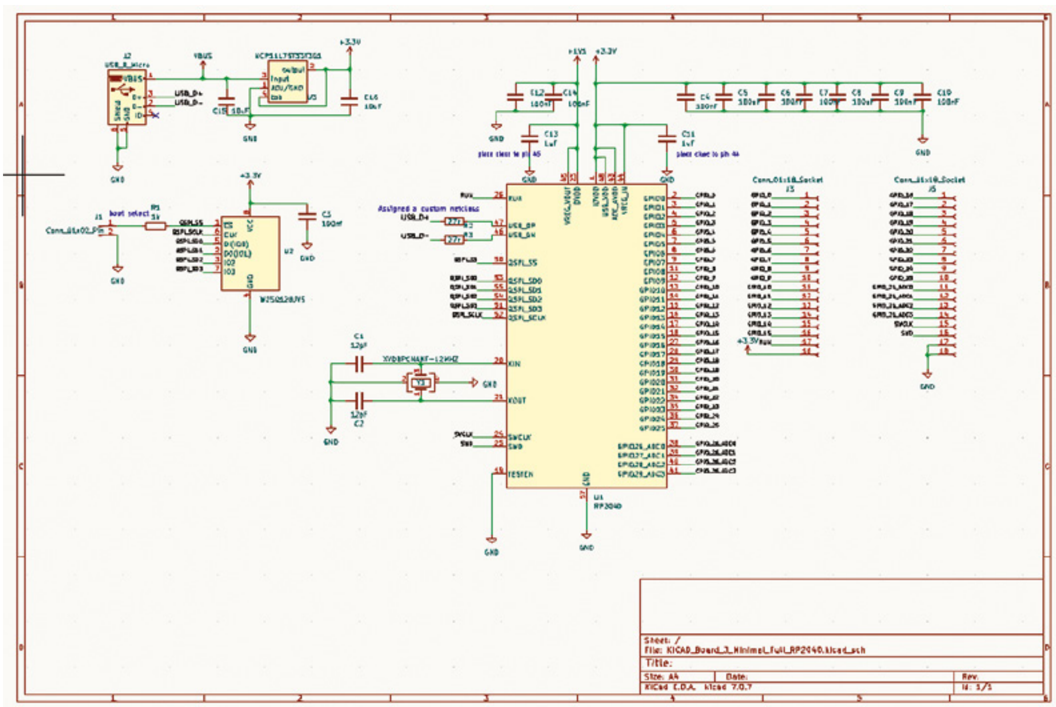


Figure 1 Our schematic from the RP2040 minimal example project

In the last issue, we laid out our largest project so far – an RP2040 minimal layout example and, in the next issue, we want to add to this design. However, the schematic is already quite full and, if we're not careful, the whole thing could become unmanageable. Let's take a look at how we can keep things clean, tidy, and easy to work with.

If you view last issue's project (**hsmag.cc/issue70**), and jump into the Schematic Editor, it looks quite cramped (**Figure 1**). One of the first things we can do is to simply increase the schematic size. Navigate to File > Page Settings and, in that dialog, you can swap the page size, currently at A4. Changing this to A3 will give us plenty more room. Whilst we are in this Page Settings dialog, we can add some detail to the Title Block fields. This is the lower right-hand corner collection of text boxes that list the title, revision issue date, and more. Whilst we don't always practise what we preach, clear titles and revision labels and dates are really useful if you plan to publish your project or use the schematic as part of documentation (**Figure 2**).

With our schematic size increased and neatly titled, we can look at other ways of organising our schematic content. One approach that we have seen,

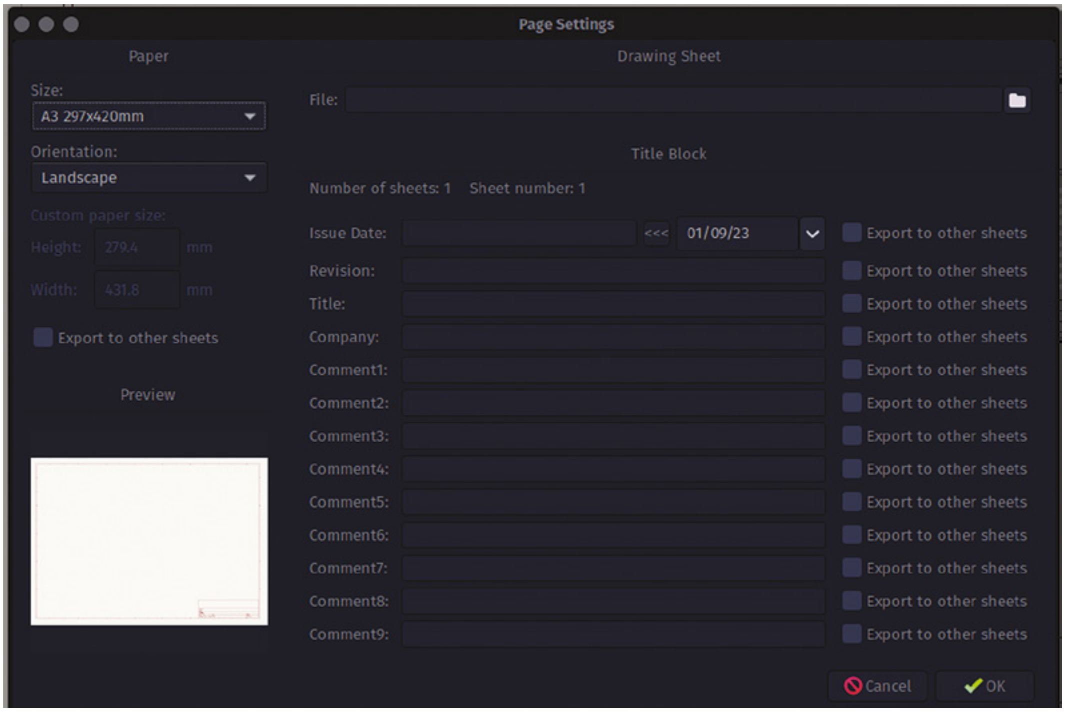


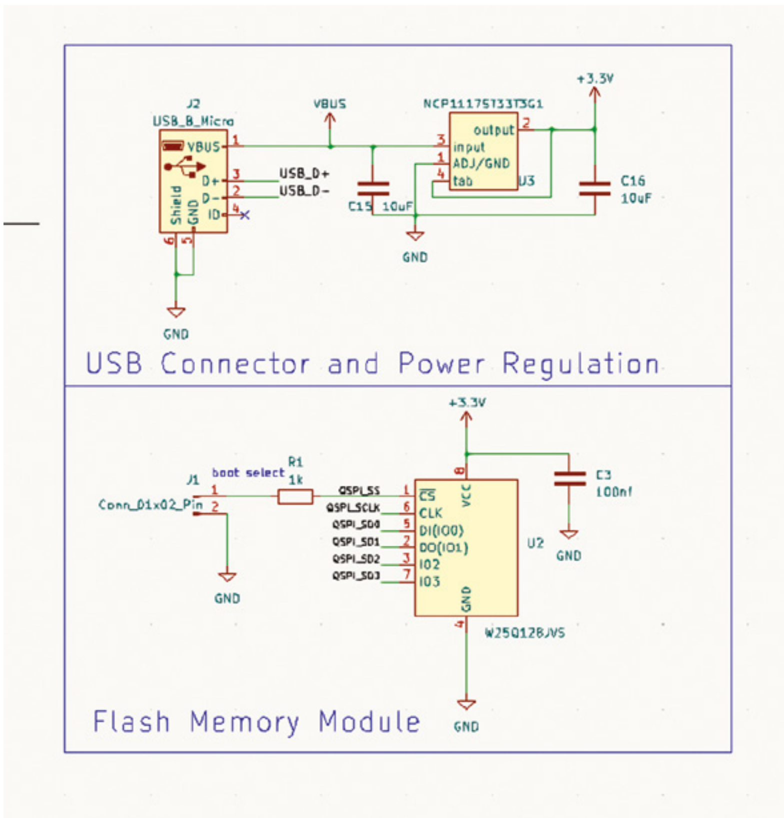
Figure 2 Increasing the schematic page size and adding clear labelling is a good start to keeping organised

Figure 3 Adding boxes and text labels can help organise sections of complex schematics

that some people like, is to create graphic boxes around different subsections of a project's circuits. For this we can use the 'Add a rectangle' tool, again in the lower right-hand side of the screen. You might need to use the selection tool to select and move parts of the schematic into a position where you can completely draw a rectangle around them.

> **Clear titles and revision labels and dates are really** useful if you plan to publish your project

To draw a rectangle using the 'Add a rectangle' tool, you do a single left-click in the start position (a corner of your rectangle) and then move the pointer to the opposite corner position you require. Once the rectangle is drawn, clicking on it with the selection tool, you can either grab the anchor points to change its dimensions, or press **M** on the keyboard to move the rectangle as you would any other object in the Schematic Editor. With a section of your schematic now encapsulated in a rectangle, we can simply use the 'Add text' tool to create and then place →



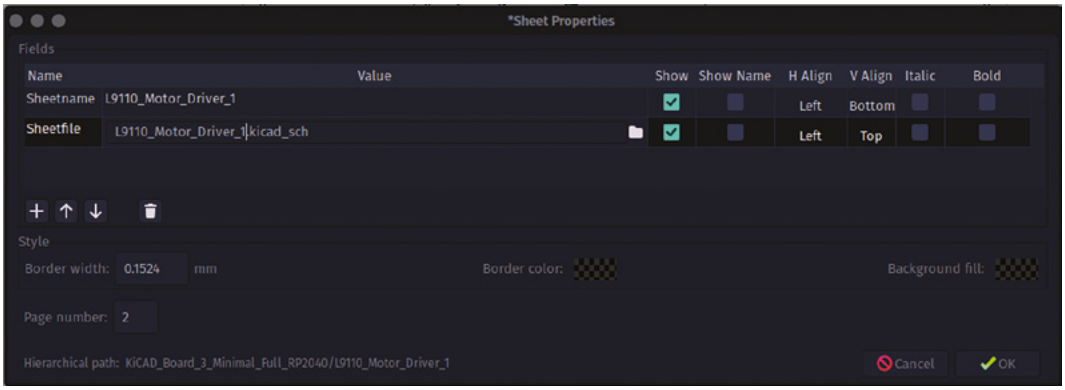


Figure 4 Adding a name for a new hierarchical sheet and a file name for the new hierarchical sheet schematic file

labels into our boxes (**Figure 3**). We did this in the original RP2040 minimal example project, in the last section of this series, when we added notes to place capacitors close to certain points etc. What we didn't make note of is that, new to KiCad 7, we can use any font for our text, allowing us to move away from the default KiCad fonts and bring a little of our own house style to our schematics.

ROVING ROBOT

We're planning to use last month's RP2040 example project as a base for a larger project. We're aiming to make a simple robot rover design with the main chassis of the robot being a PCB with all the essential components mounted on that single board. We want to keep our original RP2040 board project, so we need to make a copy of our project to develop into our robot rover platform. This is one way we can almost use KiCad projects in a modular fashion.

To copy a project in the original project go to File > Save As, then select or create a new folder to save the project to. Insert a new name for the new project copy in the Name dialog box, then click Save. Although not completely necessary, it's not a bad idea to then open the new project folder in a file browser and delete any unneeded files for the new project. For example, if you have a Gerber file folder, these probably won't be relevant to the new project. Any backup files can also be removed as you would probably return to the original source project, if required, rather than use a backup from this new branched project.

The basic premise for our robot is that it is going to have four wheels, all of which are driven by N20-style motors. This gives us options down the line in that it can run with normal style wheels and tyres, but we are also interested in mecanum wheels, which need to be driven individually to create the interesting sideways and diagonal motions mecanum wheels are known for. This means that a primary part of the design is to add four motor driver circuits to our RP2040 board, one for each wheel. Again, considering

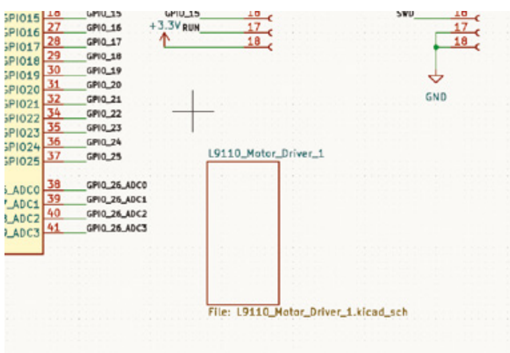
QUICK TIP

If you can't select a section of schematic completely using the selection box tool, select as much as you can, then press and hold the **CTRL** key whilst clicking any missed components or wires.

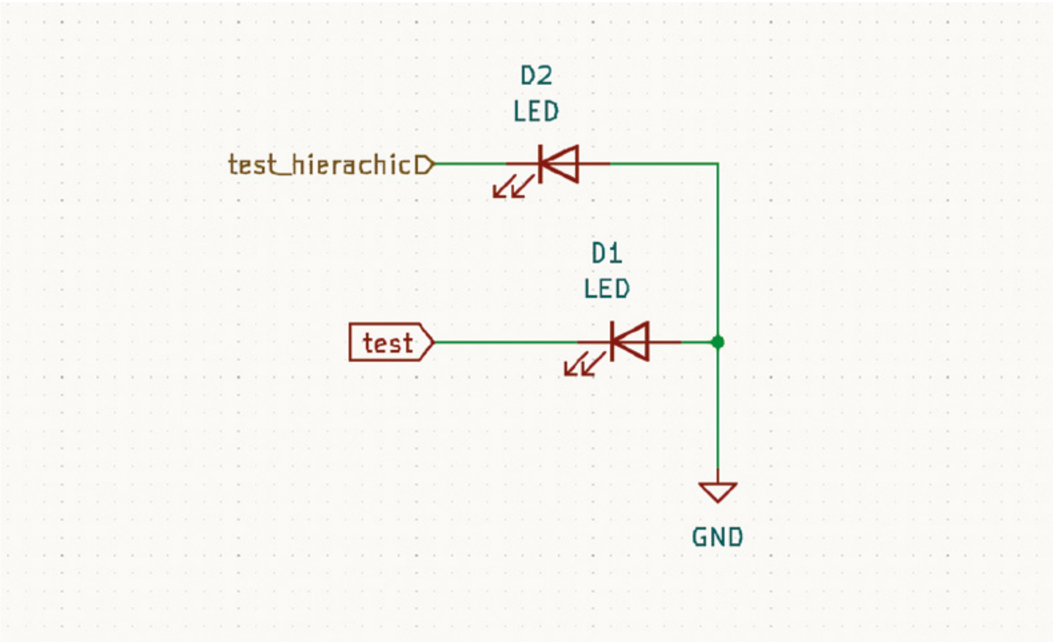
A SPLASH OF COLOUR

You can add images to schematics and, whilst you can place images anywhere on any page, a common use case for this is to add a company logo to the Title Block area. The Schematic Editor supports a wide range of image formats, including PNG, JPG, TIFF, BMP, and many more. Usefully, on import you can rescale the imported image. This is useful as you can import a larger, higher DPI image and then scale it down to fit the box. In the image, the dog logo has been created in Inkscape and exported as a PNG file with a transparent canvas. The image as exported is sized at around 700 × 500 pixels and has been exported at 300 dpi. This means that the image is good enough quality when scaled down to fit into the title block, but is still a comparably small file size at around 16kB.

To add your image to the schematic, simply click the 'Add a bitmap' tool icon on the lower right-hand side of the screen, and navigate to your chosen image file. Once the object is imported, you can move it and scale it whilst maintaining its aspect ratio by dragging the corners of the image bounding box.



Right Our empty hierarchical sheet placed in the original schematic



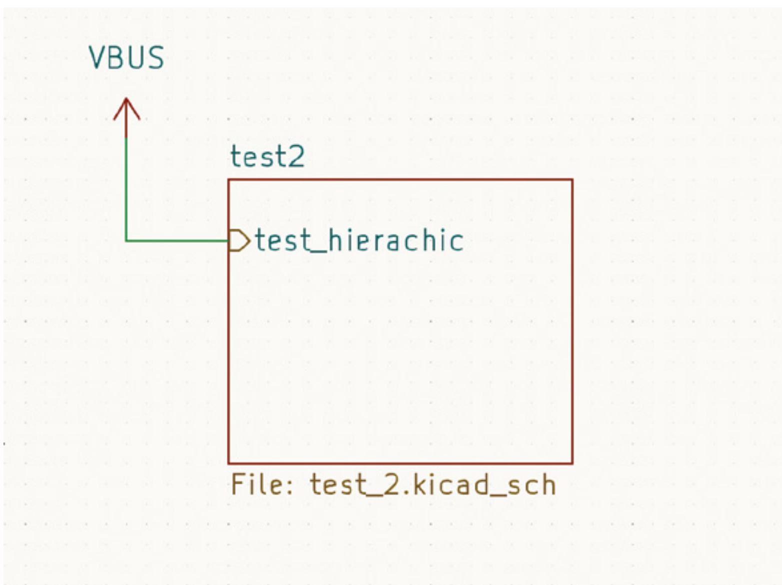
schematic clarity, we could just place the four motor driver circuits into our A3 schematic and wire them using either labels or direct wiring. However, another way we can add content to the schematic is by using hierarchical sheets. A hierarchical sheet can be thought of as a sub-sheet that exists in the schematic below the main page, into which we can insert designs which can be connected to the main top-layer schematic page.

To create a hierarchical sheet, you can use the 'Add a hierarchical sheet' tool icon found in the lower right-hand area of the Schematic Editor, or you can press the **S** key on your keyboard. Either way, you then click and drag to create a rectangle in the schematic. When you left-click to finish the rectangle, a dialog window will launch called 'Sheet properties'. Into this you can put a sheet name, which will be displayed in the rectangle on the main schematic page. We'll call our first sheet 'L9110_Motor_Driver_1'. Under this is an input box for a sheet file name. It will currently be populated with 'untitled.kicad_sch'. This is the name of a separate file that will be created for this hierarchical sheet. Again, we changed this to something appropriate, such as L9110_Motor_Driver_1.kicad_sch. Notice that there are checkboxes labelled 'show' and, by default, they will be ticked (**Figure 4**). This means that, in the main schematic sheet, our hierarchical sheet rectangle will appear with both of these pieces of information listed. It is advisable to show one, or →

> **Another way we can add content to the schematic** is by using hierarchical sheets

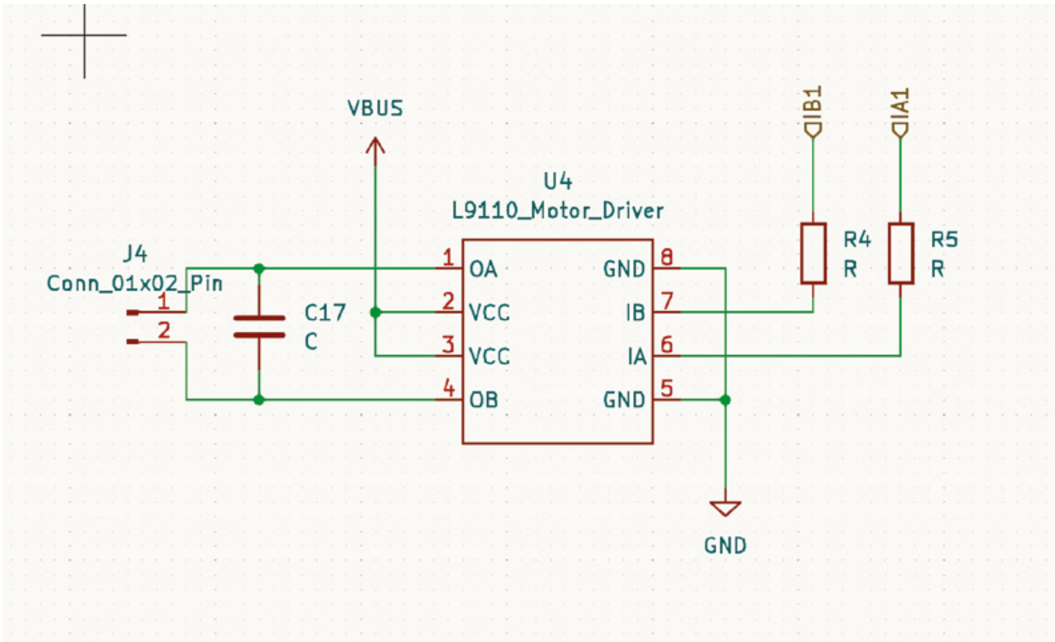
Figure 5 Inside a hierarchical sheet with a design using a hierarchical label and a global label

Figure 6 The test hierarchical sheet object viewed from the main schematic. The hierarchical pin has been connected to VBUS



QUICK TIP

As a hierarchical sheet is a separate schematic file, you can use 'File-page settings' to set the title, page size, and other details, as we did earlier.

