MAKE | BUILD | HACK | CREATE

HackSpace

TECHNOLOGY IN YOUR HANDS

hsmag.CC | September 2020 | Issue #34

SMART LIGHT

Internet-enabled illumination

POWER

Get the right voltages for your project

DISCOVER FEATHER

PLUG AND PLAY FOR EASY ELECTRONICS

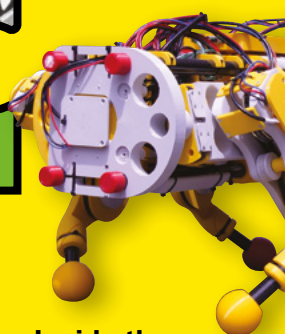
+ **RECYCLE** old clothes

+ Workshop **CAMERA**

+ Para **CORD**

OPEN DOG

Inside the world of open-source robotics



9 772515 514006 34

Sept. 2020
Issue #34 £6

RASPBERRY PI 3D PRINTING **PYTHON** BLANKETS

9.2 MILLION+ PRODUCTS ONLINE | 1,100+ INDUSTRY-LEADING SUPPLIERS | 1.9 MILLION+ PRODUCTS IN STOCK

The World's Largest Selection of Electronic Components Available for Immediate Dispatch™

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

ECIA MEMBER
supporting The Authorized Channel

Welcome to HackSpace magazine

One of the first things you have to decide when starting a new project is how much of it you're going to build yourself. Or, to put it another way, what your building blocks will be. Some people like to just add a few bits to existing hardware, some people like to work with pre-built modules, some like to work with small parts, and some enterprising folks prefer to start with raw materials (such as Sam Zeloof, who makes integrated circuits at home: **sam.zeloof.xyz/first-ic**). There is no 'right' place to start – it's all about picking what you want to work on. This month, we're looking at the Feather standard, which I find a particularly good starting point for my projects. It's bare-bones enough to be small and flexible, but not too bare-bones. There are plenty of add-ons available if you want to use them, but you don't have to. Take a look at page 34 for the inside story on how it all got started.

There is no 'right' place to start – it's all about picking what you want to work on

BEN EVERARD
Editor ben.everard@raspberrypi.org

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

get in touch at **hsmag.cc/hello**

GET IN TOUCH

hackspace@raspberrypi.org
hackspacemag
hackspacemag

ONLINE

hsmag.cc

Available on the App Store

GET IT ON Google Play

EDITORIAL

Editor
Ben Everard
ben.everard@raspberrypi.org

Features Editor
Andrew Gregory
andrew.gregory@raspberrypi.org

Sub-Editors
David Higgs, Nicola King

DESIGN

Critical Media
criticalmedia.co.uk

Head of Design
Lee Allen

Designers
Sam Ribbits, Harriet Knight, Ty Logan

Photography
Brian O'Halloran

CONTRIBUTORS

Lucy Rogers, Drew Fustini, Rosie Hattersley, Jo Hinchliffe, Mayank Sharma, Andrew Lewis, Andrew Robinson, Demitrio Pinnar, Marc de Vinck, Mike Cook

PUBLISHING

Publishing Director
Russell Barnes
russell@raspberrypi.org

Advertising
Charlie Milligan
charlotte.milligan@raspberrypi.org

DISTRIBUTION

Seymour Distribution Ltd
2 East Poultry Ave,
London EC1A 9PT
+44 (0)207 429 4000

SUBSCRIPTIONS

Unit 6, The Enterprise Centre,
Kelvin Lane, Manor Royal,
Crawley, West Sussex, RH10 9PE

To subscribe
01293 312189
hsmag.cc/subscribe

Subscription queries
hackspace@subscriptionhelpline.co.uk

recycle

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.

Contents

110

Cover Feature

34

Tutorial

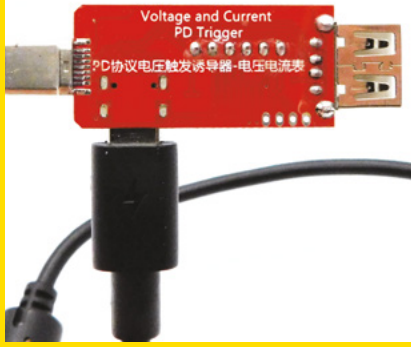
Silk filament printing

72 Silky smooth prints for flawless forms

100

Direct from Shenzhen

USB-C power



106 Get more control out of this simple power standard

06

Interview

James Bruton

52 Inside the world of homemade open-source robotics

76

20

65 FORGE

99 FIELD TEST

Off-World Bartender

By Donald Bell

hsmag.cc/27gylZ

Readers of a certain age (or those of you who are able to use YouTube) will be familiar with *The Great Egg Race*, a British children's television programme that set kids a different engineering challenge each week, and sat back and watched as they solved it. It was great telly, and showed that the best problem-solvers are people who don't know the rules that they're supposed to be breaking.

We thought of *The Great Egg Race* when we heard of the Cocktail Robotics Grand Challenge, an annual event in San Francisco where robots are judged on how well they make delicious booze. Donald Bell's entry this year is a *Blade Runner*-inspired device that uses two Raspberry Pis (a 3B, and an A+), an RFID reader, an Adafruit Feather HUZZAH ESP8266, pumps, tubing, and a heck of a lot of LEDs for that off-world 2040 aesthetic.

Right
"I've tasted things you people wouldn't believe"


FY TO
IALIZE

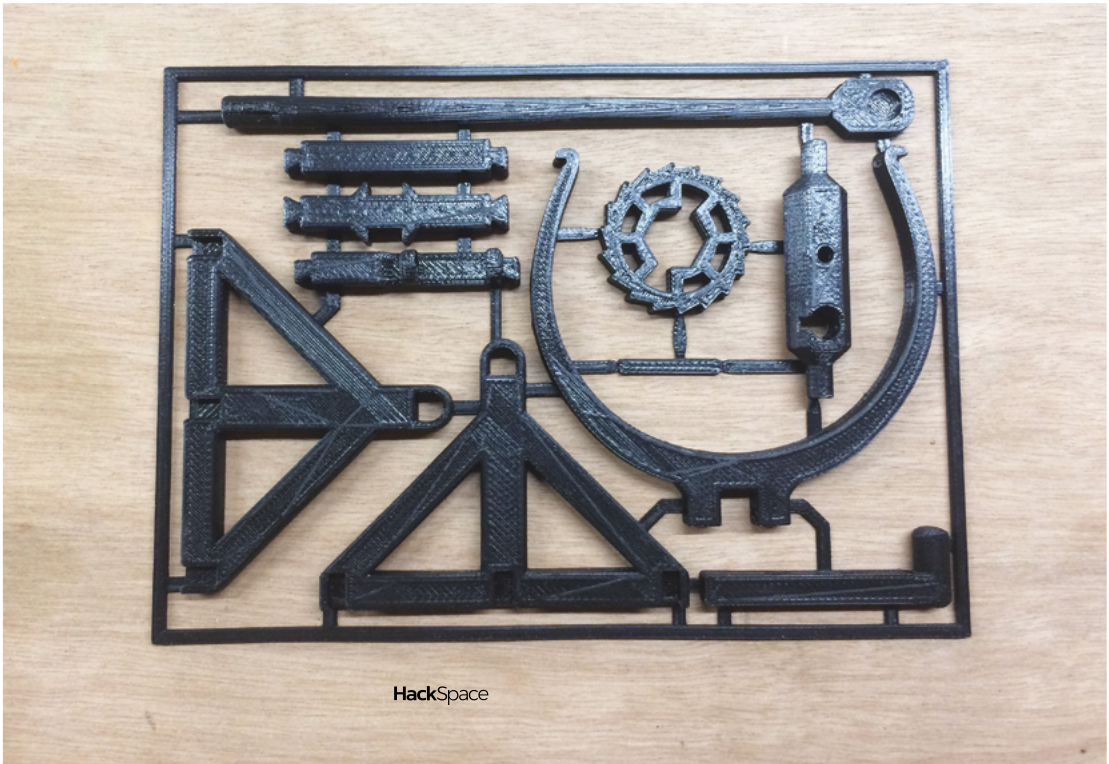
3D-printed Davinci catapult

By Brian Brocken

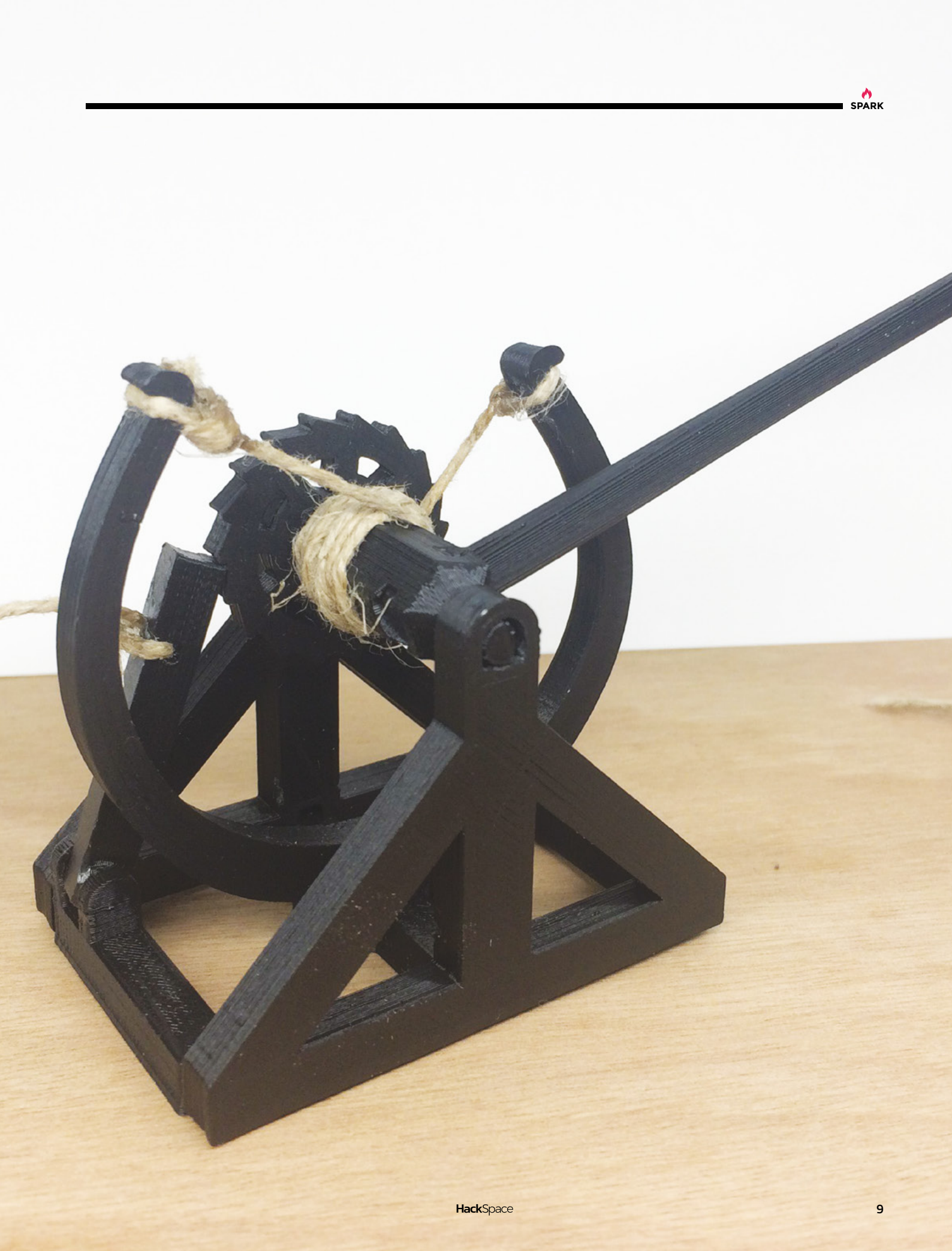
hsmag.cc/fZI7G1

Brian Brocken knows what he's doing when it comes to 3D-printed kit builds: his previous design, a wind-up car (hsmag.cc/0oSivs), went down so well that he decided to make another device that can be used in its unassembled form as a greetings card.

As it's flat, the whole thing prints in one go with no supports, and takes around three hours with a layer height of 0.2 mm. The kit snaps together with no need for glue and, as the name suggests, it's inspired by a design by the original maker genius, Leonardo da Vinci.



Right
We have yet to test whether this catapult can launch a 90 kg projectile over 300 m

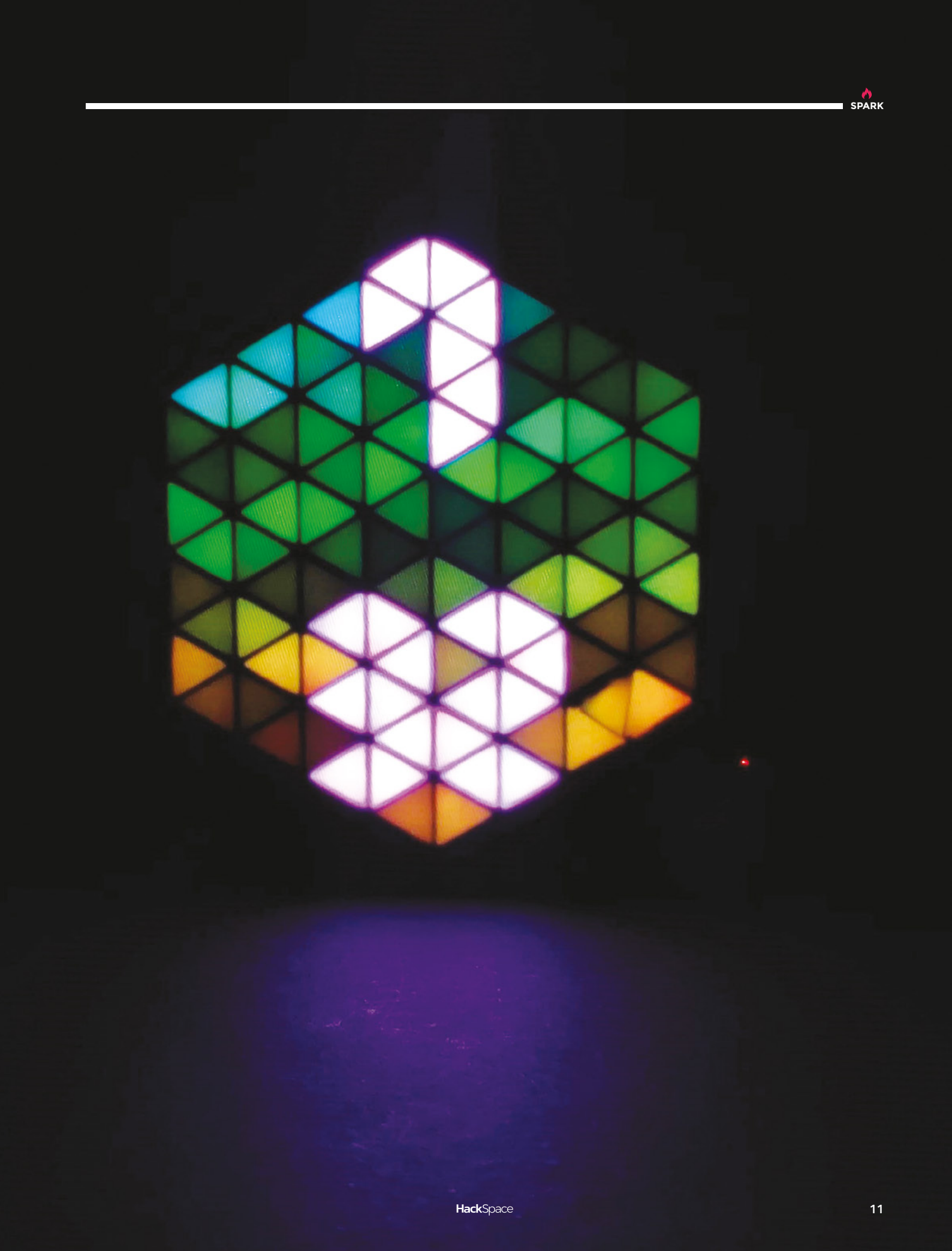


Hex clock

By **Mukesh Sankhla** hsmag.cc/feehfJ

Clock designs are an ideal way to inject some creativity into a build. There is a set of rules, and you can break them in some ways, but not others. This unique design by Mukesh Sankhla uses a matrix of triangular pixels in the shape of a hexagon to display the time, weather, or any other data on an IoT clock. The unusual display means that standard characters won't work, so Mukesh has had to design his own set of digits from 0 to 9; we think they look great.

Right
Mukesh used the FastLED library to code the animations on the clock-face



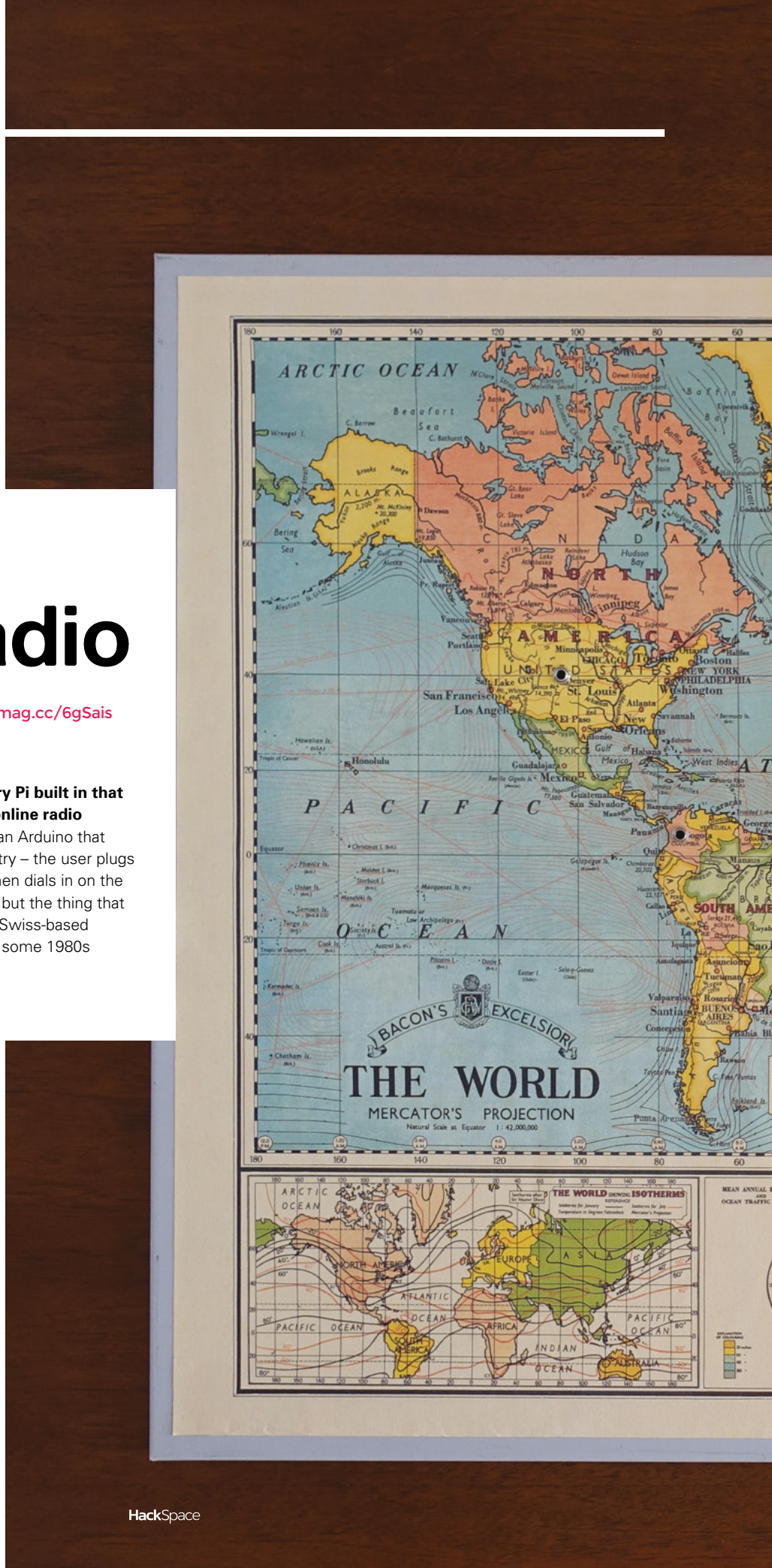
REGULAR

Rotary phone radio

By Caroline Buttet

hsmag.cc/6gSais

This rotary phone has a Raspberry Pi built in that links up to radiooooo.com, an online radio service. Behind the map, there's an Arduino that controls the selection of the country – the user plugs the phone into the jacks to use, then dials in on the phone to listen. It's a simple idea, but the thing that caught our eye was the flawless execution, by Swiss-based designer Caroline Buttet. Now, who's ready for some 1980s Sheffield synth-pop?



Right
Pick up the phone, choose a country and a decade, and listen to some great music


GREENLAND
Barents Sea
Kara Sea
UNION OF SOVIET
LENINGRAD
MOSCOW
EUROPE
ATLANTIC
OCEAN
SAHARA
AFRICA
PERSIA
Bombay
Madras
INDONESIA
OCEAN
Cape Town
RIO DE JANEIRO
THE WORLD SHOWING ISOBARS
1 2 3 4 5 6 7 8 9 0

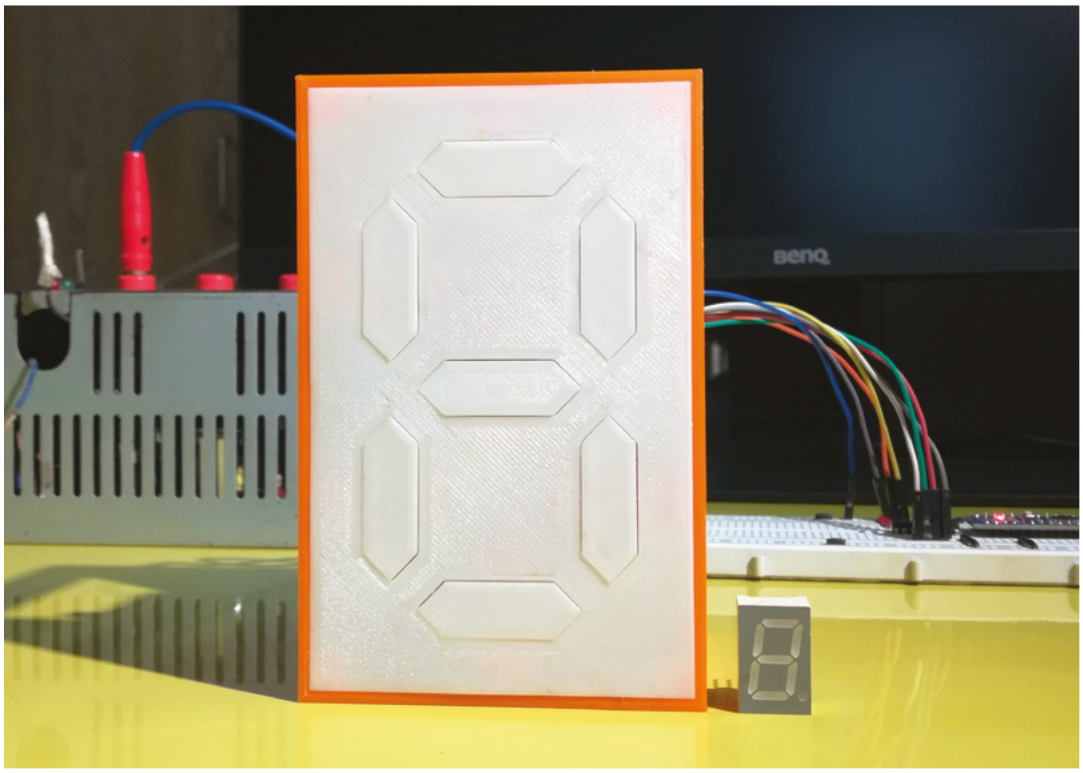
REGULAR

7-segment display

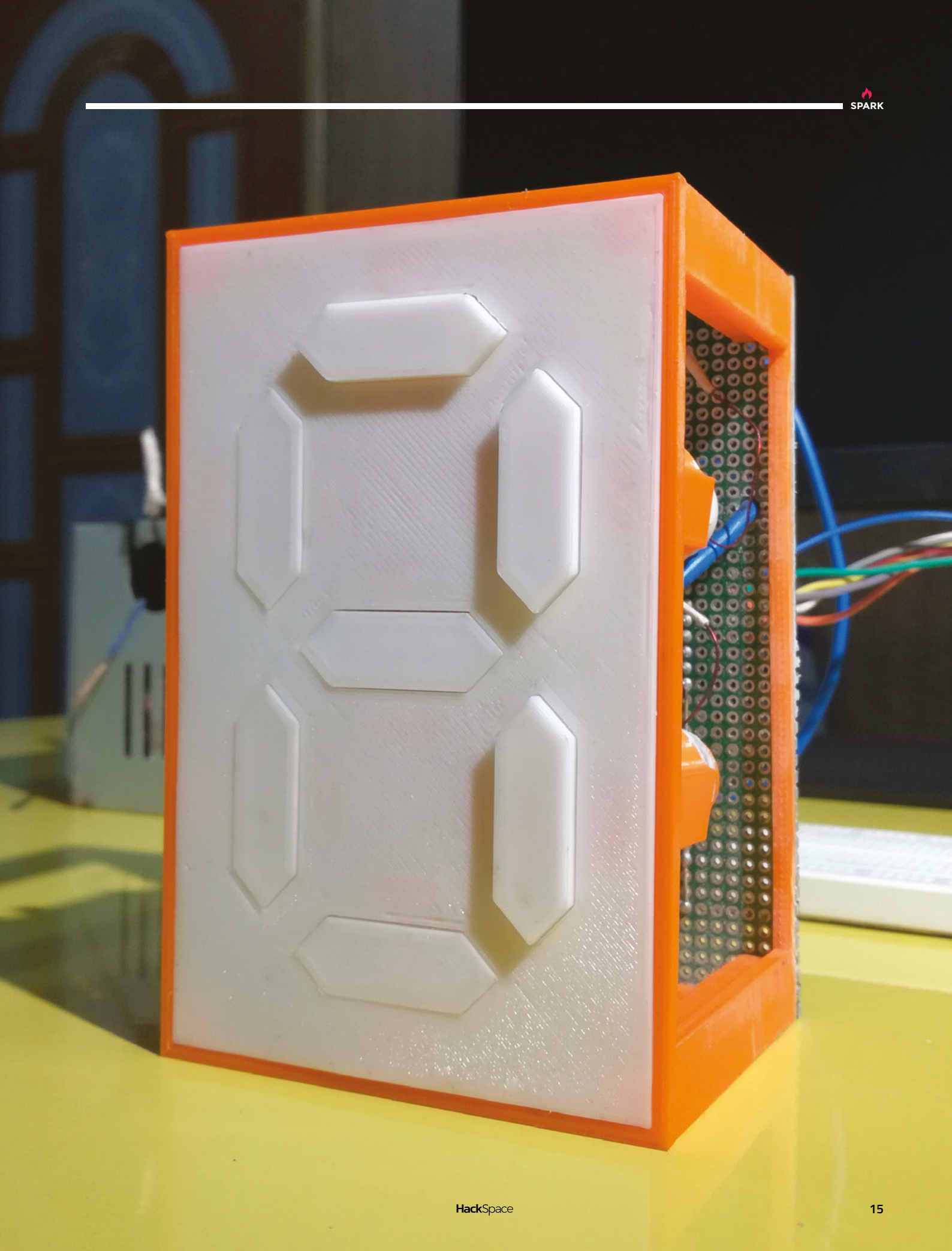
By Neeraj Rane

hsmag.cc/m4GEm0

We all spend far too long these days staring at screens. One solution to this is to replace the digits on your output devices with mechanical seven-segment displays. OK, so it's not very practical, but it looks amazing and won't give you eye strain. This design is by electrical engineering student Neeraj Rane, who replaced the servos used in similar mechanical displays with electromagnets, because he preferred the sound they make. The non-electric parts are 3D-printed and, modestly, Neeraj says it's "useless but fun".



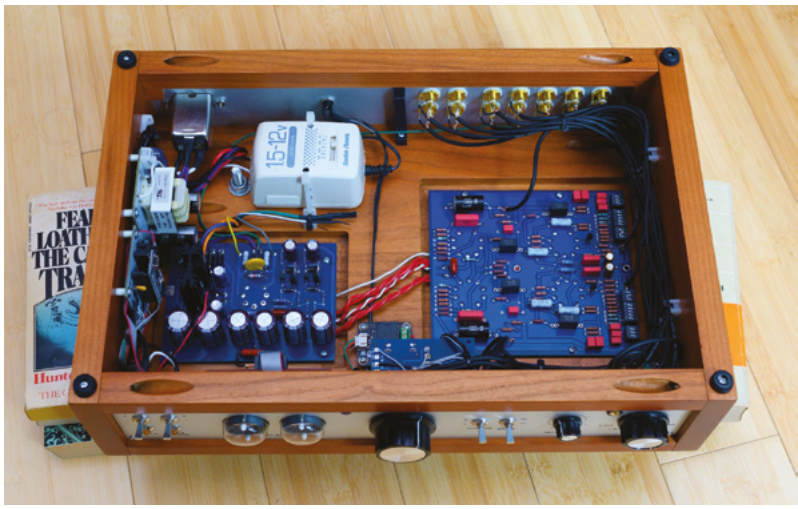
Right
Watch Neeraj's video to see the seamless digits in action



Vacuum Tube Preamplifier with Nixie Display

By Justin Scott

hsmag.cc/vrD6Gr



Audio geeks are notorious perfectionists, forever finding things to tinker with in the elusive quest for better sound. This preamp by Justin Scott combines the warm analogue sound of vintage vacuum tubes with the glowing cold war look of Nixie tubes, to show the volume level and input selection. There are loads of amp designs available – Justin used a PH-16 kit from **tubes4hifi.com** – but the enclosure is all his own.

Below
Justin added several features to the stock design, including a motorised volume knob with remote control

Objet 3d'art

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

We've pretty much got used to the idea that masks are here to stay. It took a while, and we felt funny at first, but they're normal now, with some people even turning them into fashion statements. That's why we're impressed with early adopter Bane who, back in 2012's *The Dark Knight Rises*, was wearing his mask with aplomb as part of Gotham City's autumn/winter look.

This fantastically detailed Bane mask was printed on the Project R3D RailCore II 300ZL printer, using SnoLabs Black PLA+, at a layer height of 0.2mm, and is the work of Fernando Hernandez, aka Dsk001. He modelled it himself, using a photo for inspiration. It won't protect you from COVID-19, but it will give you super-villan chic. If you're new to using Fusion, or you want to get a sense of what can be done, Fernando has shared a video of the design process here: □

hsmag.cc/mQtAfg

REGULAR

Meet The Maker: Ross Atkin

Robots for the people; people for progress

If you've ever fancied unlocking the processing power in your phone to power an AI robot made of cardboard, you should really check out the work of Ross Atkin. He's taking seriously high-spec robotics kit and packaging it in a low-cost robot, making as much as possible out of one of the most readily available materials we know of: cardboard.

We spoke to Ross live via telepresence (more on that in a moment), using his latest creation, the Smartipresence. This is a crowdfunded kit (it should still be open for backers by the time you read this) that bolts onto his ongoing creation, the Smartibot, and uses affordable materials to bring the boardroom experience to the kitchen table. It's not as good as seeing your distant relatives in the flesh, but it is a lot better than Zoom. So Ross: what's it like, splitting your time between a normal job and being a maker?

> "Introducing kids to design and engineering is massively important. **They're not exposed to it anywhere near enough in the education system**"

"Introducing kids to design and engineering is massively important. They're not exposed to it anywhere near enough in the education system. Even what we consider to be STEM education and the stuff there has been emphasis on I don't think is that good at equipping kids with the skills that they actually need. It's almost set up to exclude masses and masses of kids, because there's far too narrow a focus on programming, and not enough on making things creatively.

"That's how I got into programming. I did an engineering course, where they taught us C. I couldn't learn C like that; it was impossible. 'You're not going to make anything, but we're going to make you learn about pointers.' Great. These kinds of really abstract, hard to understand things.

"I tried again when I was at art school because I really wanted to make stuff, and in order to make the stuff I wanted to make, I had to program an Arduino. By helping people make things that they already want to make, you're so much more likely to engage a much wider cross-section of kids than if you're saying, 'We're going to make this thing that's only going to appeal to the geekiest subset of kids with the geekiest subset of parents.' And those people don't need our help: they were going to learn anyway. The people who need our help are the people who were on the cusp of doing that, or the people who will never try.

"Whereas if you show a child a toy monster truck with a camera turret on top and say, 'We're going to make this – we're going to 3D-print parts, we're going to wire up the electronics, we're going to program it so that it drives autonomously and follows a car that you're driving,' that's cool! No one is going to not want to do that, and you've learned so many skills in doing that.

"That's one of the reasons I was keen to be on a TV show called *The Big Life Fix*. I did two series and a Children In Need special. The producers had brought together a small team of designers, engineers, and computer scientists. We built one-off bits of technology to solve problems for individuals and →

Right
Ross's latest creation, the Smartipresence, is an affordable, easy to use telepresence robot

REGULAR

Left
The motor that tilts the phone is a direct drive 1:220, so there's no need for gears

groups of people. One of the groups was some farmers in the West Country who kept getting their sheep stolen. One was a village in Wales that didn't have any phone or internet connectivity. But the second series was all about addressing individual disability needs, because the dynamic of the programme worked better with an individual.

"A lot of the work I do is with disabled people using very similar methodologies, but in that work, even if it's with people with a particular impairment, you're still trying to design for a group. Whereas with the TV show you're designing for one person, which is a very cool experience, and not something you get to do very often as a designer.

"You get meaningful feedback all the time when you engage with the people you're trying to represent and design for. But it's much easier to make decisions when you're making for one person. All that matters is do they like it or not: you're not trying to balance competing needs. And you get to manifest their personality in the design, which is such a great thing to be able to do.

INFRASTRUCTURE

"My first job was designing street furniture. Litter bins, things like that. At the time, the Disability Discrimination Act had just come into force, so one of the first projects I got set was to design a range of street furniture that was compliant with the Act. It was frustrating because the legislation didn't have any guidance on how to make things work – it just said 'make a bench like this'. But there was no reasoning behind the decisions, so if you couldn't make a bench like that, there was no way to determine what was the next best thing.

"I did a master's in product design at the Royal College of Art, after which there was an opportunity to do a project looking at the needs of people with sight loss and street design. I had already worked in that field, and this was an opportunity to learn more. That project involved following a load of people with sight loss around on journeys they'd typically make unaccompanied, talking to them and analysing how the street could be designed better.

"That led to doing some design work in that space, and I then did a few research projects for Stannah, then TFH, who make special-needs toys, and a couple of projects for Scope, talking to disabled people about technology and the environments.

"When you're working on infrastructure, things move very slowly. I do a bit of work with technology with one of my clients, and that moves a bit faster – they're a really big company, and they've got loads of layers of management. I've done loads of projects with them that I've enjoyed doing; it was good work. But none of them turned into anything that anyone uses. That's a bit frustrating.

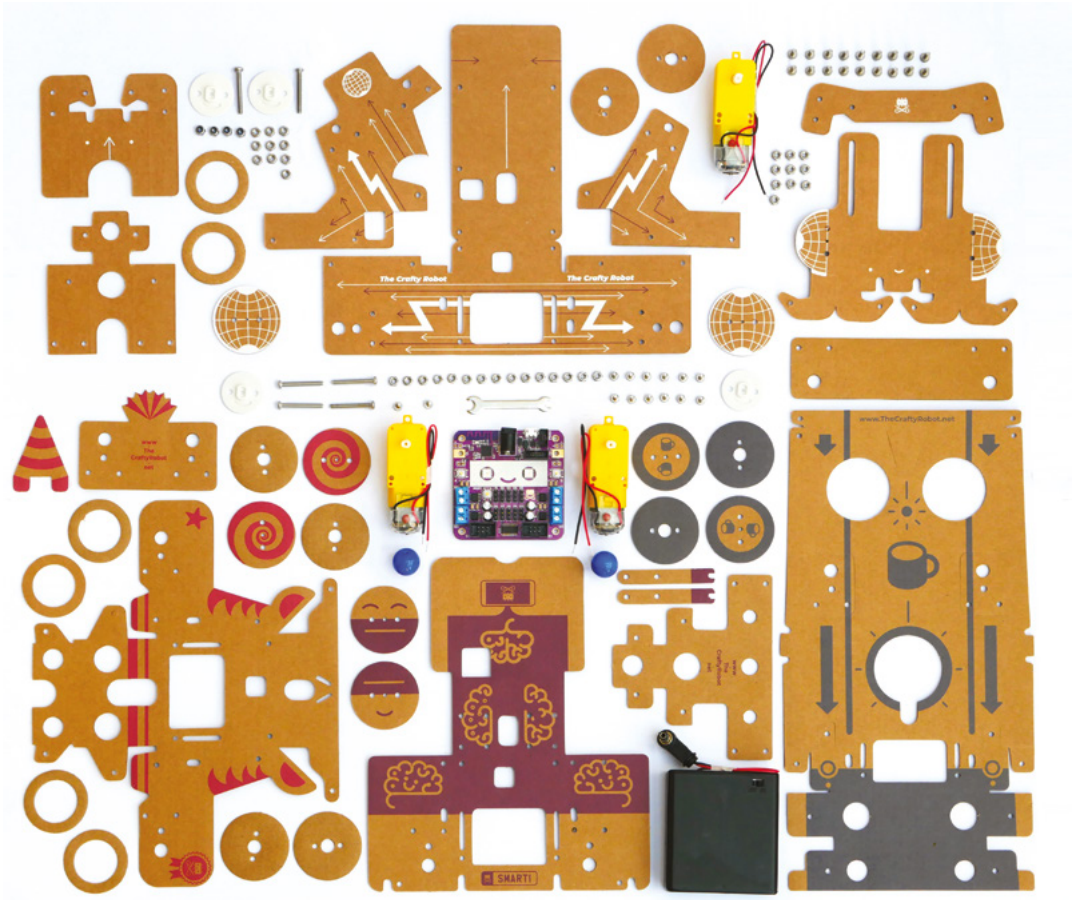
"With the infrastructure stuff, some of it is in production now, but it's really hard to get it out there and in use, because there are so many people who need to buy into it and it take a long time. And so I realise that with the robots and doing Kickstarters, you can get products out and into people's hands really quickly.

> "You get to manifest their personality in the design, **which is such a great thing to be able to do**"

SMARTIBOT

"This brings us back to part of the reason I wanted to do Smartibot. I realised that on the TV programme we kept building the same basket of electronics: Bluetooth connectivity, motor drivers etc. – and actually there are many bits of assistive tech that have the same bundle of electronics at the core of them. And you can adapt the software really easily, right? Once you've got Bluetooth connectivity, you can pair that to just about any input device. And if you can drive motors, then you can act upon the physical world. →

REGULAR



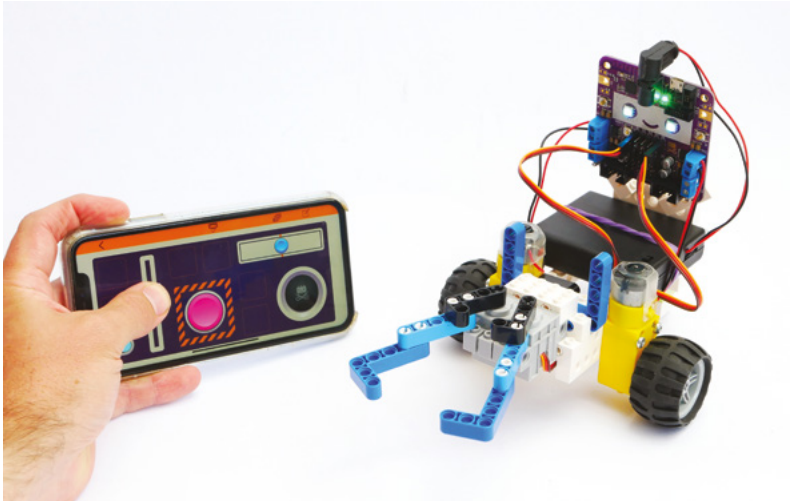
Above
To keep costs down, as much of the Smartibot as possible is made of cardboard

"The problem with the economics of assistive technology is that disabled people have a very diverse set of needs, far more so than non-disabled people. Ideally, you'd design for one person, but the more people you try to include, the less good a fit it's going to be for each person within that group.

"So you have this inherent trade-off. With traditional manufacturing, you have to make enough volume to offset the fixed cost of production, but the smaller number of people you're selling to, the lower that volume is, and the more expensive that thing ends up being.

"The way that you challenge that economic reality is by making mainstream products that are made in mainstream volumes have assistive applications. That's what Apple has done brilliantly by putting a load of assistive tech into iOS devices. They have a screen reader [VoiceOver], and they have Switch access, which is quite an esoteric feature that's only needed by a small number of people. But it's very impactful for that group, so it's really impressive that they built it in.

"Making robots for kids is a mainstream use case – that's how I came up with the Smartibot. I can do proper volumes in that. With 3D printing and stuff and the inherent flexibility of the hardware, we tried to turn what had been a product development problem into a 3D printing and code problem, because we've solved the bit in the middle. We've given you this electronic thing, and if you want to build an assistive tech product, you can integrate that with any input method you can. For the output, you've got a mechanical engineering problem, but there are a hell of a lot more people who can solve



that problem than there are people who can design a circuit board and put it into production.

"One of the things you'll find when you start dealing with factories is that a lot of the assumptions you have simply aren't true. If you get a nice company who are happy to work with you, that process of exchange is something I really enjoy. You might not get what you thought you were going to get, but you'll end up with something you still really like, by understanding what the limitations are. That's something you don't get with software.

"I was making cardboard wheels to fit the motors, but the minute you put a lot of torque through one, the shaft just rips off the cardboard. So we needed a way of putting torque into the wheel that didn't do that. I'd been trying to make the whole thing out of cardboard, and it is mainly cardboard, but we needed this tiny plastic part mounted to the shaft with holes in it so that you can mount it to the cardboard.

"The Smartibot is an Espruino-based board, which is a platform that no one seems to have heard of.

It's by a guy called Gordon Williams who's just outside Oxford, and it's a full JavaScript implementation for microcontrollers. The board runs JavaScript, you send it JavaScript, and you program in JavaScript. There's a blocks editor that works the same as MakeCode.

"You connect to it over Bluetooth, send it JavaScript, your robot's running your program, and you can send it more JavaScript any time you want.

"The app we designed is completely customisable. You can edit the control pads, and then you've got these choices of control elements that you can drop in. Say we put in a button; to keep it simple, you can just drop it in. The board has 14 motor controllers and 10 servo connectors, and you can set motor positions for those. You can map as many commands to each button as you want. Here, there's a test thing, so I can write a function in the robot firmware that works off that button, and you've got a fully generalisable control system, and you can pass arguments to it as well – whatever you want, because it's just JavaScript.

"It does things like mixing, so if you've got two axes on a joystick, it can mix between two motors for steering, or to take the monster truck example, you can mix an up motion in a joystick with side to side to control the steering servos. It gives you masses of flexibility built into the app without needing to do any programming.

"To me, that's the approach. Don't just plonk them down in front of a computer with 'Hello World' and be surprised when they don't get it. Meet people where they are and take them to where they want to be". □

How low do you go?

What do you want to make?

Lucy Rogers

@DrLucyRogers

Lucy is a maker, an engineer, and a problem-solver. She is adept at bringing ideas to life and is one of the cheerleaders for the maker industry.

Do you remember school biology lessons, and learning about the food chain? A leaf gets eaten by an insect and the insect then gets eaten by a bird. Every time you go along the 'chain', there is a loss of energy.

Over the past few months, many of us have been doing things that take us lower down the food – or purchasing – chain, by making things. For example, making meals from the raw ingredients rather than buying ready meals, baking bread, making cakes etc. There are those of us who have grown vegetables. I even bought a SodaStream so that I can make my own fizzy pop.

There are many who have made their own clothes, masks, or other textile items. Many home haircuts. Many DIY projects.

It got me thinking about how far down the chain I could go. Self-sufficiency has always appealed to me, but I also know there is a lot of 'grunt' work involved – hard manual labour that can get monotonous. And that puts me off. I have made string and rope from stripped willow and lime bark, and also from stinging nettles. I have lit a fire by rubbing two sticks together. I have made a shelter using branches, twigs, and leaves – and slept in it.

But that kind of life, for me, would mean I wouldn't have time to do all the other things I enjoy – and I am far too lazy. So it seems I like to know that I 'could' do these things if I had to – but I'd rather not have to. There's a line. How far down the chain am I prepared to go? Which things do I enjoy doing, and which will I pay to not do – which things do I depend on others to do? What things can I not do, even if I wanted to?

The last few months have made me appreciate the things I do depend upon – particularly water, electricity, gas, and internet connection. But also the ability to get food when I want it, and purchase everyday items, such as light bulbs and toilet rolls, when I want them. But I hadn't really considered my health – until I was stung by a wasp. I've been stung before and, after the initial 'ouch' and applying vinegar, I've generally ignored them and forgotten about them the next day. This time I had a reaction. I went to the chemist to get an antihistamine cream. The next day I went back to see if a tablet would be better. The pharmacist took one look at the sting – the swelling and the hot red patch and told me to see a GP immediately. After an e-consultation, where I had to send photos of the reaction, I was given antibiotics.

> **I rely on others much more than I had realised. And I now appreciate them all a lot more**

Although I like to think I am pretty independent and self-sufficient, and can turn my hand to most things, I rely on others much more than I had realised. And I now appreciate them all a lot more.

Open-source space

Hacking at the final frontier

Drew Fustini

@pdp7

Drew Fustini is a hardware designer and embedded Linux developer. He is the Vice President of the Open Source Hardware Association, and a board member of the BeagleBoard.org Foundation. Drew designs circuit boards for OSH Park, a PCB manufacturing service, and maintains the Adafruit BeagleBone Python library.

Space exploration is usually associated with national agencies like NASA, or with private corporations such as SpaceX. However, there is now a growing movement of people who believe that space shouldn't be limited to governments and companies, and that space exploration can be made more accessible with open-source technologies.

SatNOGS won the first Hackaday Prize back in 2014, with their global network of open-source satellite receivers. There were already a number of amateur satellites in space that had been designed and launched by universities and space enthusiasts from all over the world. However, until SatNOGS came along, there was no way of getting regular data for your satellite, as it would only pass within reading range a handful of times per day. The success of the SatNOGS project led to the creation of the Libre Space Foundation (**libre.space**).

> **Space exploration can be made more accessible with open-source technologies**

The Libre Space Foundation, founded in Greece, aims to make space exploration accessible by developing free and open-source technologies. Alongside infrastructure projects, including the SatNOGs satellite receivers, they work on satellites and rocketry. Their UPSat was the first open-source hardware satellite, and it was successfully delivered to the International Space Station, then deployed into orbit in 2017. This deployment was a remarkable achievement: a real milestone in open-source space exploration.

Whenever I'm in Oregon, I make sure to visit Portland State Aerospace Society (PSAS), an interdisciplinary, open-source student aerospace project at Portland State University. PSAS makes composite amateur rockets, liquid-fuelled rocket engines, and CubeSats (a type of small satellite made up of 10 cm × 3 units). Over the last 20 years, PSAS has had 13 launches of four generations of amateur rockets. Their current rocket is Launch Vehicle 3.1, a four-metre-tall, solid-fuelled rocket that goes up to about 5 km.

PSAS is also developing a CubeSat project called OreSat. OreSat is an impressive open-source system of modular, expandable satellite designs. Their first small satellite, OreSat0, should be completed in November, then dropped off in a sun-synchronous, low-earth orbit in February 2021. All of the hardware and software developed at PSAS can be found on their GitHub page: **github.com/oresat**.

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**

CUDDLY ELECTRONICS

Can I just say, I loved the tutorial on using a Furby as an output device. It's one of those things that I know I now need, but I don't know what I need it for.

Ali
Paris

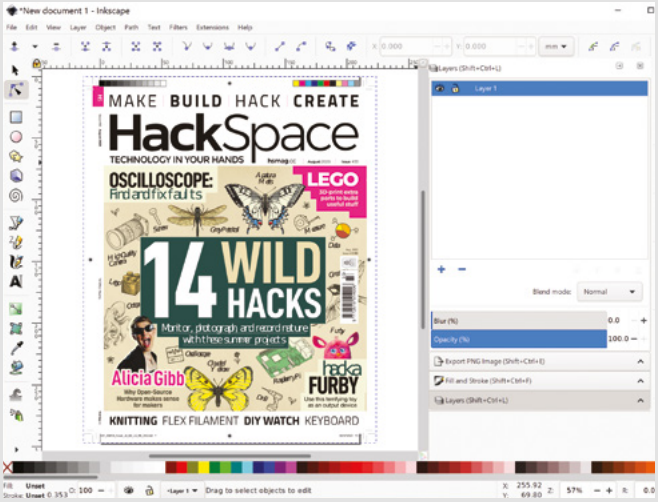
Ben says: Ah, the old solution searching for a problem. Yes, the Furby is one of the coolest output devices we've come across. They're not small though, so it needs to be a pretty chunky project. There's no reason to limit yourself to just one either – Look Mum No Computer used 44 of the fluffy critters to build an organ (the musical kind, not the internal kind). Take a look here for some inspiration: hsmag.cc/QonIf0.

VEXING VERSIONS

I've been using Inkscape for years, and it was only when reading issue 33 of HackSpace magazine that I realised they had only just released version 1.0! It's amazing software – why should they keep it at version 0 for so long? Anyway, I just downloaded the latest version, and I'm loving the new path features.

Paul
Derbyshire

Ben says: Version numbers are strange things. With commercial software, they're often used for marketing reasons as much as anything. In the open-source world, developers often start on version 0 and never quite find the right time to jump up to version 1. It's only a name and that which we call a rose, by any other name would smell as sweet; that which we call version 1 would, by any other number, work as well.



NATURE

The stereotype of the nerd is someone who loves technology and hates the outside. Sure, I've met a few people like this, but personally, I'm really interested in nature, and lots of the nerds I've met are as well. I really enjoyed learning about the ways of combining these two interests in issue 33. Oh, and I've been doing my bit for the environment by counting penguins. Am I a scientist now?

James
London

Ben says: Technology and nature are often seen as opposites, but they're really not. They're just two different things, and there's loads of ways the two interact, and if we do it properly, they interact in positive ways.

In my view, anyone who does science is a scientist, whether that's counting penguins, or performing complex experiments.

CROWDFUNDING NOW

Open Game Station

DIY handheld entertainment

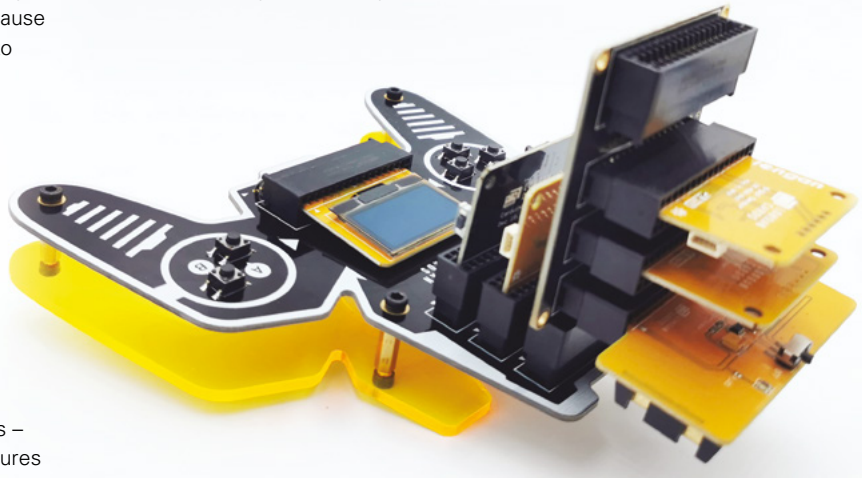
From $25 | **crowdsupply.com** | **Delivery: Nov 2020**

The Open Game Station is a modular, handheld gaming system based around the 8-bit ATmega32U4 microcontroller. While this isn't the most powerful microcontroller around (far from it in fact), it's got enough oomph for simple 2D games.

Perhaps the biggest hurdle for any maker gaming system is getting a good selection of games, because while it's fun to program your own, it's also nice to have some others to have a play with. The Open Game Station solves this problem by being compatible with the Arduboy. This means that there are hundreds of games already available (there are 277 on Erwin's Arduboy Collection, and this isn't exhaustive: **arduboy.ried.cl**).

The form factor of this portable looks like it'll fit nicely in your hand (although we haven't held one to test this out properly), though it isn't as portable as the original Arduboy. The Basic version ($25, plus shipping) comes with the basics you need for gaming – a microcontroller with screen, buzzer, and batteries – while the Pro version ($35) adds a few extra features for getting input (an accelerometer and sound sensor) that could be interesting options for creating unusual game styles.

If you're looking to test your programming skills on some limited hardware, or just want to sit around and play retro-style, 8-bit games, the Open Game Station looks like it'll be a great-value option.



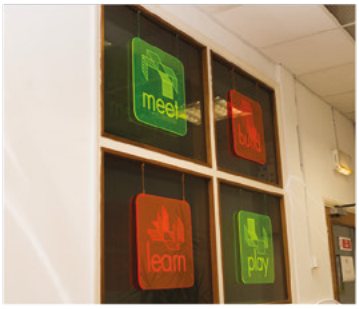
BUYER BEWARE

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.

Left
The expansion slots give you the chance to add features

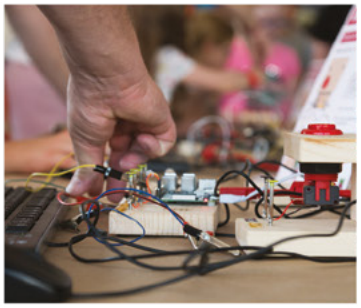










Build a Makerspace for Young People

Join our **free online training course on makerspace design** to get expert advice for setting up a makerspace in your school or community.

Sign up today: rpf.io/makerspace

HackSpace
TECHNOLOGY IN YOUR HANDS

LENS

HACK | MAKE | BUILD | CREATE

Uncover the technology that's powering the future

FEATURE

DISCOVER FEATHER

PLUG AND PLAY FOR EASY ELECTRONICS

When starting a microcontroller project, most of us don't reach for bare chips – though this is certainly possible – we start with development boards. These package up a microcontroller and a few other basic features, such as power management, into an easy-to-use module.

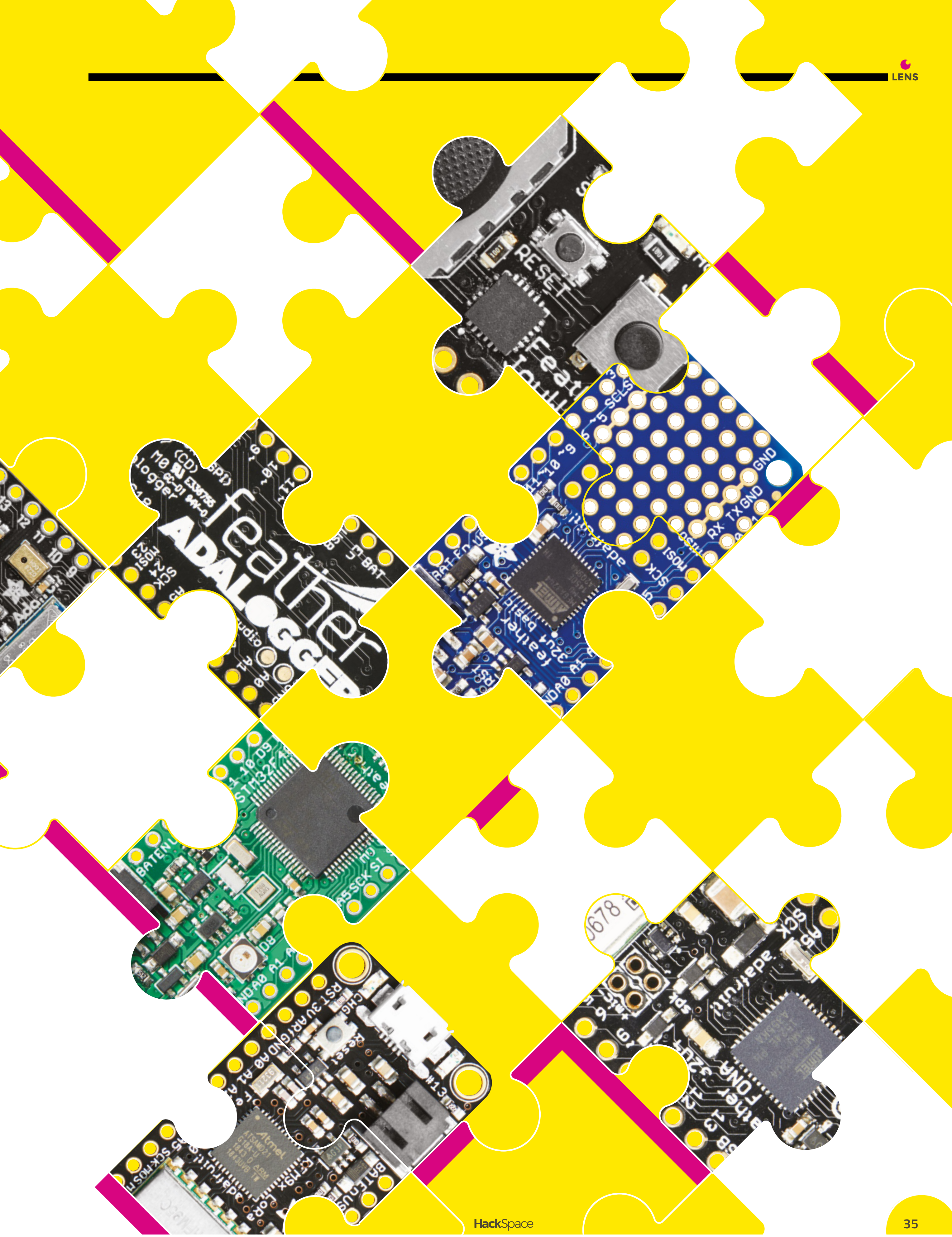
Development boards come in all shapes and sizes, from large and packed full of features, to teeny-tiny with only the barest of essentials. So, which should you choose for your next project? The form factor influences a number of things – which microcontroller you can use, what add-ons are available, what features you can fit on board, and of course, what size your final project will be.

There is no perfect form factor as we all have different needs and want different trade-offs, but we find that there's one that seems to fit well in a large proportion of our projects: Feather.

The Feather form factor is small – though not tiny – it's big enough to accommodate most microcontrollers alongside a battery charging port, and sometimes a wireless comms module.

The really great thing about Feather, though, is just how big the ecosystem is. You can use microcontrollers from most of the major families (including ARM, AVR, and ESP) and couple these controller boards with 'Wings' which add functionality.

Each Feather board is 2 inches long by 0.9 inches wide, with 16 pins on one side and 12 on the other. These pins each have a specific function that has to be in the same place on every board. That means that you can take an expansion board and it should work with any Feather mainboard (there are a few exceptions to this, but not many). This is the big advantage of Feather – a wide range of microcontrollers works with a wide range of Wings. This means that you've got a huge amount of flexibility in how you create your project. Also, for board makers, it means that you only have to design your mainboard and you automatically get compatibility with hundreds of expansion boards. Let's take a closer look at how this ecosystem came to exist. →


RESET
feather
ADALOGGER

THE BIRTH OF A NEW PARADIGM

WE CHAT WITH LIMOR FRIED TO FIND OUT HOW FEATHER CAME TO BE

Feather is the brain-child of Limor Fried, CEO of Adafruit, and the majority of Feathers and FeatherWings available are from Adafruit (though there's an increasing number of boards from other companies and individual makers).

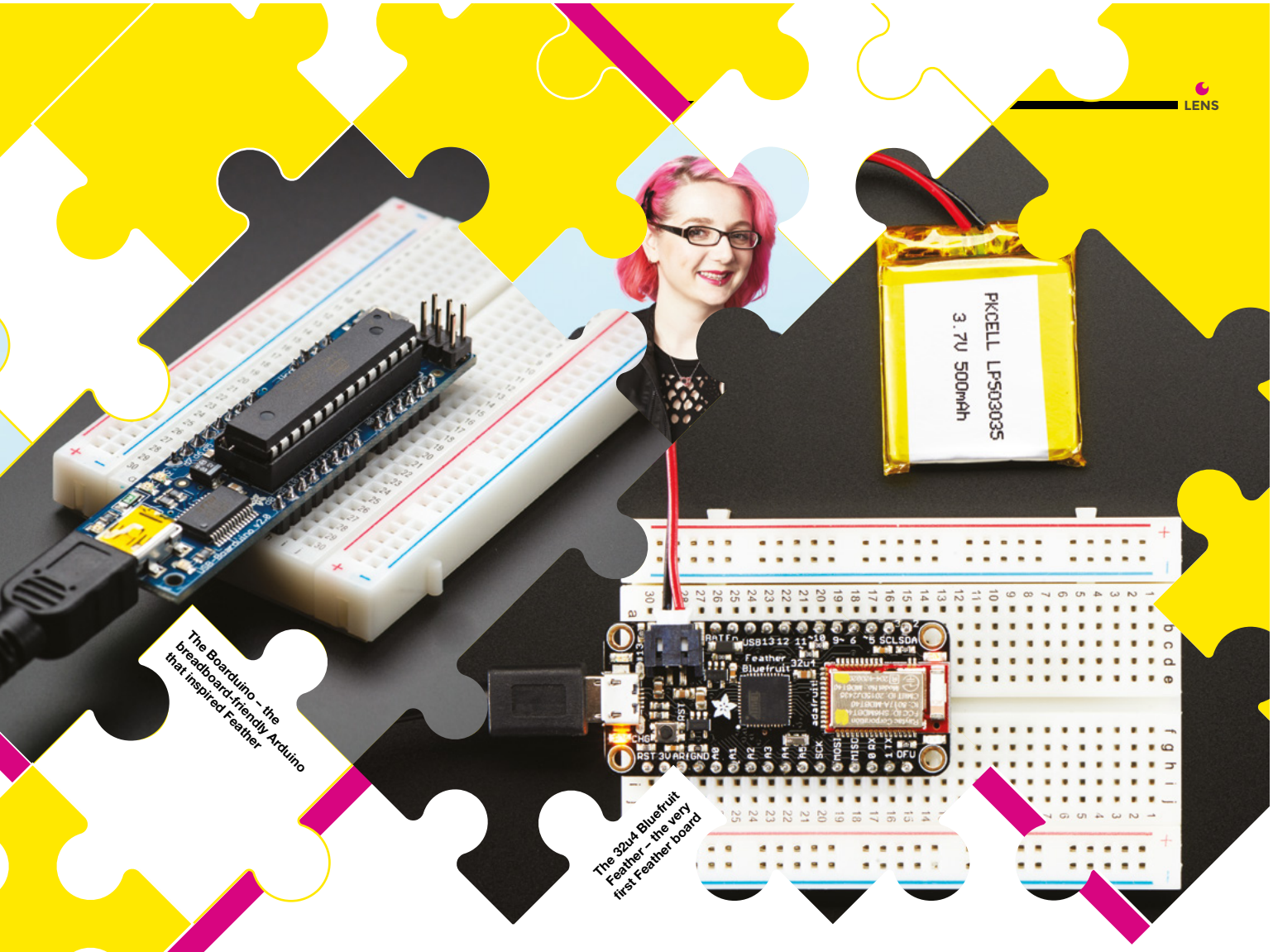
Limor started publishing electronics projects on her blog in 2005. Soon, readers started requesting kits that brought together the parts needed to create the kits. From this, Adafruit was born and gradually became one of the largest businesses in the maker world. Today, Adafruit is synonymous with hobbyist electronics, and there are few electronics makers who haven't used either their products or their software.

As well as building a business, she's found time to become a YouTube star and her weekly *Ask an Engineer* show (one of several she regularly stars on) is the longest-running live electronics show in the world.

She took a little time out from her busy schedule to chat with us about how Feather came to be the ecosystem it is today. Here's what she had to say.

HackSpace: Was there a particular moment of inspiration behind creating the Feather?

Limor Fried: The Feather comes from a very long lineage of development boards. Around 2007, when the Arduino was first released, we wanted a breadboard-friendly Arduino-compatible and got the Arduino team's blessing to call it Boarduino (**learn.adafruit.com/boarduino-kits**). Once we got a pick-and-place machine in 2010, we made an ATmega32u4 breakout (**adafruit.com/product/296**). Then in 2015, once low-cost BLE [Bluetooth Low Energy] modules became available, we stuck one onto the 32u4 breakout to make an all-in-one dev board: the Adafruit Bluefruit LE Micro (**adafruit.com/product/2661**). The pinout on the Micro is nearly the same as the Feather – all that was missing was a battery charger that would let people make wireless BLE projects. By then there were other wireless chips like the ESP8266 that would benefit from a breadboard-friendly, battery-powered board, and thus we made the Feather!



The Boarduino – the breadboard-friendly Arduino that inspired Feather

The 32u4 Bluefruit Feather – the very first Feather board

HS: I think the first Feather was the 32u4 Bluefruit back in 2015. When designing that, did you think you were creating a form factor that would take off in the way it has?

LF: Yes! It took ten years to get there, but I really thought I had a great form factor. One thing we did early was quickly spin up a variety of 'add-on' FeatherWings with different sensors, displays, GPS, to make sure that we could support complex projects.

HS: If not then, at what point did you realise that Feather was going to take off as an important form factor?

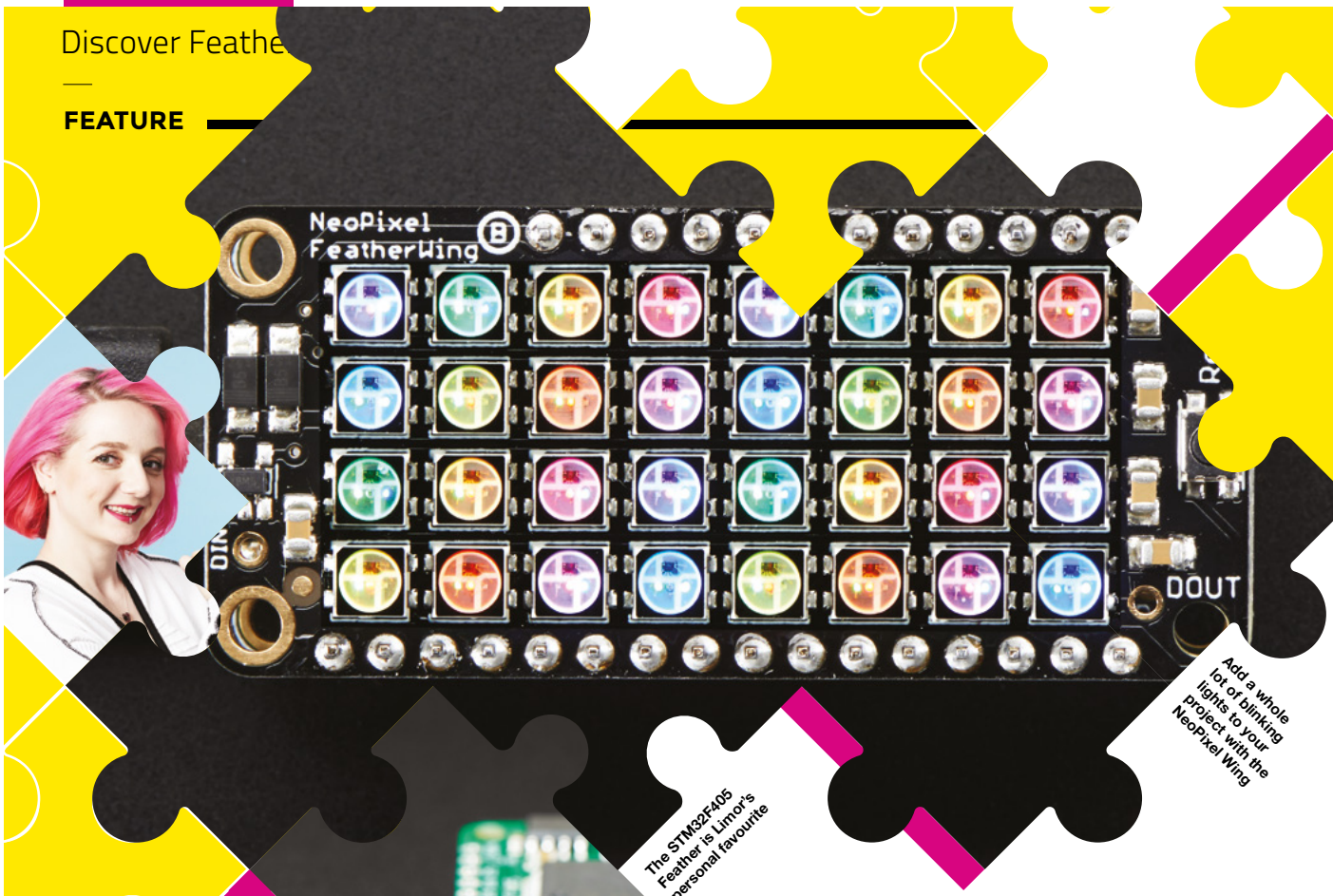
LF: A few years ago, Particle (a manufacturer of Feather-like cellular/WiFi/BLE boards) emailed us and asked if they could join the Feather ecosystem – they wanted to make dev boards for their chips and software, but did not want to design all the accessories which is very time-consuming. Joining Feather means instant access to 100+ Wings! We published specifications and the latest set of Particle boards are now Feather-compatible!

In the last two years, we've seen other electronics companies create Feather compatibles and Feather accessories, and that's been really awesome too.

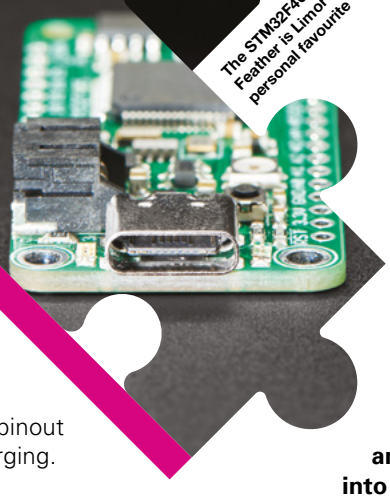
HS: Why is Feather the shape and size it is? Did you start with a size you wanted to hit, or did you start with a list of features, and see how small you could make it (or something else)?

LF: We wanted it to be breadboard-friendly, and still have a few 'holes' on a solderless breadboard, but wide enough to fit common wireless modules. So that sorta made the width 'fixed', then we didn't want it too long because we have seen that it's easy to bend pins if the aspect ratio is too high. Two inches seemed good. We wanted lots of GPIO, and the I²C/SPI/UART/analogue pins to be in fixed locations. Mounting holes are helpful for folks making enclosures. The design really just evolved from the previous layouts!

HS: LiPo charging is perhaps the one feature that Feathers have to have (beyond those provided by the microcontroller). Was there a temptation to include →



Add a whole lot of blinking lights to your project with the NeoPixel Wing



The STM32F405 Feather is Limor's personal favourite

anything else as a required thing?

LF: Nope – we wanted to keep Feather simple – the only real requirements we have are the pinout ordering, USB, and battery charging.

HS: Feather's about five years old now, and a lot has changed in the world of microcontrollers in that time. Is there anything you wish you'd done differently in the design?

LF: Yes! I wish I'd made it 1" wide instead of 0.9" – at the time, wireless modules were not as big as they've gotten now. So modules like the ESP32s really get close to the maximum width. It's OK, but a little more room would be nice. On the other hand, we really wanted to maximise available breadboard pins.

HS: Where did the name Feather come from?