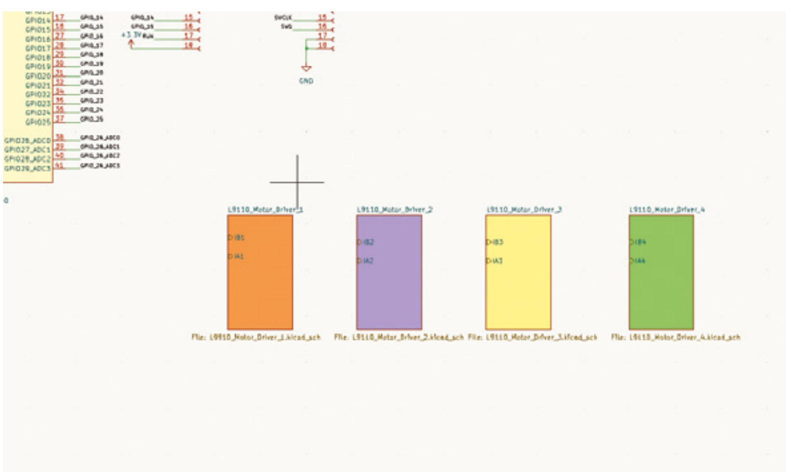


Above The L9110 circuit for each motor. Note the hierarchical labels used on the IB and IA pins

Below Each of our L9110 driver circuits sits in its own hierarchical sheet, and the individual motor driver pins are broken out, ready to be wired into the RP2040

both, of these pieces of information or else you have to navigate into the hierarchical sheet to see what it contains and it can be confusing if you have multiple hierarchical sheets. On a similar theme, notice that you can also play with the appearance of the hierarchical sheet rectangle, increasing or decreasing line width, background, and border colours. Whilst this might seem frivolous, if you end up working in a design with a lot of hierarchical sheets, being able to differentiate them by colour scheme can be an aid to productive working.

To enter your new hierarchical sheet from the main schematic page, you can either right-click on the rectangle and select 'Enter sheet', or you can double-click on the hierarchical sheet rectangle. You should be met with a brand new empty schematic.



You can now begin to place components and create your circuit in the hierarchical sheet. The first thing to note is that anything connected to a global label or net will automatically be connected to those points globally. So if, for example, you add a component and connect it to a ground 'GND' symbol and to 'VCC', they will automatically be connected to those points in the top-layer schematic and will appear as GND and VCC connected when you import those components and connectivity to the PCB Editor. If you did place and connect such a component in a hierarchical sheet, when you return to the main sheet you won't see any connections coming out of the hierarchical sheet rectangle. For general ground and power connections this might be OK, but it may get confusing if you use this approach for all your connectivity. A common way of making connections into and out of hierarchical sheets is to use hierarchical labels.

HIERARCHICAL LABELS

To place a hierarchical label, you can either click the 'Add a hierarchical label' tool icon or press **H** on your keyboard. A Hierarchical Label Properties dialog will appear. Insert a name into the Label field. You will now have a label you can place, move, and rotate and connect into your design. In **Figure 5**, we have made the hierarchical label 'test_hierarchic' and connected it to the cathode end of an LED component symbol. We've also placed a global label test to another

LED component. Moving out of the hierarchical sheet is simple: you can either right-click and select 'Leave sheet' or you can hold **ALT** and then tap the **BACKSPACE** key. Even after creating a hierarchical label inside a hierarchical sheet, you won't see that connection until you right-click over the hierarchical sheet rectangle and click 'Import Sheet Pin' from the drop-down menu. You should now see any hierarchical pins appear aligned with the edge of the rectangle. You can move these pins to any position around the rectangle, and you can connect and wire to these labels as you would any other symbol. In **Figure 6** you can see the 'test_hierarchic' label in the sheet rectangle, and we have wired it to VBUS. Note that you can't see the global label 'test', but that point inside the hierarchical sheet will be connected to any other connections with that global label anywhere in the project schematic. Note again that inside this hierarchical sheet, there is a connection to a GND symbol and, as such, that point is connected to the project's global GND label.

For our robot rover design, we are going to keep it simple and affordable and will use the L9110 motor driver IC as it's adequate for the N20 motors,

> **"One of the great benefits of using hierarchical** sheets is that we can copy them"

affordable, and there is a large amount of stock available on JLCPCB. The circuit around the L9110 is pretty straightforward, with a couple of pull-up resistors and a decoupling capacitor across the motor outputs. Creating a hierarchical sheet for the motor driver circuit, we've used hierarchical labels to create the two signal inputs. We could have opted to have the motor outputs as hierarchical labels, but it was cleaner to add the motor output connector symbols inside the hierarchical sheet, avoiding more clutter on the main schematic.

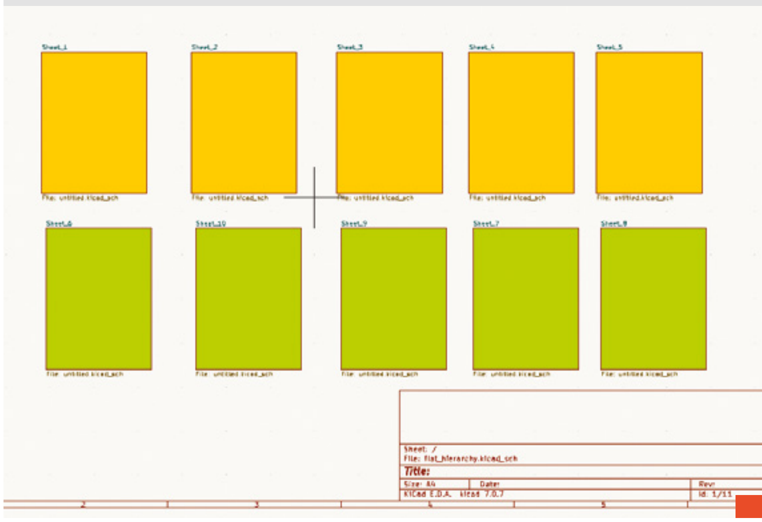
One of the great benefits of using hierarchical sheets is that we can copy them, or copy their contents, quickly to create multiples of similar modules, either within the same project or into other projects. In our robot rover project, we have simply added three more hierarchical sheets and named them sequentially as Motor_Channel_1, Motor_Channel_2, Motor_Channel_3, and Motor_Channel_4.

We then copied and pasted the contents of Motor_Channel_1 into each different hierarchical sheet. Whilst not wholly necessary, it seemed good practice to avoid confusion. So, each time we copied the L9110 circuit, we relabelled the hierarchical pins so that each motor channel was unique. We can then use the 'Import Sheet Pin' function that we used earlier on each of the motor channel hierarchical sheets, and we are ready to wire the pins to our chosen GPIO pins on the RP2040 – either directly or using labels to again keep the main schematic clean and tidy.

Hopefully you have found this guide useful. It's great to think about your practice with KiCad and how you can be organised in a way that suits you. A final good idea is to look through some open-source hardware projects that have used KiCad. You'll certainly find lots of different approaches to organisation and project management.

LARGE PROJECTS

One thing about KiCad is that you can use many different approaches, depending on your needs or your particular way of thinking. One such approach we've seen out there in KiCad-land is to use a flat hierarchy for your schematic layout. This essentially turns the first schematic sheet into a kind of holding sheet where all the hierarchical sheets are held. You can then, in each sheet, use general and global labels so you don't need to draw any connectivity on the main topmost schematic. Although this might momentarily confuse anyone else opening your project, it's actually a brilliant way to organise a particularly large project into specific sections. Of course, as each hierarchical sheet is an individual schematic file, you can simply work on each file individually as needed in development. One tip that we noticed for this approach, or for working with multiple hierarchical sheets, is that if you copy and paste a hierarchical sheet rectangle in the main schematic, it will automatically add and increment a number to the name of the sheet. So 'sheet' would become 'sheet1', then 'sheet2'. We even noticed that if you created a first sheet called 'Something_1' and copied and pasted it, the clone would increment to 'Something_2'.



Greetings card paper circuit

Paper circuits are a great way to teach basic electronics and gain an understanding of how circuits function

Nicola King

@holtonhandmade

Nicola King is a freelance writer and sub-editor. She tries her hand at most crafts, and a quick dabble in electronics has proved easier than she thought.

Perhaps one of the most basic circuits you can create, a paper circuit is a low-voltage electronic circuit created on paper or cardboard, as opposed to on a PCB. The beauty of them lies predominantly in their simplicity, and the fact that you only need a few modest components in order to build one. Instead of wiring, you create a 'trace' using a conductive material as a substitute for the wires that would normally connect components on a breadboard or PCB.

This simplicity makes paper circuits a fantastic way to either investigate circuits yourself, or to teach the fundamentals of electronic circuits to others, especially children who might be taking their very first steps into learning about circuitry.

Right
We've only used one LED here, but you could add many more to a design. Be 'LED' by your own creativity

In this tutorial, we are going to create a greetings card with a simple circuit on the back, illuminating the card and, hopefully, impressing its recipient. You really don't need a lot to get going with this project; it's quick to make, super-cheap, completely unique to you, and arguably more gratifying than buying a card from a shop. If you are of a more artistic persuasion, it can also act as a way of giving vent to your creativity, so feel free to blend that into your make, as this tutorial is less about the art per se and more about the circuitry. Let's shed some light on paper circuits.

STEP 1 DESIGN THE FRONT OF YOUR CARD

We have used here a pre-scored 5 × 7 inch card (225 gsm). Adorn as you wish, making sure you know exactly where you want the LED to shine through. Ours is a simple 'New Home' card, and we used a craft knife to make a neat cut around the hole for the LED. We then attached a small piece of vellum on the rear of the hole to 'soften' the light which will come through it, but parchment paper would work just as well. **Figure 1** shows the front of the card, which was very quick to make.

STEP 2 SKETCH OUT YOUR CIRCUIT

Before you commit your actual circuit to your now decorated card, this author highly recommends creating a 'prototype' on a scrap of card, just to test your components and check that your circuit actually works. If it does work, just repeat on your actual greetings card. (NB: We'll look at trouble-shooting a little later.)

Using a pencil, and ruler if you want very straight lines, draw out a rough idea of where you want the circuit to be on the back of the card. Begin by marking the spot for the LED. We have created a rough rectangle shape (see **Figure 2**), which includes marks where the battery will sit, where the switch will be taped to the card and, of course, where the LED will sit. Make sure to mark out the positive and negative sides of the circuit too. →

OTHER CONDUCTIVE OPTIONS

Obviously, we've used conductive copper tape here, but there are alternatives that can be used in paper circuits that might be worth trying out:

- **Conductive nylon fabric tape**
  A clear advantage of this over copper tape is that it doesn't crack when twisted, so traces won't be broken as easily. Also, as it's more flexible, you can use it on more flexible materials, such as felt, and you can sew through it. So, possibly an option to bear in mind for all those wearable/e-textile projects on your make list. One disadvantage, over copper tape, is that it can't be soldered.
  hsmag.cc/ConductiveNylon

- **Conductive ink**
  You can also purchase rollerball pens containing water-based, non-toxic conductive ink, and design your paper circuits using these. The lines that you draw will dry quickly, and you can be very precise in your drawing. While it may appear that creating a circuit this way is as easy as creating a doodle, be aware that you need to make sure the ink adheres to the particular type of paper you are using – photo paper generally works well. Also, when using ink, it can be trickier to actually attach the components for the circuit you are creating.
  hsmag.cc/ConductivePen

- **Conductive paint**
  Conductive paint traces are another option, especially if you are looking to incorporate an artistic element into your circuit. Plus, they can be used to 'glue' components to your circuit, and it's easy to layer more paint if you find an unreliable connection. However, this medium is not conductive until it's fully dry, and the paint is prone to cracking under strain, so if you bend your work a lot, the traces will likely fail over time, which could be frustrating.
  hsmag.cc/ConductivePaint

YOU'LL NEED

- **Conductive copper tape** (with conductive adhesive)
- **A switch** (optional) **– we used: hsmag.cc/Push-button**
- **LEDs** (hsmag.cc/LEDCard)
- **Sharp scissors**
- **Cardstock**
- **Craft knife**
- **A small piece of vellum** (optional)
- **Your choice of embellishments** – e.g. colouring pens, card topper, glue etc.
- **Pencil and ruler**
- **Clear sticky tape**
- **A coin cell battery** (CR2032)

QUICK TIP

You don't need to restrict yourself to using cardstock – try using a material such as felt to build your circuit on.

Figure 1
The front of our card, with a hole made at the spot we've identified where we want the LED to shine through. Your artwork can literally feature anything that you want to illuminate

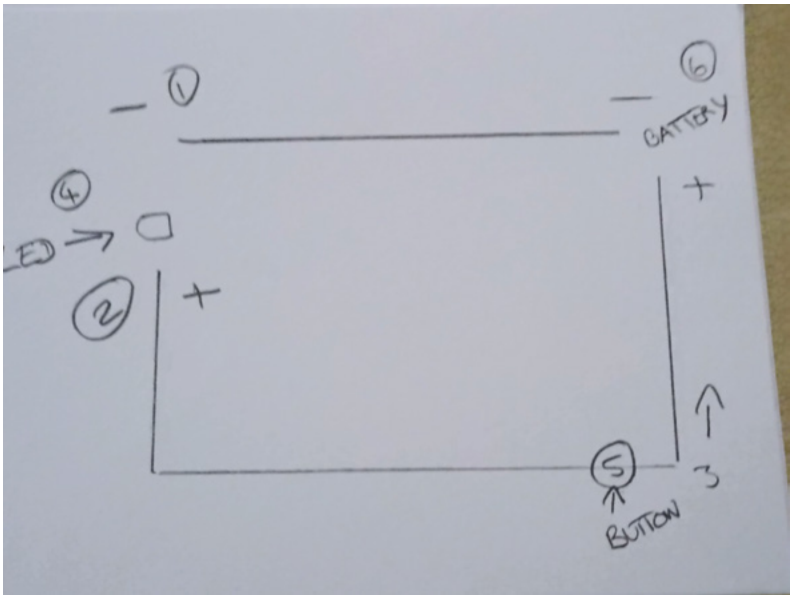


Figure 2
Once you've created the 'map' of where things will go, the rest is easy!

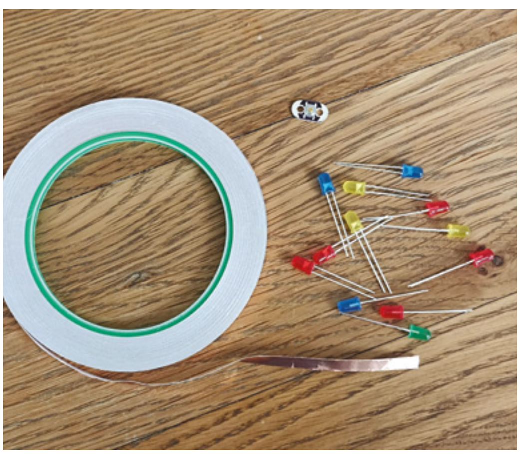
STEP 3 TAPE IT UP

We're using quite a thick copper tape here to make our circuit, mainly for illustration purposes, but you could use thinner widths of this tape if you like. The copper tape is essentially the path the electricity will follow from one side of the battery, to the LED, and on to the other side of the battery, making our complete circuit.

Begin by placing a piece of tape at the top, from where we've marked '1', along to where the battery will sit. Then tape down from the other side of where the LED will be to the point where the switch will sit – neatly folding the tape back on itself at the corner. If you use one continuous piece of tape between each component, the circuit will likely function better. The final piece of tape will sit between the other side of the button switch and the battery – attach the tape, leaving around a 3 cm piece at the end which you will fold back on itself to make a flap to make a connection with the top of the battery (**Figure 3**).

QUICK TIP

Be careful using copper tape, as you can get a nasty paper cut from it if you're not cautious. Plus, we've found it to be quite delicate, and it will tear easily.



Right
A few of your simple circuit fundamentals – inexpensive and easy to find

Figure 3
This copper flap will complete our circuit

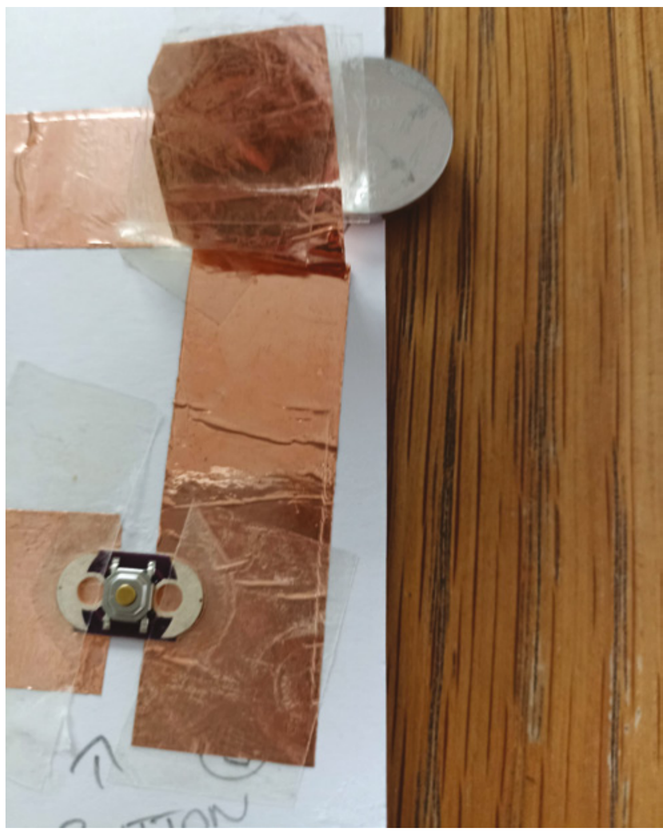
STEP 4 ADD YOUR COMPONENTS

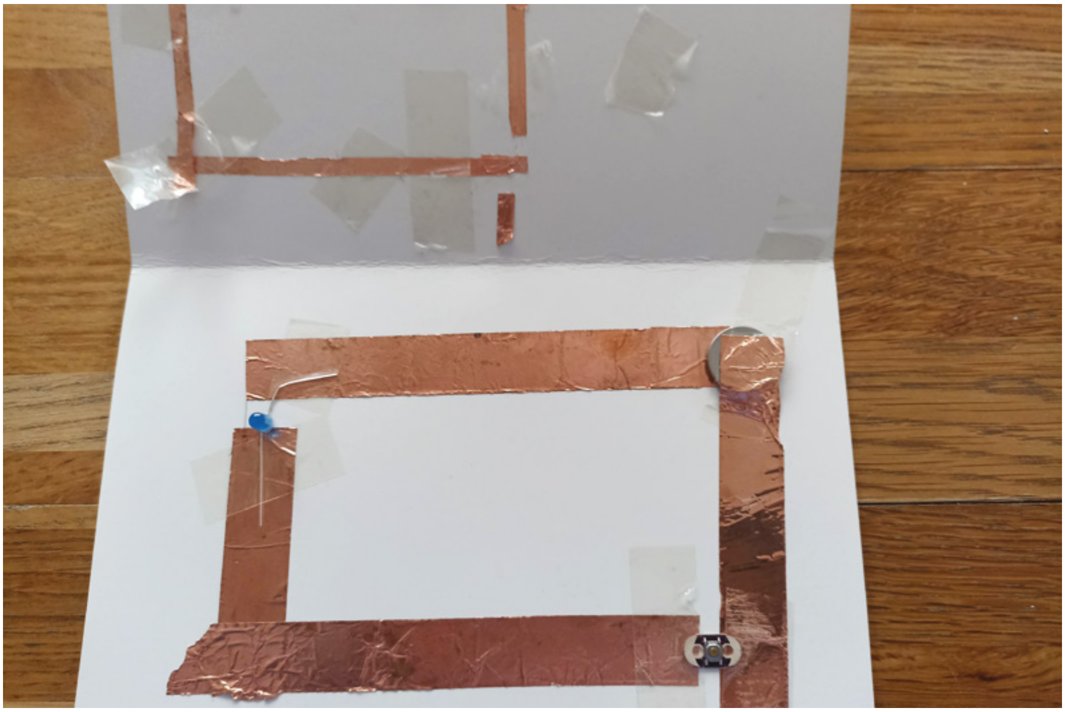
First up, the LED. You'll notice that one of the wires coming out of the LED is shorter than the other – this is the negative leg, while the longer one is the positive leg. This is important in terms of where we put the legs. On our circuit, we've marked the positive and negative sides of the circuit so, after flattening out the legs, make sure the short leg is on the negative side and attach with some tape. Do the same with the longer leg, this time on the positive side of the circuit. Next, tape your button (ours is a momentary push-button) in its allocated space.

That done, we can now power the circuit, and test all is working as it should. Place the battery positive side up in the space marked for it, sitting on the copper tape you applied earlier. Next, fold the copper flap onto the top of battery, press the button, and the LED should light up. Secure the copper flap and battery with some clear tape.

STEP 5 TROUBLE-SHOOTING AND FINISHING

If your conductive tape trace circuit is not working, there are a few things you can check, and it's most





QUICK TIP

If you wish, you can cut your tape in half lengthways to get more value, plus thinner tape is arguably easier to use for intricate circuitry.

Left Messing about with copper tape and components. This first prototype ain't pretty… but then, it's not supposed to be. Fortunately, we had a glowing LED on our first attempt

likely to be an issue with the electrical connections between the components which may not be as strong as they should be. Make sure the legs of the LED are facing in the correct direction, and are really secure against the copper tape – the same applies to the button and the battery. Alternatively, if the copper tape is somehow compromised, maybe torn somewhere, add some more tape over the top of it to patch it up. If your project is still not working, make sure there are no short circuits; for example, the copper tape on the flap could be touching the bottom of the battery.

As a finishing touch, you could also add a card insert into the middle of the card to hide your slightly messy 'workings', and it would also give you somewhere to write your greeting.

Be aware that we are using a very low-voltage battery here, and have not used a resistor (a component that regulates the flow of electrical current in a circuit) to protect the LED. However, if you start creating higher-voltage paper circuits, this is something you'll need to build in.

So, with the card completed, how about taking things a step further? The electrifying tutorial at **hsmag.cc/PianoPaper** takes you through how to create a touch-sensitive piano from a sheet of paper (oh, and a Raspberry Pi Pico W), and it cleverly utilises the fact that the pencil's graphite (a form of carbon) is actually reasonably conductive. Really, the possibilities are huge, so do a stash dive into your crafty paper/card and components, and create some circuits.

COPPER TAPE: A MULTITUDE OF USES

Copper tape, sometimes known as 'slug' tape, is basically a thin strip of copper backed with adhesive. It's popular thanks to its versatility and usefulness for a wide range of tasks, and it's also broadly available from hardware stores and online.

It's available in either non-conductive or conductive forms. The conductive tape, as we've used here, can conduct electricity on both the top and bottom of the tape, which enables electrical connections to be made easily. Apart from being very helpful in creating paper circuits, here are a few other practical applications:

- Wrapping and insulating wire circuits and electromagnetic shielding – copper tape is often used to shield electrical components from damaging electromagnetic interference, and is employed to protect a range of electrical equipment.
- Slugs and snails, generally, don't like it! Hence, a popular way of protecting precious plants is to put some copper tape just below the rim of plant pots to deter greedy gastropods. Basically, if they cross copper tape, their slime reacts with the copper, giving them a minor shock. A great way of naturally dissuading slugs, and avoiding unpleasant chemical alternatives.
- Viruses don't like it either! It's widely believed now that copper has anti-viral, anti-bacterial, and fungicidal properties, meaning that the ability of viruses and nasty bugs to thrive is severely limited when they land on copper, inactivating them. Some hospitals actually have door handles made of solid copper.
- Stained glass – some hobbyists use copper foiling as a guideline when soldering together pieces of glass in stained glass artwork. Tiffany lamps are actually made using the copper-foil method.
- This ingenious maker used copper tape to fashion a homemade antenna: **hsmag.cc/CopperAntenna**.
- Copper tape, with its colourful copper glow, also makes great wall art. These large-scale geometric designs are eye-catching and unique: **hsmag.cc/CopperTapeArt**. Or, you could use some pieces of tape to add a border to frame a picture.

Part 05

Build a Pico Keypad Mole Bop game

Create a game of Mole Bop using a 16-button, light-up keypad connected to a Raspberry Pi Pico using a port expander

MAKER

Stewart Watkiss

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

penguintutor.com

@stewartwatkiss

Understand a way to expand your electronic circuits using I2C. Learn about how to interface with a port expander and see how this can be used for a large number of switches using a switch-array. The first circuit is based around an integrated circuit that you can build on a breadboard. Then, see how a similar technique is used for a 16-button, light-up keypad. Use the keypad to create a fun game of Mole Bop.

01 Using a port expanders

Raspberry Pi Pico has 26 general-purpose input/output pins. For many projects that is more than enough, but sometimes we need more. For a 16-way keypad using a port for each switch and one for each LED, we would need 32 ports.

There are different ways to reduce the number of ports needed. To reduce the number of pins for a keypad, you could use a switch matrix which uses one port for each row and column and scan across them. An example matrix keypad is shown in **Figure 3**. A switch matrix can have a problem with ghosting when multiple buttons are pressed. Sometimes it's easier to use a port expander which increases the number of GPIO ports.

02 Using I2C

I2C is an abbreviation for Inter-Integrated Circuit, which is a protocol for communicating between microcontrollers and integrated circuits. It is a bus technology which allows multiple devices to be connected using only two wires; one for the data and the other for a clock to synchronise the devices. Each device has an address which is used to determine which device to talk to. An example is shown in **Figure 2**.

When using I2C, there needs to be pull-up resistors on the data and clock wires. These can use the internal pull-up resistors available within your Pico; if using devices over a greater distance, external pull-up resistors may be required.

03 Using a MCP23008 port expander

The MCP23008 is an 8-bit port expander which uses I2C to communicate with a Pico or other microcontroller. There are similar devices, such as the MCP23017, which have more ports, but using the smaller MCP23008 means that you can create a circuit on a single breadboard. The example circuit has just two buttons used as a demonstration. This is shown in **Figure 1**.

The address is configurable using pins 3, 4, and 5 on the IC. Connecting all those pins to ground gives an address of 0x20 (hexadecimal value). The

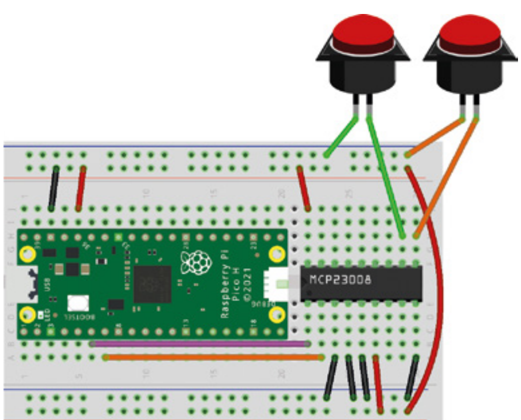
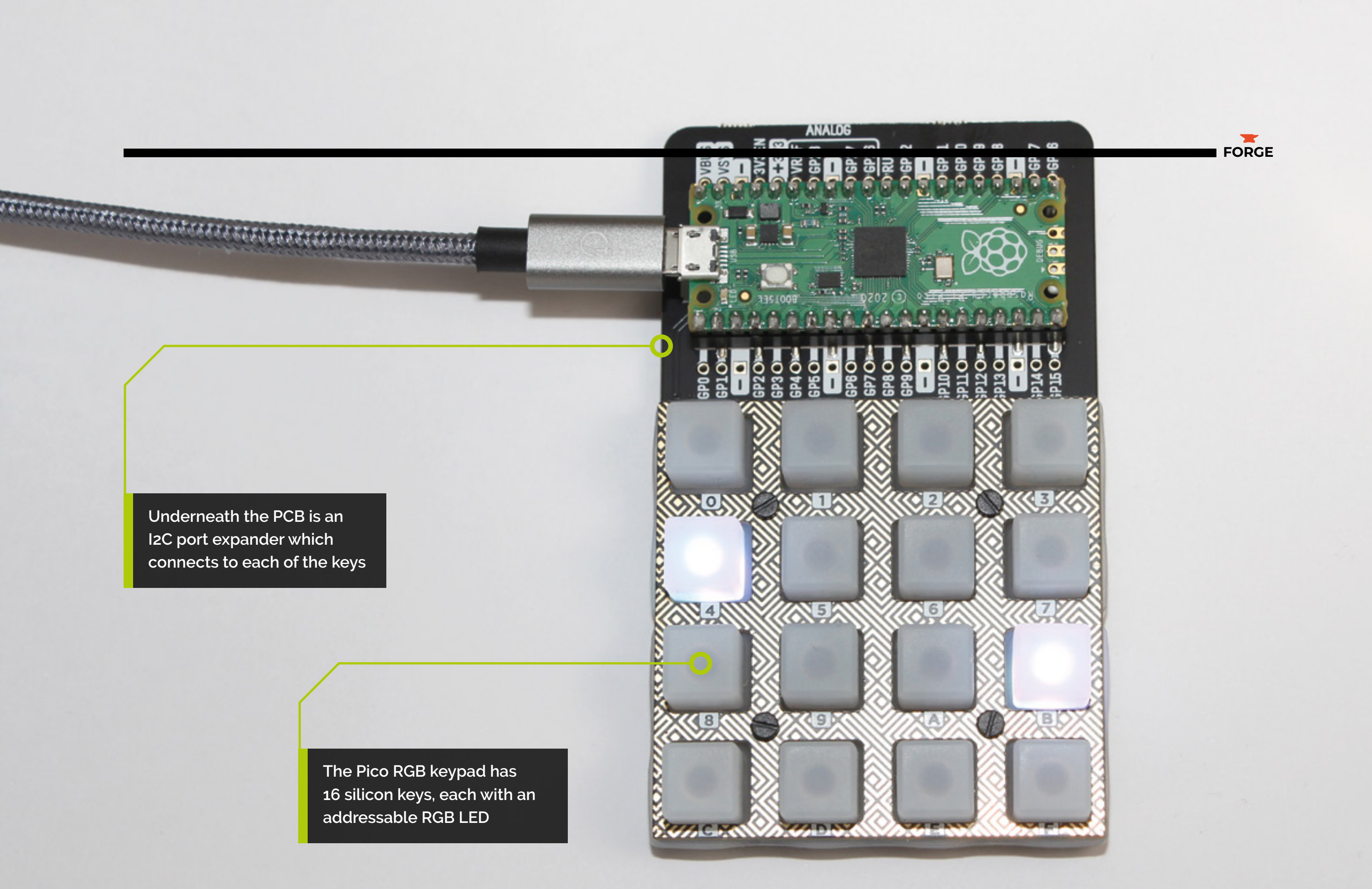


Figure 1 The MPC20008 is a port expander which can be used on a breadboard as a demonstration of how a port expander can be used

You'll Need

- 2 × button switches **magpi.cc/sanwa**
- MCP23008 port expander **magpi.cc/MCP23008**
- Pico RGB Keypad Base **magpi.cc/picokeypad**



Underneath the PCB is an I2C port expander which connects to each of the keys

The Pico RGB keypad has 16 silicon keys, each with an addressable RGB LED

two wires between the MCP23008 and a Pico are the I2C clock and data signals.

04 Communicating with the port expander

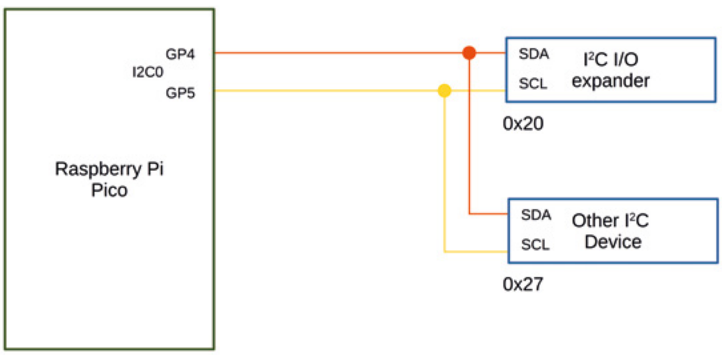
You can use MicroPython to access all the extra ports, almost as easily as if they are local pins. To communicate between a Pico and the port expander, you first need to download the library from **magpi.cc/crankshawnzmp**.

After uploading the library to your Pico, you need to create an instance of I2C and use that to create an instance of the MCP23008 library. The following uses pins 4 and 5 on your Pico and address 0x20 on the port expander:

```
from machine import Pin, I2C
import mcp23008
i2c = I2C(0, scl=Pin(5), sda=Pin(4))
mcp = mcp23008.MCP23008(i2c, 0x20)
```

05 Reading the inputs

The pins can be configured as inputs using the `setPinDir` command. This takes the port number as the first argument and the value 1 for the second argument to indicate



that it is used as an input (0 would be used for output).

```
mcp.setPinDir(0,1)
```

You can also enable pull-ups for the buttons using `setPullupOn`.

```
mcp.setPullupOn(0)
```

Once it is configured, you can query the status using `readPin`.

```
mcp.readPin(0)
```

This gives a 1 when the button is pressed, otherwise a 0.

Figure 2 I2C is a bus protocol allowing multiple devices to be connected using two GPIO ports on Raspberry Pi Pico

06 Pico RGB keypad base

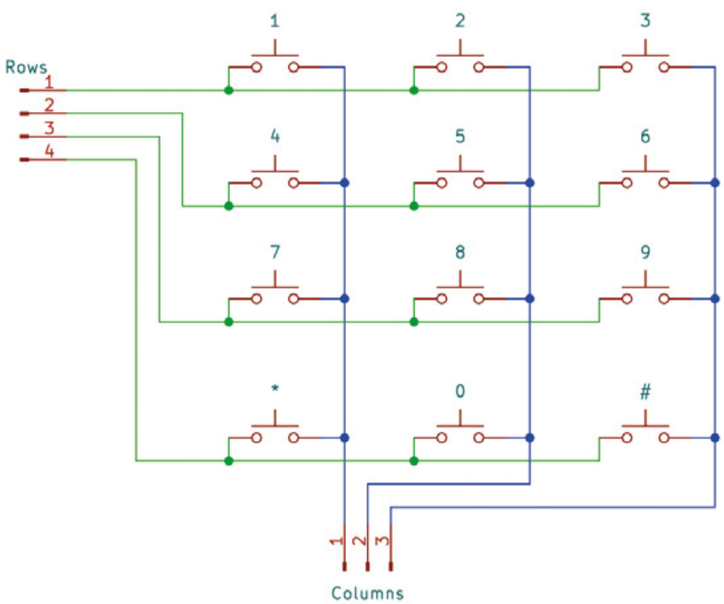
Using a port expander, you could connect that to buttons and LEDs to create a keypad. Alternatively, you could buy a Pico RGB keypad base which has already wired a keypad on a custom PCB with a 4×4 silicone keypad. The keypad uses a TCA9555 port expander on the PCB (**Figure 4**). This is like the MCP23008, but has 16 input/output ports and is a surface-mount package (better for PCBs, but not so good for breadboards). The chip even uses the same I2C address of 0x20. The PCB also includes an APA102 addressable LED for each key. These are like NeoPixels, but use a protocol based on SPI instead of the single data connection that NeoPixels use.

07 Preparing the keypad base

The base needs a little assembly for the buttons, then you plug a Pico using header pins. Instead of installing the drivers separately, Pimoroni provides a version of MicroPython with all the drivers pre-installed. Download the Pico file from **magpi.cc/pimoronipicogit**. Transfer the UF2 file to your Pico after pressing the BOOTSEL button during startup. The keypad and LEDs can be accessed by importing picokeypad and creating an instance of that class.

```
import picokeypad
keypad = picokeypad.PicoKeypad()
```

Figure 3 An alternative is to use a matrix keypad. These have the buttons arranged in a grid of rows and columns, which reduces the number of pins needed



08 Reading the keypad

The keypad is read by using the `get_num_pads` method. This returns a value between 0 and 65535 depending upon which buttons are pressed. There are 16 keys, but we've only used the first eight to simplify the explanation. **Figure 5** shows an overview of how the binary number system works in relation to the keypad numbers.

Inside the microcontroller, the status of the keys is represented by a binary digit for each key. If the key is pressed, then it has a value of 1 and, if it is not pressed, then it has a value of 0. The rightmost digit is key 0 which, if pressed, has a decimal value 1. If button 1 is pressed, that has a value of 2. That goes up to 128 if button 7 is pressed.

09 Shifting the values

To determine which button is pressed, you can read each of the bits in turn and see which are set to 1.

Starting at the right, look at the rightmost digit which is a 0, which means that button 0 is not pressed. Shift the numbers to the right and the rightmost digit becomes a 1, which means that button 1 is pressed. In Python, the shift is done using >> (two greater-than characters) followed by the number of bits to shift.

To perform the comparison, the bitwise AND function is performed. If you compare using '& 0x01' that will ignore all but the rightmost bit.

10 Creating a game

To turn this into a game, a random button is chosen which is then lit up. The player must try and press that button before the timer runs out. If the player presses the button before `lit_duration` is passed, then they score a point. If they don't, then the button turns red and the player loses

Keypad number	7	6	5	4	3	2	1	0
	128	64	32	16	8	4	2	1
Binary digit	1	0	0	0	0	0	1	0

Figure 5 An example of the binary code showing keypad numbers 1 and 7 pressed. The numbers above the binary digits are the decimal equivalent for each key

Figure 4 The TCA9555 port expander is the small square integrated circuit mounted on the rear of the keypad PCB

a life. After all lives are lost, the grid lights up green to show the final score, with each button representing four points.

11 Adding a challenge

To make the game more challenging, each time a point is scored, the time for the next button is reduced by 30 milliseconds. Whenever a life is lost, it is increased by 500 ms to give a reasonable chance to score more points. This makes it possible to get a reasonable score, with the difficulty increasing during the game.

Raspberry Pi Pico does not have a real-time clock, so to work out the time that has elapsed, `time.ticks_ms` is used. This measures time in milliseconds. The method `ticks_diff` is used to test whether the `lit_duration` time has been exceeded.

12 Further development

The game has some areas for improvement. Firstly, it will only run once and then you need to restart it from Thonny. Another `while` loop can be added to keep the game running, along with resetting the score and other variables.

Whilst it is possible to get a range of scores, many players will score around the same amount. This is due to the way that the complexity increases linearly. An alternative is to increase the difficulty using levels instead.