LF: Phil [Torrone] and I have a technique when we need to come up with names – we go to a local dim sum restaurant on a Sunday, order all sorts of dumplings and tasty fried foods, and discuss what the design is. In this case, we had something untethered, light, flat, and thin. Feather was not already in use and evoked the free-floating nature of a wireless board. Phil is an excellent brand name creator; Feather is just another one of his big hits!

HS: Feather is an open standard – anyone can make and sell Feathers. Did you consider any way of restricting it? What factors played into the decision to keep it open?

LF: No way, we were really happy to publish the specifications. We only ask that people are respectful to the users – if you change something drastically about the pinout, shape, or functionality so that other Wings won't work, please don't call it Feather, so people don't have a bad experience! We had seen the success of Arduino + shields, Raspberry Pi + HATs, heck, even PC/104 – an open standard is essential to a strong ecosystem.

HS: If someone wants to make their own Feather, what advice would you give them?

LF: Check out the Feather specification documentation at **hsmag.cc/JmdAFI** to get an idea of what is required to make something Feather-compatible. You can also look at the Awesome Feather GitHub repo to get inspiration, and when you've completed your Feather

or Wing, submit a pull request so we can have it included at **hsmag.cc/Rd6OYF**.

HS: Are there any Feathers that you'd like to exist, that don't yet?

LF: I'd love to see some RISC-V Feathers, maybe one featuring the K210 which has machine learning acceleration. For Wings, we think an RTK GPS or NB-IoT module Wing would be a very good addition. We're working on some of these, too.

HS: According to the Feather history page, you designed a Feather or Wing every week for a year. That sounds like a hectic schedule. How did it go?

LF: It was like a marathon! I tried to interleave different Wings and Feathers to do easy ones like the M0 series (where we made half a dozen SAMD21 boards with different add-ons like BLE, WiFi, SD storage, LoRa/sub-GHz radio…), and tougher ones like the FONA Feather that has cellular support (that took quite a few revisions!)

HS: Is there a Feather that you're particularly proud of?

LF: I really like the STM32F405 Feather. I think it came out very nicely with a powerful new chip I had not used before. There's an SDIO card-holder on the bottom, STEMMA QT connector for easy sensors, and it was our first USB-C Feather!

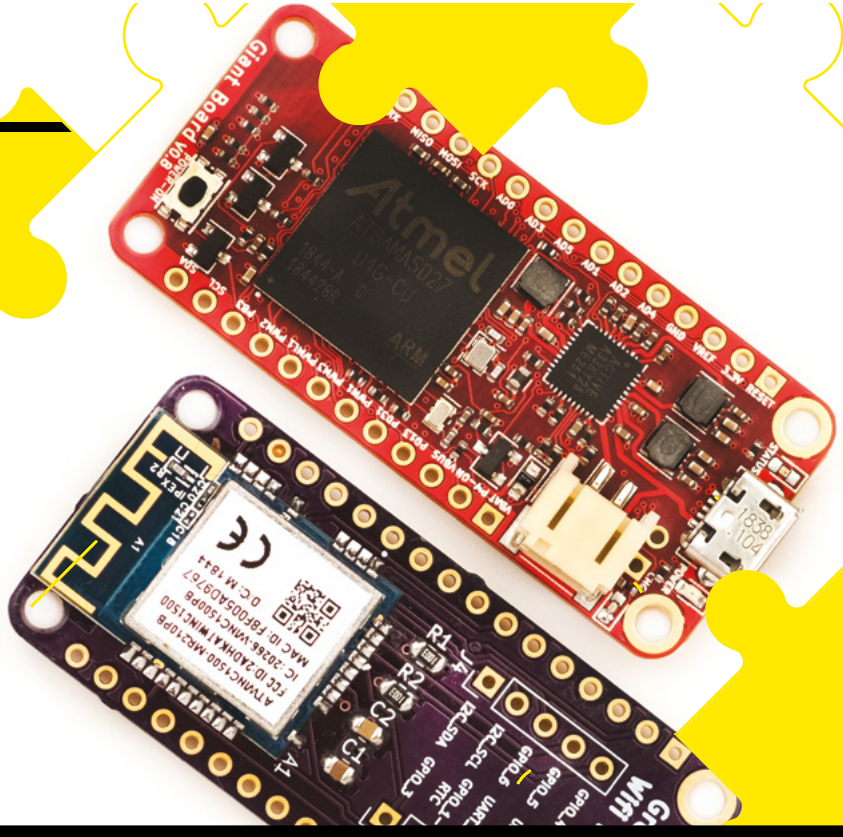
HS: Where do you see the Feather ecosystem going in the future? Is it a case of getting more processor power, RAM, and flash in chips, or do you think there are other things on the horizon?

LF: Yes, yes, yes! Powerful chips like the RT10x Cortex-M7s, maybe even STM32MP1 Linux chips (we recently reviewed the OSD32MP15x all-in-one Linux chip that would easily fit on a Feather).

We're also seeing a lot of accessibility technology (AT) for folks who want technology that can help them use their assistive devices. For example, the Freedom Wing Adapter (**hsmag.cc/KFs3HB**) can connect to wheelchair joysticks so that folks who are used to controlling wheelchairs can then turn around and use the same input device to play video games, pilot drones, browse on a computer. The Wing adds the wheelchair part, and then the Feather underneath is what adds Bluetooth, WiFi, or USB control.

HS: There are now quite a few people building Feather boards, both maker companies and individual makers. What's been the most surprising thing for you about this Feather ecosystem?

LF: Boards like the OrangeCrab and Evo M51, and other really powerful Linux-capable Feathers are really not what I expected! Folks have crammed 1GHz processors into the Feather format – I'm glad it's able to stretch to that sort of advanced functionality! →



GIANT BOARD

A PENGUIN FEATHER

Creator Chris Alessandro

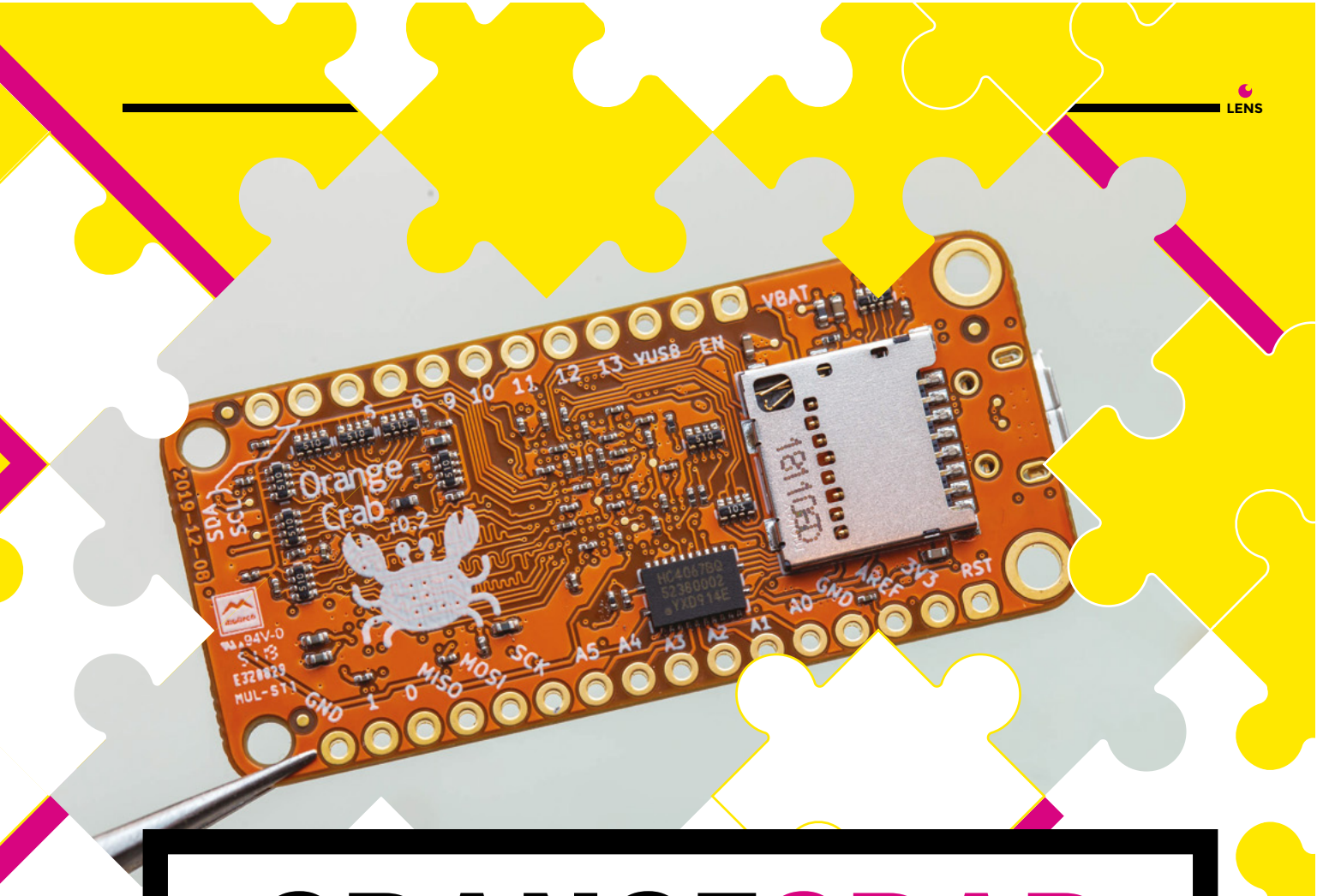
I've been making electronics and writing software for almost ten years now, and I always wanted to make my own single-board computer (SBC). I've made other SBCs in the past, but they were mostly just experiments. I had been browsing a large components supplier's website and came across the Microchip SAMA5 series of SiPs (system in package). It includes the processor and memory on the same chip, meaning that it makes it a lot easier to route the package in a small space. I had been trying to decide on a form factor when I had the light bulb moment, and had the idea to try and fit it all on a Feather-sized board. It seemed like a fun challenge, and I thought a lot of people would like it.

The Feather form factor was chosen because we like Adafruit a lot and what they've done for the community. There are also many pre-existing add-on boards, so it made a lot of sense to me to use the form factor. The size is also all super-small – smaller than most other Linux SBCs – so you could fit a fully Linux-capable board in a lot of spaces. I thought it would be interesting to try and fit a full Linux-based computer onto the Feather form factor as sort of a challenge as well. The size and pinout of the Feather worked well with the processor and other parts we had chosen.

If you're looking to make your own Feather form factor board, I would highly recommend studying the specifications provided by Adafruit.

If you're not sure where to start, having a look at existing Feather boards is also a great place to get ideas. As long as you stick to the spec, you should end up with something that works with a great line of add-on boards. Adafruit is always making new add-ons, and your board will continue to be compatible well into the future

I like the iterative design process and change things as needed. Order a board, test, test, and test. If something doesn't work right, change the design. I keep this process going until I end up with something that works well and meets all my needs.



ORANGECRAB

SQUEEZING AN FPGA INTO A SMALL FORM FACTOR

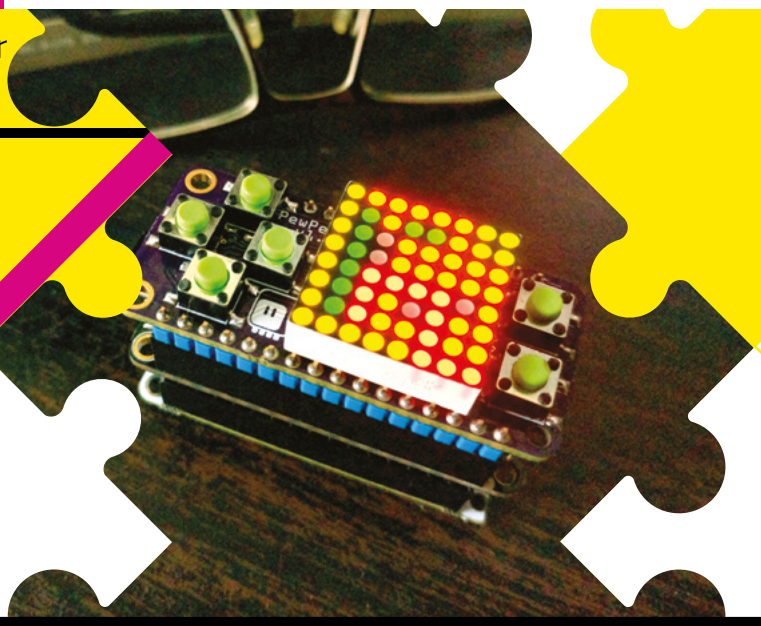
Creator **Greg Davill**

I started working on the OrangeCrab back in April 2019. Back then there were very few options available for making use of high-density DDR3 memory chips. I'd just finished working on a tiny board, the Boson Frame Grabber, a 21 mm × 21 mm PCB that had an ECP5 and HyperRAM. It was designed to fit behind a thermal camera from FLIR and add SD recording capabilities. Because I'd managed to fit everything on that very small board, I thought it would be a fun challenge for myself to make this new design on a small form factor board. I ultimately picked the Feather because it has a wide selection of available add-ons, and some very clear guidelines from Adafruit about creating custom Feather boards.

The biggest challenge on the Feather layout is just the size, in this particular design I was forced to use a very small package option for the EP5 FPGA that also makes the PCBs more expensive to fabricate. The routing for the DDR3 interfaces was also quite challenging, but achievable on a six-layer PCB.

A major comment on the OrangeCrab is that, for an FPGA board, it has very limited I/O. So a new project that I'm in the stages of prototyping is a custom board size, but using the SYZYGY FPGA interface standard. That's called the ButterStick.

If you're thinking about creating a Feather, my advice would be to go for it! There are lots of new and interesting development boards that people are creating in the Feather format, and I think it's awesome! →



PEWPEW FEATHERWING

TURN ANY FEATHER INTO A GAMES CONSOLE

Creator **Radomir Dopieralski**

I was running programming workshops [when I had the idea to build my Feather]. Python is, in my opinion, one of the best programming languages for learning, and making computer games is a great way to learn, so I wanted to teach programming by making games with Python. However, after trying several different things, it turned out that it's difficult to do. People who come to the workshops bring their own laptops with a variety of different operating systems, different versions, sometimes even without admin access. Installing and configuring everything on them takes time – you need the Python interpreter itself, you need an IDE, you need all the game libraries you are going to use, and so on. That can easily take an hour of the workshop before they get to draw a single pixel.

I tried to improve the situation in various ways, but ultimately I decided that I need a dedicated device, which I can prepare beforehand, with everything needed for the workshop already on it. However, at the time I didn't really have the skills and experience to make such a standalone device from scratch, so I used Adafruit's excellent prototyping boards, and only made an add-on, a FeatherWing, that added display, buttons, and battery necessary to turn a Feather into a simple game console.

Around that time Adafruit became interested in MicroPython, and they made their own fork, called CircuitPython, which was more focused on teaching and good experience for the beginners.

MOVING FORWARD

I have made a whole bunch since. My latest is a version of the Feather M0 Basic Proto that is as bare-bones as possible while still being compatible, called Fluff M0. It only has six components – even the USB port is just a cutout in the PCB – so that it can be made as cheap as possible. It can be useful when you want to run a CircuitPython workshop as cheaply as possible, or when you want to replace a Feather you used in a project with something that you can leave there without regret – and recover your Feather for a new project. I am also working on a couple of walking quadruped robot designs based on the Feather form factor.

OPEN BOOK

BUILD YOUR OWN E-READER

Creator Joey Castillo

The inspiration for this board traces back to the GNU Unifont project – this really ambitious effort to create one bitmap font that encodes glyphs for all the languages of the world. The idea of building an open hardware device for reading literature in every person's language was incredibly exciting to me, and I thought it would be empowering to make it an object that folks could build themselves and understand.

Lately, Feather is my first choice for most things I want to build. The compactness of the form factor is part of it, and the ecosystem of FeatherWings, but the built-in LiPo charging was what attracted me to the platform in the first place. I think it's brilliant that Adafruit made battery charging a core part of the spec, because it makes Feather match up really well with the way modern gadgets integrate into our lives. Most aren't tethered to a wall wart; they're battery-powered, and rechargeable over USB. A while back, I worked with Feather on another project, a data logger for back-country hiking. I could toss it in my backpack in the morning and plug it into a power bank to recharge in the evening, right next to my smartphone. That's possible for literally every Feather project, which is awesome!

GETTING STARTED

My advice is to just try some stuff! A standard FeatherWing-sized board costs $9 for three copies at OSH Park; you can design your idea, send it to them, and have a board in your hands a couple of weeks later. You can see what worked and what didn't, try again, and learn as you go. That's what I did, anyway! ◻

SUBSCRIBE TODAY

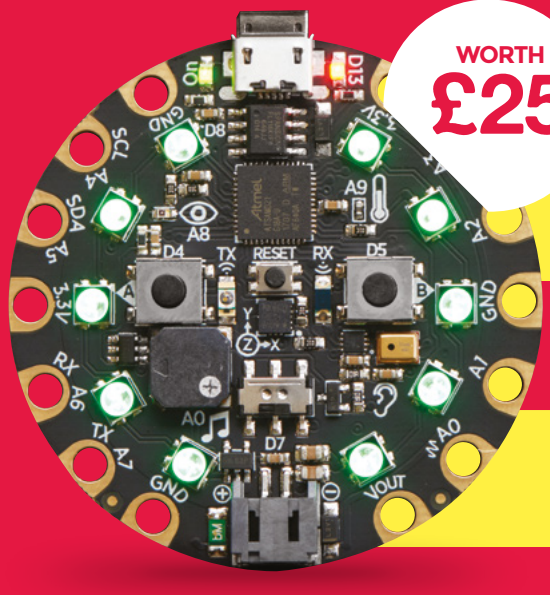
Subscribe for 12 months | **Rolling monthly subscription**

£55 (UK) **£90** (USA)

£80 (EU) **£90** (Rest of World)

Free Circuit Playground Express with 12-month upfront subscription only (no Circuit Playground Express with rolling monthly subscription)

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



FREE!

Adafruit Circuit Playground Express

With your first 12-month print subscription

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

SUBSCRIBE on app stores

From £2.29

Available on the App Store

GET IT ON Google Play

Buy now: hsmag.cc/subscribe

FEATURE

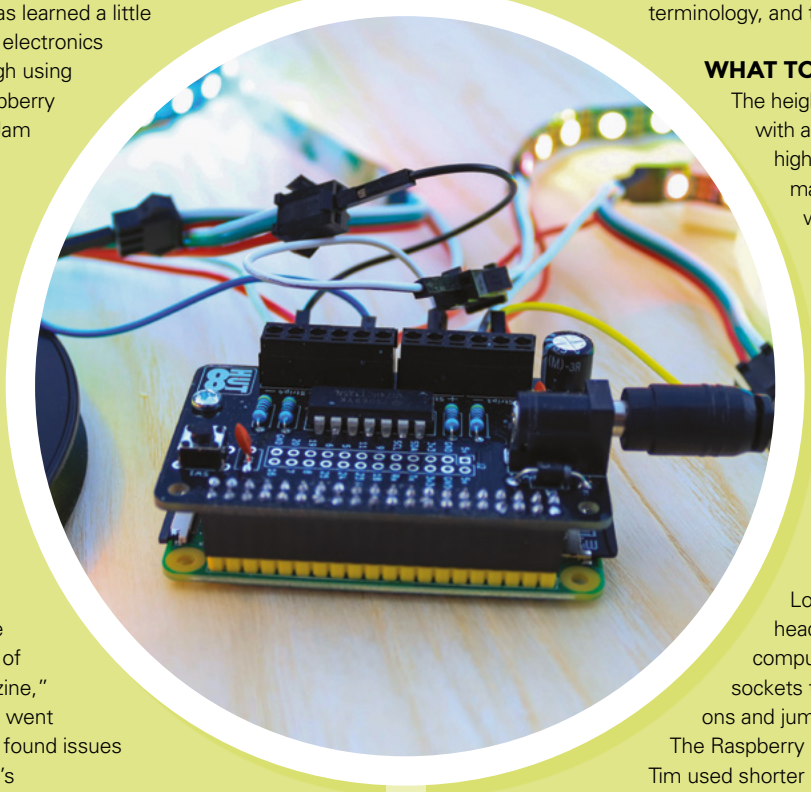
How I Learned PCB DESIGN

How Tim Richardson used an article in HackSpace magazine to learn how to design his own PCBs

WHERE DID IT ALL START?

Tim has learned a little about electronics through using a Raspberry Pi and working on the CamJam EduKits. Now, having a little extra time in the day due to not commuting, he decided to try his hand at PCB design. However, he immediately found himself staring at a cobweb of incomprehensible terminology. Eagle, KiCad, footprints, schemas. He didn't know where to start.

Not giving up, Tim asked a friend he met through Pi Wars, Paul Fretwell (**@drfootleg**), about how he got started. "From a couple of articles in HackSpace magazine," came the response. So, Tim went through his back issues and found issues 17 and 18 with Jo Hinchliffe's (**@concreted0g**) two-part article on using KiCad to produce a simple PCB.



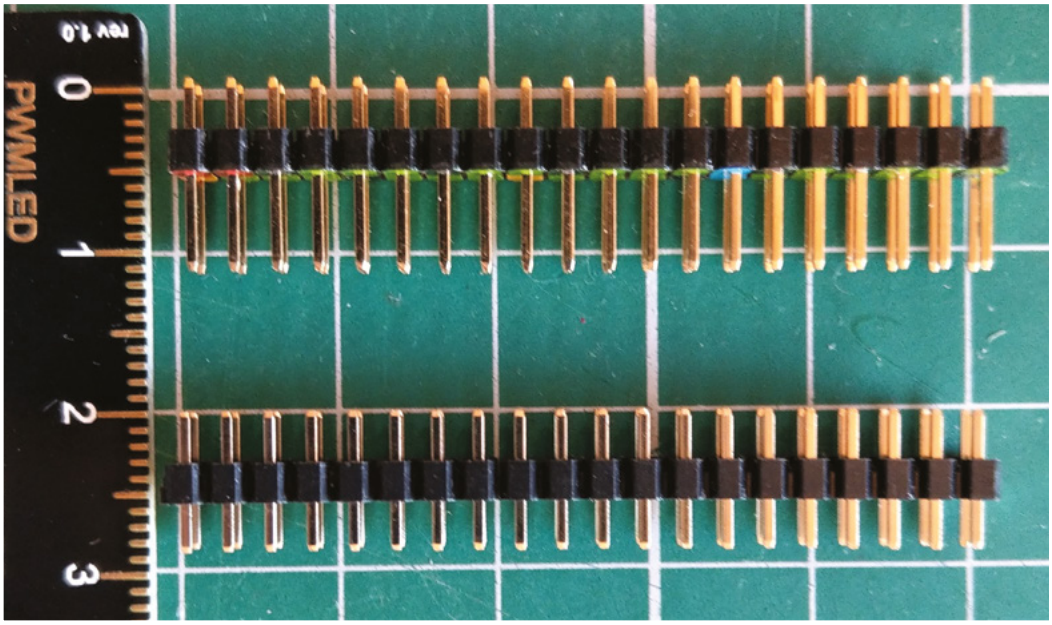
It was exactly what Tim needed to understand the design workflow, terminology, and the basics of how to start.

WHAT TO DESIGN FIRST?

The height of a Raspberry Pi Zero with a micro-HAT (µHAT) was higher than Tim wanted. He'd made a couple of projects where a Raspberry Pi Zero was placed behind a screen, but there was one project he had in mind that needed the combination to be much thinner, so how about mounting a Raspberry Pi Zero and µHAT on the same plane? To solve this, Tim decided to create a new PCB that he called the LowRider.

The major parts for the LowRider were two 'short' headers. Most single-board computers have long pins or sockets to make connecting add-ons and jumper wires more secure. The Raspberry Pi's pins are 7 mm long, but Tim used shorter pins.

The next job is to design a 'schematic diagram'. For the LowRider, this was



Left
Standard (top) versus LowRider headers

connecting two '2×20 2.54mm pitch' headers; in plain English, that is two rows of 20 pins with the pins 2.54mm (0.1") apart. More jargon to learn!

With KiCad's Eeschema open, Tim started by placing 'symbols' on the schematic diagram. Tim found the right header symbols and proceeded to connect pin one of one header to pin one of the other, and so on. Well, that was easy…

The next stage is to take each symbol (component) and assign a 'footprint' which links the symbol to the actual physical shape of the part.

The KiCad 'Assign Footprints' tool lists each component from the schematic diagram and helps you find matching footprints, or you can design your own. For a Raspberry Pi peripheral, Tim wanted to use a footprint that represented the GPIO pins. After more searching, he found a HAT template, so he imported the parts that he wanted.

Then began something that would happen many times in this adventure: starting again!

Tim replaced the generic header with a Raspberry Pi header that had all the pins correctly labelled. This meant having to reconnect each of the pins again. This would not be the last time Tim had to start over.

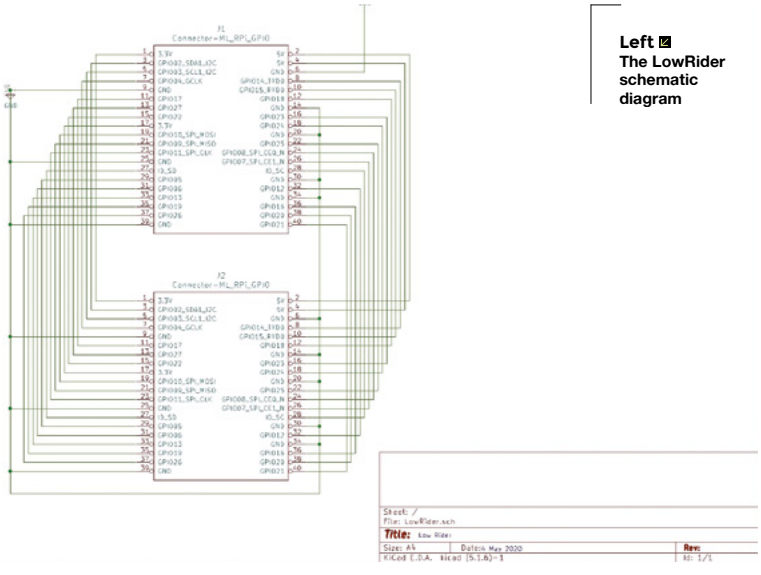
Once all the footprints were assigned, the instructions in Jo's article told Tim to generate the 'netlist'. He wasn't sure what that meant, but did it anyway.

In the next tutorial, we looked at creating a PCB. As instructed, Tim opened KiCad Pcbnew, and was presented with a blank page, as the article said. So, this is what a netlist was for! Pcbnew read the netlist and footprints then appeared, along with a whole 'rat's nest' of white lines between the symbols!

The next part of the game is to eliminate each of the white lines! You do that by drawing 'tracks' between each pin of one →

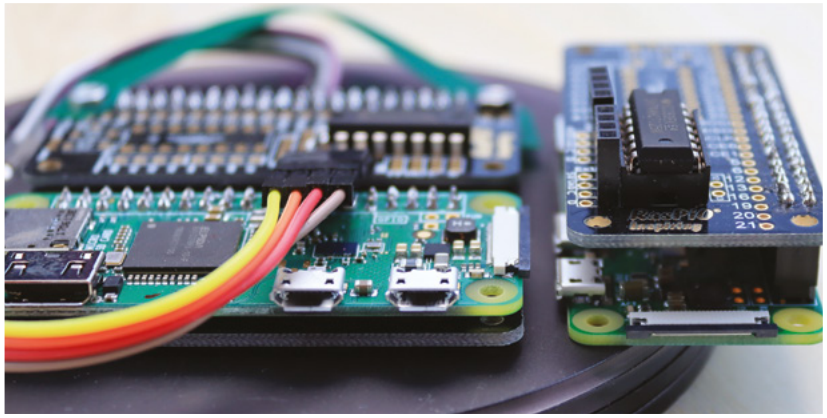
What is a 'copper flood area'?

This is the copper layer that fills both sides of a board with copper from which the tracks will be etched, and forms the 'ground plane'.

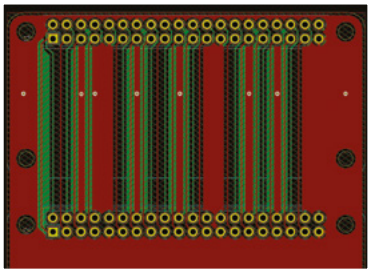


Left
The LowRider schematic diagram

FEATURE



Above
The LowRider PCB (left) compared with the same µHAT combination (right)



Above
The finished LowRider PCB

component and the corresponding pin on the other components.

Tim dutifully drew the tracks, followed by drawing the outline and mounting holes of the board. Next, Jo's article told Tim to create a 'copper flood area'.

Tim began to notice problems: he had connected each ground pin to each corresponding ground pin on the other header. KiCad kept complaining that the 'ground plane' was not actually grounded. Tim tried to fix these one at a time by connecting individual ground pins to the ground plane, but discovered that was wrong; every ground pin needed to be connected to each other via a single 'ground'. Going back to the schematic diagram and fixing all those was almost the answer. Despite all ground pins being connected, there were still areas of the 'ground plane' which were not connected to ground!

It took Tim a bit of head-scratching to realise he just had to move the tracks around to give more space, and move some tracks from the top of the PCB to the bottom and vice versa. With the addition of 'vias', Tim finally completed the board to his satisfaction.

> Despite all ground pins being connected, there were still areas of the 'ground plane' which were not connected to ground!

What is a 'ground plane'?

This is the copper layer that fills one or both sides of a board providing a link between all the ground points of the components.

It was time for manufacture. There are loads of options for this at different price points, with different lead times and qualities. Tim used PCBWay (**pcbway.com**) in China, which had an easy-to-use interface and instructions on preparing KiCad designs for manufacture. Ten boards cost $8, so it seemed worth the risk.

PIXELPI

While waiting for the first PCB to arrive, Tim started designing his second board. For a long time, Tim had wanted to control WS2812 LEDs with a Raspberry Pi. He had looked into the idea quite some time ago and had bought some 'level shifters' in preparation.

Tim gathered his components together, did lots of Googling, and eventually came up with a basic circuit that worked. With some help from the very supportive Raspberry Pi community, Tim also found a Python library which could control all four LED strips at the same time, independently. The only proviso was that it was limited to just four specific GPIO pins, but since the chip that Tim was using had four level shifters, this was perfect.

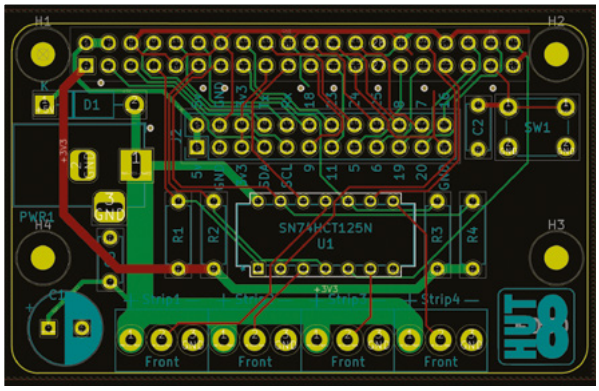
Tim once again fired up KiCad, but had finally learned what a KiCad template was! He started a new KiCad project with a predesigned µHAT template. This had the header already on the schematic diagram and a corresponding footprint for the whole µHAT on the PCB.

Even though Tim had learnt a lot from his first PCB, he still had more to learn. For example, with specific GPIO pins used to control the LEDs, almost any other pin could be used for turning the level shifter channels on and off. He initially chose pins close to the LED control pins, but when it came to routing the tracks, it simply could not be done neatly. Eventually, he chose a bank of GPIO pins that were located near the physical location of the level shifter on the board.

Tim chose to power the Raspberry Pi from the same 5V power supply as the LEDs. Reading the HAT documentation, it recommended using a 'ZVD' diode circuit to protect the Raspberry Pi, but he could not find a design anywhere. Eventually he gave up and 'phoned a friend' who designs PCBs; they recommended using a specific Schottky diode.

What are 'vias'?

Vias make a connection from one side of the board to the other.



Above
The PixelPi PCB

Tim sent the design to a few friends for suggestions. Each comment led to a redesign, but to be honest, Tim was having fun, so he did not mind.

Suggestions ranged from adding a breakout for all pins not being used for controlling the LEDs, to adding a button to turn the Raspberry Pi off. The addition of capacitors to regulate the input voltage and the 'bounce' from the button was something Tim didn't even know about.

The most valuable input was about routing tracks. Some versions of the board had tracks swapping sides multiple times using vias to avoid tracks and components. Not very pretty, and certainly not good practice. Components were moved around the board, pin assignments changed, and tracks grouped together until the board looked 'pretty'. Each iteration was breadboarded to ensure it worked.

Tim ordered this second PCB from PCBWay, but this time paid for the express service and delivery. Within a week he had his new boards!

As often happens, the PCB was not perfect: the silk screen (printing) on the underside was over through-holes and could not be read; a simple cosmetic mistake, fortunately.

But… "I soldered the first PCB and… it didn't work. The Raspberry Pi Zero did not boot," Tim sobbed. He tested the supply with a multimeter, and then worked through the components one at a time. It turns out that he had chosen the wrong footprint for the barrel jack, meaning the ground and mounting pins were swapped. The fix was simple – solder a wire between the two pins.

Those were the only two mistakes. Once he fixed the first board, Tim connected up an LED strip to each screw terminal in turn, and… they worked!

So far, Tim has controlled over 900 WS2812 LEDs using a Raspberry Pi Zero, leaving it on for hours without the board getting warm.

TIM'S ADVICE

Tim admits he has been on a steep learning curve with designing PCBs, but is extremely pleased with the results. He has some advice for others thinking about designing their own:

"If you can make a breadboard, you can design a PCB. Take time to learn the software and get a basic understanding of the terminology.

"Start with something simple – even if you don't get it made.

"Ensure you source parts before starting the design; you don't want to find parts are not available.

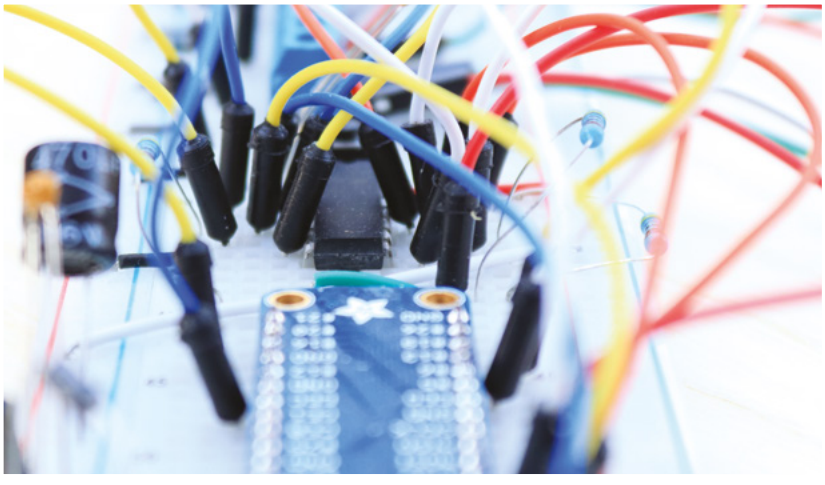
"Be ready to change the design many times.

"Think where each component is located on the board to make routing tracks simpler.

"You are not limited to the default track width; increase it to take high current. This also dissipates heat better.

"Ask someone to look at the design. They often think of things you have not.