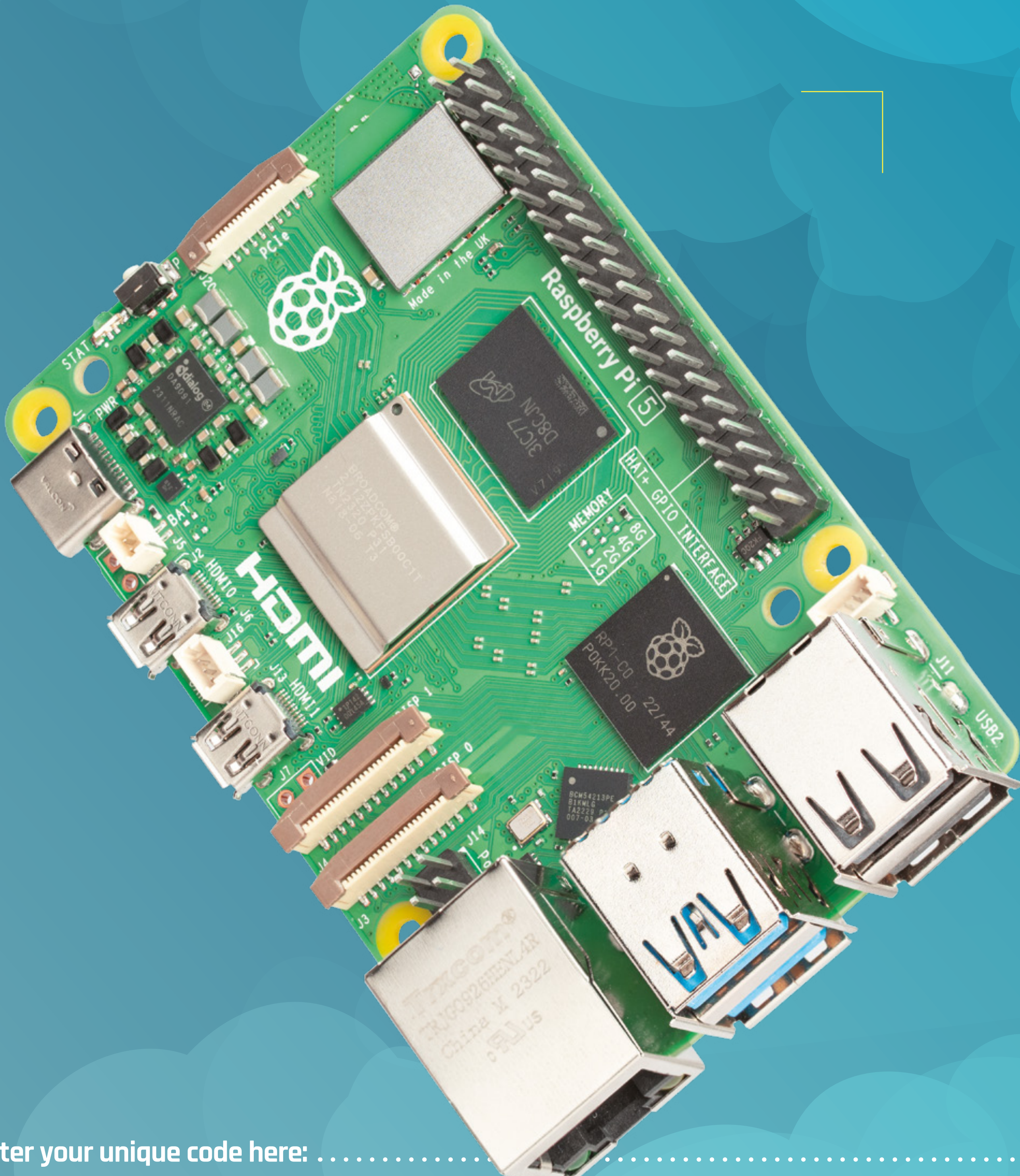
Another improvement would be to have multiple buttons light at the same time. See version two in the GitHub repository for an improved version of the game.

mole-bop.py

Language: **MicroPython**

DOWNLOAD THE FULL CODE: magpi.cc/molebop

```
import time
import random
import picokeypad

keypad = picokeypad.PicoKeypad()
keypad.set_brightness(0.5)
NUM_KEYS = keypad.get_num_pads()

lit_button = random.randint(0, NUM_KEYS - 1)
lit_time = time.ticks_ms()
lit_duration = 1000

score = 0
lives = 3

def all_off():
    for i in range (0, NUM_KEYS):
        keypad.illuminate(i, 0,0,0)

while True:
    # Check to see if the time has expired
    now_time = time.ticks_ms()
    if (time.ticks_diff (now_time, lit_time) > lit_duration):
        # light it red for missed
        keypad.illuminate(lit_button, 255, 0, 0)
        keypad.update()
        time.sleep (1)
        lives -= 1
        if (lives <=0):
            break
        # make the next one a little easier by adding 400ms
        lit_duration += 500
        # choose new button
        lit_button = random.randint(0, NUM_KEYS - 1)
        lit_time = time.ticks_ms()

    # Turn LEDs off and the lit one on
    all_off()
    keypad.illuminate(lit_button, 255, 255, 255)
    keypad.update()

    # scan keys to see if the lit_button key is pressed
    button_states = keypad.get_button_states()
    for i in range (0, NUM_KEYS):
        if i == lit_button and button_states & 0x01 > 0:
            score += 1
            lit_duration -= 30
            lit_button = random.randint(0, NUM_KEYS - 1)
            lit_time = time.ticks_ms()
        button_states = button_states >> 1

print ("Score {}".format(score))
all_off()
if (score > 64):
    score = 64
for i in range (0, score / 4):
    keypad.illuminate(i, 0, 255, 0)
keypad.update()
```



Enter your unique code here: ....

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 × 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.

3D-printed linkages

Get moving in the right direction

Ben Everard

@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.

Making a mechanism move is easy – you can add a motor or a handle to turn. However, making things move in the right way, if you don't need simple spinning, is much harder. Converting between a relatively limited set of options for input movement and whatever particular motion your project needs requires thought, planning, and more often than not, a linkage.

A linkage is a series of bars usually joined together by pivoting joints. By varying the lengths of the bars and the way they're joined, we can create incredibly complex movements. Perhaps the most famous complex movement is Theo Jansen's Strandbeest, which uses a linkage to convert the spinning of a wind turbine into a lifelike walking action. Most linkages aren't quite so on display, though. You'll find linkages in many mechanical devices. A few more common linkages are as follows.

Right Linkages are used to transfer the movement of controls on motorbikes

The Ackermann steering linkage can be used to join the front two wheels of a car so that they turn correctly. The challenge here is that the wheel on the inside of a bend needs to turn slightly further than the wheel on the outside. This is because it's slightly closer to the centre of curvature and, therefore, following a circle with a smaller radius.

Watt's linkage was born of the Industrial Revolution. James Watt needed a way of converting the reciprocating motion of his steam engine into linear motion. The solution – Watt's linkage – wasn't perfect, as the resulting motion isn't quite a straight line, but it was close enough for his purposes. It's still used today in the rear suspension system of some cars.

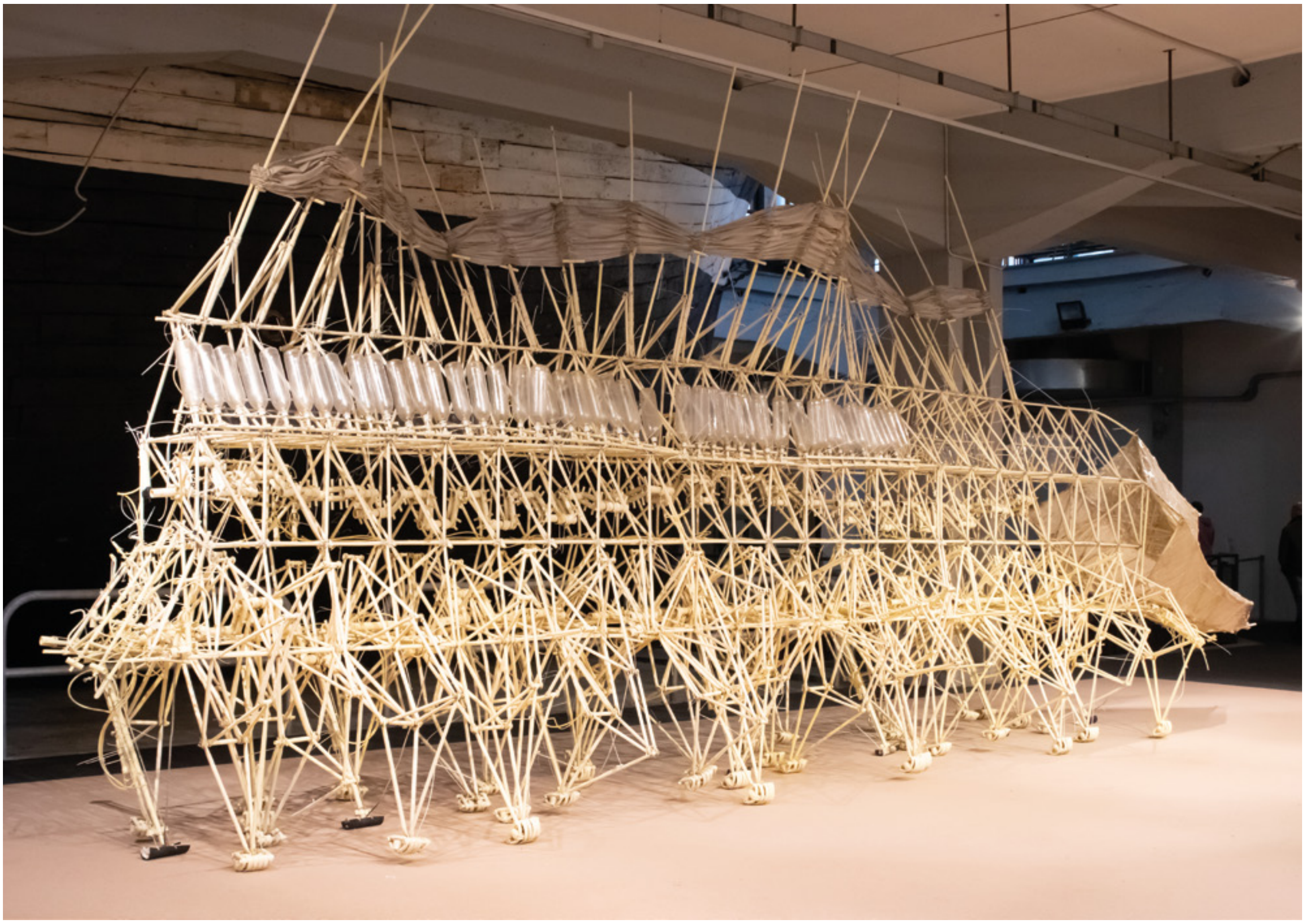
The Chebyshev lambda linkage is another approximation of a straight line. We saw this four-bar linkage in HackSpace magazine issue 66, where it was used to drive the wings of a flying eagle.

FREE AS IN FREEDOM

One of the key features of a linkage is the number of degrees of freedom. Basically, this describes the way in which it can move. A rigid structure has zero degrees of freedom.

A single point, unencumbered by any joints of fixings, has three degrees of freedom. It can move in any direction. You can reduce the number of degrees of freedom by linking it to something. This could be by making it stationary (in the real world, this would equate to it being fixed to a larger body or the ground), or by linking it to another point using a fixed bar.

For example, if you took one point and anchored it to the ground, that would have zero degrees of



freedom. If you added another point linked by a bar hinged around the fixed point, you would then have one degree of freedom.

Many useful linkages have only one degree of freedom. And this means that they move in very predictable ways.

JOINING LINKAGES

Linkages are mostly quite simple to design because they are made up of solid bars. The only real challenge is how to join them together. Usually, you want to join bars together with hinge joints that let them rotate in one axis. Generally, these want to be strong, low friction, easy to print, and small. This is quite a challenging set of requirements, and there are a few options with different pros and cons. The challenge usually comes down to the pin that joins the two parts of the hinge together. Is it part of one of the bars? A separate piece? Something additional that's not 3D-printed? Let's look at the options:

> **Linkages are mostly quite simple to design** because they are made up of solid bars

- **Attached pins:** You can simply print your linkage so that the pin that goes through the hinge is attached to one of the bars. This isn't particularly common because it's hard to secure the other bar to the pin in a satisfactory way. The pin is also the weak point, and this means that if it breaks, you have to replace the entire bar, not just the pin.

- **External pins:** In this method, each bar has a hole and you also print some pins to fit through. It's quite common to make these snap-fit, but →

Above
Though not 3D printed, the Strandbeest is one giant linkage that walks down the beach
Credit: Nigel Hoult, CC-BY-2.0

Above
A four-bar linkage gives this box a slightly unusual hinged lid

we've found these to be a bit tricky in practice. PLA is quite brittle so, while it can work, it's often a very fine line between designing something that has enough overhang to stay in place, while not requiring too much bend that it breaks. There is also no good print orientation for these. If you print them vertically, the layer lines go around the pin, which can make the snap-fit mechanism delicate. If you print them horizontally, the layer lines go along the pin and the joint can have a lot of friction. This isn't to say that these can't work – external pins are probably the most popular way of joining 3D-printed linkages – however, there are a lot of compromises that can make it tricky to design a satisfactory part.

- **Machine screws:** If it's tricky to 3D-print the external pins, then the next best option is to make them out of a standard piece of hardware, and that often means machine screws. These can work well, but they need a locking nut or a double nut to avoid coming loose. Also, these will

DESIGNING LINKAGES

If you're already familiar with FreeCAD, the Sketches tool can help you design linkages. It's not really intended for this purpose, but it's a 2D tool with an understanding of degrees of freedom and connections.

To do this, first start a new project, then go to the Part Design workbench and create a new body, then a sketch. You can pick any plane for this. This sketch won't directly lead to the final design – it just gives us a space to play with different setups and constraints. As an example, we'll create a parallel movement four-bar linkage.

The first thing you need to do is create four lines with their ends linked by coincident constraints. They should be roughly in a square, but we'll set the exact lengths later. The coincident constraints work like hinges. You can click and drag one and it'll move the whole linkages around. Make sure that you haven't accidentally created any additional constraints – sometimes FreeCAD adds horizontal or vertical constraints when you create lines.

If you're not sure how to do any of this, take a look at our *FreeCAD for Makers* free e-book, which goes through the basics of how to work with FreeCAD.

You can now click and drag any point and it'll move the linkage. However, at this point, it'll move about in all sorts of ways – there are far too many degrees of freedom, not least because the links aren't a fixed length. The next step is to add 'Fix a Length' constraints to all the linkages. The exact lengths don't matter, other than the fact that opposite sides need to have the same lengths. We went with 50 mm for the bottom and top, and 20 mm for the sides.

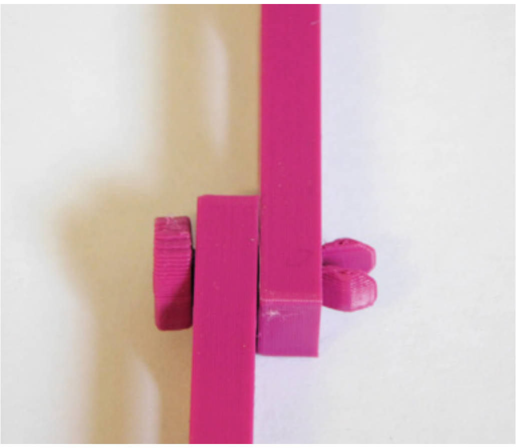
Now, if you click and drag parts, it should move like a rigid object. However, it might still move around. We need to fix one of the bottom corners in place using the Lock constraint. This fixes it in place. Locking one of the corners still allows the whole thing to rotate, though. We could lock the other bottom corner in place, but it's hard to get it lined up properly before this, so it's slightly easier to add a Horizontal constraint to the bottom bar. At this point, there should be one degree of freedom left (unlike a typical sketch, we're not looking to fully constrain it). This mechanism is called parallel movement because if you move one of the sides, the other side moves and remains parallel. You should find that if you move either of the top corners, that's exactly what happens.

This is a fairly straightforward four-bar linkage. There are plenty of others, and some can create complex movements. Once you've created a sketch with the movement you want, you can take a note of the positions, lengths, and joins, and then design the parts for your linkage.

The FreeCAD Sketches tool is a great way of getting a feel for a linkage, but it's not the only option. You could also consider:

- Cardboard and split pins. OK, this isn't the most technically advanced method, but cardboard and split pins are quick and easy to use, and are a great way to get a feel for how a mechanism will move.

- There are some linkage simulators available online, for example **hsmag.cc/desmos**. While these can be useful, you might find them limited, as they tend to be designed to create a specific type of linkage.



Above
The external pin in this join snaps into place, but leaves the joint a little loose

wear quite quickly under heavy use, as the teeth dig into the plastic. They are also very heavy in comparison with 3D-printed parts and this can affect the way in which a part moves.

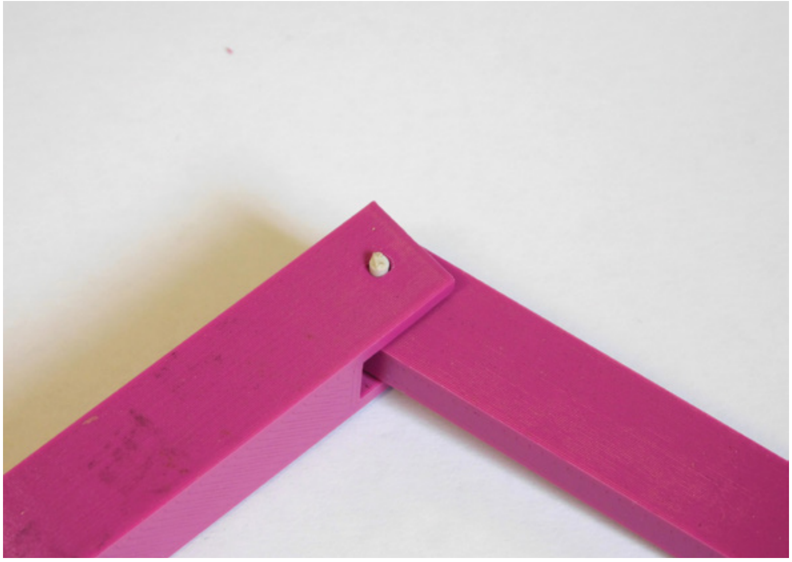
- **Filament:** There is another option that can give you the best of both worlds – use a short length of filament as the pin in the hinge. Filament has a standard and predictable width (typically 1.75 mm, though might be 2.85, depending on your printer). It's cheap, circular, and, because it doesn't have layer-lines, it's strong and has lowish friction. You'll either need to very precisely size your holes or use a drop of glue to hold it in place. When printing the holes vertically, we find 2 mm holes work well because they collapse slightly at the top. For small- to medium-sized linkages, these are our favourite joints. They're easy to design, strong, smooth, don't have much slop in them, and don't require anything other than filament and maybe a little glue.

WHAT FILAMENT?

Generally, you want linkages to be stiff, and PLA is the stiffest common 3D printer filament.

The only other significant consideration might be friction. If you need your linkage to keep running with minimal lost energy, then a material with a lower coefficient of friction, such as nylon, might be worth considering.

It's also possible to use some stock material to make the bars, and just 3D-print the hinges. Plastic, carbon fibre, and metal bar stock are all available, each with different properties, and it would be easy to 3D-print the joints between the bars to make the linkage.



Above
A short amount of filament in a 2 mm hole gives a free-moving strong joint

- **Bearings:** You can get bearing cartridges cheaply. The most common ones are standard sizes for skate wheels, but other options are available. They're cheap and very low-friction, and can be a great option if you need your linkage to run very freely or survive a lot of use. However, they are very heavy in comparison to 3D-printed parts. They certainly have their place, but are probably overkill for most cases.

- **Print-in-place:** Some of the most impressive 3D-printed mechanisms work straight off the print bed. While this can look cool, it's often not the most practical option for linkages because it's very hard to balance friction, sloppiness, and the tolerances you need for print-in-place. That's not to say that it can't work – we've seen some great print-in-place linkages – but it should be used with caution.

Linkages are fantastically useful mechanisms, as they convert one type of movement into another. With electronics, we often work with rotary motion, whether that's from servos or motors. Linkages can let us use this to control all sorts of different hardware. There are lots of standard linkages for common movements, so you may find that you can just borrow the work of some 19th-century industrialist for your next design.

The biggest challenge with 3D-printing linkages is making sure that they move freely without too much slop. We've had success using filament for the hinge, but other things can work well.

Build a Bluetooth remote for a flashlight

Make a remote Bluetooth remote control for the Pico-powered flashlight using MicroPython

Rob Miles

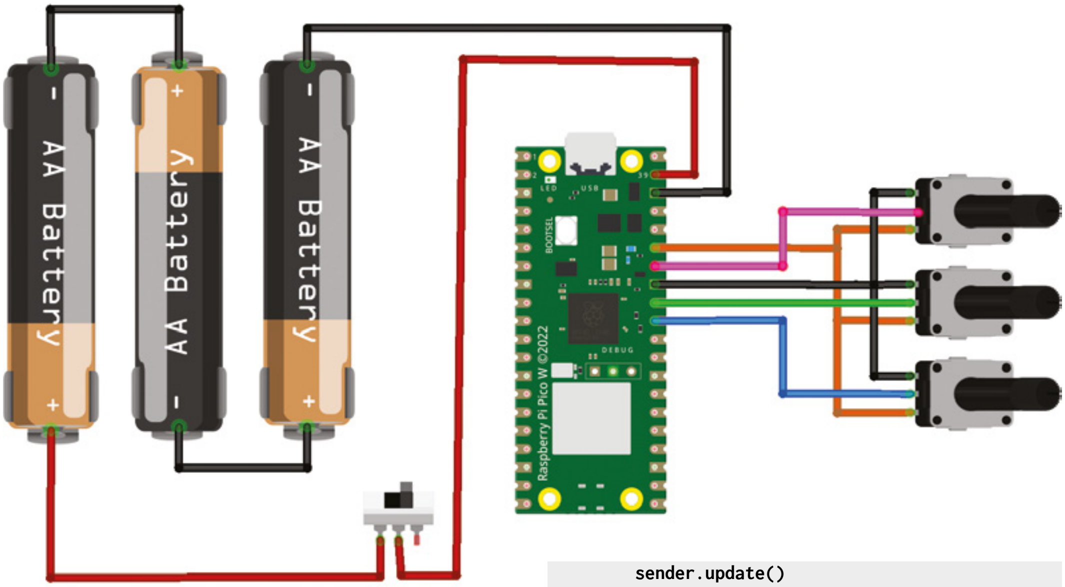
@robmiles

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

Recently, the author created a remote-controlled light based on an old camera flash he found for sale on a market stall. The light contains a Raspberry Pi Pico W programmed in CircuitPython that acts as a web server on a local network. The colour of the light is controlled via the web page hosted by the light. The Pico W also supports Bluetooth, and so in this project, a new version of the light is created which is controlled by a second Pico W-powered device sending Bluetooth commands. Continuing the upcycling theme of the previous article, this one uses an upcycled milkshake box as a case for the controller (**Figure 1**). You can find all the software for the light and the controller on the GitHub site for this project at this link: **hsmag.cc/Bluetooth_Flashlight**.