"Be prepared to fix mistakes on your board. This may mean scrapping it and ordering a redesigned one." □



Left
The final breadboard used to test the circuits and software

FEATURE

IN THE WORKSHOP: A smartwatch

By **Ben Everard**

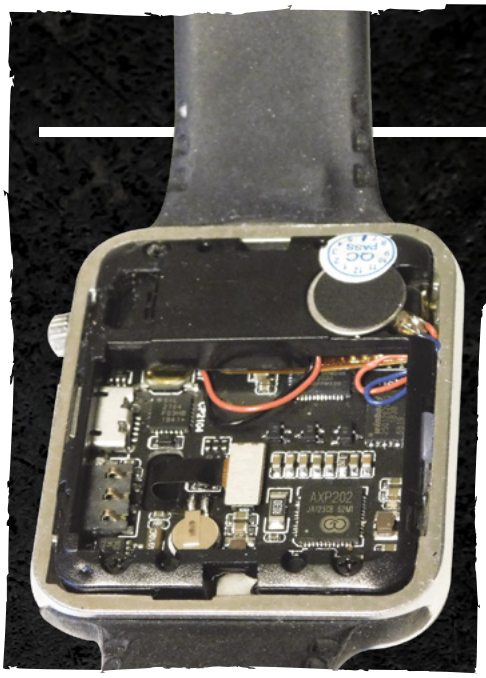
Taking control of my personal computing

Above The watch isn't small, but it is small enough to be unobtrusive

I'm fascinated by the idea of wearable computing. It's a staple of science fiction that the protagonist can access some advanced AI, or control large amounts of hardware with something minuscule strapped to their body (often their wrist). Over the last ten years or so, we've seen an explosion in the amount of computing power we can carry with us (mostly in the form of smartphones), but to me, it all feels a little lacking. Yes, there's a lot you could do with this quantity of computing power, but I can never quite seem to get the benefit from it that I'd like. Smartwatches are – for me – even more disappointing. The ones I've seen are so wrapped up in proprietary services that want to scoop up your data, and locked down, that I've not really felt they've deserved the moniker 'smart'. When I came across the LilyGo T-Watch 2020, which promised an Arduino-compatible smartwatch, I hoped that I'd found something with which I could at least start my journey towards wearable computing that delivered the experience I wanted.

I reviewed the watch in issue 33 and found that it was a capable device. As promised, it was Arduino-compatible with libraries to access all the key hardware. It has a capable processor (courtesy of the ESP32 that powers it), a battery capable of lasting all day, a touchscreen interface, and a button for user input. This was the bare bones I needed to build my own wearable computing.

Regular readers of this section will know that I favour a prototype-driven way of creating things.



Above
You can easily access the electronics inside the watch if you need to

like getting something working quickly because that way I can see how it fits into my life, and find out what I really want the object to do as I use it day-to-day. This takes a certain amount of inspiration from the 'Extreme Programming' methodology of software development, in that it lets me quickly iterate on a working product rather than spend ages trying to create what I think I want, only to find out that that wasn't what I wanted. I've spent the last few weeks doing this.

Starting at the beginning, I wanted a watch that could tell the time and count steps. As someone who sits at a desk all day, I find it important to move around and the best way, for me, of making sure I do this, is to measure my movements. Obviously pedometers have been around for a long time, but I

I wanted one that I could easily extract the data from

wanted one that I could easily extract the data from to process as I saw fit, and that wasn't tied to some proprietary back end.

Secondly, as a Brit, I'm obsessed with the weather. On our small island, the seasons seem to come and go several times in a single day, so having a heads-up about what the weather is likely to do over the coming days is useful.

Thirdly, I wanted a way to interact with other 'smart' objects around my house. The main one of these is my 3D printer. The tiny interface of a watch is not really suitable for holding a full printer interface, but being able to tell the status of the printer from my wrist means that I can always check up on a print while I'm doing other things.

None of these things are particularly complicated – the ESP32 has a real-time clock (RTC) for storing and retrieving the current time, and its WiFi capabilities mean it can grab weather and 3D printer information easily. All it needed was a bit of wrapper code to make it all work.

I've discovered that the two things you really need to pay attention to when coding for a smartwatch are battery life and usability. No one wants a watch that you can't use all day on a single charge, and no one wants to be poking around on the small, fiddly screen. Both of these are things that you can cope with quite easily on the LilyGo T-Watch 2020, but neither of them is a given. You have to carefully allot your battery's resources in order for it to last long enough, and you have to carefully think about how you display data and ask the user to interact.

I'm still experimenting, but so far, I'm pleased with the results. Starting next month, I'll go through how to create your own smartphone firmware to fit in with your particular interests and lifestyle, but if you want to get a sneak peek, you can see my current code on GitHub: **hsmag.cc/mEtS4S**.

At the time of writing, it's pretty rough, but I'm still working on it. I aim to refine it to something that's useful to me, and extensible enough to be useful to you. □

Below
The 380mAh is large by watch standards, and gives you some headroom for imperfect programming

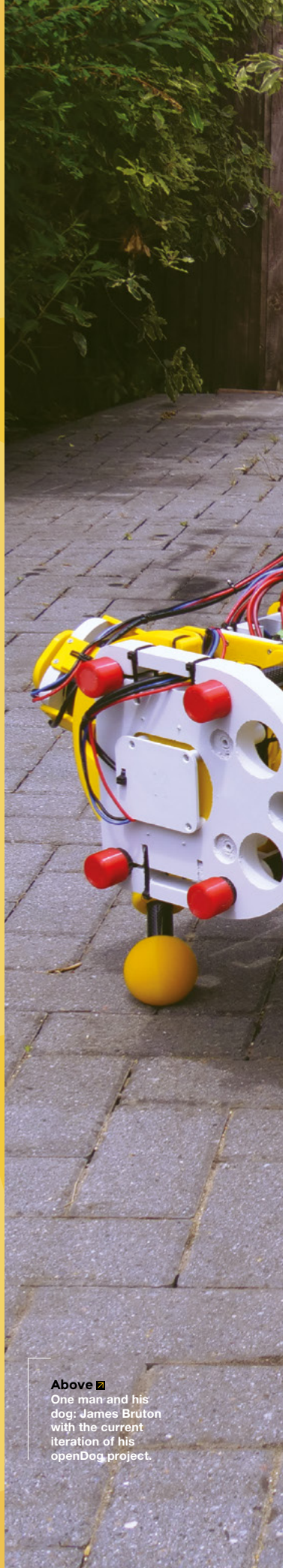
HackSpace magazine meets...

James Bruton

Who needs a multimillion-dollar research program when you've got a YouTube channel?

We've all seen the videos of Boston Dynamics' robot dogs. To some, they're an exciting glimpse of an automated future where difficult, dangerous, and dirty jobs are taken over by machines, freeing humankind for a life of constant leisure. For others, they're an awful harbinger of the day the robots take over and we have to hide from them like rats in sewers. It depends whether you prefer *The Jetsons* or *The Terminator*.

Rather than take either of these approaches, James Bruton saw the multimillion-dollar advanced robotics and thought, 'I can do that'. His project, openDog, is now in its second version, having gone through two years of development and improvement in the aim of recreating Boston Dynamics' walking, balancing quadruped. And best of all, it's completely open-source hardware and documented on YouTube and James's personal site. We caught up with him to find out how it's going.



Above
One man and his dog: James Bruton with the current iteration of his openDog project.

HackSpace Let's start at the beginning: how did you get started with building openDog?

James Bruton I started trying to build a bipedal robot back in 2004, before YouTube. My website, **xrobots.co.uk**, was a blog and I put stuff on there every few days. I built seven or eight machines – this is going back before you could get cheap stuff from China. eBay existed, but it was basically people selling junk out of their attic. And I couldn't find any websites that told you how to build human-sized walking robots, funnily enough.

So I decided to write one and figure out how to do it. That went on for a number of years until YouTube came into being. I ended up with Gonk Droid and Robot X, which are both bipedal walking robots that walked on two legs and did it reasonably OK. Boston Dynamics had come along at that time with their robot dog, so I thought, well, I can make a robot that walks on two legs not too badly, so if you have four legs, that'll be even easier, and I should be able to make something pretty convincing fairly quickly. The only thing I forgot is that the bipedal robots have big square feet and ankle joints, and the robot dogs don't, of course – they walk on points.

I'd already built this exosuit, a frame that walked when I walked [it looks a lot like the power loaders that Sigourney Weaver's character uses in *Aliens*], and it used ball screws and brushless motors, which are immensely powerful. I knew they were powerful enough to move almost anything, so I thought, let's build a really rigid thing with ball screws on all the axes. I planned out a kinematic model and built version 1, but now that the thing is built, it's a two-man lift. It's built out of 6 mm aluminium and has twelve ball screws that are massive lumps of steel, and twelve of these brushless motors. It can locomote, but it's nothing like Boston Dynamics' robot.

I kind of made that walk somehow, by putting load cells in the legs and using the motors to create a virtual spring with the ball screws, but it turned out to be very messy. Without fast processing, it's not very practical.

HS Were you pleased with the result?

JB I took a step back from it. I did two test dogs, which were back-drivable gear-boxes with brushless motors, so the spring is basically the holding power for the motor. You can push it backwards, and it's a very low gear reduction. You can back-drive the motor by manipulating it, at very low reduction, so when you push the joint back it pushes the leg, and that makes it kind of springy. But you can control that, because you can read the motor encoder position and work out where you're driving it away from the position it wants to be from, and use that to calculate the force, and you can either follow the position or resist it. You can basically make this adjustable spring.

> "I suddenly got the confidence then to say I can pretty much make this dog walk"

I did a couple of test dogs that had a motor in each leg, and they just hit the ground while I tried pushing them around with brooms and things like that. So that worked pretty well. But I'd done 20 videos on this massive thing and didn't know what to do with it. It still didn't work very well, and I'd built these two test dogs that looked really good, so then I built two mini dogs.

They were just the test to see if I could make a dog that walks without spending loads of money again. That's sort of hybrid dynamic; it does some stuff with an inertial measurement unit to make sure it keeps its centre of gravity over its feet. And it walks on two legs at a time, using the diagonally opposite legs in a trot gait. It works OK, but the servos are pretty underpowered, so it wasn't capable of anything else other than walking in a straight line.

I suddenly got the confidence then to say I can pretty much make this dog walk. It's almost rigid, very lightweight, and I've got these quasi drivable back-drivable test dogs.

And they can move fast and hit the ground, so I'm going to forget about openDog version 1, and we'll do openDog version 2, which is where I am now.

Last week I published part 6 of version 2. It's a completely new hardware design; it's all 3D-printed, apart from some carbon fibre tubes. It's got twelve brushless motors again, completely different ones this time, the big pancake ones that have lots of torque. They go straight onto the joint with a 5:1 belt reduction, so are easy to drive.

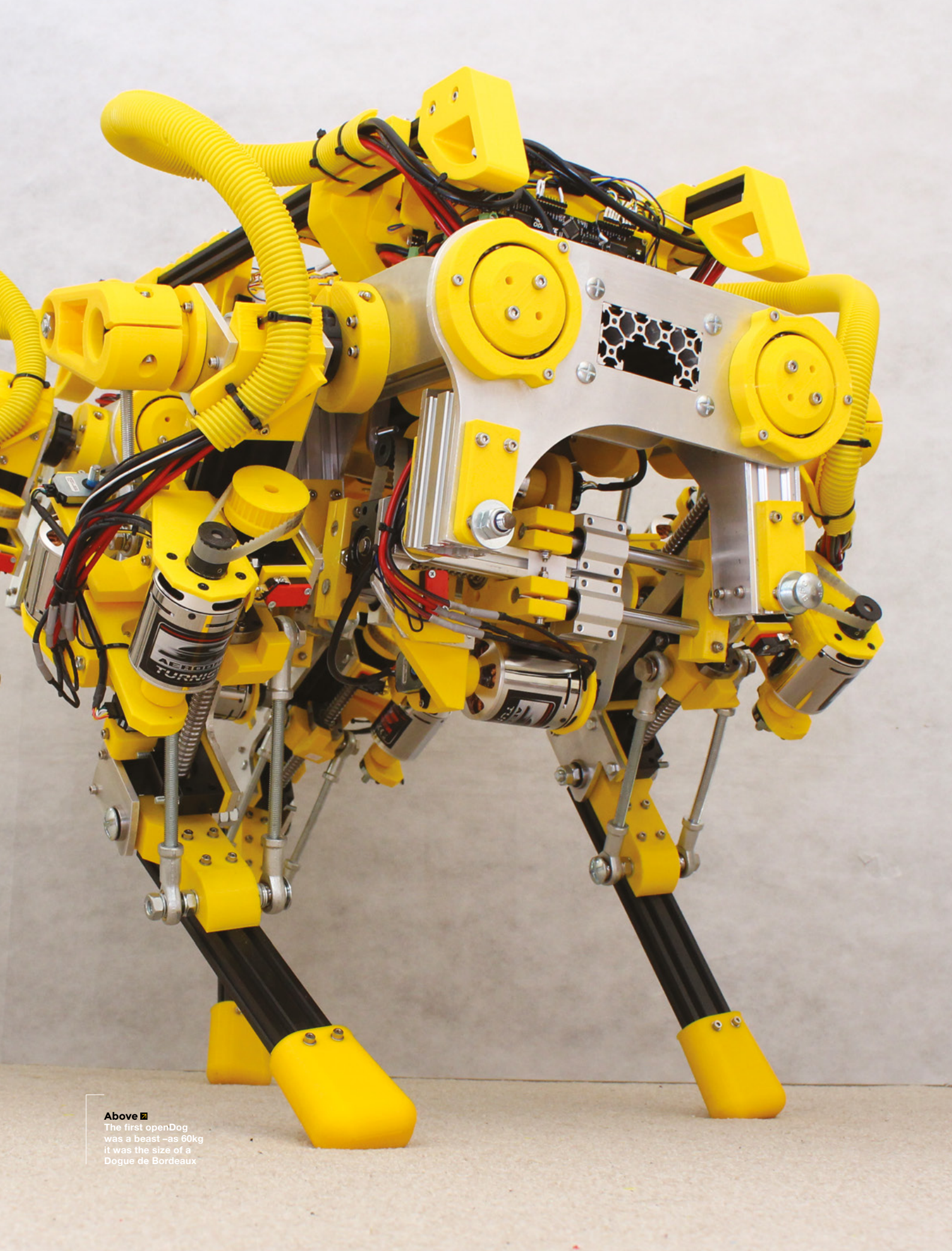
[I] took all the ODrives out of openDog version 1 – those motors are £140 each and there are twelve of them, so it's not cheap to build. The whole thing cost about £2000, with the ODrives and everything.

But it's extremely agile, and it can jump and land on its naturally springy legs and in part 6, which came out last week, it does actually walk. I did that by making the legs move really fast, which keeps it stable – it's a natural spring, like jumping on a pro stick. And then just moving the legs a little bit as I move the joystick to move it around, so that's in openDog v2 part 6. That dog weighs about 18 kg, which is probably a third of the weight of the first one, and it's extremely fast.

HS What did version 1 cost?

JB Probably about the same.

A lot of that is the ODrives which I've recycled though – those are £150, and there are six of them.



Above
The first openDog was a beast –as 60kg it was the size of a Dogue de Bordeaux

Right ◆
"Paramount contacted me and asked me if I could build them a life-size Transformer – and then we just did it!"

HS It does look, when you start cutting out sheets of 6mil aluminium, it looks more like 'Open Rhino' than openDog. It's a big unit.

JB I was just going from what I'd done before. All the bipedal robots were rigid with ball screws.

I just used a bit of inertial measurement data to sway the hips and ankles, and it kind of worked. Development has come on a lot since then.

HS Is that because a biped will have all its weight on one leg, rather than bouncing along?

JB I don't know – they [the bipedal robots James made] looked really dynamic, they took quick steps. When the Gonk droid walks, it looks like a child or something.

I guess it was lighter. If you get lighter and smaller then the problem's not so bad. You can make a small dog with radio-controlled servos, and that's fine because it doesn't need to absorb such massive inertia. As soon as you start scaling up, things spiral bigger and bigger. If you've made something twice as big, it's bigger in three dimensions, so it's eight times the volume and eight times heavier. So every time you double it, then double it again is 64 times, it's a cube function.

So my little servo dog, there are loads of those on YouTube and they look fantastic, but if you scale them up, then there's suddenly so much inertia pushing its rigid leg down on the ground. You've got to have that spring in there.

HS I guess these are all considerations that you'd never thought of when you started.

In openDog version 1, video 1, you start out saying, "I'm going to build something like the Boston Dynamics dog. Obviously it's not going to be as good as that, because I'm not as good as Boston Dynamics."

Do you think that's true now? Have you caught up with them yet?

JB Oh no. Look at the millions of dollars of DARPA and Google funding they've had. But I'm pretty happy with openDog version 2 for something that I've just made.

There are a lot of people complaining in YouTube comments that it's all down to software.

But actually the inherent hardware is what's made it work more easily in the end. This back-drivable, low-reduction joints, and massively powerful brushless motors... the software that makes openDog 2 work will not make openDog 1 work, because it's not really anything – it's just moving joints to fixed positions at fixed times. But because of that spring that we get out of the motor holding power is what makes it work inherently. It's what makes it dynamic where it moves sideways, when it wobbles a bit and brings itself back – it's totally down to the mechanical build.

> "If you've made something twice as big, it's bigger in three dimensions, so it's eight times the volume and eight times heavier"

HS One thing that struck me about the version 1 videos – that you started it in software and you were writing code before you'd even written any hardware. Did you take a different approach for version 2?

JB Not really, no. All I did there was the kinematic model, which was me making sure I could move the joints in Cartesian coordinates and work out all the joint axes. But that's as far forward as I'd thought with that; I'd never thought about it moving.

The mini dogs and version 2 are using exactly the same kinematic model that I've copied and pasted every time, with minor modifications. I've learned a lot doing it, because I'd never even done a kinematic model for a robot before.

HS OpenDog is open-source hardware. Do you get much constructive feedback from people who have downloaded the code?

JB There are very few people contributing. I mean, there are people who want to make a pull request because the indentation isn't in line in the code and things like that...

That said, I have a couple of patrons who subscribe on Patreon, and one of them is building it, and there's another who's implemented a machine learning setup – he's making a robot dog walk better by giving it rewards. So you can have, it's a little servo dog, but he's got an optical mouse tracker underneath that can work out how fast it's walking.

That's something I'll be working with him on probably to make my dog walk better. That's something for the future.

HS Let's get on to some other big things you've made. What else have you been working on?

JB I've done a few projects with Colin [Furze]. One was an Amazon Alexa-activated bed shaker, and we also did two eBay promos (everything has to be bought off eBay). Colin built the giant TIE fighter, and I built BB-9E, which was the bad BB-8. We did a promo with the kids who find the TIE fighter in the middle of a field. That was with eBay and Disney, for the launch of *Star Wars 8*. We did another one which was Marvel *Avengers: Infinity War*, which was when we built the Hulkbuster.

HS The rolling ball version of R2-D2? It's obviously CGI...

JB For the movie, they had a mixture of things. They had these ones with big stabiliser rigs on them, with a central pivot and an arm that held the head, so →

Right ◆
James' driveable BB-8 droid, from the latest Star Wars films

those were practical effects, but they CGI'd out the rig. They had some that were pushed by a person; one that just sat on the ground and its head bobbed around and didn't move anywhere.

When Harrison Ford broke his leg and everyone had to go home, that's when they built the free-roaming one that they use for red carpet events. And, of course, that's the one that everyone wanted to replicate, and that's how mine works: there's no supporting rig on it – it drives around by itself.

HS How does that work? Is the head held on with magnets?

JB It is, but what's going in inside is obviously a lot more complicated. It's based on a single axis that drives in one direction. On that, there's a pendulum that swings sideways, so it leans to steer. And on that pendulum is a flywheel so it can spin on the spot. And all those things have the same centre of rotation. Also with the same centre of rotation is the head control arm, which then moves in another three axes so that it can turn and move the head all around. It's all controlled with an inertial measurement unit and PID controllers. Because otherwise, it will just wobble around like crazy.

So regulating the speed and the pendulum to hold the head forward at the perfect angle as you drive, all of that motion control has to be integrated dynamically – it varies the motor speed and positions 50 times a second, or more. So it's a bit of a challenge.

HS How big a leap is building something like that, with all those mathematical considerations, from starting out with flashing all LEDs on and off, the 'Hello World' of robotics?

JB I relied heavily on Arduino libraries. The inertial measurement unit has a library that just gives you an angle. There's a library for the PID controller to control the motor speed. I used two servos to make a V-shaped lever to move the head around, and there's a library for that. Remote-control, I used Bluetooth on a serial port, but there are other ways. It's just a case of understanding how they all tie together, and what you're trying to achieve.

There's the mechanical build of course, which has to have motors with the right torque and speed to be able to compensate quick enough to hold it stable.

It shouldn't be your first project! If you were going to try building one of these, then your first project should be one of those two-wheeled balancing robots, because that uses a lot of the same principles. BB-8 has a heavy weight in the bottom so it stays upright when it's switched off, but when you drive you still have to keep that mass moving dynamically so it doesn't wobble around like a Weeble

HS And the Iron Man suit that you made, do you still have that?

JB I do, it's all packed away in boxes. I don't know if I'll ever wear it again. I think probably, looking back on it now, that was before I had a 3D printer when I started that.

I started trying to build an Iron Man suit in 2008, and it was all clay sculpts and moulds and casts. I used templates made from foam, and then I made moulds from foam and made fibreglass and resin-cast parts. I only had a 3D printer right at the end, for some of the catches and bits and pieces. Other than that, it's all handmade.

It took four years. That's when I started seriously making YouTube videos. My top-viewed video on YouTube has had around 67 million views, and that's four years compressed into four minutes of me building the Iron Man suit. Some of those early clips are the first videos that I started making; the resolution is 4:3 on an old DV camcorder. And it still gets, maybe, half a million views a month today. There's a bit of a legacy there with YouTube.

HS And how about right now?

JB Most recently, I've been working on a robot that can do mapping and navigation. It's got a Raspberry Pi on top, and an Arduino. It's running ROS, the Robot Operating System. It's got lidar on top, and it can do mapping and navigation. I've got something on speech recognition coming out at the end of August, but I haven't done anything else for a while, because of a huge, big, secret project that's taken all my time. It's a big build. And I can't tell you about it.

HS Using somebody else's operating system, ready-built as ROS is, does that feel like a step down after doing everything from scratch, as you did with openDog?

JB You say that, but then I've used Arduino libraries and so on, and the ODrives to control the brushless motors, they've got their own processors on them to do all the positioning for you. I'm a big believer in living on example code anyway.

ROS itself actually isn't that easy, although it's all built for you. It's quite hard to get your head around, so it feels like quite an achievement getting it to work.

> There was no buying an Adafruit module and downloading a library, sticking some variables in, and it working back then

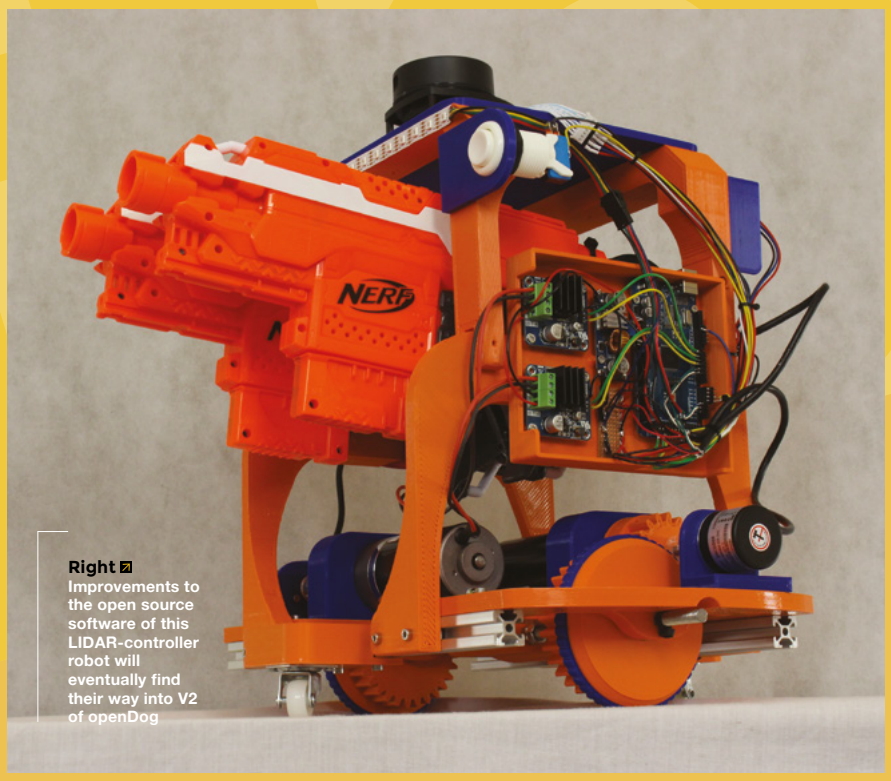
I mean if you think about Arduino and Raspberry Pi, and everything that's around these days, and how it used to be in the old days. So, I did a degree in electronic engineering in the 1990s, before open-source infrastructure existed. We basically did programmable logic controls and all sorts of stuff – that really was doing it yourself. There was no buying an Adafruit module and downloading a library, sticking some variables in, and it working back then. That really has made things a lot more accessible.

All my projects rely on third-party Arduino libraries – you don't have to get into the nitty-gritty, low-level stuff. Things like the ODrive to control brushless motors – you could just attach an encoder to a brushless motor, and there's an Arduino library to drive that motor to a position. Trying to control brushless motors from scratch is pretty tricky. Shared libraries is the reason openDog exists.

HS So I guess that's why you were so keen to release your CNC files, 3D-printing files, and all the other code yourself?

JB I tend to find that drives traffic. Ultimately, I'm a content creator, and I find that putting that out as open-source drives subscribers, it drives people to become YouTube channel members. Even if they don't want the code, the CAD drives that sort of community. People like to support open-source.

But there's more to it than that. LulzBot, which supports my channel, is a completely open-source company. They did nearly go out of business recently, and then they got bought and they're still going. But if they had gone out of business, it would have been OK, because the firmware is open-source. You can keep maintaining your printer, keep using another slicer, or whatever you want to do. So, it's quite useful when you're making a big purchase like a printer – if it's open-source, it's better than if it's proprietary and the slicer's based in the cloud, then the company goes out of business, and your expensive device is essentially bricked.



Right Improvements to the open source software of this LIDAR-controller robot will eventually find their way into V2 of openDog

FEATURE

IMPROVISER'S TOOLBOX

PARACORD

Bale out your projects

Mayank Sharma

@geekybodhi

Mayank is a Padawan maker with an irrational fear of drills. He likes to replicate electronic builds, and gets a kick out of hacking everyday objects creatively.

Paracord, the word and the tool, comes from its use by American paratroopers in the Second World War. On landing, the soldiers would salvage the lines from their parachutes and then later use them for all kinds of emergency fixes in the field.

Owing to its usability and durability, paracord soon became a standard military supply and part of the standard deployment package. Over time, more people, even those outside of the military, discovered paracord could be used for all kinds of outdoor projects.

Paracord owes its origins to the invention of nylon in 1935 by Wallace Carothers. Nylon became the durable replacement for the Japanese silk for making the chords of the parachutes. A paracord is essentially just a nylon cord that's made up of a core of threads placed inside a woven nylon jacket.

> "Despite their exotic history, paracords are fairly accessible, and you can find them in your favourite crafts store"

The US military established standards for the production of paracord. The most common type is made up of seven inner nylon strands, that are in turn made up of three smaller strands each, packed inside a nylon outer shell. This arrangement of cords together has a breaking strength of 550 pounds.

Of course, that isn't the only kind of paracord available on the market, and they now come in all kinds of colours, patterns, and thickness. Their thickness determines their type, though sometimes it's also their breaking load, measured in pounds, that is used as an identifier. For instance, the cord described earlier is commonly referred to as Paracord Type III, or a 550 Paracord, since it has a breaking load of 550 pounds.

Due to its popularity, the cord is available in many variants of varying quality. Type II is also widely used for more elegant knotwork, and Type I is often used as an accessory cord. The term 'Mil-Spec Paracord' is used for paracord that is made to strict US military specifications, and is quite popular for its use in all kinds of extreme outdoor sports.

Irrespective of how they are used, paracords are often braided or weaved. Knots play an important function when working with paracords as well. In fact, the knots are often used to stylise the outcome. Many makers also join paracords by melting their ends together. The nylon cores fuse together easily when heated to create joints that appear seamless and are quite strong.

Despite their exotic history, paracords are fairly accessible, and you can find them in your favourite crafts store. If you haven't worked with paracord before, here are some projects to get you started.

CELL PHONE CASE

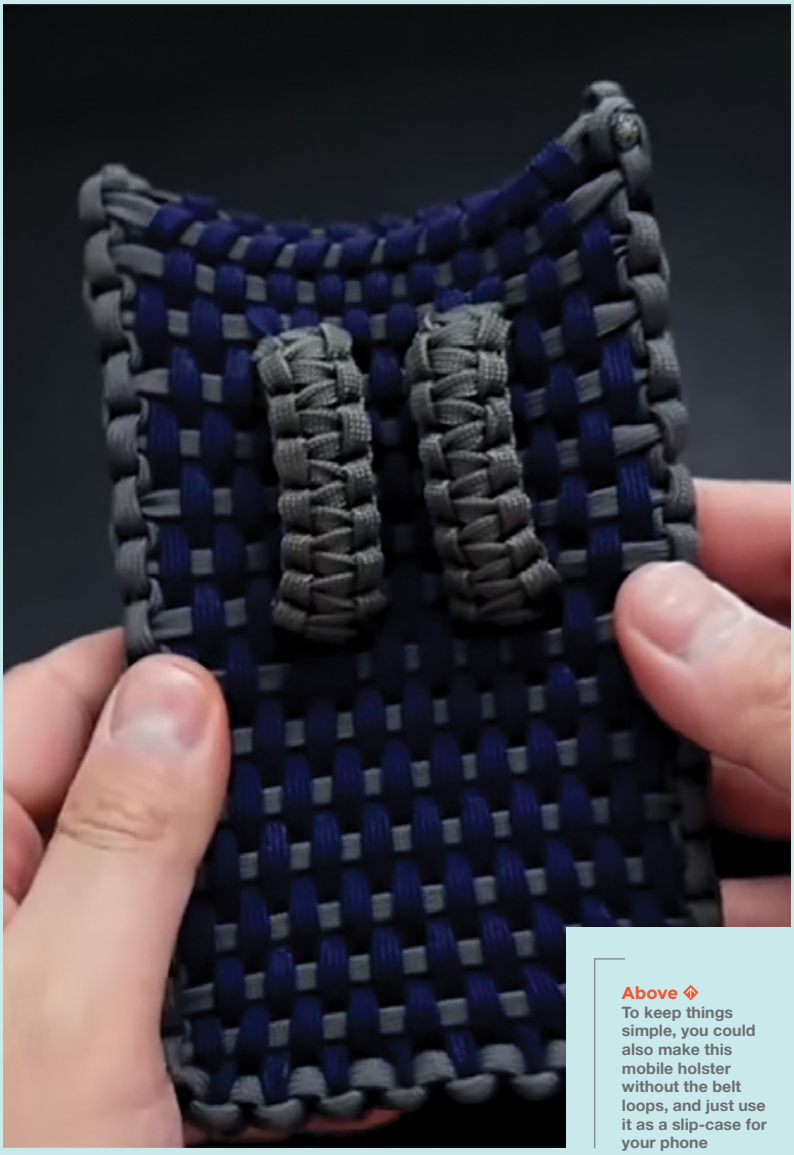
Project Maker
Tim

Project Link
hsmag.cc/RrYA13

Tim loves working with paracord, and although he has tons of projects on his blog, we like this one in particular. He's used different lengths of grey and blue coloured paracords – the exact amount of paracord you will need will vary depending on the size of your phone. He first makes a cardboard template of his phone, which he then uses for measurements. Tim knits the outer perimeter of the case by using a length of paracord and tying it up in cobra knots. Don't worry if you don't know how to tie the knot because Tim has done a great job of explaining and demoing it in the video. The rest of the case is created by flattened lengths of gutted

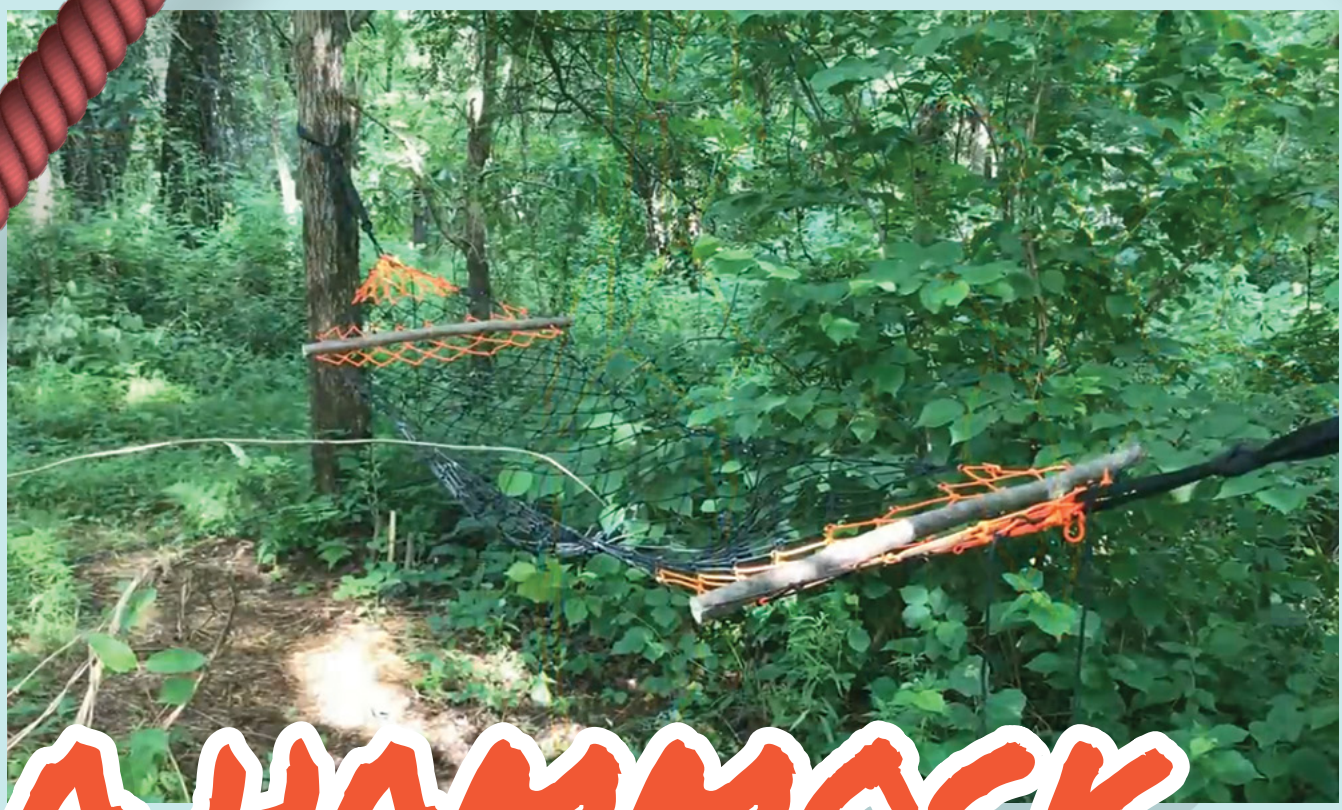
> "You'll have to make sure the cords are nice and flat as you weave them"

grey and blue paracords. You could use ungutted paracords as well, but they'll just make the case a lot bulkier. Attach the grey flattened cord to the cobra knots on both sides and finish by weaving the other flattened cord over and under the grey one. It's a time-consuming process, since you'll have to make sure the cords are nice and flat as you weave them into the case. Finish the case by adding a couple of belt loops. Make sure you watch Tim's video to better understand the process. →



Above
To keep things simple, you could also make this mobile holster without the belt loops, and just use it as a slip-case for your phone

FEATURE



A HAMMOCK

Project Maker
DAVID CANTERBURY

Project Link
hsmag.cc/9WfVGI

Want to take your paracord knitting skills to the next level? Follow survival instructor, and former television presenter, David Canterbury as he uses a few hundred feet of paracord to weave himself a hammock. The good thing about his hammock is that it's made with a simple netting method using any type of strong cordage.

David starts by forming a cord line at the top where he ties a simple girth hitch knot over the top line and then pulls two lines through it. It's important to understand how big the hammock needs to be as that'll dictate how much paracord you'll need. His has a width of up to his neck, which is about five feet. He's using four-inch meshes throughout the hammock, which dictates how many down lines he has to put in his cross-line at the top.