NO NEED TO PAIR

The Wi-Fi-controlled flashlight is configured with network credentials, and the user must visit a specific web address to connect to the flashlight controller page. The Bluetooth light and remote control use a Bluetooth addressing scheme in which there is no need to explicitly pair two devices that wish to communicate. There is no need to configure either the light or the control. However, if you wanted to use multiple lights with a single controller in the same room, things get a little trickier. You will need to use a different Bluetooth addressing scheme if you want to enable multiple



Far Left Figure 1. The light output has been reduced for the purpose of photography.

Figure 2 The remote control works well with only three alkaline batteries, as the power requirements of the device are not high

connections. However, if you just want to send data from one device to another over short distances, the method described here works very well, and it is easy to change the type and amount of data sent.

CREATING A REMOTE CONTROL

Figure 2 shows the circuit diagram for the device. The three potentiometers are connected to the analogue-to-digital converter (ADC) inputs on the Pico. The input from the ADC is scaled to produce intensity values in the range 0–255. These values are sent via Bluetooth to the light.

SENDING VALUES WITH CLASS

The Pico W in the remote control runs a MicroPython program called **ble_sender.py**, which repeatedly reads the ADC devices connected to the red, green, and blue potentiometers and sends a block of data containing the colour intensity values. If the user adjusts one of the colour settings the updated value is sent out over Bluetooth to the light. The `ble_sender` program uses a class called `BLE_sender`. This is supplied with an instance of the `ColourControls` class which interacts with the potentiometers:

```
def run():
    print("Light sender starting")
    controls = ColourControls()
    sender = BLE_sender(controls)
    while True:
        sender.update()
        time.sleep_ms(20)
```

The statements above create a `BLE_sender` instance and feed it a `ColourControls` instance that reads the control inputs. It then repeatedly calls the `update` method in the sender, which reads colour values from the control. The `ColourControls` class reads the control inputs. The code for the class is shown below.

```
class ColourControls:
   def __init__(self):
        self.red_ADC = machine.ADC(0)
        self.green_ADC = machine.ADC(1)
        self.blue_ADC = machine.ADC(2)
        self.old_red=-1
        self.old_green=-1
        self.old_blue=-1
        self.update()

    def update(self):
        self.red = round(self.red_ADC.read_
u16()/256)
        self.green = round(self.green_ADC.read_
u16()/256)
        self.blue = round(self.blue_ADC.read_
u16()/256)
        if self.red==self.old_red and self.
green==self.old_green\
            and self.blue==self.old_blue:
            return False
        self.old_red=self.red
        self.old_green=self.green
        self.old_blue=self.blue
        return True
```

YOU'LL NEED

- **A suitable box** (you can put the control into anything you fancy)
- **A Raspberry Pi Pico W** (remember to get the 'W' version as this has Wi-Fi and Bluetooth capability)
- **3 × linear 10K rotary potentiometers** Other resistance values will work as well. Make sure you get potentiometers, not rotary encoders. The potentiometers will have only three connections, whereas a rotary encoder will have four
- **A single pole single throw power switch**

QUICK TIP

Using the analogue power and ground connections on the Pico will make the analogue readings more stable.

```
    def pack(self):
        return struct.pack("<BBB", self.red,self.
green,self.blue)
```

The `__init__` method is called when an instance of `ColourControls` is created. It sets up the ADC interfaces on the Pico and sets initial values used in the class. The `update` method is called to request an update of the red, green, and blue values held inside `ColourControls` class. The class stores previous values of red, green, and blue readings and `update` returns `True` if it detects the inputs have changed. The `pack` method returns the byte values to be sent to the light. It packs the red, green, and blue intensity values into three bytes. The `BLE_sender` class calls `update` and then calls `pack` if update returns `True`.

> You don't need to know how the BLE_sender class works **to send different kinds of information using it**

TWO CLASSES ARE BETTER THAN ONE

You may be wondering why two classes are used, when it would be perfectly possible for a single class to do everything. This is to provide flexibility. If you want to send values from a different device (for example you might want to send values from a distance sensor) you can create your own version of the `ColourControls` class (perhaps called `DistanceControls`) to read and pack the data from the distance sensor and use this class with the `BLE_sender` class instead. You don't need to know how the `BLE_sender` class works to send different kinds of information using it. As long as the class you create contains `update` and `pack` methods (and the `update` method returns `True` if the inputs have changed) it will just work. The `BLE_sender` class itself is based on the temperature sensor example in the Pico examples you can find here: **hsmag.cc/MicroPython_Examples**.

POTENTIOMETERS AND ROTARY ENCODERS

There are two types of twiddly knobs you can use with microcontrollers: potentiometers and rotary encoders. The former, which we've used here, create a changing voltage as they're turned. This voltage can be read by an analogue input. They're easy to use and hold a physical memory of their position (i.e. if you power off the project, they will read the same value when you power it back on). However, they can only be turned a fixed amount. Rotary encoders emit a series of pulses as they're turned, often with a soft click as they do. You can keep on turning them forever in either direction and they'll keep on pulsing. They're great for user interfaces (scroll wheels are one type of rotary encoder).

Below The control is set to around halfway. You can tell this because the little plastic finger over the wiper is connected to the centre of the circular track connecting the outer two contacts

RECEIVING LOUD AND CLEAR

Figure 3 shows the Pico from the flashlight being reprogrammed. The Pico was loaded with MicroPython, as the original Wi-Fi-controlled light uses CircuitPython. Below, you can see the MicroPython code that runs when the light starts running. The Pico W in the controller acts as a server, serving out red, green, and blue intensity values which can be used to set the intensity of a light. The program **ble_reciever.py** runs in the light to receive these values. It uses a class called `BLE_receiver` to receive data from the controller and use the data to set the colour of the light.

```
def run():
    print("Light Receiver Starting")
    pixieLight = PixieLight(0.2)
    receiver = BLE_receiver(pixieLight)
    while(True):
        receiver.update()
        time.sleep_ms(50)
```

The `run` function above implements the receiver behaviour. It creates a `BLE_receiver` instance called `receiver`. The receiver is given the device to control, in this case, an instance of the `PixieLight` class. Once the receiver has been created the `run` function repeatedly calls the `update` function of `receiver`. The `update` function creates a connection to a transmitter and then fetches blocks of data from Bluetooth which are then passed onto the `pixieLight`.

```
from machine import Pin,UART
from Pixie import Pixie
import time
import struct
```

```
class PixieLight:

    def __init__(self, brightness, txPinNo=4,
rxPinNo=5):
        uart =
UART(1,tx=Pin(txPinNo),rx=Pin(rxPinNo),
baudrate=115200)
        self.pixies = Pixie(uart, 1,
brightness=brightness)
        self.old_red=-1
        self.old_green=-1
        self.old_blue=-1
        self.pixies[0] = (0, 255, 0)

    def render(self):
        self.pixies.fill((self.red,self.green,self.
blue))

    def set_colours(self,red,green,blue):
        if red==self.old_red and green==self.old_
green\
            and blue==self.old_blue:
            return False
        self.red = red
        self.green = green
        self.blue = blue
        self.old_red=self.red
        self.old_green=self.green
        self.old_blue=self.blue
        self.render()
        return True

    def unpack(self, data):
        value = struct.unpack("<BBB", data)
        if self.set_
colours(value[0],value[1],value[2]):
            self.dump()

    def dump(self):
        print("Red:%3d Green:%3d Blue:%3d"%(self.
red,self.green,self.blue))
```

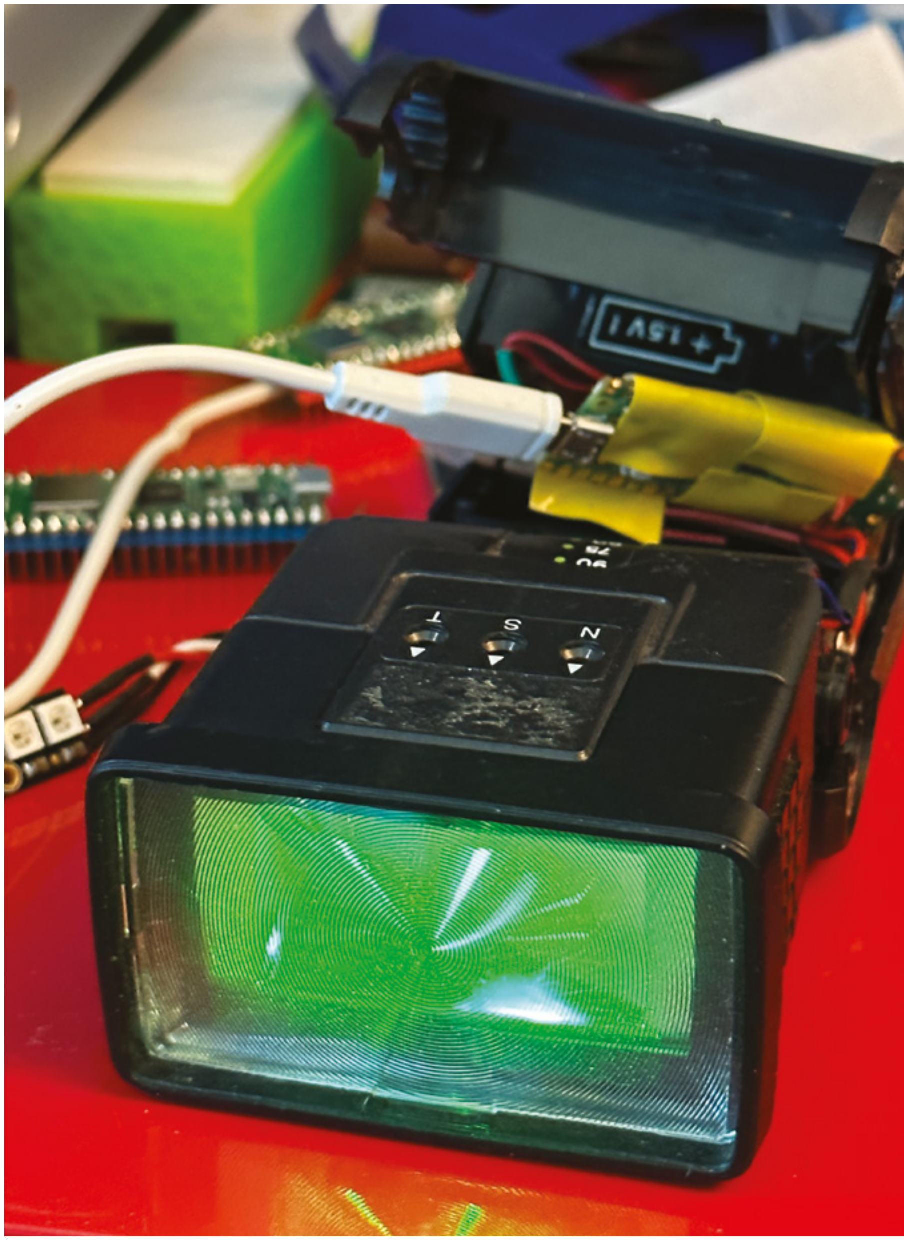


Figure 3 If the light output is not set very bright, the light software can be tested using just a USB connection for the power

The **PixieLight** class is shown above. The receiver will get blocks of data from Bluetooth and pass them to **pixieLight** by calling the **unpack** method. The **unpack** method is supplied with a block of data which must be unpacked and then used to set a new light colour. It sets the colours of the light and, if the colours have changed, it prints the new values to the terminal.

TAKING OUT A CONTRACT

The **BLE_reciever** and **BLE_sender** classes are not aware of the content of the data they are transferring. They just move blocks of binary data from one place to another over Bluetooth. The form and meaning of the data is determined by a 'contract' between the **ColourControls** and the **PixieLight** classes. The **pack** method in **ColourControls** assembles a block of three bytes which contains the red, green, and blue intensity values selected by the user. The **unpack** method in **PixieLight** then takes these three values from an incoming block and uses them to control the colour intensities in the lights it is driving.

Both the classes need to agree on the encoding (8-bit values) and the order (red, green, blue) of the values. Any misunderstandings will result in wrong, →

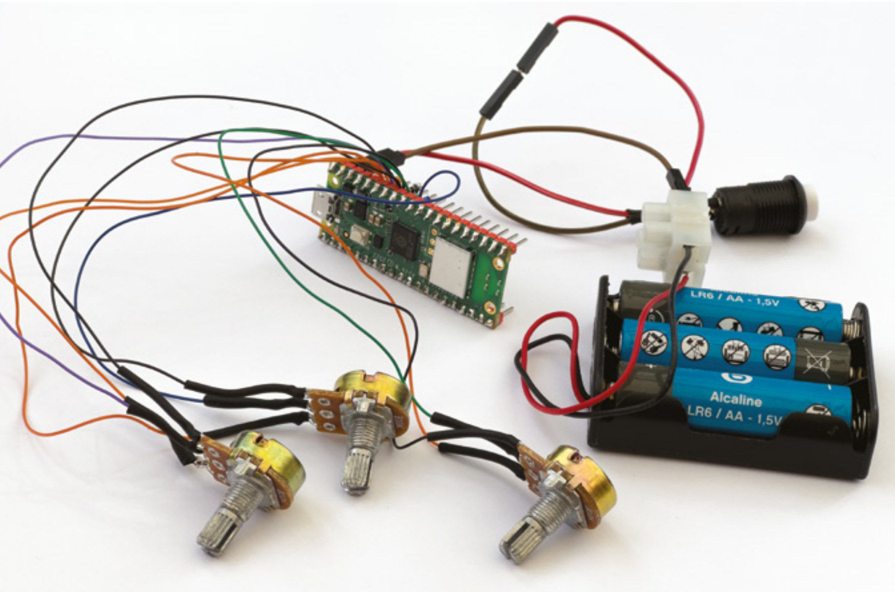


Figure 4
The potentiometer connections were wrapped around the Pico pins and then soldered to the potentiometers

or no colours being displayed. We are sending 'raw' data values like this because Bluetooth limits the size of data packets. If we were sending the data over a network connection with more capacity, we could use JSON (JavaScript Object Notation) to encode the data. JSON data includes a description of the data.

```
{
  "red_byte":0,
  "green_byte":255,
  "blue_byte":0
}
```

The JSON encoded data above could be used to send the colour bright green to a JavaScript application. The name of each value is part of the data, making it easy to understand what is being sent. However, this message contains many more bytes than the three that are used by the light. With low-capacity connections such as Bluetooth, we must make sure that both sides of the conversation agree on the data format.

If you want to use the incoming data to control a different device, you can swap **`PixieLight`** for a class of your own which contains an **`unpack`** method that acts on the data that is received from the **`BLE_receiver`**, but you would need to make sure that the send and receive objects agree on the form of the data being transferred.

TIME FOR AN UPDATE

Most light devices, for example NeoPixels, are easy to use. You just tell them the colour you want, and they light up with that colour until they are given a new colour value. However, the Pixie light used in the flashlight needs to be repeatedly told the colour to be displayed, otherwise it will shut down. To achieve this, we add an **`update`** function to the **`PixieLight`** class. This is called regularly by **`BLE_receiver`** and will refresh the colour settings on the light when required.

```
def update(self):
    millis = time.ticks_ms()
    interval = time.ticks_diff(millis, self.last_
update_ticks)
    if interval>self.update_tick_interval:
        self.render()
        self.last_update_ticks = millis
```

The **`PixieLight`** class contains two member values: **`last_update_ticks`** and **`update_tick_interval`**. The **`last_update_ticks`** value contains the time in milliseconds since the light was refreshed. The **`update_tick_interval`** value contains the interval between refreshes, in our case 200 milliseconds.

CIRCUITPYTHON VS MICROPYTHON

The code to control the original light was written using CircuitPython. However, at the time of writing, there are no Pico W Bluetooth libraries available for CircuitPython. Bluetooth connections are presently only supported by MicroPython. CircuitPython and MicroPython are very similar; they differ mainly in the availability of different libraries for specific devices. If you are serious about becoming an experienced embedded developer, you should aim to be familiar with both implementations. The author originally chose CircuitPython because of the availability of a library for the Pixie lamp he was using. However, he was able to convert this library to MicroPython and then use MicroPython for this project.

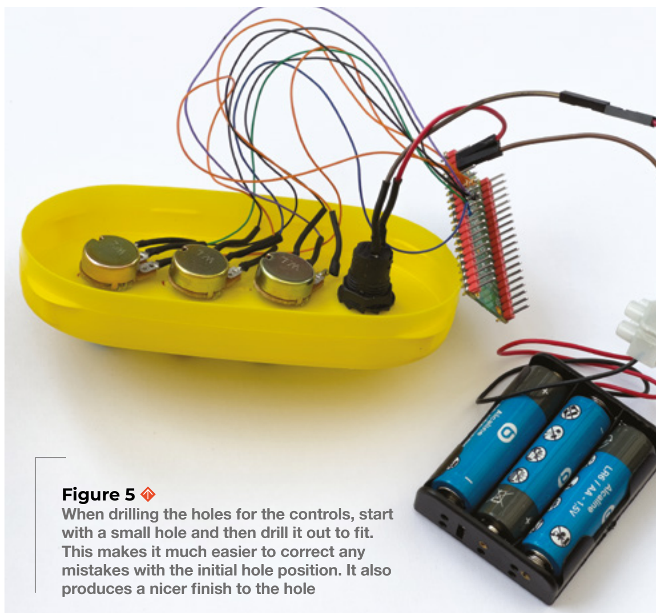


Figure 5
When drilling the holes for the controls, start with a small hole and then drill it out to fit. This makes it much easier to correct any mistakes with the initial hole position. It also produces a nicer finish to the hole

Each time `update` is called, it gets the current time and then works out how many milliseconds have elapsed since `update` was last called. If the interval is greater than `update_tick_interval`, the `render` function is called to display the lights and keep the Pixie shining.

FINAL ASSEMBLY

Figure 4 shows the wiring for the light controller. It was first built on the desk and tested before being transferred to the case. The power is controlled by a push-button which is connected in series with the battery box. Once the circuit had been tested, it was then ready to be put inside the case, which was a box of milkshake which was in no way specially purchased for this project. The author really likes strawberry milkshake.

Figure 5 shows the controls fitted to the top of the milkshake box. The switch was pushed through the top of the box and then connected to the Pico and the battery. The Pico and battery holder were then secured to the inside of the box using adhesive Velcro. The resulting box looks quite pleasing (at least to the author), and the device inside works well. It would be possible (and indeed quite easy) to use the same technique to create other Bluetooth-connected devices.

Above Only a good idea if your device doesn't use radio communications

BOXING CLEVER

The author likes the idea of upcycling, where a project reimagines an existing product. The controller for the light is fitted inside a box that previously contained milkshake powder. Here are some tips for upcycling.

- **Don't use metal boxes for devices that contain radios.** The author's first choice of box was a tin that had previously contained mints. After a bit of thought, this was abandoned as a bad idea, but the mints were delicious.
- **Use longish connecting wires.** The author used to take great pride in how all his wires between components were exactly the right length. This, while aesthetically pleasing, did tend to make fitting the components into the box much harder and more likely to cause damage. Longer wires give a lot more leeway in case construction and can be safely tucked inside the case.
- **Don't always twist all the connecting wires together to make a wiring harness.** This sounds counter-intuitive, but it does make sense (at least to the author). If you have ten wires going to a component, you could twist them all together to make a single cable. However, because all the wires will be different lengths, this means that the weight of all the wires may end up being borne by the shortest cable connection, which will then break. If all the wires are separated, each connection will only have to support the weight of one wire. If you really do want to group cables together, make sure they are supported inside the case.
- **Construct and test your circuit outside the case.** It is very hard to check connections once you have put them in a box. Create the entire circuit and test it first. Then put it in the box.
- **Use heat-shrink tubing to cover connections and make sure that connections have physical integrity.** If you are soldering a wire to a terminal, make sure to hook the wire through a hole to give the connection physical strength, and then use solder to hold the wire in place and provide the electrical connection. Then cover the whole thing with heat-shrink tubing. This makes everything look a lot tidier and means that loose wires won't cause problems. This is particularly important if you are using a metal case, as it prevents signals shorting.
- **Use connectors with components that fit into the case from the outside.** Some components, for example, some switches and lights, are fitted from the outside of the case and then secured internally. In the light control, the push-button power switch is fitted into the case from the outside, and so it is connected using push-fit connectors rather than being connected directly.
- **Don't stick components into your case using a permanent adhesive.** The author is a big fan of self-adhesive Velcro, which can be used to hold processors and batteries in place. The components are held firmly, but they can be removed if required.

Overkill Bluetooth headphones

Modifying our ear candy in the most ridiculous way

Ben Everard

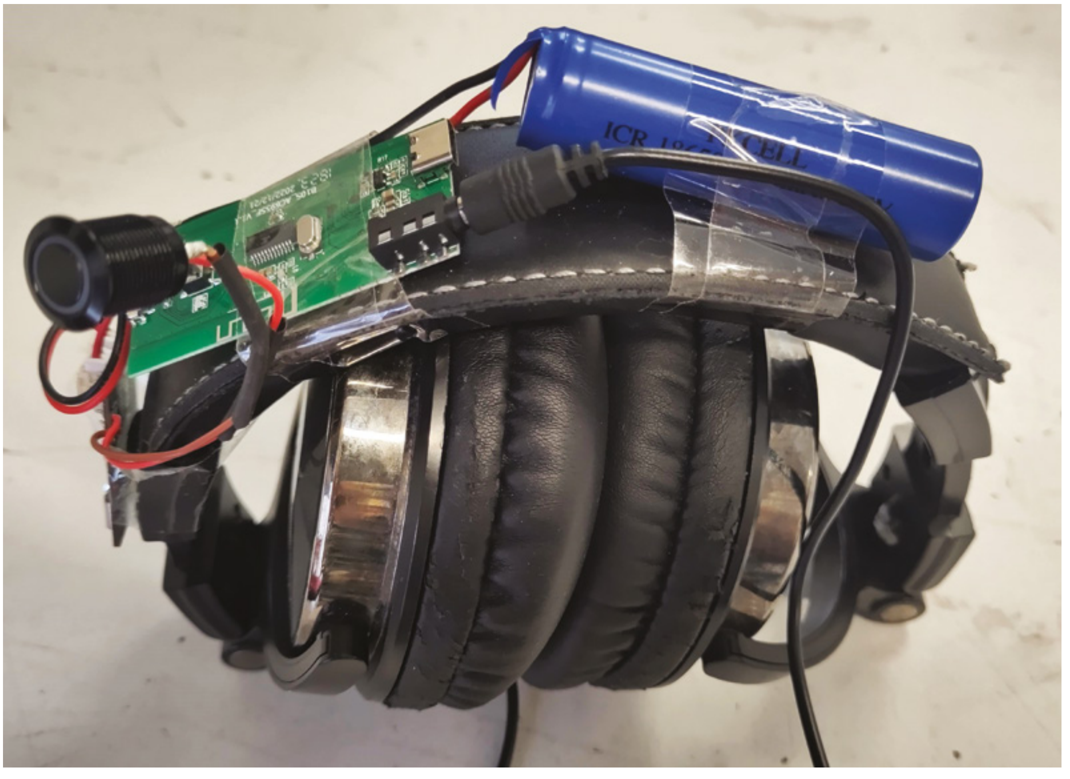
@ben_everard

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.

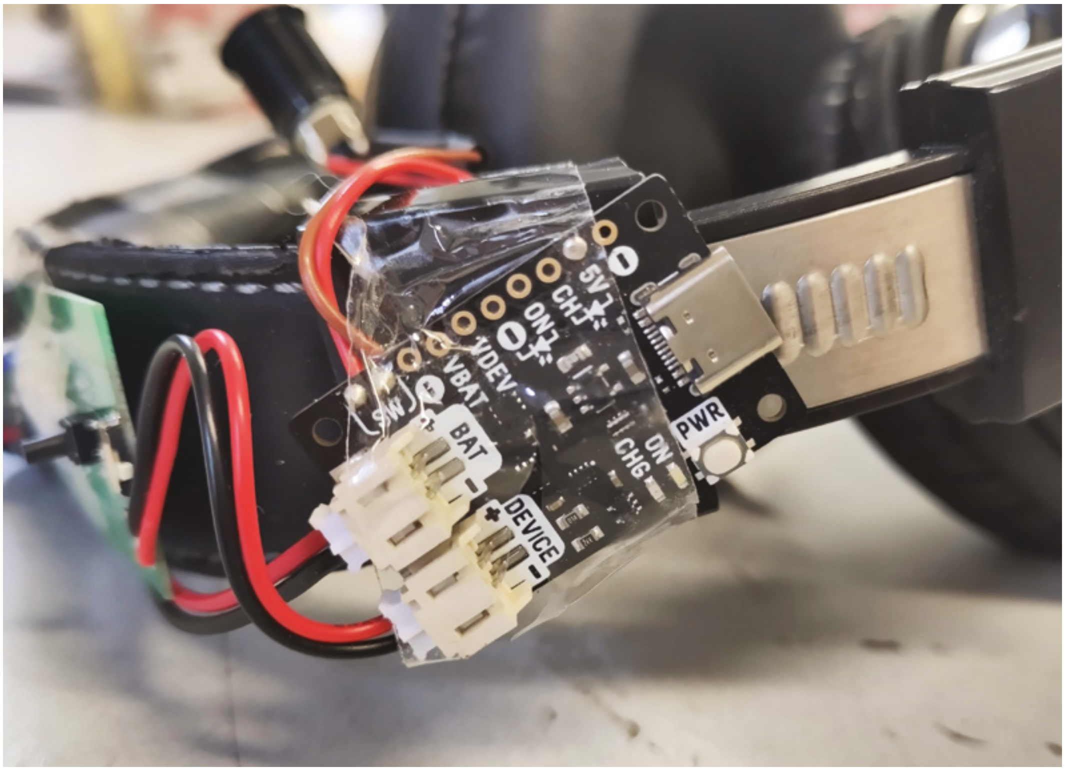
Modern phones have famously abandoned the headphone jack, and it seems that there's nothing us consumers can do about it. Like many people, this writer found himself in possession of several pairs of headphones that no longer work with his portable devices. And, like many people, he bought a small battery-powered Bluetooth receiver that plugs into the jack port on it.

This kind of worked. It didn't mount cleanly on the frame of the headphones (so he was planning a 3D-printed mount). However, after a couple of weeks, disaster struck and the battery stopped taking a charge. A sensible person would probably return this to the retailer for a refund. However, it was only a few quid and he was interested in finding out exactly what went wrong, so instead, he pried it open to see what was inside.

There was nothing obviously broken on the board: no wires snapped, nothing with obvious magic-smoke escape marks. There was a single PCB. On one side, there was what looked like a battery charging chip, and on the other, a Bluetooth controller. With nothing



Right OK, they look a bit shonky, but they work!



obvious broken, there was nothing obvious we could do to fix it, at least in its current form. We had a scour through some previous projects to see if there was anything we could borrow to get this up and running again, and found a Pimoroni LiPo Amigo. This was a bit overkill for this project (in fact, this part was more expensive than the Bluetooth adaptor), but it's what we had, and it works well. In the previous project, it had a large, waterproof power switch, and we left this on purely because it looked so ridiculous.

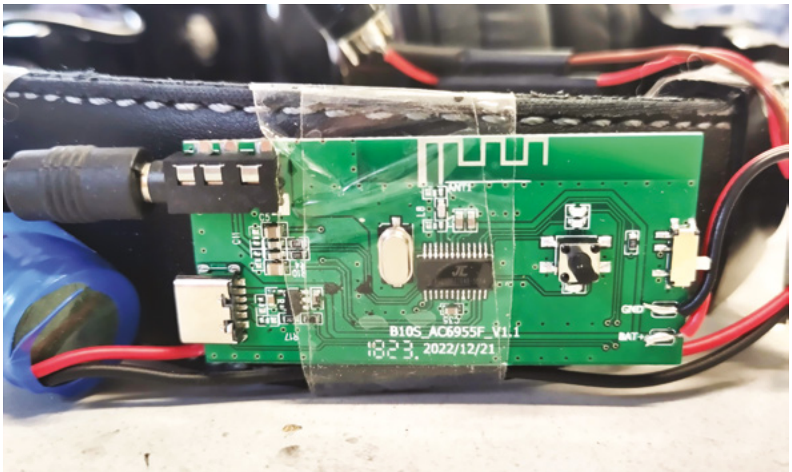
> **The biggest downside of it was that headphones** are surprisingly sensitive to weight

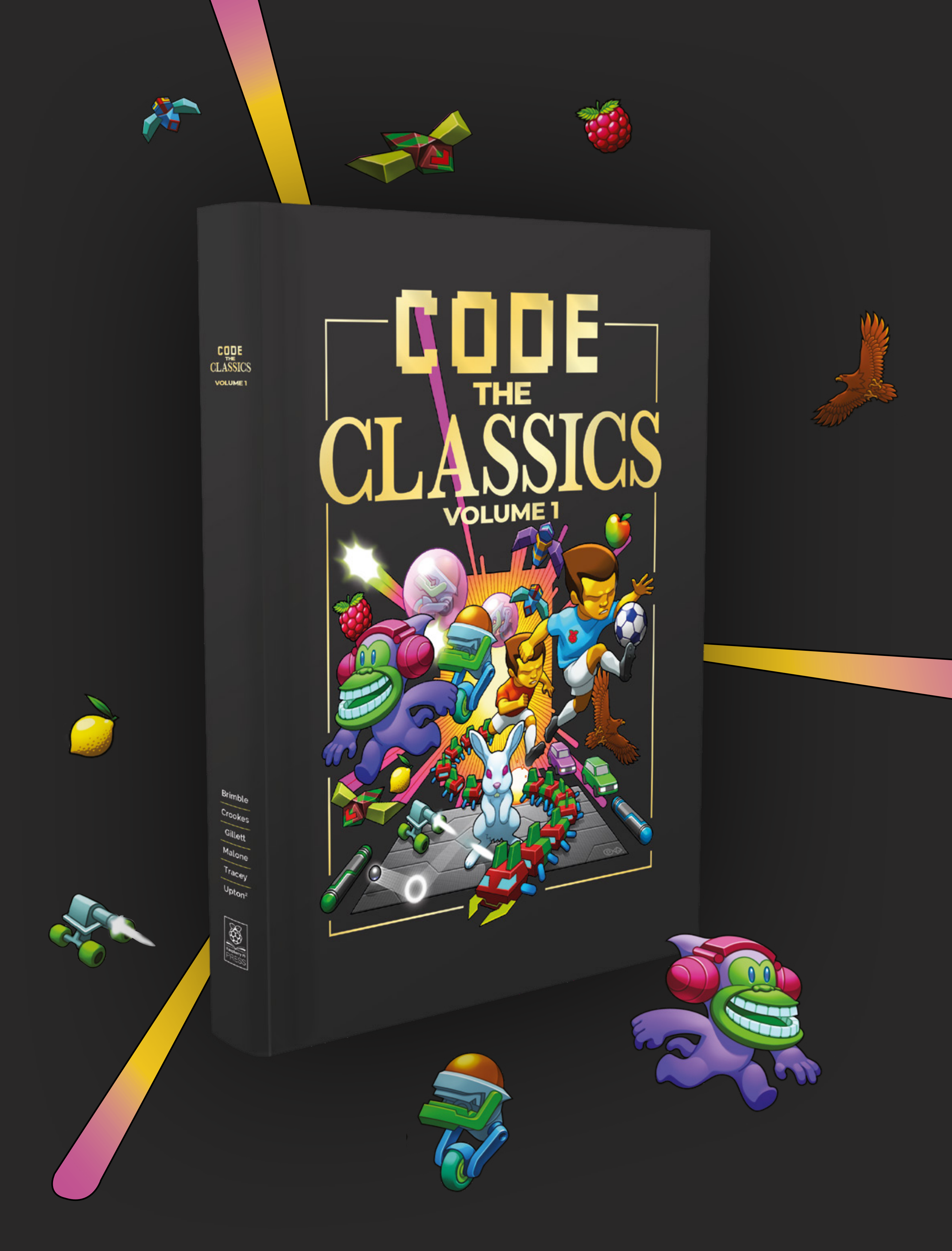
We soldered this in to the power and ground lines on the audio chip. We then needed a battery. We could have salvaged the one from the original project, but it was a bit on the small side, so we grabbed a spare lithium-ion battery that we had. This may not be the best choice for a head-mounted device because they can be a fire risk if they are abused. However, it's robust enough for a test, and we can decide what to do when everything is up and how we like it. Obviously, there is a better way of making it, but we just taped everything onto the band of the headphones. We were a little surprised when it worked first time, but it did.

The biggest downside of it was that headphones are surprisingly sensitive to weight, and the pressure on the top of our head gave us a headache after wearing it for a short time. We'll keep an eye out for better battery options, and it's now time to have a think about how to make them look more visually pleasing (or perhaps we'll lean into the cyberpunk aesthetic), but at least we can listen to music and podcasts when we're out and about.

Above
The LiPo Amigo is a great board for adding battery power and recharging to a project

Below
The internals of the Bluetooth adaptor was a single board that handled battery charging and connectivity, and only one part of this was broken




CODE
THE
CLASSICS
VOLUME 1
Brimble
Crookes
Gillett
Malone
Tracey
Upton²

The sublimation and the ridiculous

PART 02

Sublimation printing on difficult materials

Dr Andrew Lewis

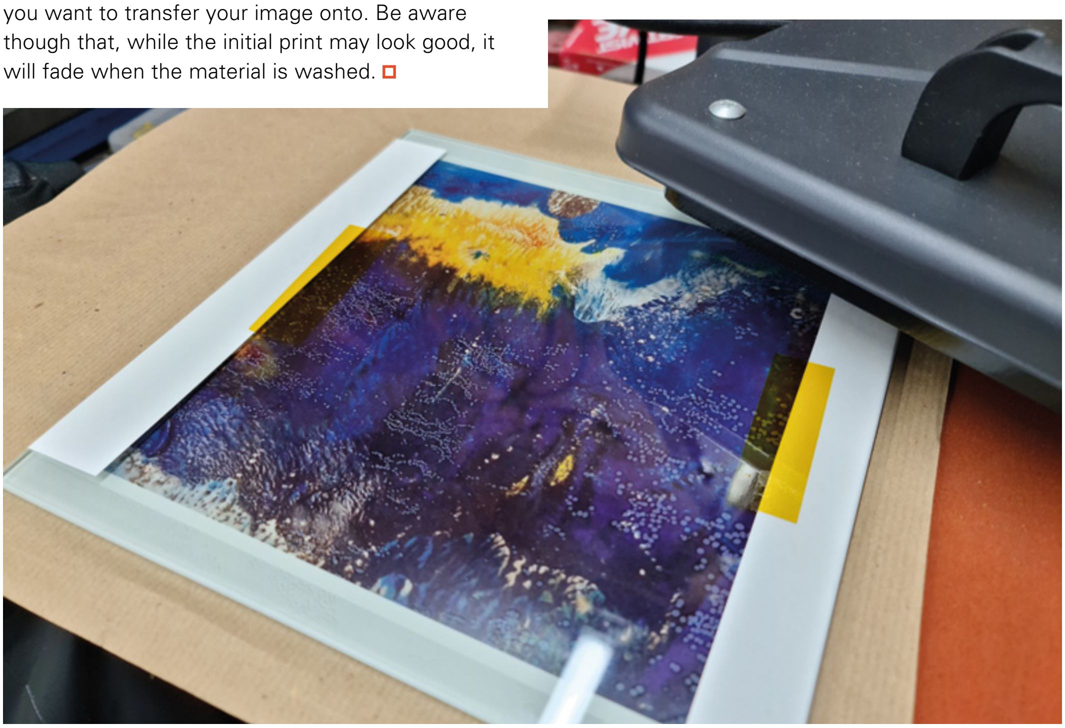
Dr Andrew Lewis is a specialist fabricator and maker, and is the owner of the Andrew Lewis Workshop.

Sublimation printing works by heat-transferring coloured plastic onto plain plastic. But what happens when you need to transfer an image onto something that isn't made from plastic? Commercial sublimation blanks are available for some items, but for others you'll need to prepare the material you are working with so that an image will transfer properly.

Cotton blends and purely synthetic fabrics are mostly suitable for sublimation printing without any extra preparation, but natural fibres like cotton aren't. If you don't have any other choice but to work with cotton, you can apply a fabric sublimation spray, or even just a clear acrylic spray paint to the area that you want to transfer your image onto. Be aware though that, while the initial print may look good, it will fade when the material is washed. □

FEEL THE HEAT

If you don't have a sublimation oven, you can create a printable surface on solid materials by using a hot laminating pouch. Lamination pouches are essentially a sheet of thin plastic with a transparent glue coating on the inside. If you split the pouch into two separate sides, you can hot-press the lamination plastic onto the glass, creating a plastic layer that you can sublimate onto. The result isn't as resilient as using a sublimation spray, but you can get decent results from this technique, and it will work on most rigid materials. You can even get gloss and matte pouches, so you can choose the most appropriate surface finish for your product.



Right When you're transfer printing, it's normal to reverse the image so that it's the right way around when it transfers. The exception to this rule is printing on transparent materials. If you're looking through the material, then there's no need to reverse the image before printing

GLASS AND CERAMIC

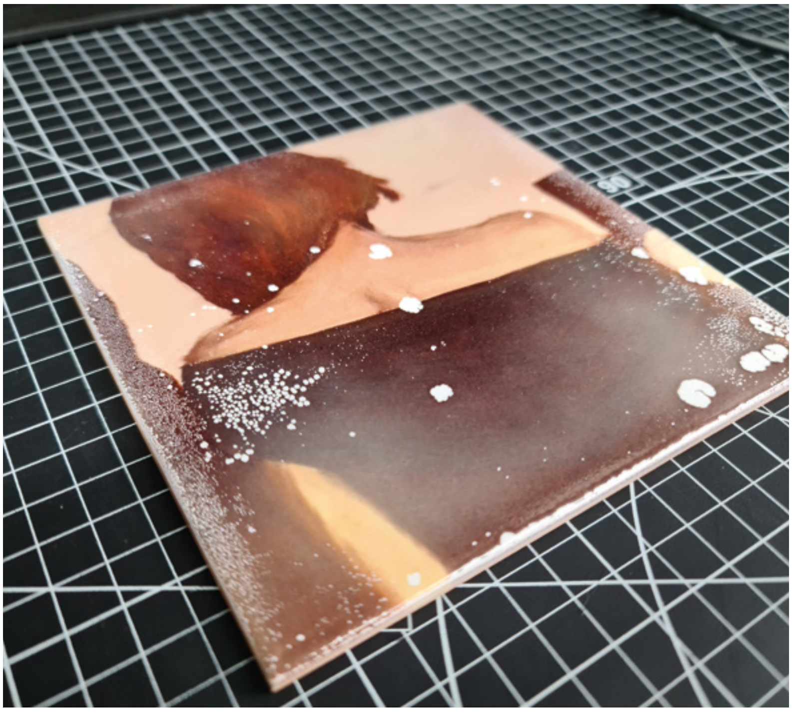
You can sublimate onto transparent glass or ceramic in three ways: applying a Polycrylic or acrylic varnish; applying a vinyl sheet to the glass; or using a dedicated sublimation spray. The best results come from using a sublimation spray, but you'll need a sublimation oven to cure the spray before you can apply a print to the surface. It's also difficult to apply any form of coating onto glass without it settling unevenly, because glass is a very smooth material. To counteract this problem, you have two options: either use a dedicated adhesion promoter like Subli Glaze, or very lightly key the surface with abrasive paper (around 400 grit will suffice), then clean it with isopropyl alcohol. Taking an abrasive paper to a transparent material might seem crazy, but the light scratches caused by the paper will improve adhesion of your sublimation coating, and the clear coating will effectively even out the scratches and hide them. If you want to create a backlit image on glass, you'll also need to finish with an opalescent white spray, or use what's known as a pass-through sublimation spray like Subli Vista Pass Through. Pass-through sublimation spray is applied to glass before you transfer your image, but after you've coated the glass with a sublimation coating. The name implies that the image 'passes through' the coating and is visible from the other side. So, the normal process for printing on glass with a sublimation coating would be to prepare the glass with adhesion promoter or abrasive paper, apply two coats of sublimation coating, and wait for them to dry for a few minutes. Apply two coats of a pass-through spray and let it dry for a few minutes, then cure the spray in a dedicated oven at 160°C for about 20 minutes.

Once you've prepared your sublimation glass, you can proceed to printing in the normal way, by printing your design onto sublimation paper and transferring it to the glass with a heated press. For the best results, preheat the glass to around 80°C in the press first, then apply the transfer sheet, holding it in place with heat-proof tape. Place the glass in the press with the transfer sheet furthest away from the heating element. You get the best results by heating the whole sheet of glass for several minutes under gentle pressure so that the image transfers onto an already hot piece of glass. Let the glass cool slowly, then remove the transfer sheet. A couple of sheets of baking paper will help protect the glass from scratches while you're pressing it.