For the meshes, he makes a two-inch mesh at the top with a four-inch mesh for the rest of the hammock. He uses his fingers as a gauge and creates the mesh using reef knots. This gives the mesh better uniformity. The process is time-consuming, but David explains it in great detail and also shares useful tips that'll help you during the build process.

"THE GOOD THING ABOUT HIS HAMMOCK IS THAT IT'S MADE WITH A SIMPLE NETTING METHOD USING ANY TYPE OF STRONG CORDAGE"

SURVIVAL BRACELET

Tom Hardy's bracelet in *Mad Max* is quite a rage and can be easily made with some paracord.

Project Maker
SHAWN MATTHEWS

Project Link
hsmag.cc/yUncTC

Shawn takes an eleven-foot length of paracord and folds it to find the centre point. He then takes the centre point and wraps it around his wrist to gauge the diameter of the bracelet and marks the endpoint. The remaining length of paracord is then tied using cobra knots all the way through the bracelet until you are left with a tiny loop at the end. To secure the bracelet, Shawn uses a couple of paracord fids to create stoppers at the ends of the paracord bracelet. He then takes the fids off, determines where he needs to place the stopper by trying on the bracelet, ties the stopper knot, and singes the ends to complete the adjustable bracelet. Watch Shawn's video to better visualise the process.

Above
If this bracelet doesn't catch your fancy, the BoredParacord channel has got lots of others

TANGLE-FREE HEADPHONES

If you haven't yet switched to wireless headphones, you can use some paracord to make your annoying corded ones, that always get tangled, irritate you a little less. You'll first have to cut the headphone wire, and then string the pieces through a gutted length of paracord. Once you can't push the wires through any further, Matt suggests you instead scrunch up the paracord and then extend it over the wire. You'll then have to strip and connect the wires back together, which Matt has explained in detail in another video tutorial that he's linked to in this one. Make sure you thread some heat shrink tubing, which will come in handy later. Then connect the wires as instructed, and check the headphones, before insulating the connections with some electrical tape. Wrap up the process by covering the joints with the heat shrink tubing that had been piped earlier, and your tangle-free headphones are ready.

Project Maker
MATT CARL

Project Link
hsmag.cc/vEQbFd

Above
Matt suggests adding some hot glue over the joints for some structural integrity

HackSpace
TECHNOLOGY IN YOUR HANDS

FORGE

HACK | MAKE | BUILD | CREATE

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

CircuitPython on ESP32-S2

A new microcontroller joins the CircuitPython family

Ben Everard

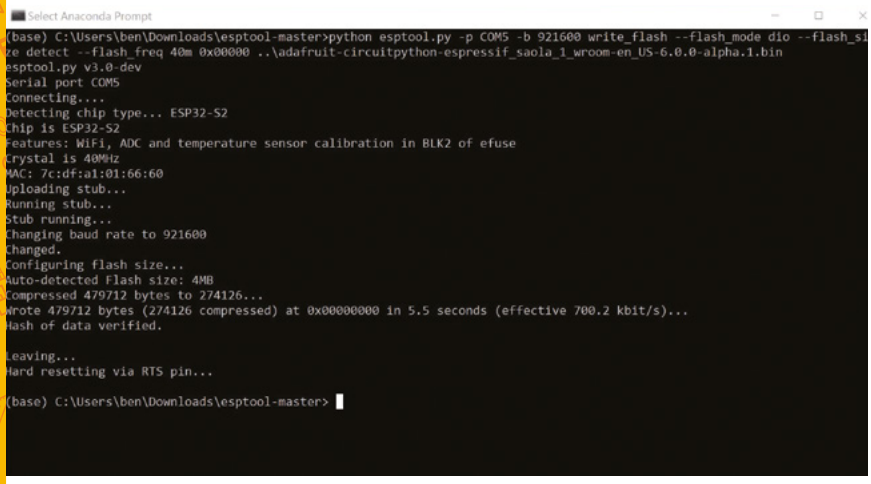
@ben_everard

Ben loves cutting stuff, any stuff. There's no longer a shelf to store these tools on (it's now two shelves), and the door's in danger.

The ESP32 microcontroller has been around for quite a while, and offers great value for money. For around £7 you can get a powerful dual-core microcontroller with on-board Bluetooth and WiFi. It's compatible with Arduino and MicroPython, but not CircuitPython. The reason for this is that it lacks a native USB port. While there is a USB port that you can use to send data to the device, it's a little clunky to work with. You need to use a command line tool to upload files (this is built into the Arduino IDE), whereas with CircuitPython, devices are mounted as removable storage you can drag and drop files to.

Fortunately, there's a new version of the ESP32: the ESP32-S2. This is a stripped-down version of the microcontroller – it has only a single processor core, for example. However, it does have a native USB port, and this means that it's possible to run CircuitPython on it and mount the drive as removable storage. We decided to try it out to see what it's like. This is more complex than other CircuitPython devices, and we wouldn't recommend it at the moment unless you're familiar with setting up microcontroller development environments. Hopefully, this will change as more CircuitPython-specific hardware comes on the market.

Below
Make sure you're using the latest dev version of esptool when flashing the device



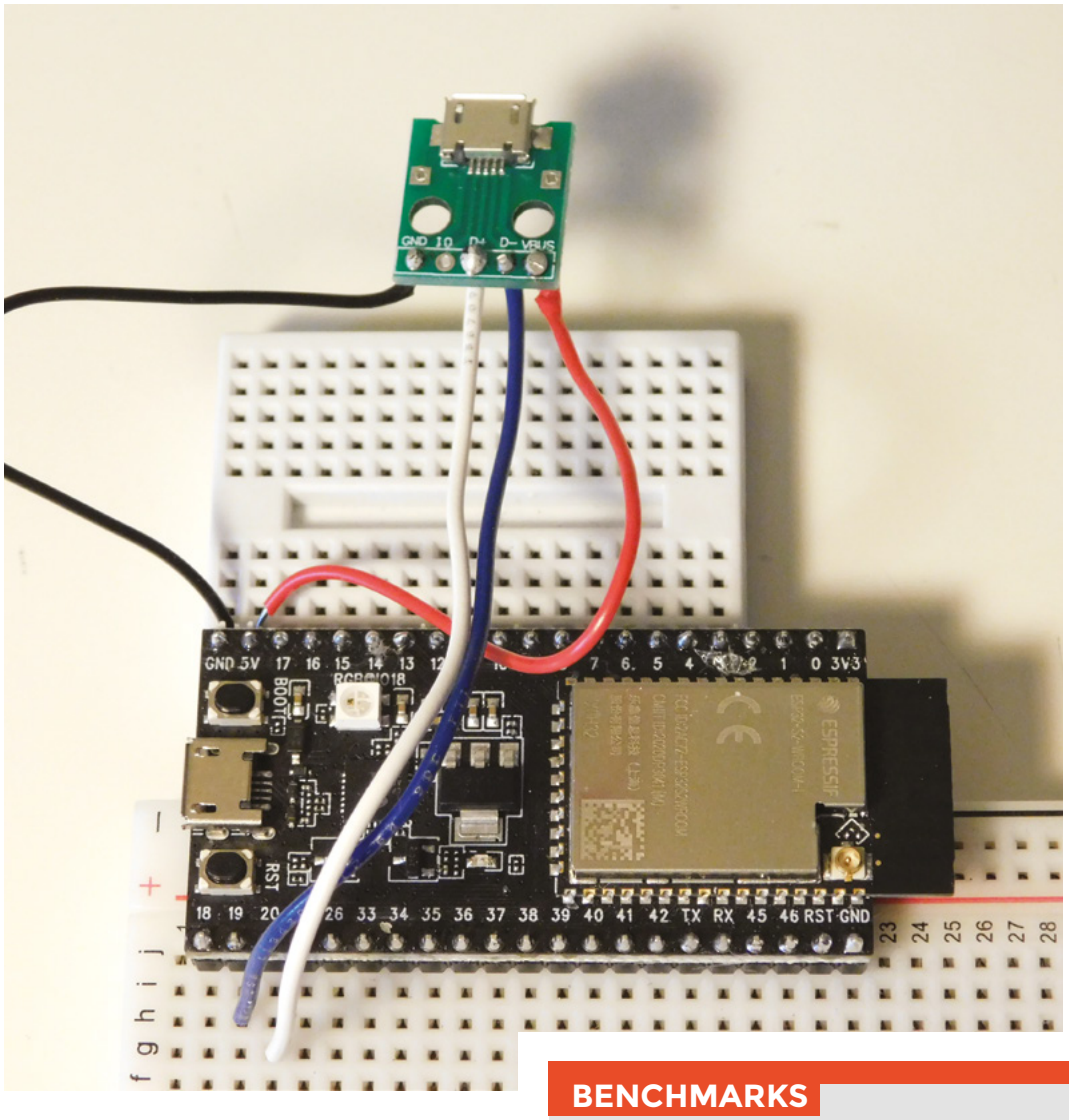
At the time of writing, there are only a few development boards built on this microcontroller. The Saola-1 comes with either a WROVER or WROOM module, and Unexpected Maker is selling a pre-release version of the FeatherS2. We tested out a Saola-1 with WROOM.

This module does have a USB port, but it's set up in the same way as previous ESP32 boards – in other words, incompatible with CircuitPython. The new native USB connections are exposed on the pin headers, so the first thing we need to do is attach a USB port to them. You can get USB breakout cables, but we soldered up a micro USB port to expose the connections we needed. In a USB cable, you'll find a ground, 5V, D-, and D+. D- and D+ both carry the same signal, but inverted so D- is always the opposite of D+. To wire up the board, simply connect 5V to 5V, GND to GND, D- to pin 19, and D+ to pin 20.

> "There's no support for either Bluetooth or WiFi, **though that may change**"

To make matters a little more complex, the actual CircuitPython binary that you need to flash on the board needs to be loaded over the original USB port, not our new addition. To do this, you need the latest version of esptool – you can download this from GitHub at **github.com/espressif/esptool**. If installing from a different source, ensure that you have version 3.0-dev or higher to include support for the ESP32-S2. You'll also need Python installed for this to work (see GitHub README for details – you'll need the Development Mode installation). With that installed, you can download the CircuitPython binary from **circuitpython.org**, then flash it to the device using esptool with a command something like the following:

```
python esptool.py -p <com port> -b 921600
write_flash --flash_mode dio --flash_size detect
--flash_freq 40m 0x00000 ..\adafruit-circuitpython-
espressif_saola_1_wroom-en_US-6.0.0-alpha.1.bin
```



Left
You have to add your own extra USB port if using one of the Saola-1 boards

You'll need to change `<com port>` to the relevant serial port, and you may need to change the path or name of the downloaded CircuitPython binary. If this runs without errors, you should now have CircuitPython running on the device. Switch the USB cable to the new USB port, and you should be able to access the file system and REPL as you would with any other CircuitPython device.

Perhaps the major difference between the ESP32-S2 and other CircuitPython boards is that pins on the ESP32-S2 are named IO1, IO2, etc., while on other boards we've used they're named D1, D2, etc.

If you're looking to use the connectivity features of the ESP32-S2, then you'll be disappointed. At the time of writing, there's no support for either Bluetooth or WiFi, though that may change in the coming months (this is still an alpha release after all). However, you do have access to lots of CircuitPython goodies.

BENCHMARKS

The ESP32-S2 is a powerful microcontroller, so we decided to compare it against two other common, powerful microcontrollers: the SAMD51 (in the form of an ItsyBitsy M4 Express) and an nRF52840 (in the form of an Adafruit CLUE). You can see the full results below, but broadly speaking, this processor is somewhere between the two in terms of performance (in all these benchmarks, lower is better).

It's worth taking this with a pinch of salt, as this is the very first version for the ESP32-S2 chip while the others have had a few years' worth of improvements. We may well see performance increase on this platform in the coming years.

Benchmark	ESP32-S2	nRF52840	SAMD51
GPIO on/off	3.743	7.34802	2.237
integer sum	5.41299	8.534	3.05
integer multi	12.154	10.438	4.023
float sum	2.96001	6.03198	1.671
float multi	2.99501	6.021	1.647
float divide	3.106	6.05801	1.664

Make a rag rug from cast-off clothes

Create useful homewares from clothes you've discarded

Rosie Hattersley

@RosieHattersley

Rosie Hattersley writes tech, craft, and life hacks and tweets @RosieHattersley.

If you've spent the last few months reorganising things and chucking out items you no longer need, chances are you've been waiting for charity shops to signal that they're again able to accept donations.

Right now, they're overwhelmed. Rather than add to the deluge of cast-offs, you could turn unwanted clothes and bedding into something useful instead. It's a slow-paced, mindful process that you can do in front of the TV, or while listening to a podcast.

Although an actual rug is usually what's made when rag rugging, you could just as easily use this upcycling technique to decorate the edge of picture frames, a lampshade, or even make a piece of eco-art to create an instant shabby chic effect. Here, we'll look at three different methods of rag rugging, starting with the classic shaggy rag rug. The materials and preparation are similar for all three options, and you can mix and match techniques.

Start by assembling the clothes and other fabrics you want to use, ideally choosing colours that complement each other. For a doormat-sized project, you'll need about five pieces. The fabrics can be a range of types and thicknesses. For example, the dark blue edges of our project piece come from brushed cotton pyjamas, while the bright blue is a much lighter, wrinkle-prone nylon. T-shirts, shirts, jeans, and cotton sheets and duvet covers are also ideal for turning into rag rug pieces.

Right
Assemble tools to help you poke fabric through a hessian backing. Round up old T-shirts and prepare to attack them

Far Right
A latch hook snags pieces of fabric, then holds them in place while it draws them up

TOOL TALK

If you're going to make a largish project such as a doormat-sized rug like ours, or a decorative rug for a room, you might want to buy to dedicated latch hook tool or spring tool. Both have prominent hooks, plus a means of trapping your fabric on it so that it doesn't fall off as you pull it through the hessian. They cost less than £5 from craft or sewing stores.

Another option is to use a crochet hook. These are cheaper and more widely available, but don't have a means of securing your fabric to them. Finally, you could use a small pair of needlenose pliers. You'll make larger holes in your hessian, so

they're best for projects with thicker fabrics and for loopy rag rugging because there's less chance of the fabric being pulled out.

PREPARING YOUR FABRIC

For shaggy rag rugging – which looks like a sea of fabric fingers – you need to cut your source fabric into oblongs 1.5 cm to 2 cm wide, and around 7 cm long. Cut thicker fabric, such as denim or heavy cottons, narrower than stretchy T-shirt material. You can't use seams, so cut just inside the seamed edges of arms, side seams, and necklines.

Fold your fabric into 1.5 cm or 2 cm pleats, and cut along the length of several layers at once. For use as shaggy rag rug pieces, cut the strips into equal-sized lengths approximately 7 cm long.

> **Cut thicker fabric, such as denim or heavy cottons,** narrower than stretchy T-shirt material

For the loopy rag rugging technique, you need to create long strips from your source fabric. For maximum cutting and rug-making efficiency, lay your fabric flat and snip along the longest side to make strips. The Ragged Life Blog (**raggedlifeblog.com**) has lots of fabric prepping tips.

For the third technique, braided rag rugging, you also need long strips. As an alternative, you could cut up T-shirts and other sizeable items into balls of 'yarn' by cutting a series of diagonals into either →

Top Left
For loopy rag rugging, you need to cut up clothes into long strips

Left Middle
Roll up your fabric strips into balls of fabric

Bottom Left
Poke your pliers, or latch hook tool through the hessian and grab the fabric below it

Top Right
Shaggy rag rugging consists of lines of individual pieces of fabric

YOU'LL NEED

- **Selection of old T-shirts, shirts, bedding, or other fabric**
- **Hessian fabric, cut to size**
- **Sharp scissors**
- **Wooden latch hook, crochet hook, or needlenose pliers**
- **Strong needle and thread**
- **Masking tape for the braided fabric technique**

Above
Our rug progressing as we snip up various unwanted clothes

side of the fabric, and then cutting the fabric as one continuous strip. Instructions for using the braided rag rugging technique are on the next page.

SHAGGY RAG RUGGING

Start at one corner of the hessian and a few squares in. Insert your chosen tool through the front of the hessian so that the hook (or pliers head) protrudes. With your other hand, take your first piece of cut-up fabric and place it, pattern-side upwards, on the hook. Place or hold it underneath the hessian. If using a latch hook tool, bring the small piece of metal above the hook down so that it latches into the hook, snaring the fabric.

Draw the fabric back up through the hessian, and wiggle it so that approximately half the fabric strip is on the top side of the hessian. Press the latch hook arm again to release the fabric. Now, push the hooking tool down through a square two or three along from the first one, snare the other end of the fabric, and draw it up to the top of the hessian. If you look on the underside of the hessian, the fabric strip should be snug against it, but shouldn't pull the hessian out of place. Repeat the process of hooking oblongs of fabric to create a line of rag rugging.

> **To begin loopy rag rugging, take a long strip of fabric** and pull it through from the underside of the hessian

QUICK TIP

It's sensible to hem the edges of the hessian (or other open-weave fabric) you're using as the backing for your rug, so the strands don't get in the way while you attach the pieces of fabric to it.

CHECK AS YOU GO

At the end of each row, turn the hessian over to check there are no huge gaps. This can happen easily as the fabric disguises them as you work. Add pieces of rag rugging to fill any gaps. Floaty fabrics, such as light nylon, are the most likely to be hiding big gaps. Gaps can also appear if you leave several rows between loopy rag rug rows.

STRETCHING EXERCISE

Stretchy fabrics, such as cut-up old T-shirts, are an obvious upcycling choice, as many of us have lots of them. Stretch fabric tends to twist as you work, leaving a narrow, quite flat row of loops. You'll need to create a couple of these rows in this fabric if you want it to stand out.

Alternatively, create two rows at once by working in a grid, making the first loop, then starting the next one adjacent to it before moving on to a new pair. Because you don't have to create one lengthy row then another, it seems psychologically faster and more satisfying.

For a purely shaggy rag rug effect, create more rows of rag rugging using exactly the same technique, varying the colours or fabric types as you wish. Rows should be around three squares away from the first row if making a tightly woven rug, but can be spaced a little further apart if your strips of fabric are quite thick or wide. Three or four squares apart is ideal.

LOOPY RAG RUGGING

Loopy rag rugging is a faster technique than shaggy rag rugging, but it creates a much flatter design. To begin loopy rag rugging, take a long strip of fabric and pull it through from the underside of the hessian. Tie a knot at the end of the fabric strip that is on the underside. Now create a line of fabric by pulling the fabric strip through evenly spaced squares.

Left
Use different coloured fabrics to create a pattern in the rug

Below
The hessian disappears behind the rags as you complete the rug

It's easiest if you pull the fabric up through the first square and allow enough fabric to sit on top of the hessian to form while you then poke the rest of the fabric strip down through a square close to the first one. Next, push your ragging tool down through a third square another couple along the row, and pull the fabric underneath taut against the hessian. Continue the loop-making process, leaving loops on top and pulling tight underneath. When a strip of fabric runs out, simply knot a new one and continue making loops! Unlike shaggy rag rugging (which ends up with fabric ends splayed in all directions), you can make a feature of how you angle your loops.

> **When you've plaited all your fabric, lay out the** plaits so you can see which is the longest

Begin by preparing the braids. Plait strips of coloured fabric together, securing them at one end by taping them to a tabletop with masking tape. Add new strips when one runs out by cutting a slit in the end of the existing plait strand and another in the strand to be added. Pass the new strand through the slit in the other strand. When you run out of fabric strips of that colour, tie off the plait at the end.

When you've plaited all your fabric, lay out the plaits so you can see which is the longest. Start with the fabric of which you have the least (so it seems to go further) and coil it around itself and secure it by sewing it tightly together through the centre of each plait. Wrap or sew subsequent plaits to the one you're running out of to continue the coil. ◻

QUICK TIP

A plaited coil, made from random offcuts of fabric, is an ideal way to edge your hessian-backed rag rug to give it a neat finish. Simply sew it onto the hessian.

BRAIDED RUG MAKING

Your cut-up fabric strips lend themselves well to making circular items, such as round rugs or coasters. If the material you have available is a real mishmash, rather than being sourced from three or four distinctly coloured items, consider a braided or plaited design.

PATTERN FORMING

Fabric in several different colours is a clear contender for a pattern, rather than just coloured strips. A balanced, symmetrical pattern is both easy and effective, since you just need to split each fabric pile in half and can work inwards, changing colour as needed.

If you're creating a pattern, draw it on the hessian with a marker pen as a guide. Large shapes, letters, and numbers all work well; intricate patterns less so, except with the loopy rag rugging technique. Organise your cut-up fabric so you can see how much of each you have, decide roughly what design or arrangement of colours you want to use, and allocate the colour by fabric available and amount needed.

TUTORIAL

Beautiful prints with silk PLA

Shiny plastic for great-looking parts

Above
Three versions of the Low-poly Rose Twist Vase by riskable on Thingiverse. The layer height varies from 0.1 mm to 0.3 mm

Ben Everard

@ben_everard

Ben loves cutting stuff, any stuff. There's no longer a shelf to store these tools on (it's now two shelves), and the door's in danger.

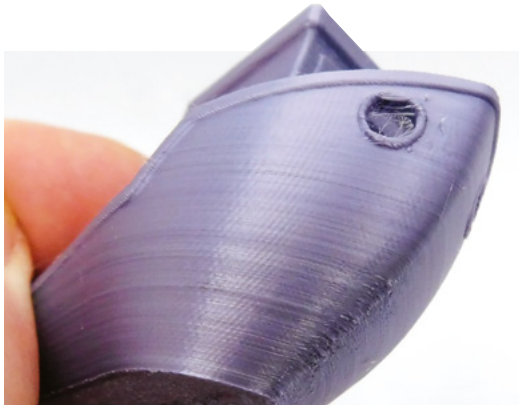
PLA is the most popular plastic for 3D printing, and it comes in some fantastic colours. But the surface finish is always a little matt. Silk PLA has added magic that makes it shine. OK, it's not magic, it's elastomers that leave your prints looking glossy and gorgeous.

For the most part, silk PLA prints just as regular PLA does, and you might be able to use your slicer's normal PLA settings. However, the elastomers that make silk filament shiny also affect its physical properties slightly. It tends to be a little more flexible and stretchy, and this can affect how it prints. We can't give exact advice as each silk filament is a little different, but you may find that you need to play with the retraction or extrusion multiplier settings to get it to print reliably.

Some people report clogging issues with silk, but this isn't something we've experienced. With a wide range of different silk filaments on the market, and each one having a different blend of PLA and elastomers, it's hard to pinpoint the cause of this. If you experience this, it's worth going through the usual processes for cleaning your extruder and perhaps checking that you're printing at the correct temperature for your filament (it may need to be a little warmer than PLA).

CHOOSE YOUR STYLE

Silk filament can create excellent-looking prints, but it can also highlight every imperfection in the print, depending on how it's used. It looks best on prints with a lot of features, such as sharp edges or tight curves – the reflections then pick out these features.



Above
Silk filament highlights any imperfections, such as the wobble in this overhang on Benchy

Above
The raw filament looks great even before it's printed

> It looks best on prints with a lot of features, **such as sharp edges or tight curves**

If your print has large flat areas or gentle curves, the reflections will pick up any issues, such as ghosting or a Z seam.

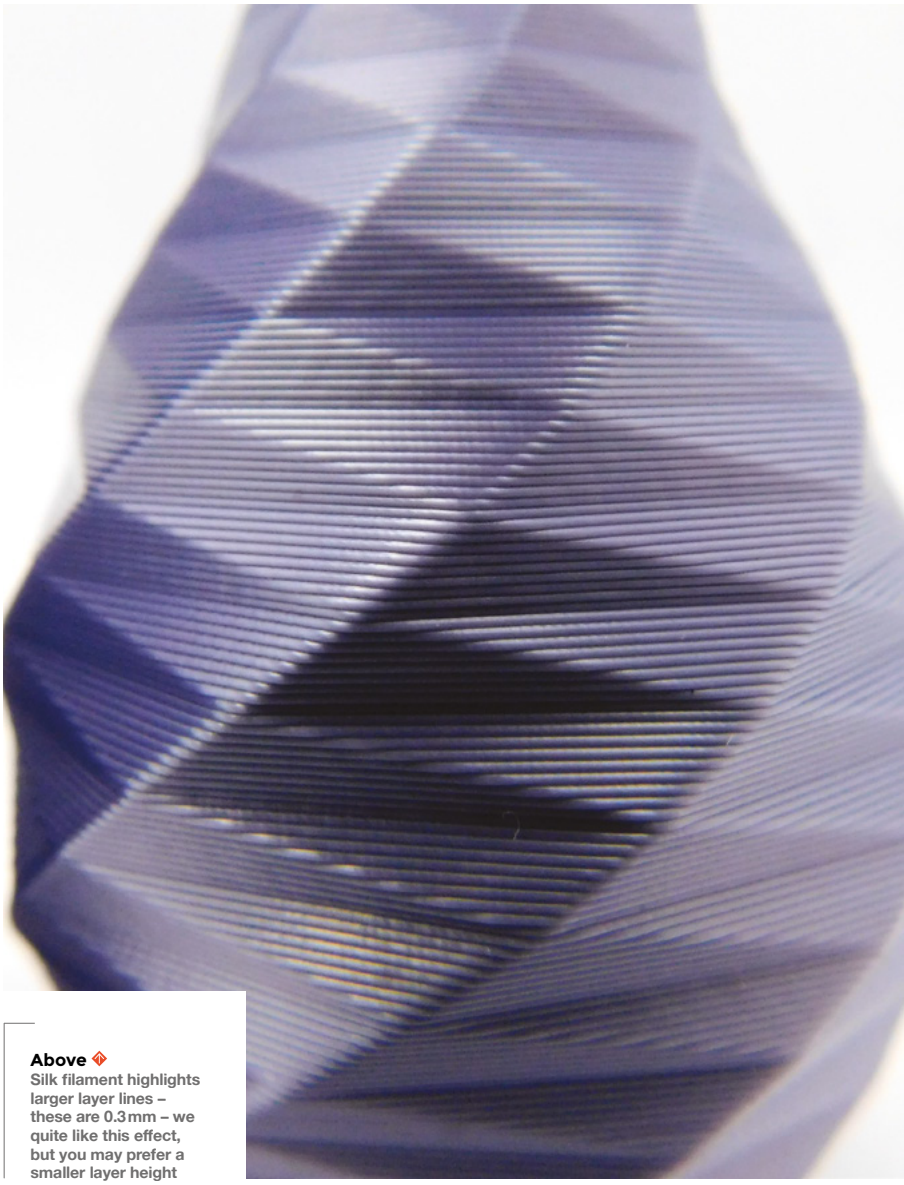
We wouldn't recommend silk PLA for any mechanically important parts – the elastomers that make it shiny affect its strength. The slight increase in flexibility may be useful for some prints, but if this is what you're after, a different plastic, such as PETG, may be more appropriate.

When used well, silk filament produces stunning prints straight off the print bed – no need for finishing or painting. We're big fans of low-poly prints in silk, but that's just us.

ALTERNATIVES

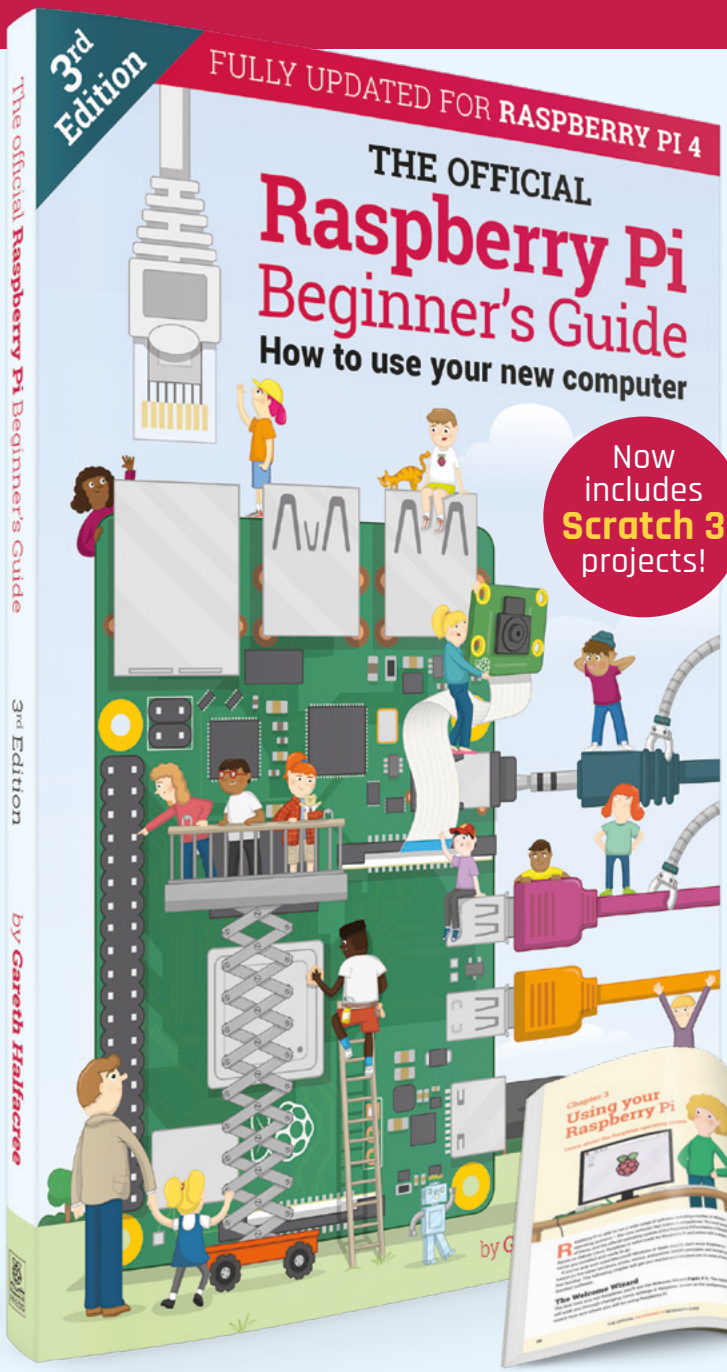
If you want high-gloss prints, silk PLA is a great choice, but there are a few alternatives:

- PET and PETG are both a little shinier than PLA, though not as shiny as most silk filaments.
- Vapour smoothing is where you place your print in an enclosure with some solvent vapour that dissolves the surface of the print – when done correctly, it can lead to a very smooth finish. The chemicals required can be quite unpleasant though. This is most commonly done with ABS, though others can work.
- You can simply paint your print using high-gloss paint. This works particularly well if you need to sand or fill parts of your print first.



Above
Silk filament highlights larger layer lines – these are 0.3mm – we quite like this effect, but you may prefer a smaller layer height

THE OFFICIAL Raspberry Pi Beginner's Guide



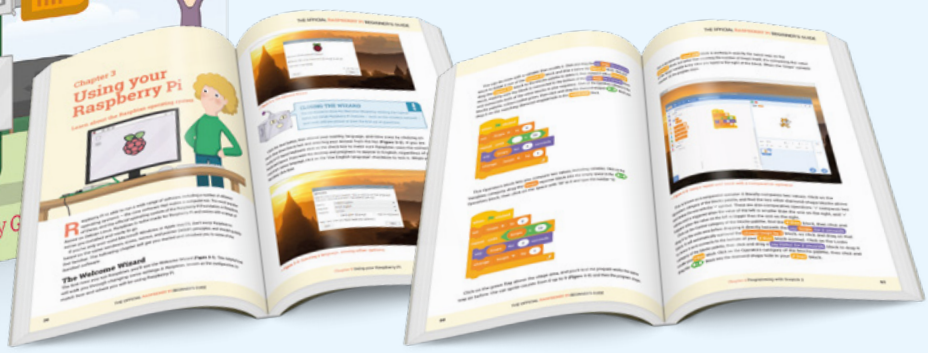
The only guide you need to get started with Raspberry Pi

Inside:

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

Plus much, much more!

£10 with FREE worldwide delivery



Buy online: magpi.cc/BGbook

BOOK OF MAKING

VOLUME 2

ONLY £10

WHSmith

BARNES&NOBLE

Available on the App Store

GET IT ON Google Play

THE BEST PROJECTS FROM HACKSPACE MAGAZINE

THE ULTIMATE SKILLS, TRICKS, AND MAKES

AVAILABLE NOW

hsmag.cc/store

FROM THE MAKERS OF **HackSpace** MAGAZINE

TUTORIAL

Workshop-ready camera kit

Add a suit of armour to your digital camera and keep it safe it from workshop accidents

Dr Andrew Lewis

@monkeysailor

Dr Andrew Lewis is the owner of **Shedlandia.com**, a restorer of old tools, a fabricator for hire, a research scientist, and a founder member of the Guild of Makers.

Right
Oil, paint, and glue can damage your camera. A silicone cover will protect your camera from the worst of the grime. Choose a distinctive colour so you can see (and avoid) it easily while you're in the workshop

If you're thinking of getting more serious about documenting your projects, then you're eventually going to end up investing in a decent camera and microphone.

When you do get that new piece of kit, it's worth taking a few extra steps to protect that investment and make sure your camera is workshop-ready.

Making things can be a messy and dangerous occupation. Fire, acid, aerosolised paint, glue, solvent spills, metal particles, and high-speed tools don't make a workshop the ideal environment to immerse your shiny new camera into. Here are a few simple things you can do to give your new kit a fighting chance.

YOU'LL NEED

- Silicone camera cover
- Tempered glass screen protector
- UV filter to suit your camera lens
- Lens hood to suit your camera lens
- Two-handed camera stabiliser

Left
A camera steady grip serves multiple functions. It keeps your dirty hands off the camera when you move it, and anything that's attached to your camera (external mics, recorders, and even the built-in screen) gets a protective ring around it that prevents damage from sideswipes and falls. The downside of this type of grip is that you need to remove it to change the battery

Below
Lens hoods are designed to stop side lighting from causing flare effects in your images, but they also make a physical barrier that protects the lens from knocks and scrapes. You can use the lens hood to anchor a cylinder of card, paper, or thin plastic to shield the exposed sides of the lens when you're working in a really messy environment

Above
Always keep at least one ultraviolet or neutral-density 0 filter on your lens. Projects do occasionally launch themselves and take unexpected flights through the workshop, and extra layers of glass on the end of your camera lens can prevent a very expensive accident

Below
You probably need to see the screen when you're filming, and that means you need to expose the delicate surface to messy fingers and mechanical perils. A tempered glass screen protector keeps your LCD safe, and can be changed when it gets stained or broken

TUTORIAL

Variable power for your projects

DC step-down or 'buck' converters are a flexible and cheap way of getting electrons

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!

Lots of the time, when experimenting with electronics, we need to feed a particular voltage into a system, and it's a rite of passage for many to wire up a voltage regulator to create a fixed voltage supply that we need. Linear regulators, such as the ubiquitous LM7805, are great in that they can take an unregulated voltage between 7 volts and 25 volts, and regulate the output to a fixed 5 volts. However, linear voltage regulators aren't the most efficient approach and often have to dissipate a lot of energy as heat, and you may have to add increasing sized heat-sinks for them to be able to cope if you draw a lot of current. Also, whilst linear regulators have many great uses, sometimes you might want to have the ability to vary the output voltage – this is where variable DC 'buck' converters can be very handy. Variable DC step-down or buck converters are a type of switch-mode converter and can be incredibly efficient. They are available in lots of different forms, and they have become cheap and easy to source online.

One of the first we looked at is a cheaply available board built around the LM2596 IC (**Figure 2**). This board is available in a few different form factors, with the most common and cheapest being one that has a simple input end, an output end, and a small multi-turn variable resistor to set the regulated output voltage. Taking an input voltage between 3.2 V and 40 V, and outputting a lower voltage between 1.25 V and 35 V, it's capable of delivering up to 3 amps.

The basic LM2596 is simple in operation – attach a supply to the input, then your multimeter and hold the probes to the output. Using a small flat-head screwdriver, adjust the small pot, and you can vary the voltage to suit your needs. A common hack with these, to make them more useful for experimenting with use across a range of projects, is to replace the tiny potentiometer on the module and wire in a larger multi-turn potentiometer. In **Figure 1**, you can see that we built a simple, yet accurate, variable power supply

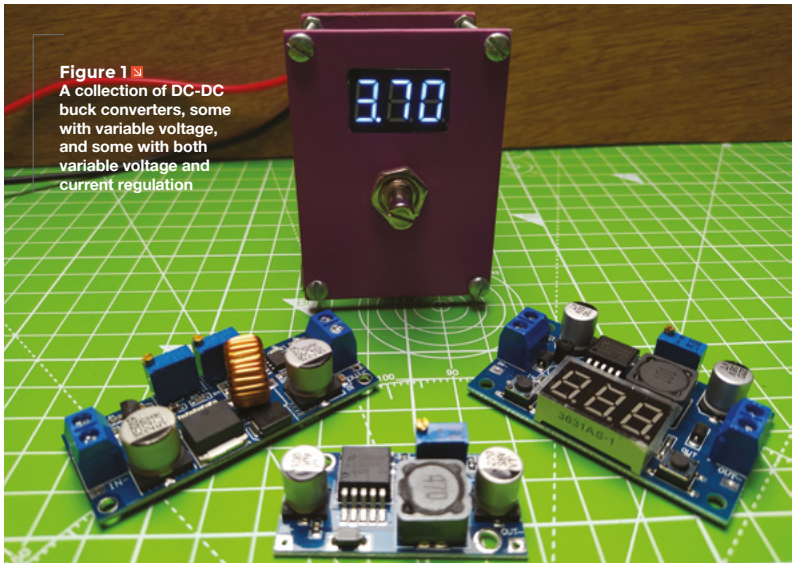
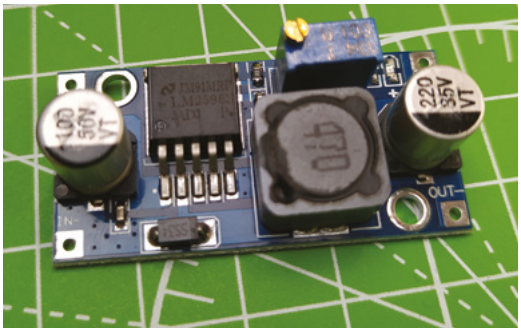


Figure 1
A collection of DC-DC buck converters, some with variable voltage, and some with both variable voltage and current regulation

Figure 2
One of the commonest boards is this one that uses the LM2596 IC, and has a single trimming potentiometer to adjust the output voltage



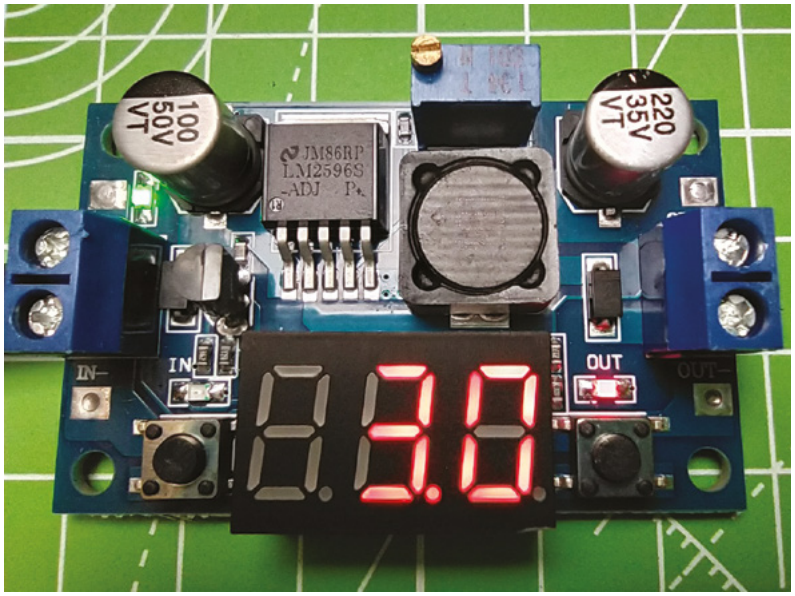


Figure 3
Another variant of the LM2596 board with a switchable LED voltmeter that can show the input voltage and also the variable output voltage

suitable for small breadboard projects by adding a ten-turn potentiometer and also a small three-wire voltage meter LED display, which can be found online for very little cost. It seems this popular use for the LM2596 has been noticed by manufacturers, and they are now sold with built-in LED displays, as seen in **Figure 3**.

More recently, cheap buck converter modules have appeared that can not only regulate voltage but also have the ability to regulate current output. This can be a very useful function to have in a power supply and can, for example, create a very efficient method of driving high-power LEDs. The module we bought is based around the XL4015 IC, and can regulate an input voltage of 8 V–36 V to an output of 1.25 V–32 V with current output adjustable up to 5 amps. The basic module we bought is sold as having a battery charging function which we haven't explored, but it's pretty simple to work with to supply a constant voltage and a constant current. The module that we have (**Figure 4**) has two small multi-turn trimpots again, to adjust the voltage and the current, but these are unlabelled, so the first task was to discover which was the voltage one! We hooked up a supply voltage and connected our multimeter, set the multimeter to measure DC voltage, and then we adjusted both trimpots to identify which one changed the voltage.

As an example, a great use for these constant-current-capable boards is to drive high-power LEDs. We wanted to drive a 1-watt 'warm white' LED we had, which required a forward voltage of 3.3 V. So, to give this LED 1 watt of power, we rearranged the power formula where power in watts (W) is equal to voltage in volts multiplied by current in amps ($P=V\times I$). Rearranging this formula, power divided by voltage equals current ($P/V=I$). We can then substitute in the values of 1 W and 3.3 V to return the current as 0.3 A.

Having set the voltage to 3.3 V, we then need to swap the probe connections on the multimeter to configure the latter to measure current. The multimeter we used has connections for a range of currents up to 600 mA, and a connector up to 10 amps. While the 600 mA range would be enough, we were aware we could easily turn the trimmer a little too vigorously and go out of range, so we used the 10 amp connection.

Connecting the multimeter and turning on the power supply, the module detects that the output is essentially shorted, indicated by the red LED, and it indicates that it is in constant current mode by the blue 'load present' LED. We then used the current-adjusting trimming potentiometer to set the current to 0.3 A (**Figure 5**).

Our supply was now ready to use with our 1 watt LED and, as such, we connected it and powered it on. The module now detects that there is a load present on the output and lights the blue LED and, of course, the 1-watt LED lights up, having been fed the correct voltage and current.

There are lots of variants of these types of cheap modules out there, and they can be utilised in projects in lots of ways. If you find an interesting variant, or come up with an interesting use for one, do get in touch.

Below
Success! The variable voltage and constant current supply delivering 1 watt to the high-power LED

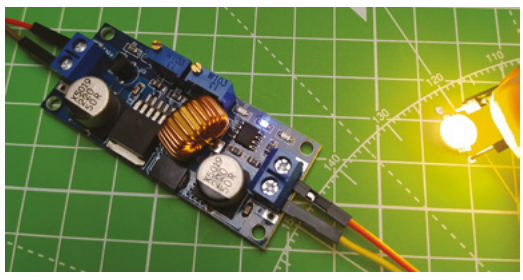
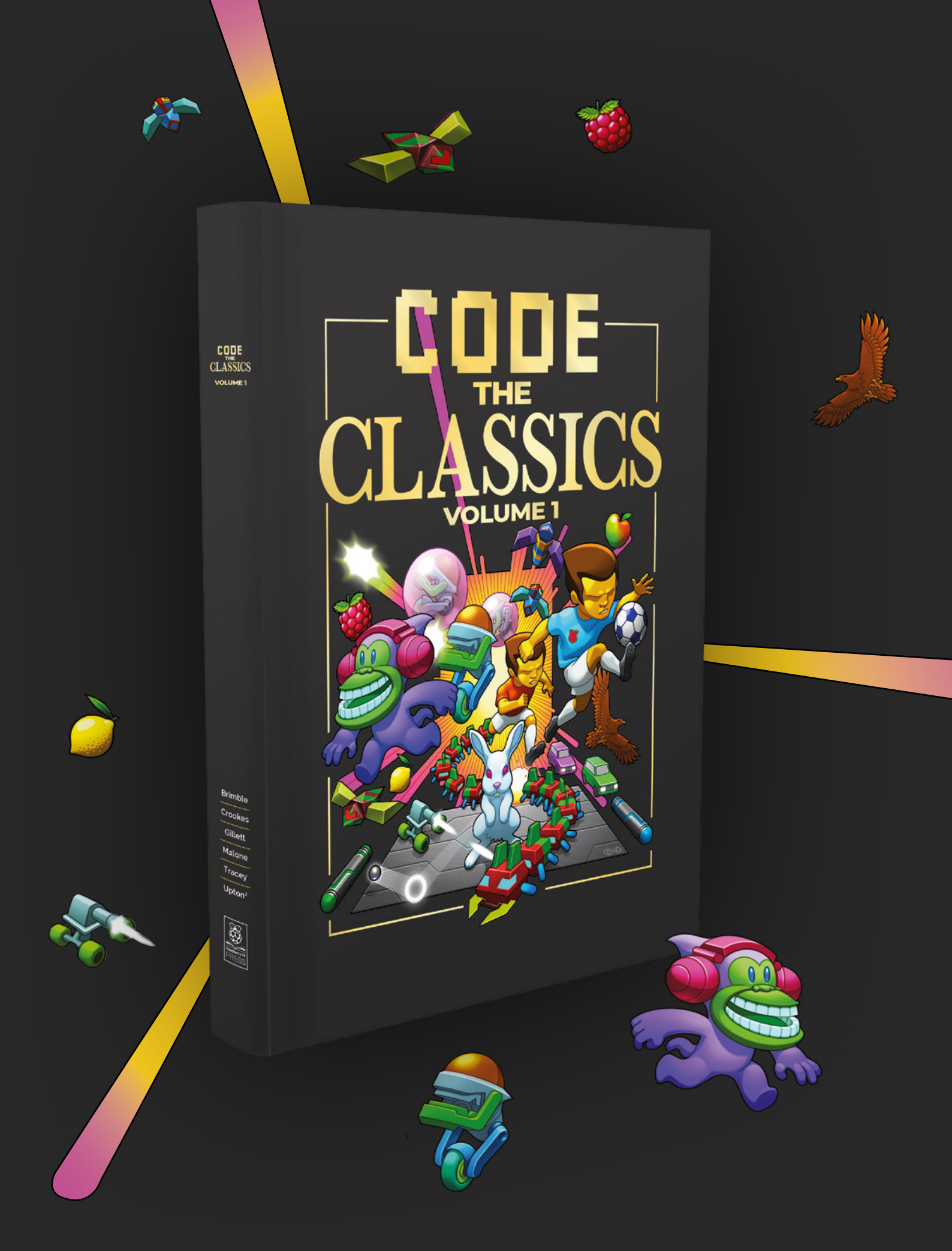


Figure 4
Identifying which trimming potentiometer adjusts the voltage, and which adjusts the current

Figure 5
Adjusting the current to the desired amount that will deliver 1 watt to the LED at the correct voltage

YOU'LL NEED

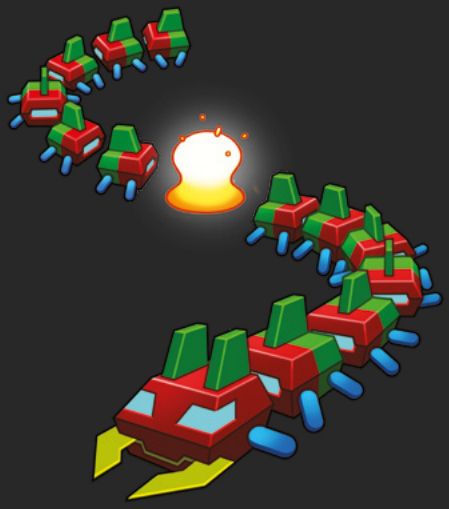
- **LM2596 or XL4015 DC-DC converter module**
- **Power source** (a 12 V battery, for example)
- **A multimeter, capable of measuring voltages and currents**
- **1-watt high-power LED** (optional)


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

CODE THE CLASSICS VOLUME 1

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

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



Available now **hsmag.cc/store**

Debugging digital signals

Learn to fix the most common causes of faults in inter-component communications

Above
Probing a circuit with a Saleae logic analyser to snoop on the SPI communications

Dr Andrew Robinson

Dr Andrew Robinson is a part-time university lecturer and runs his own design and manufacture electronics consultancy in Manchester. He's responsible for creating CodeBug, PiFace, and the Quizit.net interactive quiz site.

Discover how to eavesdrop on the messages sent between components in your circuits. Learn the common causes of problems and how reading the contents of the messages can facilitate diagnosis and save days of debugging time.

In my previous article, *Debug Techniques with Multimeters and Oscilloscopes*, we gave our golden rules for debugging: break the system down into subcomponents and observe the behaviour. To achieve this, you'll need to monitor the messages being sent between systems.

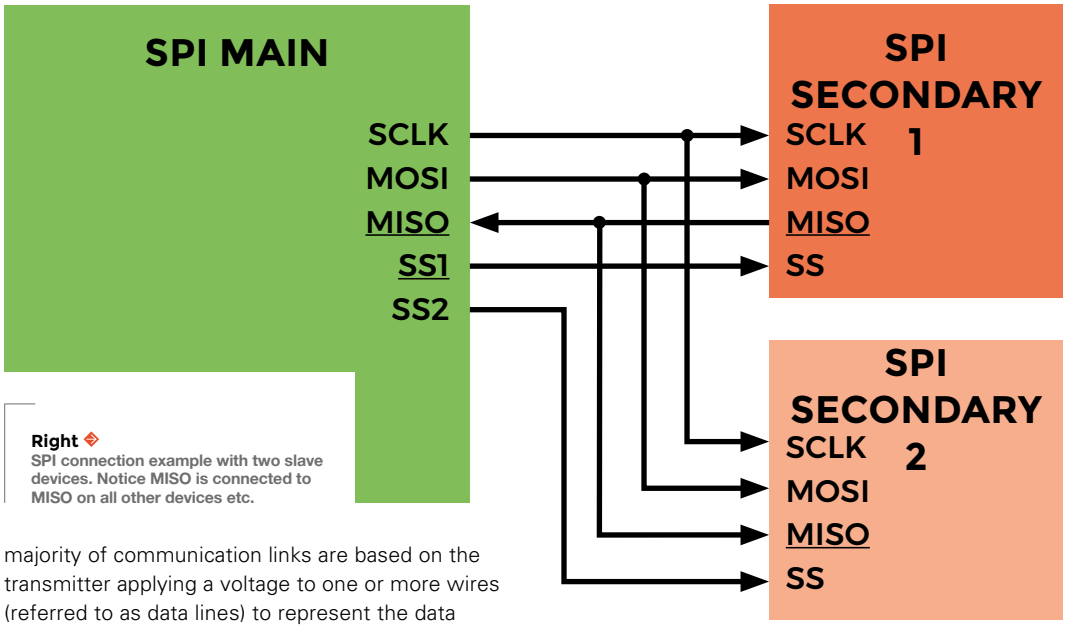
We also mentioned how planning for debugging at the design stage can make things easier. You need to be able to probe communications signals to take measurements, or temporarily break and inject stimulus data. As such, design your PCB accordingly; provide pads or space for headers to connect test wires and include zero ohm resistors, or wire links where you might need to temporarily break a connection. Consider adding a spare output pin you can use to generate a trigger signal, or a reference marker to indicate when a point has been reached in your code. And a final tip: label everything on the silkscreen so it's easy to probe the correct wire!

EAVESDROP ON THE INTER-DEVICE COMMUNICATIONS

Before we jump into practical details, it's worth considering how data is sent in circuits. The vast

ACTIVE HIGH AND ACTIVE LOW

Protocols differ in how they interpret the voltage on a wire. An active high signal will read a higher voltage as 'on', while an active low signal will interpret a low or lack of voltage as 'on'. To interpret communication waveforms, you need to know if a signal is active high or low. Convention states that signal names have a line above or are prefixed with an exclamation mark to indicate active low. Signals are said to be asserted if they are made active.



Right
SPI connection example with two slave devices. Notice MISO is connected to MISO on all other devices etc.

majority of communication links are based on the transmitter applying a voltage to one or more wires (referred to as data lines) to represent the data being sent. A receiver reads the data by measuring the voltage (or absence of) on the wires. Since the data being sent will change over time, so too will the voltage, and the receiver needs to know when the voltage on all the lines should have settled and be ready to read. There are two schemes for this: clocked and self-clocking.

CLOCKED VS SELF-CLOCKING

Clocked protocols have an additional signal line that indicates when the other signals have settled and data is ready. Commonly encountered clocked protocols include I²C and SPI. Other protocols, such as serial, do not have a separate clock signal and derive the timing from the data lines instead. Data transmissions have a data rate (sometimes called baud rate) which relates to the number of times a signal is allowed to change in a second. The receiver and sender must be set to the same data rate otherwise no or incorrect data will be transferred. The physical connection must be such that it allows the electrical signals to transition fast enough and reduce susceptibility to interference.

FIND OUT WHAT THE RECEIVER IS REALLY RECEIVING

If you can, a good place to start when debugging a misbehaving communication is to get an understanding of what the receiver is actually receiving. Add code to log it to memory, or somehow display it (e.g. re-sending it over a serial monitoring console, or flash an LED for particular messages or characters). Compare what is received with what the sender thinks it is sending by also getting the sender to log outgoing data. Typical causes of no or completely incorrect data are misconfigured baud rates or pin configurations (e.g. if a pin is set inappropriately for input or output) on microcontrollers. If the data being received matches what is being sent then you're probably sending the wrong data and you need to check your software. If possible, compare the data that your code sends with a known working example.

> **If possible, compare the data that your code sends** with a known working example

CHECKING THE DATA

Sometimes you can't see what a device is receiving, or you need more insight into what is happening on the wires. The next level of debugging is to read the data on the wire with another device and check that it is as expected. There's nothing to stop you attaching another microcontroller and comparing what it is receiving. However, while this is cheap, if you've used the same receiving code, you can't be confident that the same bug won't affect the second microcontroller too. Some chips have hardware design faults in communication controllers, so using the same device won't reveal this either. A more →

QUICK TIP

The quickest check that a serial port is functioning is to disconnect the transmit and receive lines from the rest of the circuit and connect them to each other (called a loopback). If the data sent out matches what's received, you know the device is at least capable of sending and receiving. If not, you need to check that the port is enabled.

Above
SPI capture of master sending characters 'hi', while slave doesn't send anything (interpreted as 00)

robust approach would be to use a logic analyser to record and decode the messages and show where the signal is invalid.

EQUIPMENT BUYING GUIDE

Most modern mixed-signal digital scopes can offer protocol decoding as add-ons. This has the advantage of integrating with other features; for instance, triggering an analogue capture. Alternatively, logic analysers are available as standalone devices.

This author has found Saleae a good choice for a standalone logic analyser. Just over ten years ago, when the founders couldn't find a suitable, affordable logic analyser, they built their own, with the first units being hand-assembled by the CEO. With a passion for making test equipment that's highly usable, their software is powerful, yet simple and clear to use. It works out of the box for many protocols, and can easily be extended through Python. Data is gathered on the device, and then streamed over USB to be recorded and analysed in Linux, Windows, or macOS. As such, the memory depth is only limited to the memory of your computer, which is vastly larger than current oscilloscopes. They often have discounts available for makers and startups.

> "To debug a protocol **you need to know what it should look like first**"

OTHER CONSIDERATIONS

Very low-cost analysers are available, but it's worth considering the specifications carefully, and checking reviews regarding their reliability. Some Saleae clones lack input isolation, which puts your PC and circuit at risk of damage. When debugging, you don't want doubts as to whether your test equipment is not seeing a fault or, even worse, reporting a phantom fault that's not really there.

When choosing an analyser, it's worth considering similar features as covered in the last article on choosing an oscilloscope: memory depth, sample rate, max signal speed, input isolation, usability of software and range of protocols supported.

DEBUGGING SPI

To debug a protocol, you need to know what it should look like first. SPI is perhaps the simplest, so we'll look at that first. You'll commonly encounter SPI where moderate amounts of data have to be transferred quickly, e.g. for memories, SD cards, and small LCD/OLED displays.

SPI consists of one main, and at least one Secondary device, connected by four signals:

- SCLK Serial CLocK (generated by the main to synchronise timing)
- MISO Main Input Secondary Output (used to receive data at the main)
- MOSI Main Output Secondary Input (used to send data to the secondary)
- SS Secondary Select (used to indicate the main wishes to communicate with a device and unique per secondary device).

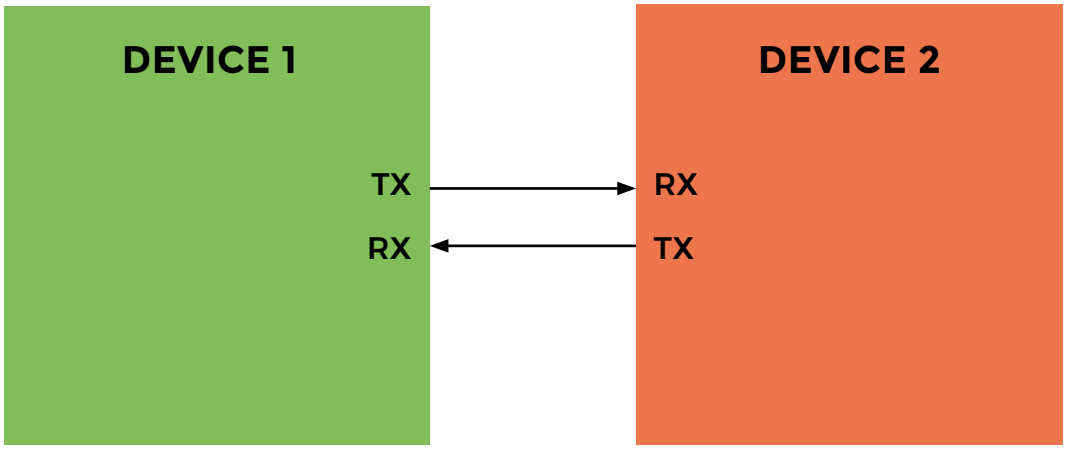
The main selects the secondary it wants to communicate with by asserting the appropriate secondary Select line. SS is almost always active low. The main drives the Serial Clock line with pulses that indicate when the data is ready to be read. Data is sent one bit at a time on the MISO and MOSI lines. Data is transferred to and from the main and secondary simultaneously. If there is no data to be sent, the line is left idle, and 0 will be transmitted. At the end of a transmission, the main will stop the SCLK and usually deassert the SS.

QUICK TIP

Sigrok is an open-source project that provides protocol decoding and remote control of a range of test equipment. This community effort makes it possible to decode data gathered by very low-cost analysers or oscilloscopes. However, unless someone has already created an interface for your equipment, you'll have to write some interfacing code first.

BIT ORDER

To further add to confusion, data can be sent most significant bit (MSB) first or least significant bit (LSB) first. For instance, the binary value for six is 00000110. This can be sent in the order 0,0,0,0,0,1,1,0 (MSB first), or in the order 0,1,1,0,0,0,0,0 (LSB) bit by bit. For historical reasons, serial links use LSB.



Left
Example serial port connection, TX is connected to RX

When debugging SPI, check:

- at the start of a transaction that the SS line is asserted;
- there are regular transitions on the SCLK line and transitions on the MISO/MOSI lines that vary with the data being sent;
- that the state of the data lines are constant when the clock indicates data is ready to be read;
- on the data sheet that the clock polarity and clock phase match for the main and all secondaries. Different devices interpret the clock as active high or active low, and clock the data in at the leading or trailing transition of the clock pulse;
- that the bus is running below the maximum clock speed that the devices and cabling can support. If in doubt, try slowing the main clock down to see if data transmission becomes more reliable.

The most common causes of problems on SPI are mixed up signal line connections, misconfigured ports on microcontrollers, or too fast a clock. Examining the waveforms should reveal all these errors.

SERIAL INTERFACE

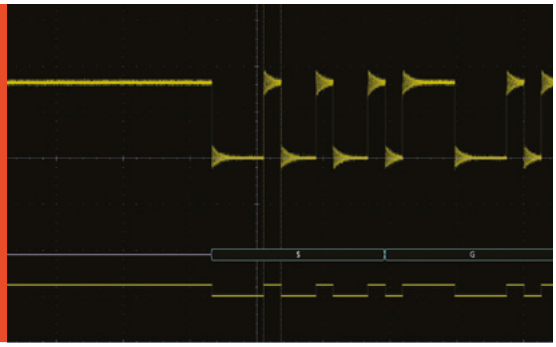
'Serial' is a bit of a woolly term. Although the term serial interface can describe any class of interface where data is sent sequentially, 'serial port' (which sometimes confusingly is shortened to just serial) is often used as shorthand to refer to a specific communication link. Some devices will call serial ports RS232 (even when technically they don't meet the RS232 standard), or a UART (universal asynchronous receiver transmitter). Serial ports are frequently found on circuit boards to print debugging information, and used to interface GPS receivers or 3G and 4G modems. Serial ports can have a range of configuration settings, but the vast majority have the characteristics as detailed below. If necessary, check the data sheets for the exact settings for the serial port on the device you're debugging.

Serial ports transmit data on their TX signal and receive on RX. These names are relative to each device, so the TX of one device is connected to the RX of another. There should only ever be one TX but →

Below
The impact of long cables causes signal transitions to be less clean, which will show on an oscilloscope

RISE TIMES AND **EDGE SPEED**

Wires in the real world are not perfect. They behave as little chains of inductors and resistors, with tiny capacitors (which behave as if they are coupled with every surrounding conductor nearby). When a signal on a wire changes, it has to charge or discharge this capacitance through the resistance, which takes time. Though the delays may be small, for a signal that may be changing 40 million times per second (for a typical 40MHz SPI bus) or more, these delays can become significant compared to the time between signal reads. The time taken to transition is called the rise and fall times, or edge speed. You can keep rise times fast by reducing the resistance and capacitance of interconnecting wires by keeping them short and away from other interfering conductors. Alternatively, reducing data transmission speeds makes the rise time less significant, compared with the time the signals are settled and stable for.



Above
Decoding characters from a serial port

this can be connected to multiple RX. Being asynchronous, there is no separate clock and the sender and receiver synchronise from the transitions on the data line. Typically, the sender keeps the line high when it is idle.

To send a byte of data, the sender sends a start bit followed by eight bits to represent the data and then a stop bit. Basic error detection can be performed if a parity bit is included, but this is not commonly used. Configurations are expressed in a shorthand; 8N1 refers to eight data bits, no parity and 1 stop bit, and is almost always used. The time to send each bit is a factor of the baud rate. From the baud rate and the configuration, you can calculate the timing of each bit. The sender and receiver must have the same baud rate and configuration otherwise no or erroneous data will be received.

To debug a serial port, first you need to know the approximate timing of an individual bit. For example, with a baud rate of 9600 characters per second and 8N1, in one second (8+1+1) × 9600 bits will be sent, meaning each bit period will have a time of 1/96000 = 104.17 µs.

Find the start of transmission by looking for the line transitioning after idling for a period. Mark off the bit periods (every 104.17 µs), and record if the line is at the same level as when idle. This example shows data being sent from a GPS. Breaking the captured waveform into 104.17 µs blocks, we get:

	Data								
0	0	0	1	0	0	1	0	0	1

	Data								
0	1	1	1	0	0	0	1	0	1

	Data								
0	0	1	1	1	0	0	1	0	1

Use the waveform to check the settings, use the pulse period to confirm the baud rate, and that there are the correct number of bits for the start, stop, and parity bits.

READING THE SERIAL DATA

It's not too difficult to read the data being sent from the waveforms. However, for historical reasons the bits of each character are usually sent backwards (or LSB first), so need to be reversed.

LSB First	MSB First	Decimal	ASCII
00100100	00100100	36	$
11100010	01000111	71	G
01110010	01001110	78	N

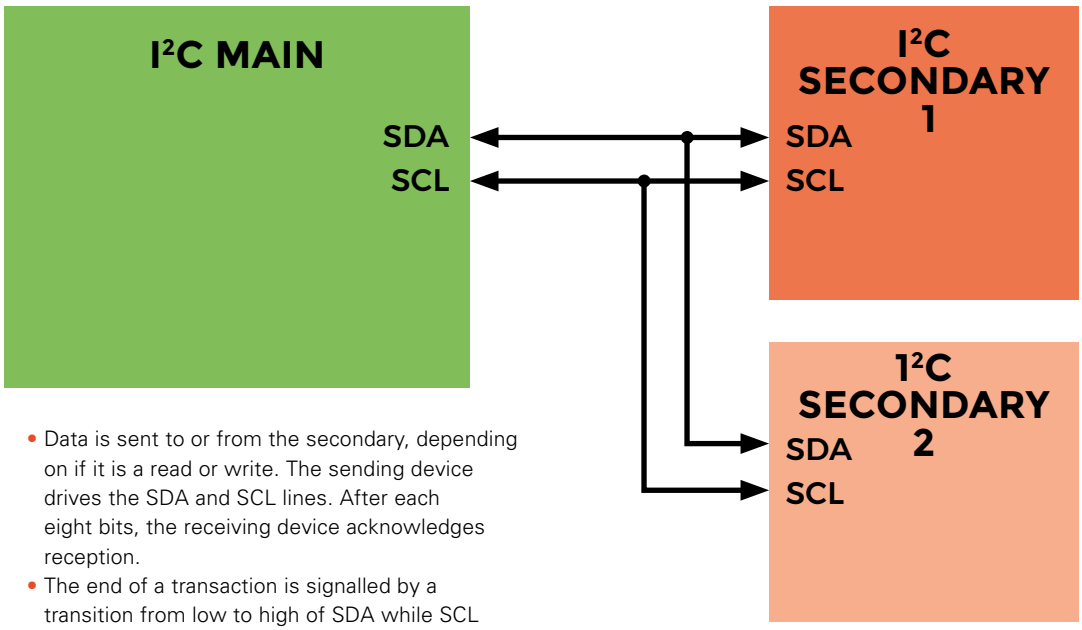
Here, we show an example decode where we reverse each eight bits of data (to make it MSB first), and then convert it to decimal before looking it up in an ASCII table to get the characters. Here we can see we have received $GN, which is the start of a message from the GPS.

Part of the reason serial port communications are so commonly encountered is that they're simple and very robust. The majority of problems will be due to misconfigured settings. Luckily, you can glean all the settings you need by looking at the waveform.

I²C INTERFACE

The I²C interface has just two wires, SDA (Serial DAta) and SCL (Serial CLock), and allows bidirectional communication between the main and multiple secondaries. Each device has a unique address and secondaries only respond when invited by the main.

- The main signals the start of a transaction by driving SDA low while SCL is high.
- The main sends the address of the secondary it wants to talk to and if it wants to read or write.



Left Example setup for I²C

- Data is sent to or from the secondary, depending on if it is a read or write. The sending device drives the SDA and SCL lines. After each eight bits, the receiving device acknowledges reception.
- The end of a transaction is signalled by a transition from low to high of SDA while SCL is high.

You should find details of how to interpret data you recieve from a secondary in the data sheet.

WITH I²C, DON'T FORGET THE PULL-UPS!

I²C data lines are never driven high by a chip; instead they rely on resistors connected to the power supply to pull the signals up. As such, a common cause of problems is forgetting to include the pull-up resistors. Other causes of problems include incorrect secondary address, or interference between SCL and SDA. You can check the address of a secondary device, and that it's responding, by doing a scan of the bus. On Raspberry Pi, this is done with the `i2cdetect` command.

Enable the I²C interface in raspi-config

```
sudo raspi-config
```

Install the i2c scan tool

```
sudo apt-get install i2c-tools
sudo i2cdetect -y 1
```

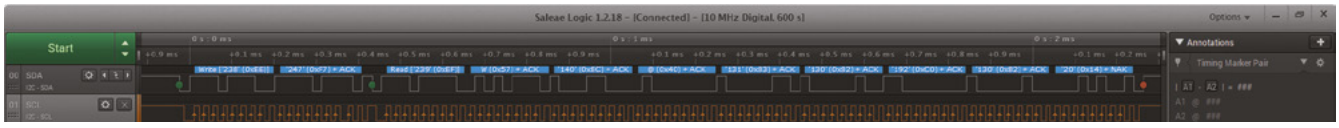
Or, for an old 256MB Raspberry Pi

```
sudo i2cdetect -y 0
```

> **By having some knowledge of these three protocols,** you should be able to diagnose the vast majority of communication problems

I²C is unusual as the secondary device can slow down the main by holding the SCL line low until it is ready. Unfortunately, some devices (such as early Raspberry Pis) do not recognise this clock stretching correctly, which can result in errors. The other likely causes of problems with I²C are a lack of or incorrect pull-up resistors, poor cabling resulting in interference, or (like the other protocols) too fast a clock speed.

By having some knowledge of these three protocols, you should be able to diagnose the vast majority of communication problems you'll likely encounter as a maker. This author used to attempt to fix communication problems by repeatedly making little changes to code, recompiling, and flashing. Once he got in the habit of recording and reading communication waveforms, often he was able to quickly pinpoint exactly what was wrong, and save days of debugging time. ◻



Below I²C waveform capture. Example write of 0xF7 followed by read of eight bytes. Green blobs indicate start condition, and red blobs shows stop condition

ESP8266 smart light

Add a light fitting to the Internet of Things

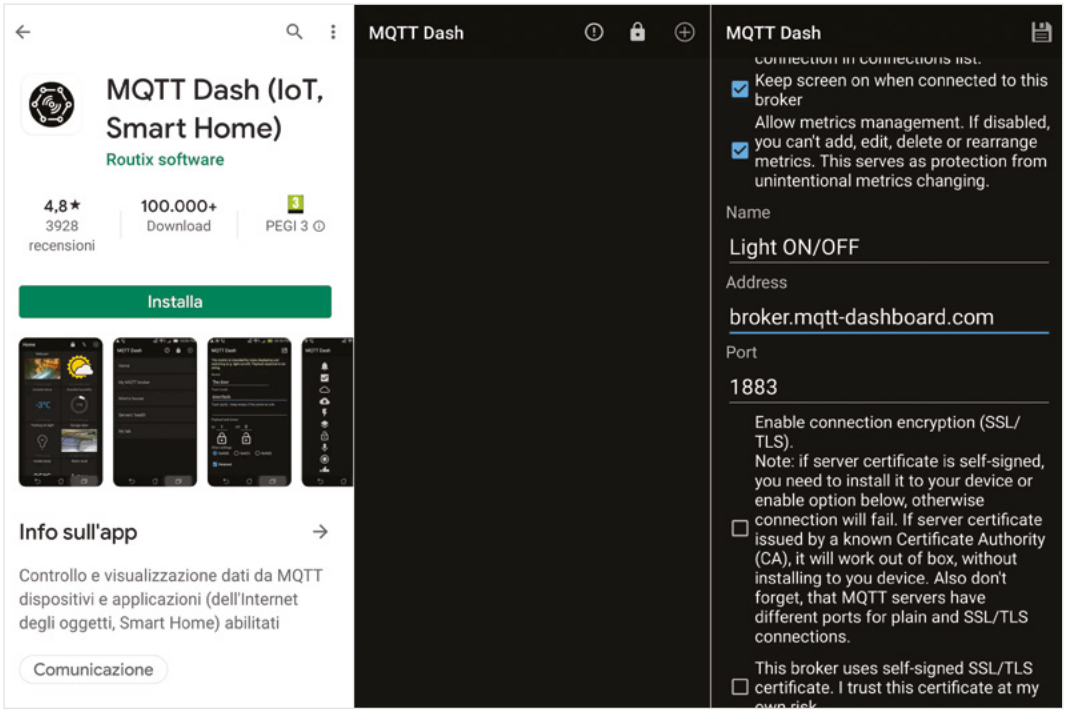
Demetrio Pinna

Demitrio spends his working days wrangling big computer systems and his free time playing with little ones.