Above Sublimation printing is easy enough if you have ready-made sublimation blanks to hand, but transferring images onto unusual materials generally requires some sort of intermediate carrier. This MDF sheet was prepared with a white primer paint and an adhesive plastic front made out of a glossy lamination sheet. Alternatively, Polycrylic or sublimation spray would have worked as the intermediate carrier

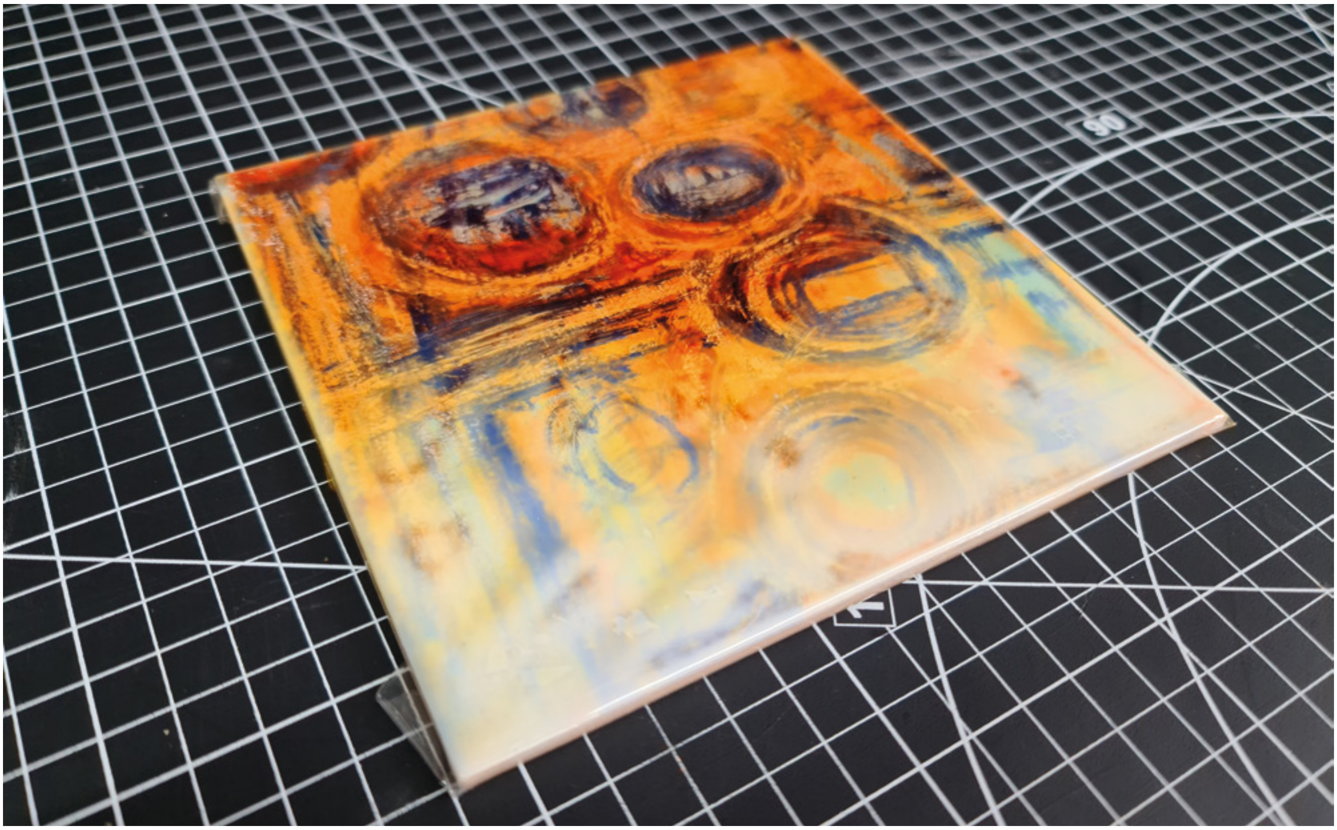
Right
Glass and ceramic are probably the trickiest materials to sublimate onto. They're both brittle and can shatter under pressure or because of thermal shock, and they're very smooth. Between the two materials, transparent glass is probably the least forgiving but most rewarding to work with

Below
This transfer failed because the tile surface wasn't prepared properly before the sublimation spray was applied. The sublimation spray clumped together and didn't stick evenly to the smooth surface of the ceramic tile. Light sanding with wet and dry paper, or using a specialist adhesion promoter, stops this from happening



WOOD ISSUES

Untreated wood is tricky to work with for sublimation for a few reasons. Wood isn't usually pure white, and heating wood can release sticky resins that affect the colour of your image, completely blacking out some areas and staining others. Also, the texture of wood is uneven, and that can lead to wrinkling and poor adhesion in some cases. As an additional problem, materials like wood and cardboard are hygroscopic (absorb moisture), and that can give you issues even after you've printed your item. To prepare wood and thick cardboard for printing, press it for 10–15 minutes at 140°C to drive off any excess moisture, then let it cool. It should settle as a flat sheet. If it doesn't, flip it over and give it another few minutes in the press. If you want to make sure your colours are close to true, apply a couple of layers of titanium dioxide-based white primer. This should help prevent any bleed-through discolouration caused by wood resin. Finally, add a transparent Polycrylic paint or a dedicated sublimation spray, then let it dry thoroughly before pressing as normal, with the print side closest to the hotplate of your press. Finally, seal the finished item with clear acrylic on all sides to prevent the material from warping as it absorbs moisture from the air.



HTV TIPS

Heat Transfer Vinyl (HTV) is a special type of vinyl that does exactly what its name implies. It's a handy tool to have in your arsenal when working with stubborn materials like cotton, because it can perform a few handy functions. It will stick to natural fibres quite nicely, and act as an intermediate layer between the cotton and your printed design. It's opaque, and white HTV can be used to block out an area on a coloured material to make coloured prints stand out. However, it does have a few drawbacks. It's a relatively thick material that will completely cover most surface textures, so if you're trying for a coarse canvas look, it's probably not the best choice. It's also very pliable, and any imperfections will imprint on it. For example, adding a sublimation print smaller than the size of the vinyl would create an imprinted pattern around the edge of the sublimation paper. Even the texture of craft paper or baking paper will transfer visibly onto the vinyl. It's also quite sticky, and will definitely make a mess of your hotplate if you forget to protect it with silicone paper. The normal method of using HTV with sublimation printing is to apply the vinyl as instructed on the package and let it cool, then remove the protective plastic sheet and apply the sublimation transfer using a sheet of transfer paper larger than the vinyl. Removing the transfer paper after pressing can be tricky, as the vinyl will stick to it, and some experimentation may be necessary to discover whether hot-peeling or cold-peeling gives the best results.

> "**Removing the transfer paper after pressing can be** tricky, as the vinyl will stick to it"

Above
Inadequate or uneven heat and pressure will leave your transfers looking pale and uneven. You can see that the foreground portion of this print is very pale. You can also see that excess plastic wrapped behind the tile looks untidy. Cropping plastic covers neatly, as though wrapping a gift, leaves a much more even finish, and the edge seams can be hidden behind a cork or felt backing fabric

Left
Transfers onto cotton work well in the short term as long as you prepare the material with an acrylic spray or a dedicated sublimation promoter. However, some 'clear' acrylic sprays are actually slightly yellow when you apply them, while others will start to yellow above a certain temperature. It's best to test the spray on a spare piece of material first if you've never used it before

FIELD TEST

HACK | MAKE | BUILD | CREATE

Hacker gear poked, prodded, taken apart, and investigated

ONLY THE BEST

Build your own timepiece

Got time? You do now: here's our selection of the best clocks for home electronics tinkerers

By **Marc de Vinck** @devinck

It's no secret that I love clocks, DIY electronic clock kits in particular. I've built at least a dozen of them over the years – most of them ended up being given as gifts that were always well-received. I've covered clock kits in the past, but it's time to take a second look to see what's new. Surprisingly, most of the big players in the DIY space haven't added any more clock kits to their stores, but you can find a lot of new kits from independent and smaller-scale manufacturers.

So, why do I like clock kits so much? It's simple. Once completed, it's one of those projects that you can keep enjoying for a long time. Many other kits are fun for a while, but most don't have the longevity of a clock. I have one clock made up of several old Nixie tubes that I made about 15 years ago on my desk, and I still enjoy it every day. I also like the fact that you can use them as a learning tool for anyone interested in electronics. You can find simple kits that teach soldering and the basics of electronics at a reasonable price. And the evidence of your efforts can be enjoyed for years to come.

7 segment LED clock Kit vs S14Clock LED Word Clock Kit

HOBBY HOME ELECTRONICS $130 | tindie.com

STORE139 $69 | tindie.com

Laser-cut wood enclosure, what's not to love? The 7 segment LED clock Kit by Hobby Home Electronics is a functional and fun clock that you solder together yourself. At first glance, you might think this is a good beginner kit, but that would be a mistake for someone new to soldering. This kit is mostly surface-mount components. Now, if you've mastered through-hole soldering but are new to surface-mount soldering, then this would be a good kit. There is plenty of room on the PCB. And the components are fairly large for surface-mount electronics.

Once you've soldered everything together, add the preprogrammed Arduino-compatible microcontroller and screw together the laser-cut case. Now you just need a 5V power supply like one that typically powers your mobile phone and you'll have a beautiful clock to display in your home or office.

Left We love the juxtaposition of modern LEDs and traditional wood

I have seen quite a few word clocks out there in the DIY community. Most of them are a bit bulkier than the S14Clock LED Word Clock Kit. This beautifully sleek clock features a web-synchronised clock with 14-segment LED displays which is highly configurable to whatever messaging you'd like to show.

The clock, thanks to its integrated ESP32-S2, shows the time and a selection of preprogrammed preset messages that are configurable via the built-in web interface. You can pick up a smaller 12-character display, or you can really go for it with the larger 24-character version. I know which one I want!

Below Tell the time just like we did in the 1980s

VERDICT

7 segment LED clock Kit

A complete kit, ready to make and enjoy.

9/10

S14Clock LED Word Clock Kit

Unique and sleek.

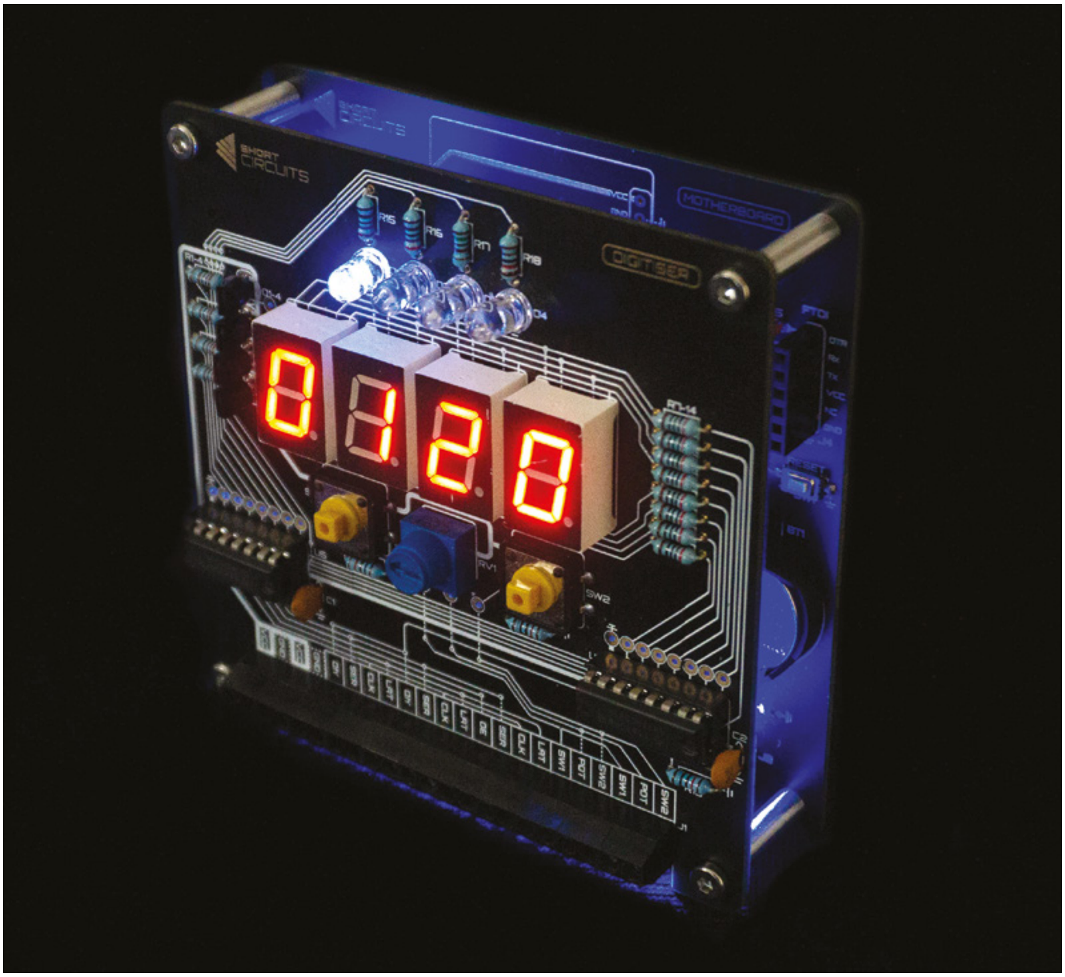
10/10

Digital Clock Pack

SHORT CIRCUITS ◈ **$47 tindie.com**

If design aesthetics are important to you, then this is your clock kit! The Digital Clock Pack from Short Circuits in the United Kingdom is one of the most beautiful clock kits I have seen. From the black PCB to the carefully considered locations of the vias and traces, It's hard to imagine a more aesthetically pleasing kit.

The kit includes a motherboard and a digitiser board that couple together to create a beautiful seven-segment clock that doesn't require an enclosure. Why would you want to hide those perfectly placed components? The kit includes two buttons and a potentiometer to let you change what is being displayed and the brightness. There are also special display modes like a countdown timer and stopwatch, along with the standard time and date. I really like everything about this little clock.



Left ◈ You'll either love or hate the look of this timepiece. We love it

VERDICT

Digital Clock Pack

Beautiful and functional.

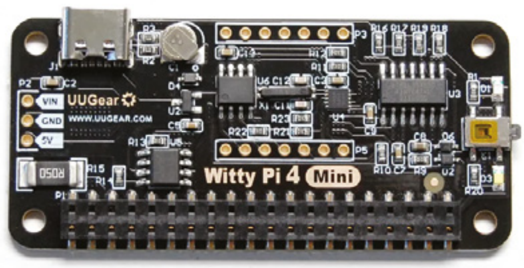
10/10

Witty Pi 4 Mini

UUGEAR ◈ $26.43 | pimoroni.com

For some of you, the idea of a ready-made kit doesn't sound like your idea of fun. You're the more adventurous DIYer that wants to fully customise your build. Well, if that's you, then the Witty Pi 4 Mini is the perfect starting point for your Raspberry Pi-based custom timekeeping build.

This Witty Pi HAT features a real-time clock coupled with power management circuitry. If you are planning on a complex Raspberry Pi-based project that needs accurate time, and you want to run intermittently, and with very low power requirements, then this is the HAT for you. Check out the Pimoroni page for more on what you can make with the Witty Pi 4 Mini.



Below The Raspberry Pi 5 has an RTC, but you can add this feature to earlier models

VERDICT

Witty Pi 4 Mini

A great start for your own build.

8/10

DC 5V Green LED Electronic Clock DIY Kit

ICSTATION ◈ $12.31 | tindie.com

Despite the rather generic name, the DC 5V Green LED Electronic Clock DIY Kit by ICStation is a perfect example of a kit that I love to use for teaching soldering. Even a beginner can easy solder this through-hole kit and, once complete, they have a fun and functional desktop clock. Even though it's small and simple, it packs a bunch of features, including 12-hour and 24-hour time modes, an alarm, automatic brightness, and it even announces the time on the hour. An affordable clock kit that can speak the time? Fun!

Below Learn to solder and get yourself a clock

VERDICT

DC 5V Green LED Electronic Clock DIY Kit

Affordable and simple to build.

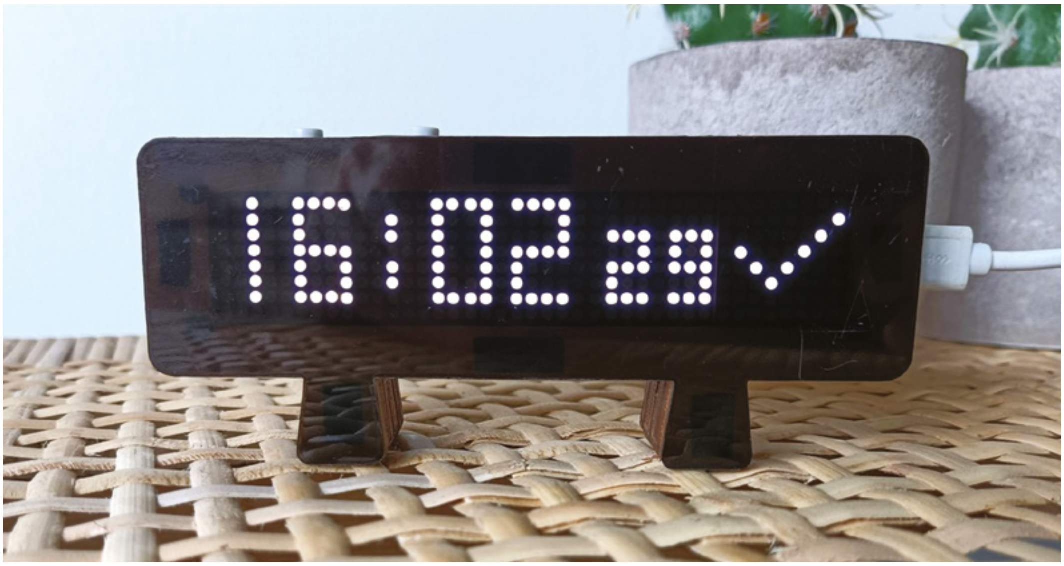
9/10

ESP8266 LED Matrix Clock Kit

ELEKITSORPARTS $42 | tindie.com

So, you don't want to solder, but you still want a clock kit? Well then, the ESP8266 LED Matrix Clock Kit from elekitsorparts may be a good choice. You get two PCBs – one with an ESP8266 and supporting components, and another with five 8×8 LED matrix displays. All you need to do is put it all together in the custom laser-cut wood and acrylic case. Being an open-source project, all the source code is ready to be downloaded and modified. It also supports MQTT, so more advanced users can make it display data gathered from the web.

Below
Click together your own clock



VERDICT

ESP8266 LED Matrix Clock Kit

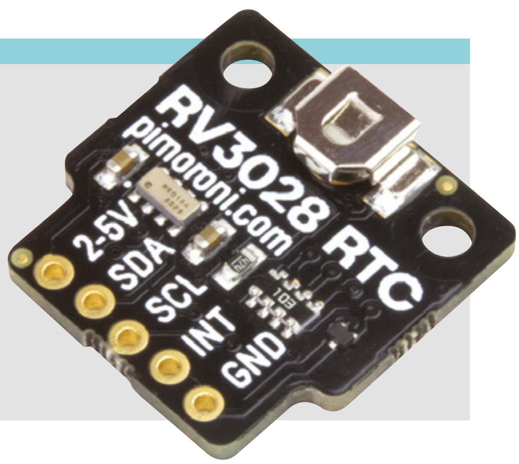
Fun and retro.

9/10

RV3028 REAL-TIME CLOCK

PIMORONI $16.24 | pimoroni.com

The RV3028 RTC is an extremely accurate clock module that has a drift of just ±1 second per million seconds at 25 degrees. You communicate with it via I2C interface, and it's 3.3V- or 5V-compatible. Pimoroni designed it so you can solder a 5-pin right-angle header onto it, and then you can insert it into the bottom left five pins on your Raspberry Pi's header. If you need accurate timing, this is a great little breakout board.




FROM THE MAKERS OF The MagPi THE OFFICIAL RASPBERRY PI MAGAZINE
NEW 2022 UPDATE
RETRO GAMING WITH RASPBERRY PI
2ND EDITION
164 PAGES OF VIDEO GAME PROJECTS
+ BUILD AN ARCADE MACHINE
PLAY & CODE GAMES!

Shaper Trace

Convert your sketches into vector graphics for cutting, engraving, and more

SHAPER ◆ £109 | shapertools.com

By Ben Everard @ben_everard

The Shaper Trace is a drawing toolkit that converts sketches, or drawings, into scalable vector graphics (SVG) via a plastic frame and an app on your phone. The device is available for pre-order for £109, and is set to ship in December 2023.