Since smartphones have become part of our lives, our way of life has changed. They have become objects which we can no longer separate ourselves from. In fact, it is often the first thing we put in our pocket before leaving the house. With a smartphone in hand, we can contact anyone, create and exchange multimedia content, surf the internet, play games, work, travel with maps, and much more. The future ahead of us is full of surprises because this tool will allow us to do things that were previously unthinkable, especially with the Internet of Things (IoT), where all objects are connected to the internet and are constantly monitored from anywhere and at any time. Everything and everyone always connected in a big way on a network.

For some years now, it has been possible to use smartphones to replace remote controls via special apps, for example, your TV, air conditioning, DVD players, etc. This prompted us to create a simple project to use the smartphone to remotely switch on/off other equipment such as lights, irrigation systems, temperature sensors, domestic appliances, and much more.

We did this with a NodeMCU 1.0 ESP8266 (ESP-12E) card, an SRD-05VDC-SL-C relay, and one smartphone with an app installed that would



Right
The phone app makes it easy to send messages to your device and can be used for far more than a light

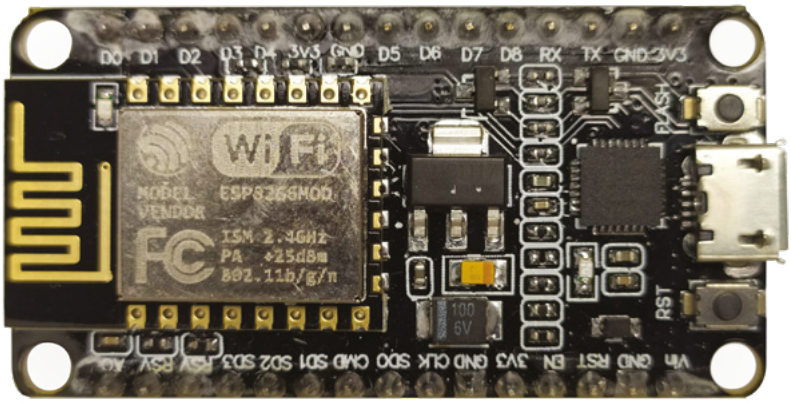
transform it into an MQTT client, using the Message Queuing Telemetry Transport protocol.

For Android smartphones, there's the MQTT Dash app; for iOS, there's MQTT Probe; and for Chrome, there's MQTTBox. Many other platforms also have suitable apps.

MQTT is a very light and flexible asynchronous data transmission protocol that works according to a system of publication and subscription of messages. The messages are sent to a special server called 'message broker'. The devices involved in the transmission of messages are called 'clients', and in our case they are the smartphone (or computer) and the NodeMCU board. The user, via smartphone, 'publishes' messages in the broker – a message, in our case, corresponds to a specific command.

> "The broker sends the message to the clients **who have 'subscribed' to the channel, in this case the NodeMCU**"

The broker sends the message to the clients who have 'subscribed' to the channel, in this case the NodeMCU, which converts the message into a command ("1" = closes the relay contacts = light on, or "0" = opens the relay contacts = light off). A moment later, the NodeMCU 'posts' a message to the broker corresponding to a state. The broker sends the message to the clients that have 'subscribed' to the channel – in this case, the smartphone (or computer), which converts the message into a status warning ("1" = "The Light is ON", or "0" = "The Light is OFF").



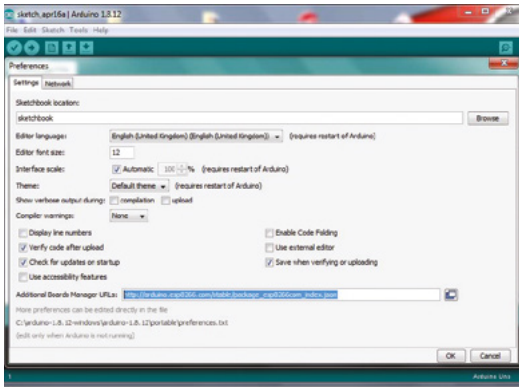
Above The ESP8266 comes on many different development boards. The one we've used here – the NodeMCU – is a common option, but this technique should work with others

Bottom Left Setting up the Arduino IDE to work with ESP8266 boards

The MQTT protocol is a standard protocol, and the dedicated TCP/IP port is 1883.

It is possible to run your own MQTT server, but we have used a publicly available one at **broker.mqtt-dashboard.com**. You could also run a private proker which may be more secure.

Obviously, it is essential that the NodeMCU card is also connected to a WiFi network, so don't forget to set the SSID and password of the access point to be used in the sketch.



SAFETY

This project uses mains electricity, and so there is a risk of electrocution. We recommend using low voltages for testing it out, and then mounting it securely before applying a high voltage. The circuit should be properly enclosed before a high voltage is applied. If you're in any way unsure about working with mains voltages, you could use this project to switch lower voltages – plenty of devices run at 12V or lower and will work safely with this device.

FIRST, YOU NEED TO INSTALL THE APP AND CONFIGURE THE BROKER

- From Google Play, search and install 'MQTT Dash (IoT, Smart Home)'. Open the newly installed app and tap on the '+' at the top right. This will open a window where you can enter the following:
- Name = Light ON/OFF;
- Address = broker.mqtt-dashboard.com

Then tap on the floppy disk icon at the top right, to save the configuration. You can now configure the on/off button. Tap on 'Light ON/OFF' to connect the smartphone to the broker, and use the '+' button to add a new widget. Select Switch/button as the type and enter the following configuration:

- Name = ON/OFF;
  Topic (pub) – keep empty if the same as sub = inTopicCommand →

- Again, use the floppy disk icon to save the configuration.
- The final part of the app configuration is to set up the status topic. Again, use the '+' button, but this time select Text as the type, and enter the following:
- Name = STATUS;
  Topic (sub) = outTopicStatus;
  Main text size = Medium
- A final click on the floppy disk icon, and you're ready to start work on the hardware.

WHAT IS NODEMCU 1.0 ESP8266 (ESP-12E)?

The NodeMCU is a mini card that mounts the ESP8266, a system on a chip (SoC) that includes a WiFi module with integrated TCP/IP stack. It's a microcontroller that, due to its small size and low price tag, is very versatile in IoT applications, home automation, wearable applications, automotive, etc.

To program the NodeMCU, we will use the Arduino IDE by connecting it to the PC via a cable micro USB. First, however, the correct USB driver must be installed, which the manufacturer indicated through the abbreviation on the chip (USB > TTL converter) placed on the card.

- In our case, the 'SILABS CP2102' chip and the drivers can be downloaded from **hsmag.cc/bA91uW**.
- If Arduino IDE is not installed on your computer, download it from **hsmag.cc/HXA72T**.
- Start the IDE, and in the Additional Board Manager URLs field (Arduino IDE > File > Preferences (Settings)), insert: arduino.esp8266.com/stable/package_esp8266com_index.json

Through the Boards Manager (Arduino IDE > Tools > Boards Manager), install version 2.6.3 of the module called 'esp8266 by ESP8266 Community'. Close the Arduino IDE and restart it, then enter the following options in the tools menu:

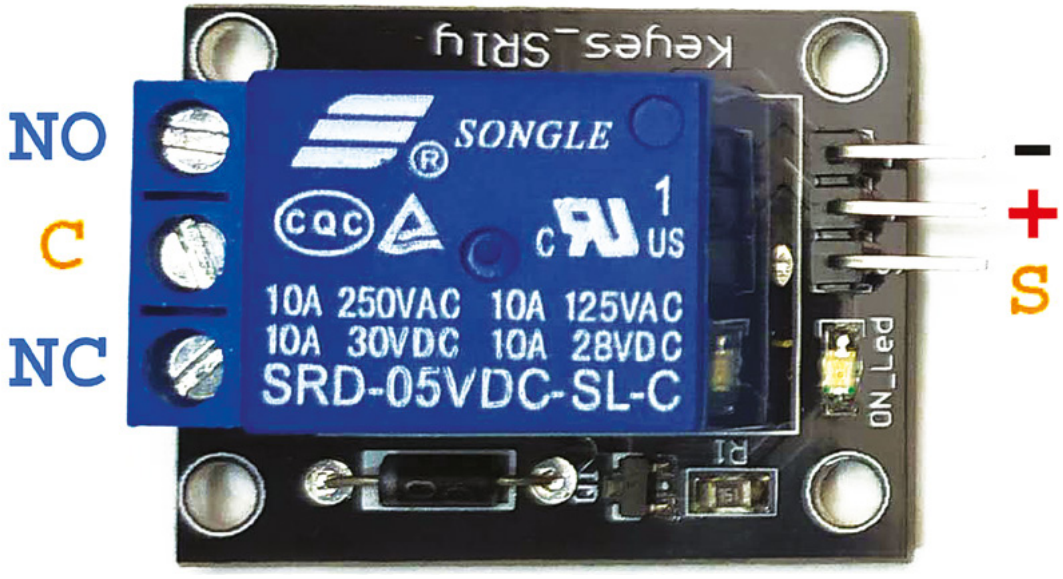
```
Board: "NodeMCU 1.0 (ESP-12E Module)";
Builtin Led: "16";
Upload Speed: "115200";
CPU Frequency: "80MHz";
Flash Size: "4M (3M SPIFFS)";
```

And select the relevant port.

- Finally, download the 'PubSubClient 2.7' library (**PubSubClient-2.7.0.zip**) from **hsmag.cc/wNUb8f**, and unzip it in the **Arduino-1.8.12** library folder.

Now that we've got our microcontroller set up, the second half of our project is a relay. We opted for an SRD-05VDC-SL-C. It's a relay able to work with maximum current to the contacts of 10A, maximum voltage to the contacts of 250V, and working temperature from 25°C to 70°C. The internal electromagnet that opens and closes the internal contacts is powered up at 5Vcc and is controlled by the high pulses received in the Signal pin. It can work in NC (normally closed) or NO (normally open).

In NC mode, the relay opens the circuit and then interrupts the passage of current when a high pulse from the NodeMCU arrives in the Signal pin. To close



Right
This is the pinout of our relay module, but check yours before wiring it up. You may need to adjust our wiring to match your module.

the circuit and return to the original state, it needs an impulse low.

In the NO mode, however, the exact opposite occurs.

```
"NO": Normally Open 120-250V
from gpiozero import Buzzer
"C" : Common terminal
"NC": Normally Closed 120-250V
"-" : Connects to the ground pin on the NodeMCU
"+" : Connects the NodeMCU's 3.3V pin
"S" : Carries the trigger signal from the NodeMCU
that activates the relay
```

NOW THAT WE'VE GOT ALL THE BITS, LET'S CONNECT THEM UP

The assembly of this circuit is very simple since there are only two components: the NodeMCU 1.0 ESP8266 (ESP-12E) and relay. Despite its simplicity, make sure you pay attention as we'll be using this with mains voltages (see safety box). However, it's best to get everything set up and tested before you add any high voltages. We've given the circuit layout on a breadboard in figure 1 which is for low-voltage testing. You should put this circuit on perfboard or similar secure setup for mains use. Note that

> Before adding mains power, you need to **make sure that your project is wired together securely**

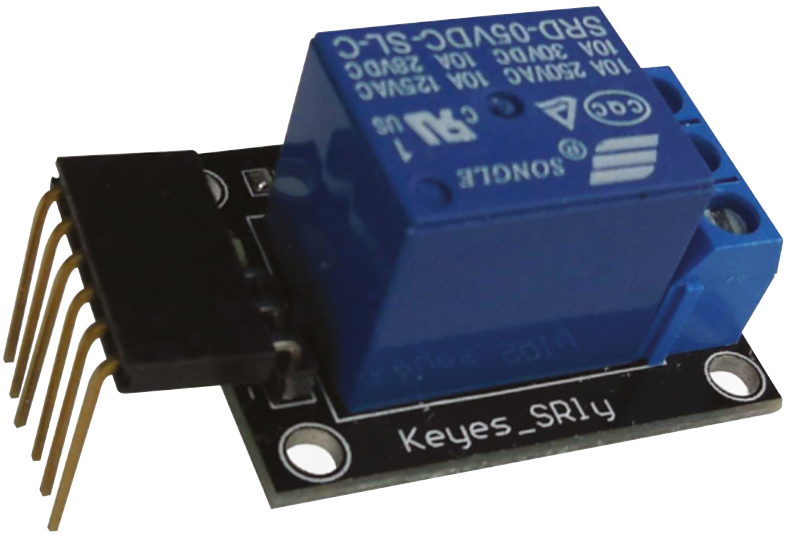
mains won't flow through the main circuit (only the terminals on the relay). Remember that some connections on the bottom of the relay board will be connected to the lightbulb voltage as will the screw terminals on the top.

Let's start by connecting the '-' and '+' of the relay respectively to GND (black wire) and 3V3 (red wire) of the NodeMCU, while pin S of the relay will be connected to D1 of the NodeMCU. Connect the lamp by carefully following the diagram (**Figure 1**), to avoid possible short circuits. Connect it only to connectors C (common) and NO of the relay, but not at the power.

At this point, make sure everything is correct and, only after this check, connect the NodeMCU to the computer via a micro USB cable. Run the Arduino IDE, and load the sketch (which you can download from **hsmag.cc/issue34**), through which you can test the circuit operation by analysing the IDE Serial Monitor. If, when using the app, you hear the noise of relay contacts that close and open, secure your circuit in a container, and only after, connect the lamp to alternating current. Whenever NodeMCU receives from the broker **broker.mqttdashboard.com** the message '1', the relay contact will be closed, allowing the lamp to turn on. A moment later, the smartphone will show the command execution status on the display: 'Light On'.

Before adding mains power, you need to make sure that your project is wired together securely and fully enclosed in an insulating box. You need to make sure that there's no chance of fingers, or anything else conductive, coming into contact with the high-voltage parts of the circuit. You can buy plastic project boxes and other forms of enclosure, or make your own.

Once everything is secure and safe, plug everything in and reap the rewards of your labour by controlling your lighting from the comfort of your couch.



Above
Right-angled headers may make it easier to develop the circuit

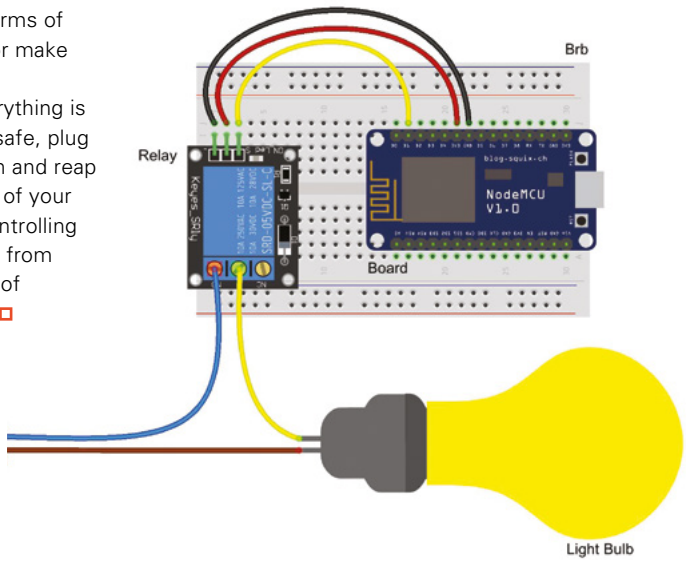


Figure 1
The layout for low-voltage testing. Put this on perfboard or similar and enclose before applying high voltages

Part 04

Rotary encoders: Raise a Glitch Storm

Control a storm of sound with rotary encoders

MAKER

Mike Cook

Veteran magazine author from the old days, writer of the Body Build series, plus co-author of *Raspberry Pi for Dummies*, *Raspberry Pi Projects*, and *Raspberry Pi Projects for Dummies*.

magpi.cc/mikecook

A Glitch Storm is an explosive torrent of musical rhythms and sound, all generated from a single line of code. In theory, you can't do this with a Raspberry Pi running Python – this month, we throw theory out the window and show you how.

01 What is a Glitch Storm

A Glitch Storm is a user-influenceable version of bytebeat music. We love definitions like that here at the Bakery: something you have never heard of is simple a development of something else you have never heard of. Bytebeat music was at the heart of the old Commodore 64 demo scene, a competition to see who could produce the most impressive graphs and music in a very limited number of bytes. This was revived/rediscovered and christened by Viznut, aka Ville-Matias Heikkilä, in 2011. And then JC Ureña of the 'spherical sound society' converted the concept into the interactive Glitch Storm.

02 So what is it?

Most random music generators work on the level of notes; that is, notes are chosen one at a time and then played, like our Fractal Music project in *The MagPi* #66 (**magpi.cc/66**). However, with bytebeat music, an algorithm generates the actual samples levels that make up the sound. This algorithm performs bitwise operations on a tick variable that increments with each sample. Depending on the algorithm used, this may or may not produce something musically interesting. Often, the samples produced exhibit a fractal structure, which is itself similar on many levels, thus providing both the notes and structure.

> "The algorithm contains variables that a user can change in real-time while the sample is playing"

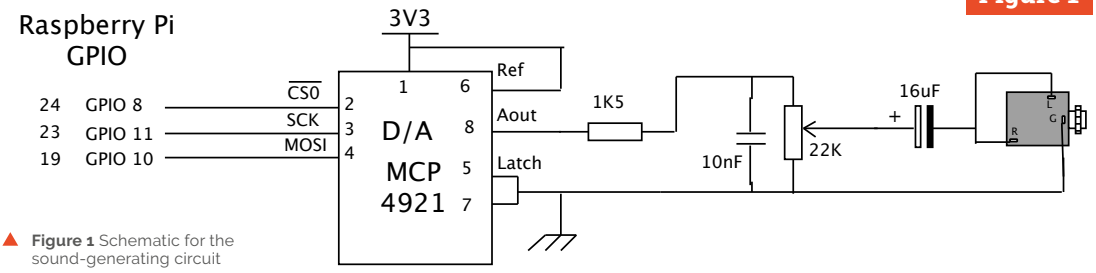
03 Enter the 'Glitch Storm'

With a Glitch Storm, three user-controlled variables – a, b, and c – can be added to this algorithm, allowing the results to be fine-tuned. In the 'Algorithms' box, you can see that the bytebeat algorithms simply run; they all repeat after a certain time, but this time can be long, in the order of hours for some. A Glitch Storm algorithm, on the other hand, contains variables that a user can change in real-time while the sample is playing. This exactly what we can

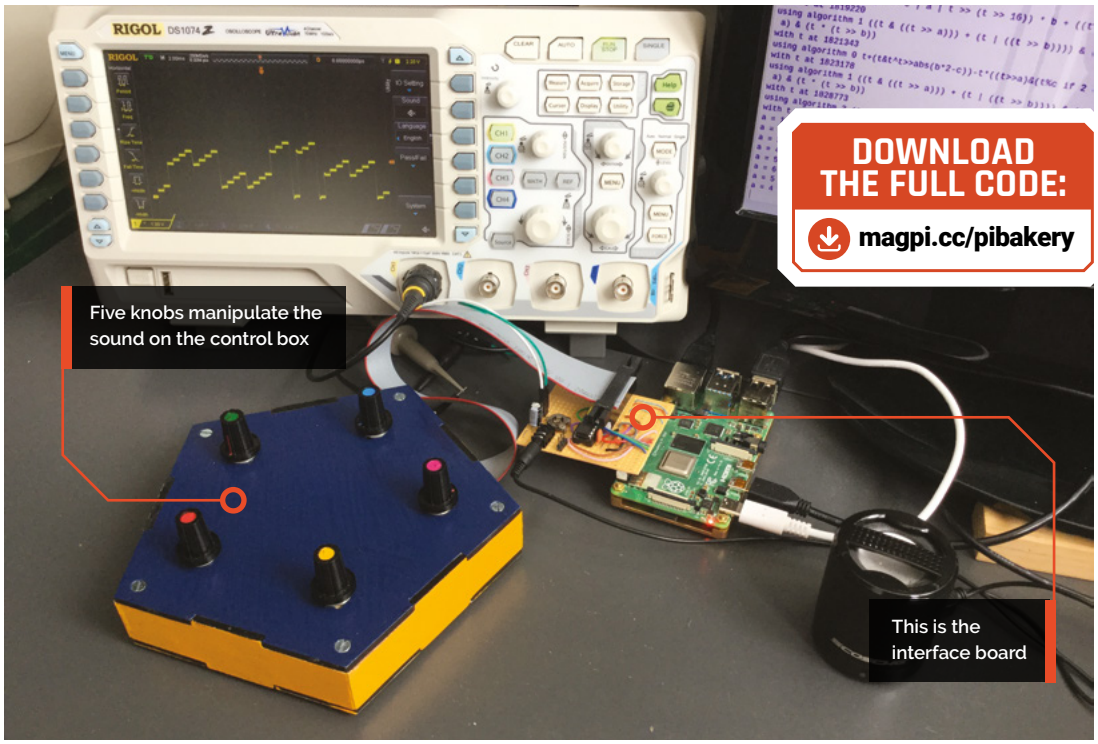
You'll Need

- MCP4921 D/A converter **magpi.cc/mcp4921**
- 5 × KY-040 rotary switches, with nuts **magpi.cc/rotary**
- 3.5 mm stereo jack socket **magpi.cc/stereopcb**

Figure 1



▲ **Figure 1** Schematic for the sound-generating circuit



THE MAGPI

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

Algorithms

After each sample is calculated, t is incremented.

Bytebeat

```
Sample = t  # this produces a simple sawtooth wave
Sample = t & t >> 8 # a minimal Sierpinski harmony
Sample = t * (42 & t >> 10) # "the 42 melody"
```

Glitch Storm

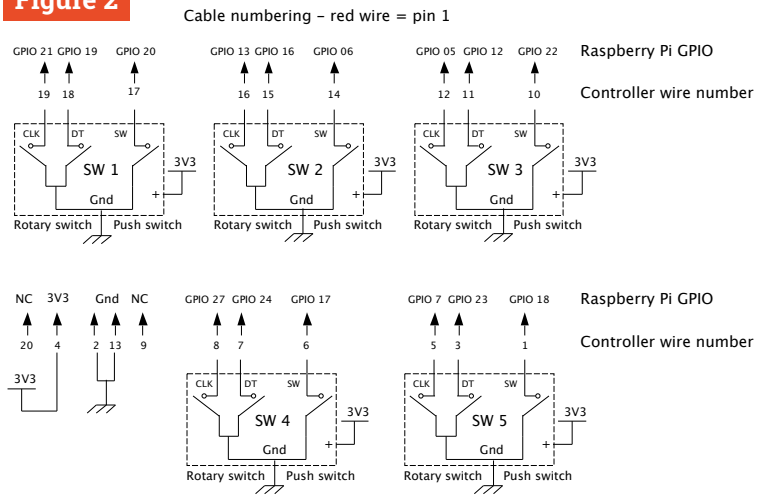
```
Sample = t * (t >> a) & (b * t >> 7) & (8 * t >> c)
```

do with rotary encoders, without having the algorithm interrupted by checking the state of them all the time.

04 What hardware?

In order to produce music like this on the Raspberry Pi, we need some extra hardware to generate the sound samples, and also a bunch of rotary encoders to control things. The samples are produced by using a 12-bit A/D converter connected to one of the SPI ports. The schematic of this is shown in **Figure 1**. The clock rate for the transfer of data to this can be controlled and provides a simple way of controlling, to some extent, the sample rate of the sound. **Figure 2** shows the wiring diagram of the five rotary encoders we used.

Figure 2

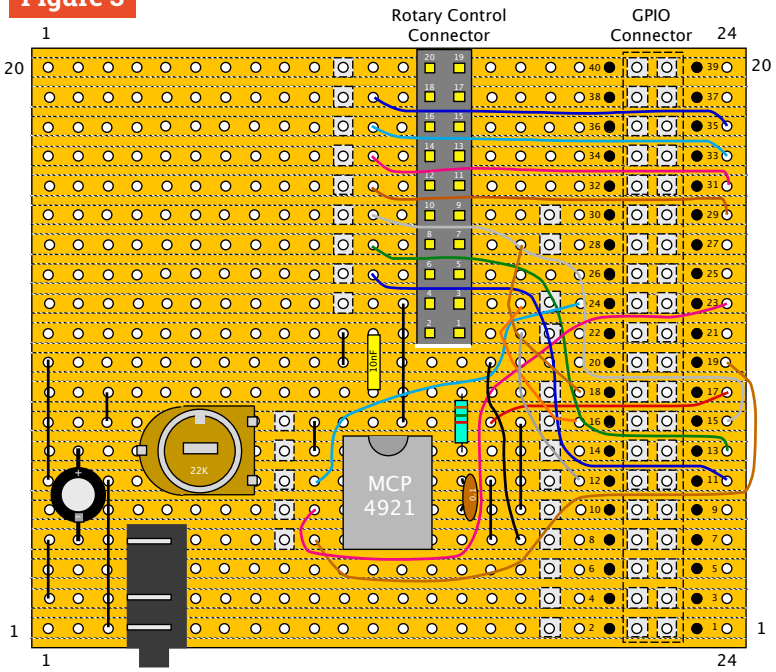


▲ **Figure 2** Schematic for the control box

05 Making the hardware

The hardware comes as two parts: the D/A converter and associated audio components. These are built on a board that hangs off Raspberry Pi's GPIO pins. Also on this board is a socket that carries the wires to the control box. We used an IDC (insulation displacement connector) to connect between the board and the box, as we wanted the D/A connection wires to be as short as possible because they carry a high frequency signal. We used a pentagonal box just for fun, with a control in each corner, but the box shape is not important here. →

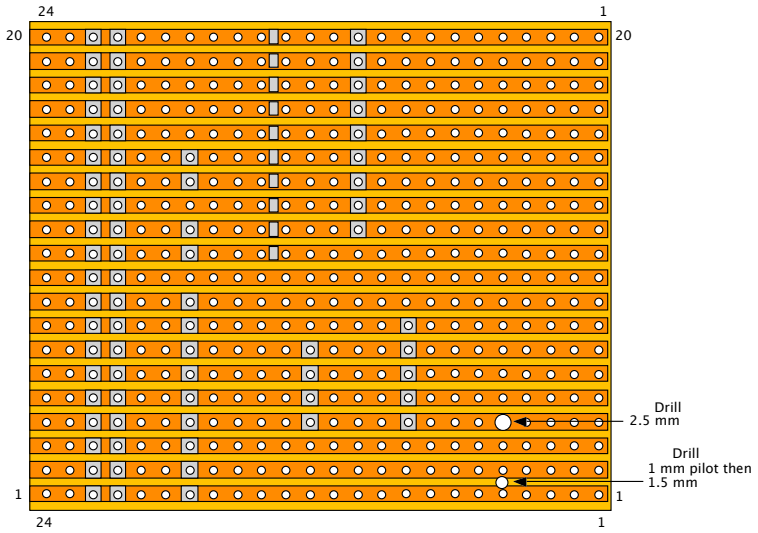
Figure 3



▲ **Figure 3** Front physical layout of the interface board

> The knobs control the user variables as well as the sample rate and what algorithm to use

Figure 4



▲ **Figure 4** Rear physical layout of the interface board

06 Construction

The board is built on a 20-row by 24-hole piece of stripboard. **Figure 3** and **Figure 4** show the physical layout for the front and back of the board. The hole number 5 on row 4 is enlarged to 2.5 mm and a new hole is drilled between rows 1 and 2 to accommodate the audio jack socket. A 40-way surface-mount socket connector is soldered to the back of the board, and a 20-way socket is soldered to the front. You could miss this out and wire the 20-way ribbon cable direct to the holes in these positions if you want to economise.

07 Further construction notes

Note: as always, the physical layout diagram shows where the wires go, not necessarily the route they will take. Here, we don't want wires crossing the 20-way connector, so the upper four wires use 30 AWG Kynar wire to pop under the connector and out through a track hole, without soldering, on the other side. When putting the 20-way IDC pin connector on the ribbon cable, make sure the red end connector wire is connected to the pin next to the downward-pointing triangle on the pin connector. **Figure 5** shows a photograph of the control box wiring.

08 Testing the D/A

The **live_byte_beat.py** listing on GitHub is a minimal program for trying out a bytebeat algorithm. It will play until stopped by pressing **CTRL+C**. The variable `v` holds the value of the sample, which is then transferred to the D/A over SPI in two bytes. The format of these two bytes is shown in **Figure 6**, along with how we have to manipulate `v` to achieve an 8-bit or 12-bit sample output. Note that all algorithms were designed for an 8-bit sample size, and using 12 bits is a free bonus here: it does sound radically different, and not always in a good way.

09 The main software

The main software for this project is on our GitHub page (**magpi.cc/pibakery**), and contains 24 Pythonised algorithms. The knobs control the user variables as well as the sample rate and what algorithm to use. You can add extra algorithms, but if you are searching online for them, you will

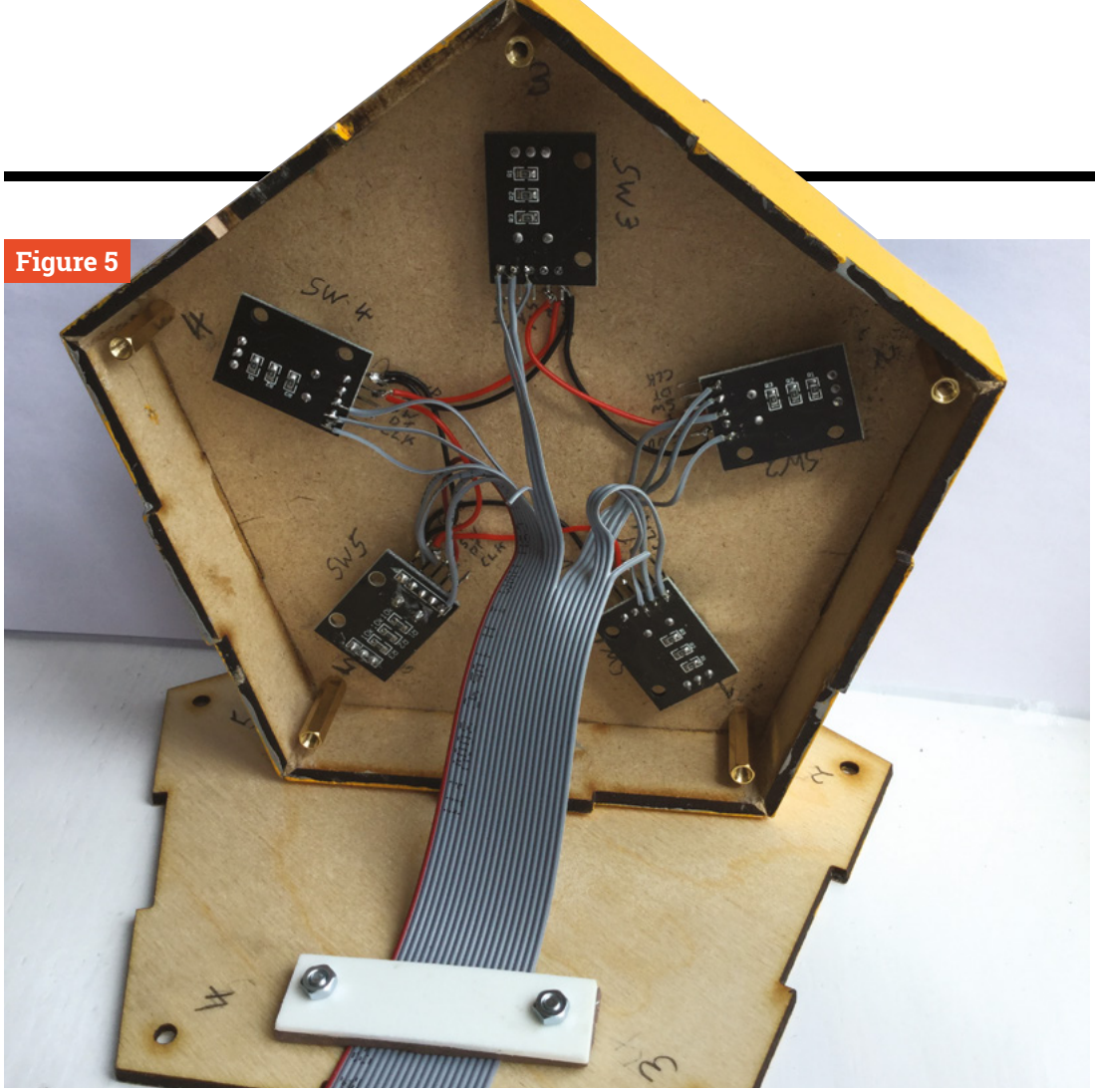


Figure 5

Top Tip

Ribbon cable connector problems

If you get red wire in the wrong pin, you will have to compensate by wiring the encoders differently. A revised schematic for this is on our GitHub page.

Figure 5 Wiring of the control board

Figure 6 How to program the registers in the D/A converter

find they are written in C. There are two major differences you need to note when converting from C to Python. The first is the ternary operation which in C is a question mark, and the second is the modulus operator with a percent sign. See the notes that accompany the main code about these.

10 Why does this work?

There are a few reasons why you would not expect this to work on a Raspberry Pi in Python. The most obvious being that of the interruptions made by the operating system, regularly interrupting the flow of output samples. Well, it turns out that this is not as bad as you might fear, and the extra 'noise' this causes is at a low level and is masked by the glitchy nature of the sound. As Python is an interpreted language, it is just about fast enough to give an adequate sample rate on a Raspberry Pi 4.

Make some noise

You can now explore the wide range of algorithms for generating a Glitch Storm and interact with the sound. On our GitHub page there's a list of useful

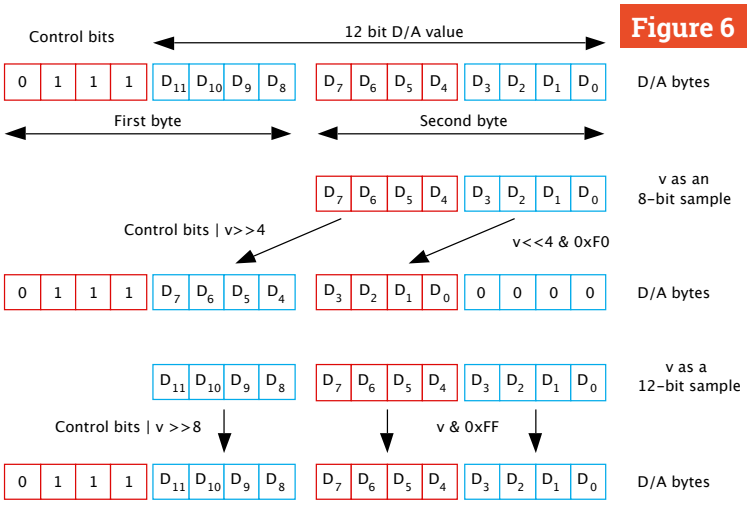


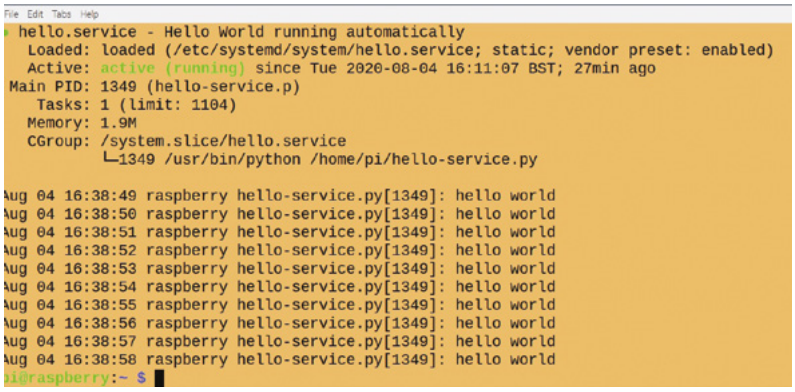
Figure 6

links allowing you to explore what others have done so far. For a sneak preview of the bytebeat type of sound, visit **magpi.cc/bytebeatdemo**; you can even add your own algorithms here. For interaction, however, there's no substitute for having your own hardware. The best settings are often found by making small adjustments and listening to the long-term effects – some algorithms surprise you about a minute or two into a sequence by changing dramatically.