The key selling point of the Shaper Trace is its ability to swiftly convert hand-drawn sketches into vector graphics. By placing the Trace Frame over a drawing and using the Trace App, users get an SVG file, compatible with various maker tools like laser cutters, vinyl cutters, and CNC machines. You can use any pen or paper for the drawing, but you'll get best results with a black pen on white paper. The Trace does come with a pen, but others will work as well.

Below ◆
As the name suggests, you can trace the outline of parts

The Trace offers a maximum drawing size of 178 by 268 mm, which may not be A4-sized but should suffice for most projects. Moreover, users can choose to create the vector using either the outline or the centreline of the image, allowing for different levels of detail and customisation.

In use, the Shaper Trace delivers on its promise. It smoothly translates sketches into vector graphics, has the ability to trace real-world shapes, and is really easy to use. Just pop the frame over the image, open the web app, take a photo, and select the option you want (centreline or outline). The only advanced options are level of detail and smoothing, but in many cases the defaults will work fine.

The Trace opens up possibilities for both aesthetic and functional design elements. For example, you could trace around leaves to add a realistic aesthetic element to your design, or you could accurately trace the outlines for screw holes, ensuring precise placements when laser-cutting.

It's one of those tools that doesn't do a whole job itself (you need to pair it with some form of manufacturing tool like a laser cutter, vinyl cutter, CNC mill, etc.), and, in most cases, it won't even do the whole design job itself – you'll most probably need an additional vector graphics tool like Illustrator, Inkscape, or Shaper's own Studio. It complements a workflow that is already mostly complete, but that leaves you a bit frustrated.

The ease of use is the real feature of Trace. Once you have your drawing, it's only a few seconds until you have the vector image that you can manipulate and prepare for whatever tool you want.

However, the Trace does have its limitations. Control over the vector creation process is somewhat restricted, with options limited to adjusting detail and smoothing. One notable missing feature is the ability to place the vector on the inside or outside edge of

Left The device works well even with relatively complicated shapes

a line, which would be really useful if you'd traced around the edge of something that you wanted to include in your design.

Some of the Shaper Trace's features can be replicated without the device. Scanning or photographing drawings and then converting them into vectors is possible, and many people do this regularly. The Trace offers dimensional accuracy, and the tools for centreline and smoothing do pair well with machines like CNC mills and laser cutters. We've certainly never come across a tool that does it as well or as easily as the Trace. Whether or not this ease is enough to justify the price tag will depend a lot on how important these features are, and how often you use it.

> "The Shaper Trace presents a practical and efficient way to **convert hand-drawn sketches into vector graphics**"

The price may deter hobbyists who only need occasional conversions or have minimal image processing needs, and it might be hard to justify this cost if you don't have a specific use case in mind. However, for frequent users and professionals dealing with vector graphics regularly, the Trace could prove to be a valuable time-saving tool.

Trace's software setup is linked to a single Shaper account, requiring a username and password. This works well for individual users, but may pose some inconveniences for users with multiple computers, as everything must share the same account. This is a bit of a shame, as we can certainly see this as being a useful tool for shared workshops.

It's also worth noting that Trace is entirely dependent on a web app that you can't save for offline use. While there is no reason to suspect any issues, the reliance on a web app means that if the app becomes unavailable for any reason (for example, if Shaper closes down at some point in the future), the Trace would become a useless plastic rectangle.

The Shaper Trace presents a practical and efficient way to convert hand-drawn sketches into vector graphics, making it a valuable tool for those skilled in sketching and working with vector images. We're not aware of any other tool that works as well or as quickly for this. While its price and potential dependence on the web app may raise some concerns, the Trace's overall utility and ease of use make it a valuable asset in any workshop, especially for those heavily involved in vector-based design and fabrication.

VERDICT

Works well and it's easy to use, but it's expensive and dependent on a web app.

9/10

Inkscape 1.3

Free and open-source vector graphics for makers

INKSCAPE ◆ **Free** | **inkscape.org**

By Ben Everard

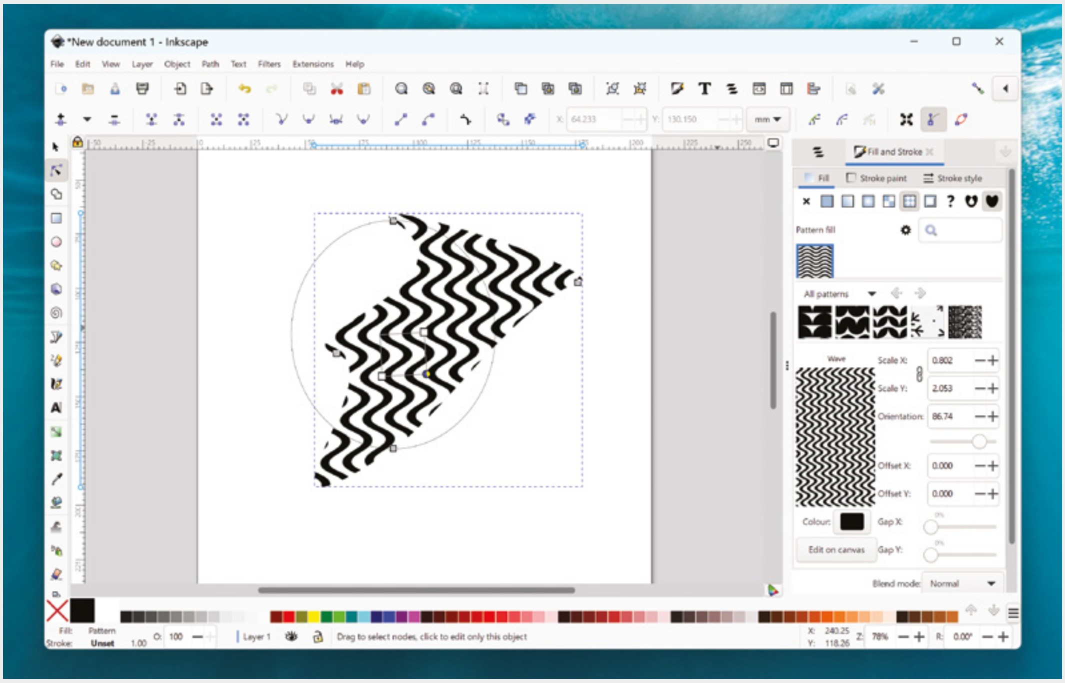
@ben_everard

Inkscape is probably the most common vector drawing tool for makers. It's one of those bits of software that spans multiple disciplines. We use it for laser cutting, plotting, CNC milling, PCB design, vinyl cutting, and 3D printing – we've even used Inkscape for embroidery. It is probably the most widely used piece of maker software.

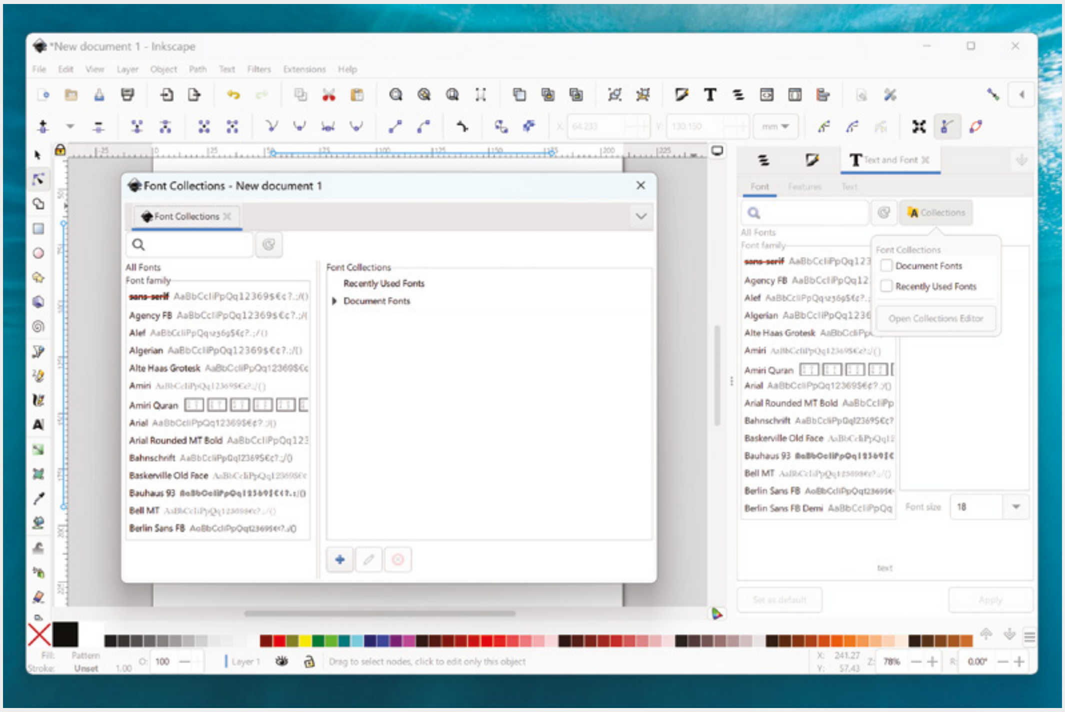
At a basic level, vector graphics are about lines. You join these lines together to make shapes, and these shapes might be anything – illustrators use them to create pictures, while makers might use them to define the cut routes of a laser, but to the software, they're just lines. This is all opposed to raster graphics (which are the other main type of computer graphics). In raster graphics, you deal with pixels where each pixel has a specific colour and is (as far as the software is concerned) completely disconnected from other pixels.

This means that, if you want to work with lines and shapes, Inkscape could be for you. If you want to work with pixels, then you probably want a different bit of software.

Within Inkscape, you can create paths (which are basically lines made up of either straight or curved sections). You can draw these or build them up from shapes and text, and you can add fills, patterns, or gradients within these paths. You can also arrange



Right ◆ It's a bit easier to work with fill patterns, but these won't be converted into paths



Left You can now organise the myriad of fonts on your machine into useful collections

these paths into layers to group features. For instance, we use different layers to define separate paths for different pens on the plotter.

Most of the time, Inkscape works with SVG (Scalable Vector Graphics) files, but it can import and export to a range, including PDF and DXF (which is sometimes used for laser cutters, vinyl cutters, and CNCs).

PROS AND CONS

The generality of Inkscape is simultaneously its best and worst feature. You can use it for almost any vector image-related problem, and this means that you only have to learn one tool to accomplish a lot of tasks. But, it also means that it can be a bit clunky as it's not set up for any specific workflow (though you have a bit of freedom to configure the toolbars and sidebar). Once you've got to grips with the basic concepts, it's usually fairly straightforward to get it to work in the way you want. So far, we've been talking about Inkscape in general, so let's take a more detailed look at the key features of the new version.

The headline change for many users will be performance, and much of this comes from new multithreading. How much of a speed-up this brings depends on how many cores your computer has, but most modern machines have more than one. Benchmarking the performance of a graphical application is almost impossible. The developers claim a two to four times speed-up, and it certainly feels much snappier, particularly when working with complex drawings.

Many of the new features in Inkscape 1.3 aren't particularly maker-focussed. For example, the new pattern builder is great but, since patterns can't be converted into paths, they don't work particularly well with most maker tools. The Lasso Selector, and redesigned Live Path Effects, are both useful tools for manipulating shapes, but neither is likely to dramatically alter your experience of the software.

There's also a notable, but minor, improvement in DXF export that means non-path objects will automatically be converted to paths. This will be a

"The headline change for many users **will be performance**"

boon for laser cutter users who find their text magically disappears when they go to etch it (or is that just us?)

The most useful of the new features is the Shape Builder tool, which makes it easy to combine primitive shapes into more complex offerings using Boolean operations. This could be accomplished before, but the process is now less cumbersome.

Performance aside, there's not much in the new release to really excite makers, but this doesn't detract from the fact that Inkscape is a fantastic tool for makers. It's free, powerful, and, while it can take a bit of time to really get to grips with it, that time will pay dividends across lots of different tools and machinery.

VERDICT

A great, free vector graphics program just got faster.

9/10

CROWDFUNDING NOW

Computer Engineering for Big Babies

Big switches and comforting lights

From £23 | **hsmag.cc/cefbb** | **Delivery:** December 2023

Our one big issue with *Computer Engineering for Big Babies* is that it's not about computer engineering – it's about digital logic. It builds on the earlier *Computer Engineering for Babies* book which introduced logic gates. This book expands this into standard components such as shift registers and memory read. Each page has two switches and five lights that can be used to illustrate the concept. The crowdfunding campaign doesn't detail how many pages there are but, from the images, it looks like about five.

Let's be perfectly honest – no, your one- or two-year-old isn't going to learn a meaningful amount about digital electronics from this book (just as they're not going to learn anything about zoology from *That's Not My Otter*, or much about robbery on the high seas from *Ten Little Pirates*). However, many toddlers love a switch and, for some reason, most toy manufacturers seem intent on attaching a sound effects chip to the other end of the switch. For sticking to light effects alone, this book deserves interest from parents.

This particular reviewer rigged up some light switches to LEDs to give mini humans some switches to toggle without an auditory assault and, thanks to *Computer Engineering for Big Babies*, future parents will be able to get this peace without having to do the wiring themselves.

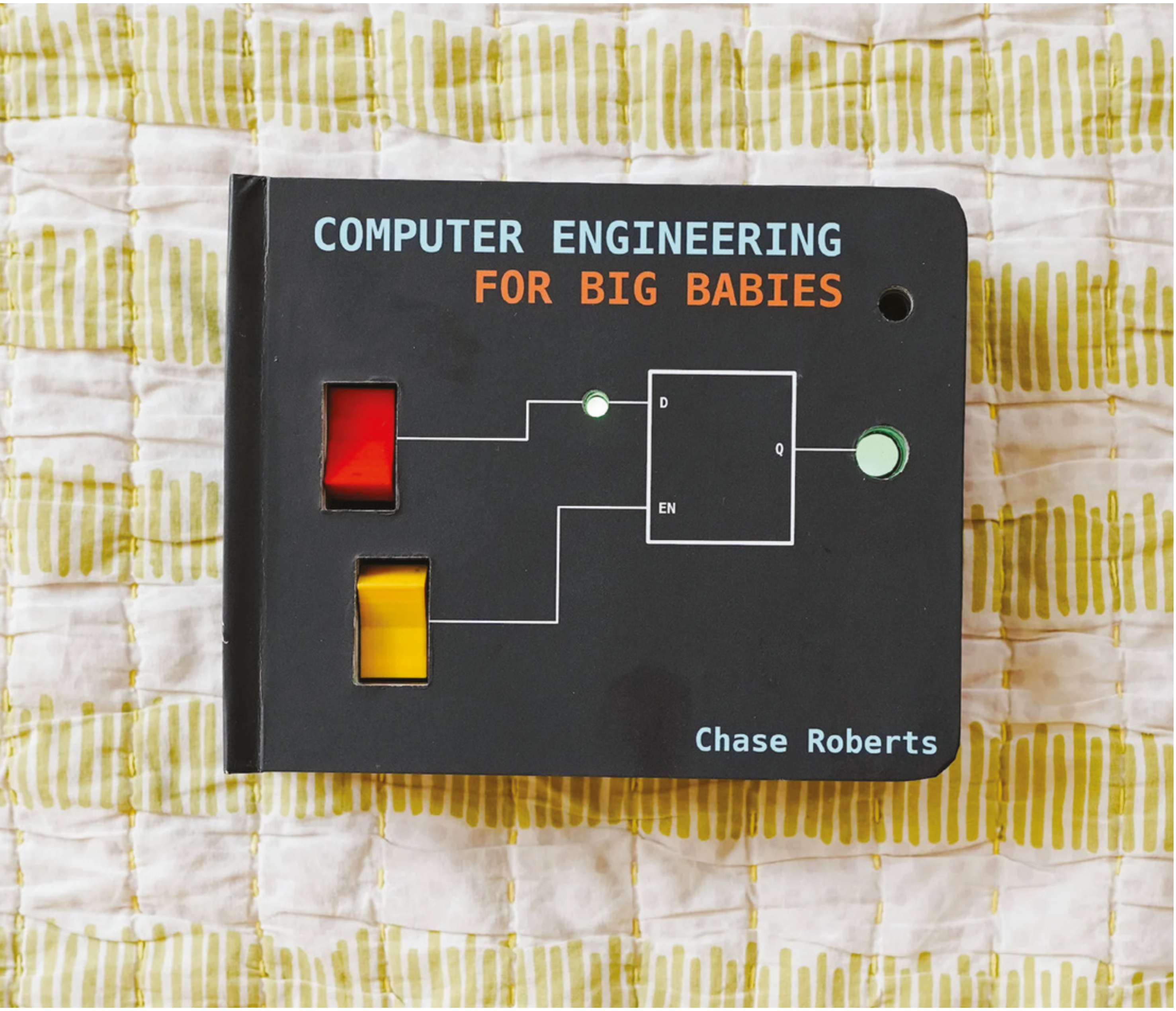
"**For sticking to light effects alone, this** book deserves interest from parents"

Computer Engineering for Big Babies, like many books of the genre, is really for parents rather than children. If you're sleep-deprived and want some calming switches to flick while talking affectionately to a small human, then this could be just the book for you. Just don't expect your child to remember any of it.

Above Clicky switches and blinking lights are all the budding engineer needs

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.



next month

issue #72

ON SALE
26 OCTOBER

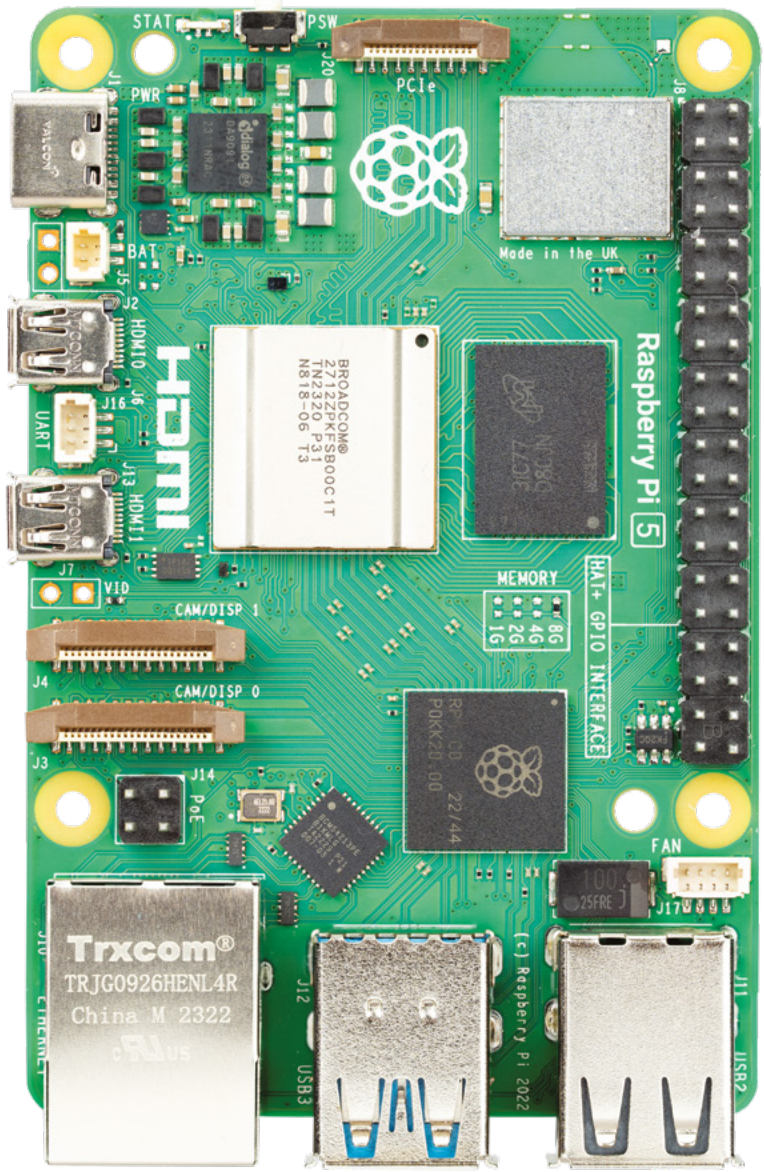
RASPBERRY PI 5 PROJECTS

ALSO

- → ARDUINO
- → ROBOTS
- → 3D PRINTING
- → AND MUCH MORE

DON'T MISS OUT

hsmag.cc/subscribe



Raspberry Pi 5

We have guaranteed stock of Raspberry Pi 5s for all subscribers.
See page 62 for details.