TUTORIAL

Run your Raspberry Pi code automatically

Turning a script into a service



Ben Everard

@ben_everard

Ben is a former sysadmin who's spent longer than he cares to admit fiddling with running services on Linux boxes

Above
You can easily access the status and output of your script from the command line, either via a screen or a remote session (such as SSH)

So, you've written some code for your Raspberry Pi that runs your project. You need to run it every time your system starts, but how do you do this? Perhaps the easiest method is adding a line to the **/etc/rc.local** file. This is a Bash script that runs each time the computer boots up. It's quick and easy, but there are a few drawbacks to it, namely:

- If your script outputs any information or errors, where do these errors go?
- If your script crashes, how do you know, and how do you restart it?
- Can you stop or reload your script easily?

Fortunately, the operating system used by Raspberry Pi computers (Linux) has a mechanism for managing things in the background: systemd. This is a service layer that's widely used to manage all sorts of bits of essential software on your Raspberry Pi or other Linux computer, from databases to window managers. It provides a way of controlling software that runs in the background. Let's take a look at what this means in practice.

Systemd can handle a huge range of things, but we'll be looking at 'services' which is the name used for software that runs in the background because it typically provides a service.

Our really simple script will simply use Python to output the phrase 'hello world' every second. This is, admittedly not the most useful script, but it'll show the technique of running our code automatically, and capturing the output.

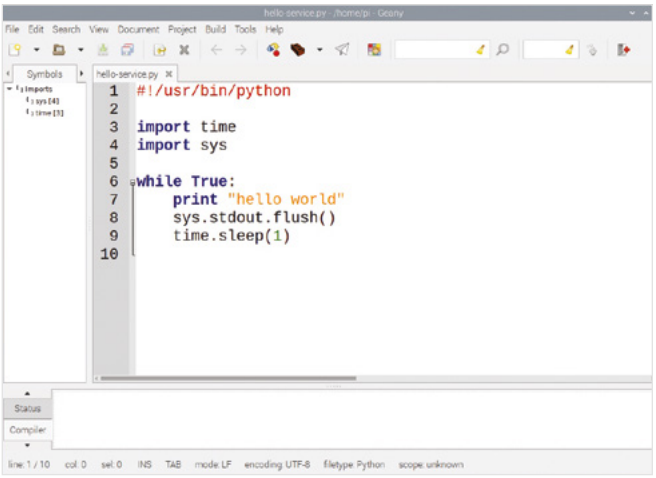
This can be done with the following code:

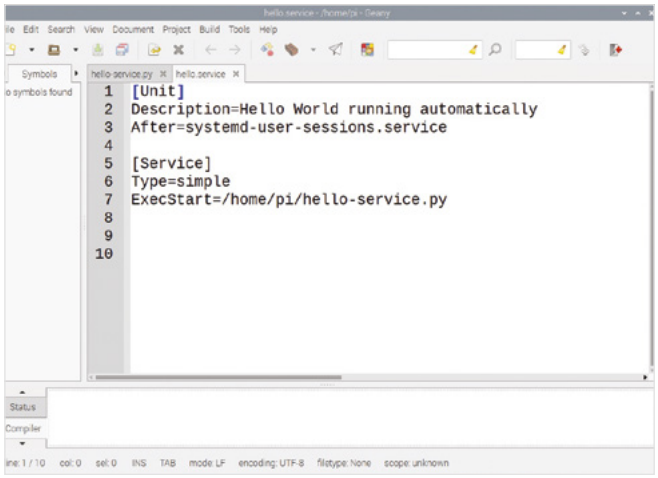
```
#!/usr/bin/python

import time
import sys

While True:
        print "hello world"
        sys.stdout.flush()
        time.sleep(1)
```

There are a couple of unusual bits in this. The first is the first line – this is known as a shebang, and it's used to tell the operating system what program to use to run the script. This is the complete path to the binary for Python. We got this on Raspbian OS. It might be a little different on other Linux distros. If you're unsure, you can run the command 'which python' and it will tell you.

The second unusual thing is the call to `sys.stdout.flush()`. This isn't strictly necessary, and it just tells the operating system to make sure all the output has made its way through the output buffers and on to its destination. We found we had a large lag between the script running and output making its way to the logs if we didn't use it. In practice, you may find that you can live without this in your programs (or just include it once in a main loop).

We saved this in a file called **hello-service.py** in **/home/pi**.





Above
Systemd service files are straightforward and easy ways of encapsulating the information needed to run your script

Left
Flushing the output means you always have the latest information

Finally, we need to make this file executable by running the command:

```
chmod a+x /home/pi/hello-service.py
```

RUN THE SCRIPT

That's our script ready. Let's now get it running as a service. In order to let systemd know what we want to do with our simple service, we need to create a unit file. There's a lot that can go into a unit file, but for basic usage, they can be quite simple. Ours will be as follows:

```
[Unit]
Description=A service to say hello world
After=systemd-user-sessions.service

[Service]
Type=simple
ExecStart=/home/pi/hello-service.py
```

This tells us a bit about our script. The `After` section tells systemd when we want our script to start – in this case, after the `systemd-user-sessions` (which is one of the system services that starts every boot).

> There's no fussing around trying to **find PIDs of processes in order to stop or restart them**

Type `simple` (as opposed to forking) tells systemd that the command will continue to run in the session it was started.

The final line tells systemd what command to run.

Save this unit file as **hello.service** in your **Home** directory, then copy it to the **systemd** directory with:

```
sudo cp /home/pi/hello.service /etc/systemd/system
```

That's everything set up and ready to go. We just need to let systemd know what we want to do.

You can start your service with:

```
systemctl start hello.service
```

see its status with:

```
systemctl status hello.service
```

and stop it with:

```
systemctl stop hello.service
```

It's this management of the running code that makes this method of running code much easier. There's no fussing around trying to find PIDs of processes in order to stop or restart them. What's more, once you start a service, systemd will monitor it, and if it crashes for some reason, it will attempt to restart it. Obviously, this isn't a reason to create code that's unstable, but it is an extra line of defence if your code has to run all day by itself.

If you want to see the output of the service, you can use `journalctl`:

```
journalctl -u hello -e
```

What we've done so far will get the service up and running, but it won't automatically start it every time you boot the system. For that, you need to enable the service with:

```
systemctl enable hello.service
```

With this done, you can restart your machine, and it will automatically start. If you want to stop it starting automatically, you can disable it with:

```
systemctl disable hello.service
```

There's far more to systemd than we've looked at here, but with these basics, you can get your code up and running and make sure it's looked after properly as it whirs away in the background.

HackSpace
TECHNOLOGY IN YOUR HANDS

FIELD TEST

HACK | MAKE | BUILD | CREATE

Hacker gear poked, prodded, taken apart, and investigated

ONLY THE BEST

When you need to keep it small

Accessories for your Raspberry Pi Zero

By **Marc de Vinck** @devinck

The Raspberry Pi Zero and Zero W are diminutively small, as far as single-board computers go. This author still remembers the day he plugged one in, hooked up a monitor, keyboard, and mouse for the first time. The OS started to boot up and showed a lot of scrolling lines of text. After a few quick configurations, I had a working computer. Not a microcontroller, but a real working computer, running a decent operating system! It's amazing to think about just how small Raspberry Pi Zero really is compared to its predecessors – and maybe even its contemporaries! Even more impressive is the price, which hovers just around the cost of a pizza.

Most of us have come to know and love the Raspberry Pi platform. We've used it, understand it, and it has become an essentia ltool that we use. I recently had some friends over, and they were interested in what I was working on: a Raspberry Pi Zero running RetroPie. For those of you who don't know, RetroPie allows you to run classic video games on Raspberry Pi. The look on their faces when they saw a table with just a Raspberry Pi Zero sitting there plugged into a monitor, running games smoothly, just like they did in the 1980s, was priceless. It brought back that feeling I had the first time I used a Raspberry Pi Zero.

They couldn't believe such a small 'thing' could run such complicated software so well. Yeah, we all know retro games from the 1980s aren't really complicated to run, but I didn't want to get into all the details! However, I did have a little fun when I also showed them that it was running a keyboard and mouse via Bluetooth, and it had built-in WiFi. It blew their minds, and hopefully I ignited a spark of curiosity. Wait until they learn about Raspberry Pi 4!

> **Most of us have come to know and** love the Raspberry Pi platform – it's an essential tool

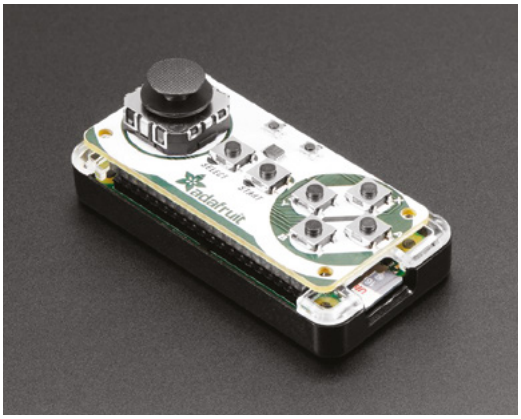
In this Best of Breed, we'll be taking a look at some of our favourite – and small – Raspberry Pi Zero accessories. These are all useful accessories that won't make your project get too big physically or break the bank. They all fall within the already small form factor of Raspberry Pi Zero. Or at least close enough!

Adafruit Joy Bonnet for Raspberry Pi vs ZeroSeg

ADAFRUIT $14.95 | adafruit.com **PIHUT** $13 | thepihut.com

As we mentioned in the intro, **Raspberry Pi makes a great retro gaming system.** Yes, the new Raspberry Pi 4 can emulate a lot more advanced gaming systems, but Raspberry Pi Zero can hold its own on the classics systems from the 1980s. And what's a gaming system without a few buttons and a joystick? That's where the Adafruit Joy Bonnet for Raspberry Pi Zero comes into play.

There are times when you want a compact and portable gaming system, and Raspberry Pi Zero, coupled with the Joy Bonnet is a perfect match. Take it on the go when travelling, plug in a hotel TV via a mini HDMI cable, and you've got a pretty robust gaming system on a large TV. The Bonnet comes fully assembled and ready to go. Just add a simple case and stow it away in your bag. And if you're not into gaming, you can use the Joy Bonnet to navigate a media centre running on Raspberry Pi, or any other application where you need some basic inputs. Check out the product page to learn more about building your own Raspberry Pi-based gaming system.



Below
A computer in your game pad

VERDICT

Adafruit Joy Bonnet for Raspberry Pi

Gaming on Raspberry Pi Zero is easy with this Bonnet.

10/10

ZeroSeg

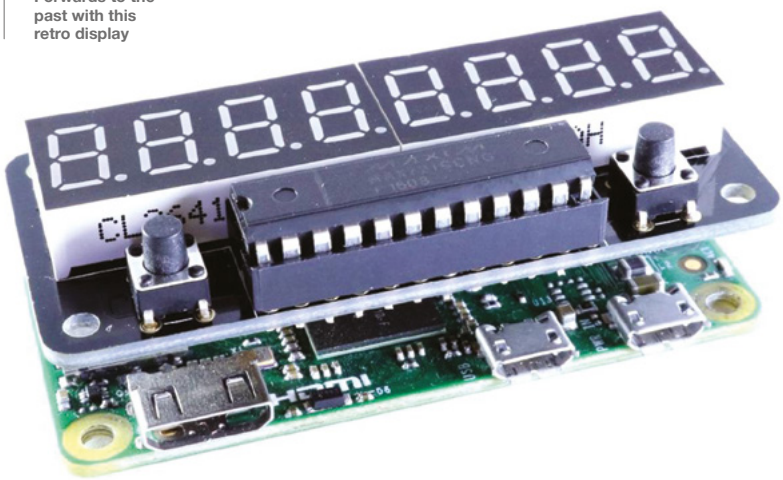
A retro-like accessory for your Raspberry Pi.

8/10

Speaking of retro fun, check out the **ZeroSeg from The Pi Hut.** It's a classic, eight-character, seven-segment display add-on board for your Raspberry Pi. It also includes two handy buttons for resetting, or any function you may need for your project. We've always loved the classic look of a red seven-segment display, and we find it kind of interesting to pair it with the latest tech like Raspberry Pi Zero or Pi Zero W.

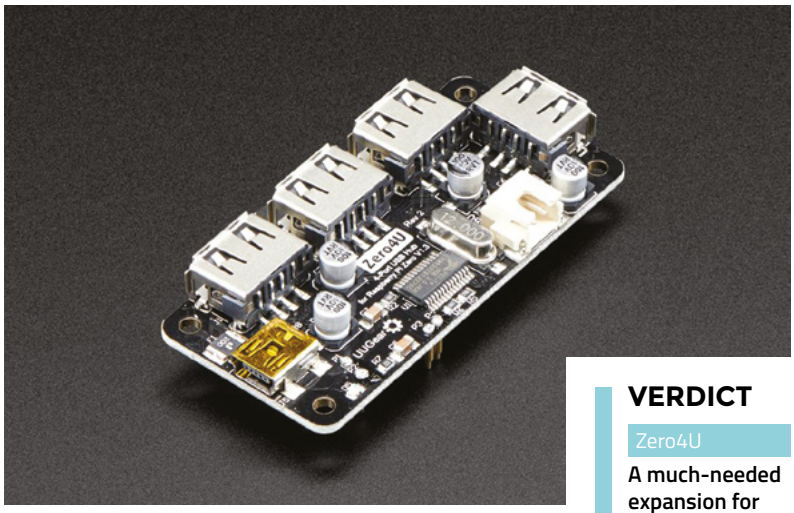
The displays are controlled via a MAX7219CNG, which handles all the grunt work for your Raspberry Pi and alleviates the need for a lot of pins. And the best part about this board is that it handles the data from your Raspberry Pi in a traditional way, so most existing code for a seven-segment display will work without many changes. The Pi Hut has a full code library and example code to get you up and running retro-fast with the ZeroSeg. →

Below
Forwards to the past with this retro display



Zero4U

ADAFRUIT ◈ **$9.95** | **adafruit.com**



The tiny size of the Raspberry Pi Zero means that there's not much space for connectors and USB is a bit limited. For many projects, this won't be a problem, but sometimes you need just a few more peripherals. This is where the Zero4U really saves the day. By simply plugging the board into your Raspberry Pi Zero, you instantly add a convenient four-port USB hub.

The Zero4U gets its power directly from your Raspberry Pi Zero, so you won't need to add an extra power supply. Now when you go to add a keyboard, mouse, or other USB accessories into your tiny Raspberry Pi project, you'll be ready to go, thanks to the Zero4U. Be sure to check out the product page over at Adafruit to make sure it's compatible with your specific version of Raspberry Pi Zero.

VERDICT

Zero4U

A much-needed expansion for your Raspberry Pi Zero.

10/10

Zero Spy Camera for Raspberry Pi Zero

ADAFRUIT ◈ **$19.95** | **adafruit.com**

The Raspberry Pi Zero has a lot of power for such a small form factor; Enough horsepower to even run a full video camera system. And although most cameras for Raspberry Pi ecosystems are small, the Zero Spy Camera from Adafruit is extra small and makes a perfect companion for Raspberry Pi Zero. The module itself measures in at only 8.6mm × 8.6mm, and the attached ribbon cable is only 2" (52mm) long, making for a compact little camera system.

Because of the small size of Raspberry Pi Zero, and the fact that you don't need anything other than this camera module itself for it to capture video, you can easily make a very compact security system. Of course, you'll have to figure out a power supply, but that isn't too difficult thanks to a convenient USB plug on the Raspberry Pi board itself. And if you use this camera with a Raspberry Pi Zero W, you can even add some wireless connectivity and remote control.

Above ◈
No need to skimp on the USB ports

Left ◈
A tiny camera for a tiny computer

VERDICT

Zero Spy Camera for Raspberry Pi Zero

A small camera for a small board.

8/10

Inky pHAT

PIMORONI ◈ **$18.34** | **pimoroni.com**

Left ◈
Displays need not hog lots of power

Below ◈
Displays and buttons all in one handy package

Just because Raspberry Pi Zero is small doesn't mean you can't have a big display. The Inky pHAT from Pimoroni is a Raspberry Pi-sized, energy-efficient e-ink display. It's available in three different colour combinations: black/white/red, black/white/yellow, and a simple black and white. The screen features a 212×104 pixel area, allowing for crisp text and graphics. And, since it is an e-ink display, it's very sunlight-friendly and a great choice for any outdoor project.

And speaking of outdoors, where power can be limited, the Inky pHAT only sips about 8 mA power when refreshing the contents of the display, so a battery will last a long time compared to a traditional LCD or TFT display. Just keep in mind the refresh of the screen can take up to 15 seconds, so don't expect to run any animations or videos!

Pimoroni provides a great 'getting started with Inky pHAT' tutorial, featuring lots of code examples and everything you need to get started. Head on over to the product page to see more examples of the available colours and the tutorials.

VERDICT

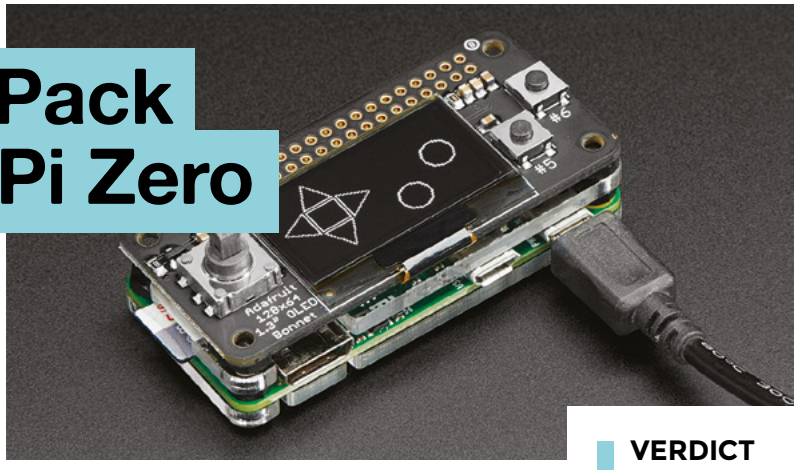
Inky pHAT

Available in multiple colour combos and at a great price.

10/10

OLED Bonnet Pack for Raspberry Pi Zero

ADAFRUIT ◈ **$34.95** | **adafruit.com**



The OLED Bonnet Pack for Raspberry Pi Zero is a great little pack collection of components that everyone could use. Not only do you get the Adafruit 128×64 OLED Bonnet, but you also get a Raspberry Pi Zero W and enclosure. This pack of parts can do a lot right out of the box!

The OLED Bonnet includes a very readable 128×64 pixel OLED display, along with a five-way joystick and two push-buttons. Now you can easily include navigation and inputs to your Raspberry Pi project. Adding an interface with multiple menus couldn't be easier thanks to the example code and libraries. Adafruit has been able to get up to 15 frames per second, so you should be able to add some fun little animations too!

Also included is a Raspberry Pi Zero W, which has built-in wireless features and everything else you love about the Raspberry Pi ecosystem. And last but not least, the kit also includes a nice little case, allowing for a much better hand-held experience, and it provides some protection too! If you don't have a Raspberry Pi Zero W yet, this would make a great little bundle to add to your collection. →

VERDICT

OLED Bonnet Pack for Raspberry Pi Zero

All you need for a quick project, including a Raspberry Pi Zero!

10/10

Flirc Raspberry Pi Zero Case

FLIRC $12.95 | sparkfun.com

At first glance, you might be thinking that you have seen smaller cases than the Flirc Raspberry Pi Zero Case, and you'd be correct. And if you thought this was supposed to be all about the smallest of small accessories for Raspberry Pi Zero, you'd also be correct. But in this case, no pun intended, we bent the rules. Why? Heat. That's why!

Raspberry Pi Zero can run a little warm if you are crunching a lot of numbers, and this case helps dissipate all that heat. The case is made from aluminium, and it acts as a large heat sink. Other cases might allow you to add a heat sink, but this one is integrated, and it does a great job.

Another nice feature, other than its good looks, is the fact that you have access to most of the ports of Raspberry Pi Zero (not the microSD card). So, in situations where you need a little extra security, you don't have to worry about someone grabbing and removing the card. And yes, in other situations this could be a problem, but we find that in a lot of our projects we tend to not remove the card very often.

Another cool feature is the two different tops that are included. One allows access to the GPIO pins, and the other is solid and does not. Again, this can add a lot of extra security to the Raspberry Pi inside. We can definitely see a lot of uses for this nicely designed little enclosure, even if security isn't your top priority.

Below Keep cool without noisy fans

VERDICT

Flirc Raspberry Pi Zero Case

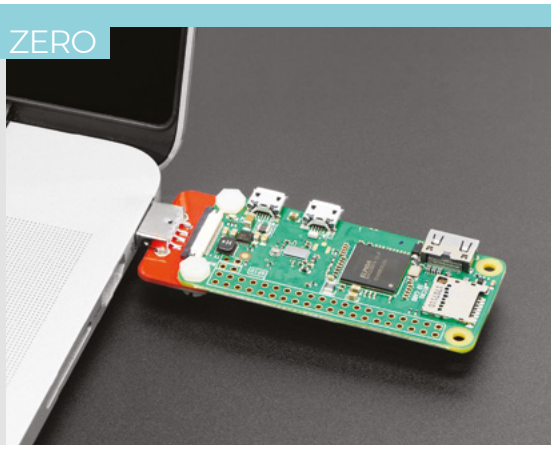
Rugged with added features and security.

9/10

ZERO STEM FOR RASPBERRY PI ZERO

ADAFRUIT $5.95 | **adafruit.com**

Adding a full-size USB port to your Raspberry Pi Zero couldn't be more compact. The Zero Stem for Raspberry Pi Zero from Adafruit allows you to connect Raspberry Pi to a USB port, and it can act as a USB dongle. It requires a little bit of soldering, and only works with the latest versions of Raspberry Pi Zero, so check the website before you buy one for yourself. If you need to grab power or data via USB and your computer, this might be a very handy little accessory.



GET STARTED WITH ARDUINO®

Robots, musical instruments, smart displays and more

£10 with FREE worldwide shipping

Inside:

- Build a four-legged walking robot
- Create a Tetris-inspired clock
- Grow veg with hydroponics
- And much more!

AVAILABLE NOW hsmag.cc/store

plus all good newsagents and:

WHSmith BARNES&NOBLE

Available on the App Store

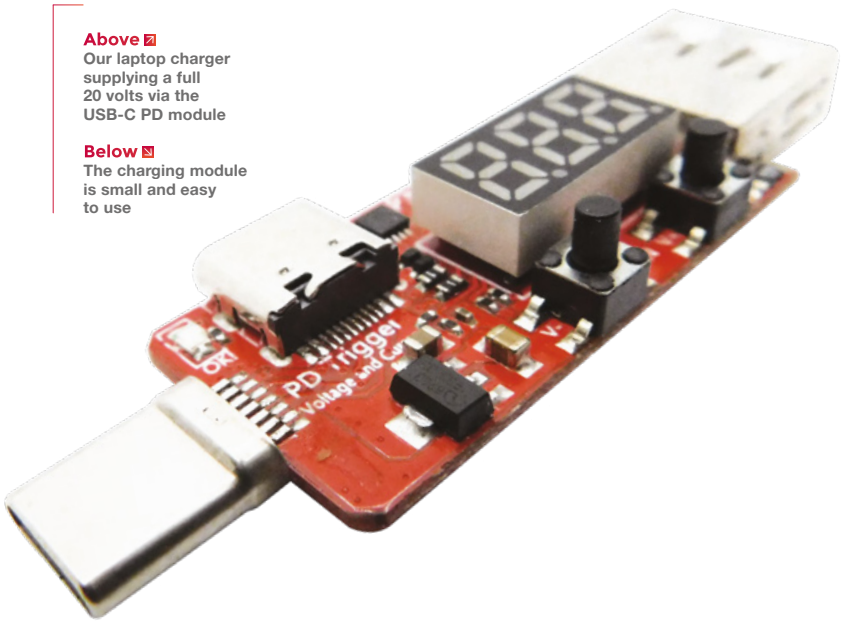
GET IT ON Google Play

FROM THE MAKERS OF HackSpace MAGAZINE

USB-C power supply

Get flexible power from this universal standard

By Ben Everard @ben_everard



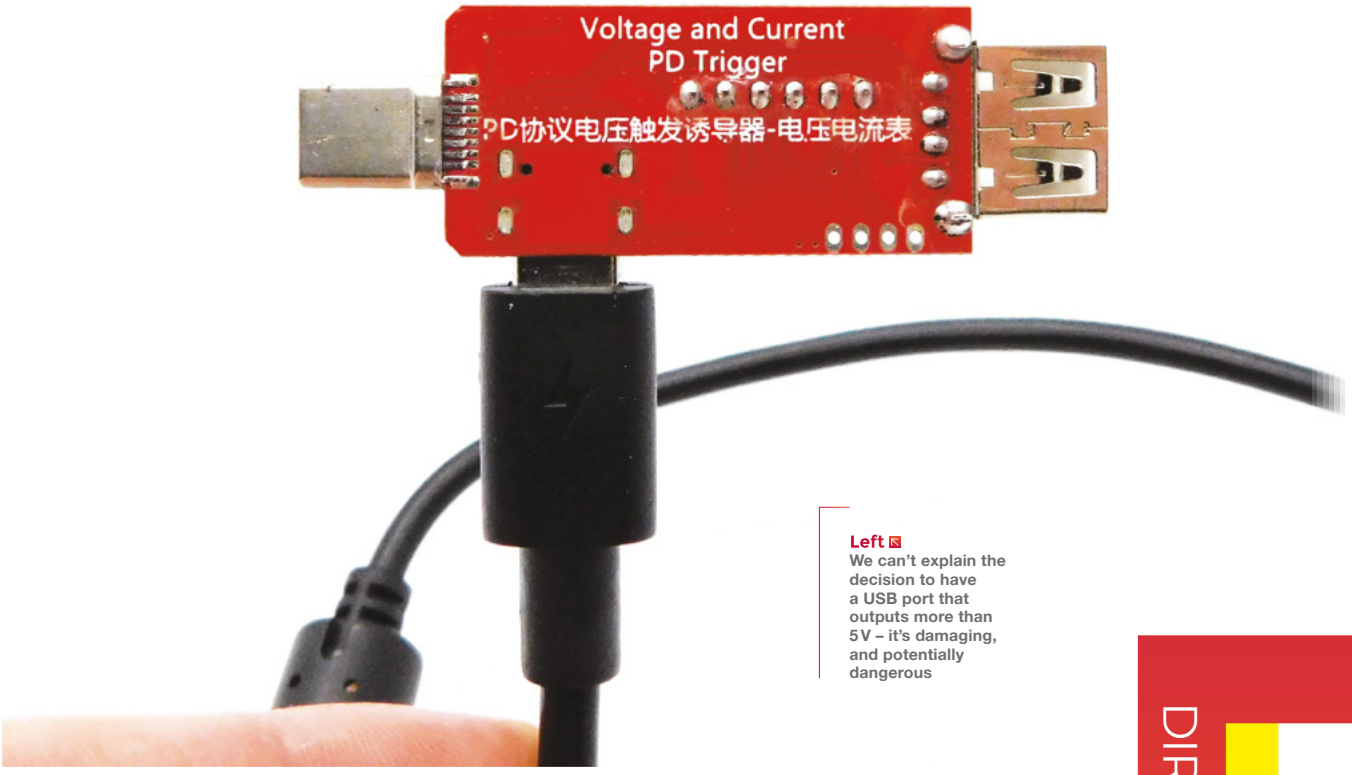
Above Our laptop charger supplying a full 20 volts via the USB-C PD module

Below The charging module is small and easy to use

Getting electricity into your projects can be surprisingly challenging. For low voltages and small currents, USB has become a very common standard. It gives you 5 V and, if you're careful with your power supply, you can probably get about 2 A. Most people have a few of these power supplies hanging around and, if you need to take your project on the road, then there are lots of rechargeable power banks available at different capacities.

USB-C has the potential to provide much more power through something called Power Delivery (PD). When you first plug a USB-C device into a power supply, the power supply will send over 5 V. This should be enough to run the power supply circuity and get everything started. Your device can then request more power. Not all USB-C power supplies can supply more than 5 V, but your device can negotiate to receive up to 5 A at 20 V, if the supply can handle it. This is potentially a lot more flexible, but also requires some more sophisticated electronics than just a power socket.

Fortunately, there are off-the-shelf modules that help do the hard work for you. We tested out a PD 3.0 protocol current voltage trigger that came from Banggood (**hsmag.cc/cr0UR2**). For £9.36, we got the board and a USB-to-barrel-jack cable, including delivery to the UK.



Left
We can't explain the decision to have a USB port that outputs more than 5V – it's damaging, and potentially dangerous

DIRECT FROM **SHENZHEN**

The first thing to say about this board is that the output is via a regular USB port, but it outputs the full voltage from USB-C PD (up to 20V). In other words, **DO NOT PLUG ANYTHING INTO THIS USB PORT THAT IS EXPECTING A STANDARD USB VOLTAGE.** You will damage it, and you might set it on fire. Why the designers chose this as an output connector, rather than having a barrel jack, is a mystery to us.

That warning in place, let's take a look at what the module actually does.

There are both male and female USB-C ports, but they both work in the same way, and you can supply power via either one. Having both is handy, as it means you can connect it directly to either a computer or a power supply. Once the board is powered up, the three seven-segment displays light up, displaying the voltage. There are two buttons on the board: one requests a higher voltage, the other a lower voltage. These allow you to scroll through the available power profiles.

In this usage, you can select the power you need for your project and get it to come through a compatible USB-C device. In theory, it can supply up to 100 watts (20V, 5A); however, we would have some concerns about running this much power through the device. If you want to push the limits of USB-C, then we'd strongly recommend a module from a reputable manufacturer (there's not even manufacturer's information with this board).

This module works well for prototyping, but it would be nice if there was a way to set the voltage a little more permanently. It does remember the power profiles between power-offs, but if anything taps the button, it'll shift the power. In principle, you could remove the buttons once you had the power profile you want selected. It would also be nice if there were a digital interface to this so you could interact with it from a microcontroller, but we've only seen this capability in more expensive boards.

> **You'll probably want a more robust solution** if you're planning on using the power for any length of time

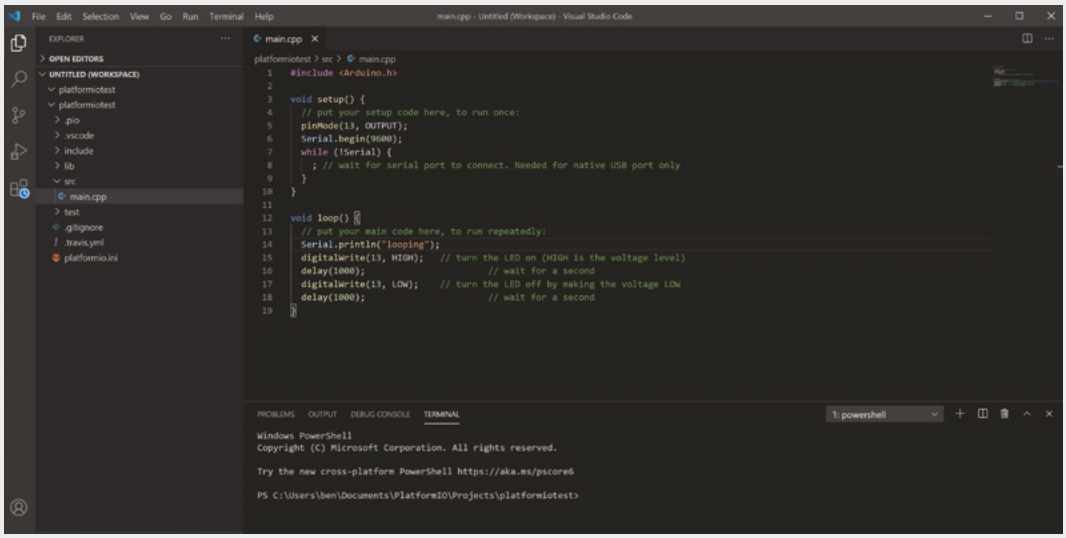
If you have a spare USB-C power supply, or USB-C portable power bank, this is a cheap way of accessing its power modes for prototypes, but you'll probably want a more robust solution if you're planning on using the power for any length of time. ◻

PlatformIO

Automatically configure your development environment

PLATFORMIO Free (or paid support) | platformio.org

By **Ben Everard** @ben_everard



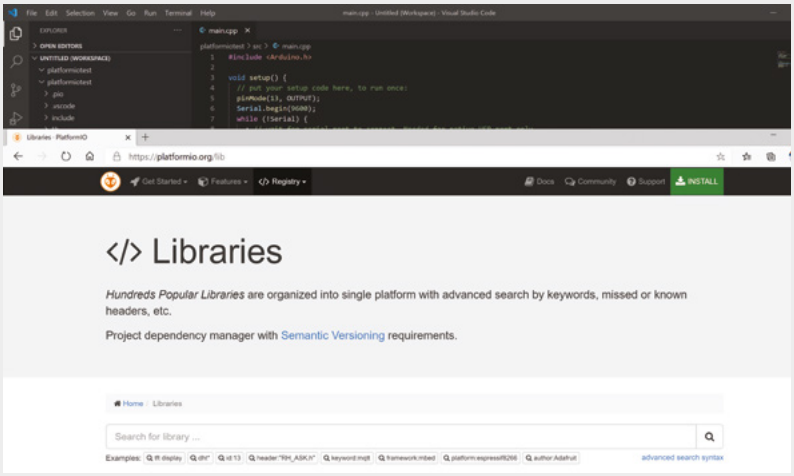
Right
PlatformIO running on Visual Studio Code gives you plenty of power, but still keeps things reasonably simple

PlatformIO is a little hard to describe, as it falls between the gaps in the language we use to describe development tools. It's not an integrated development environment (IDE), but it is a plug-in for many popular IDEs (such as VS Code and Atom). It's not a framework, but it does package up popular embedded development frameworks (such as Arduino). According to the documentation: "PlatformIO is a cross-platform, cross-architecture, multiple framework, professional tool for embedded systems engineers and for software developers who write applications for embedded products." However, that tells us very little about what you'd use it for.

The best way we can describe it is a wrapper that lets you connect different frameworks (which include things like compilers and core libraries), board definitions, and libraries with development environments (that help you manage the source code). On top of this, there are also some tools designed for embedded development such as remote device management, unit testing, and debugging.

Essentially, it lets you pick your development environment and turn it into an embedded development environment. We tested this out with VS Code, but it will also work with a wide range of others including Eclipse, Vim, Emacs, and Cloud9. There is also a command line version if you want more flexibility.

Once it's installed, you can create new projects. For each project you select the board and framework you're using and PlatformIO will scurry off to the internet and fetch all the bits you need to compile and run it. There's also a built-in library manager for installing the various bits of additional code your project needs.



Above
The libraries manager gives you access to a mind-boggling amount of code to add to your projects

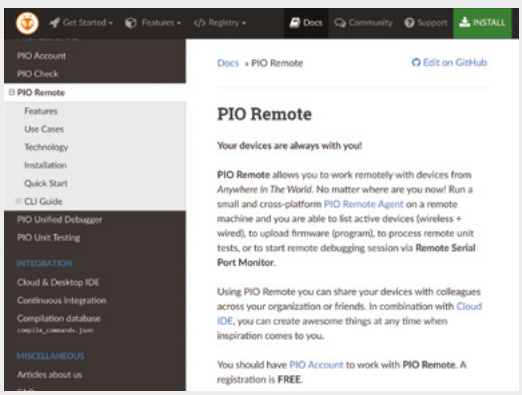
Below
The thorough documentation makes it easy to get up and running with PlatformIO

At this level, PlatformIO is a competent development environment that offers a unified approach for working with a wide range of platforms. This is great if you find yourself skipping between different technologies for different projects. However, there is a deeper level of PlatformIO, offering more advanced features for people working on larger projects.

PlatformIO Remote allows you to control embedded devices from anywhere in the world, with one caveat – they must be connected to a computer capable of running PlatformIO Remote Agent. The easiest way of running this is on a Raspberry Pi, but other Linux or Windows computers also work. PlatformIO lets you use this machine as a proxy for uploading code to, or interacting with, connected microcontrollers. This, for example, lets you create a build farm for easily testing out code on many devices without having to have all of them physically with you.

If your board supports it, you can use many of the popular debugging tools with PlatformIO in a unified way. As with other parts of PlatformIO, you are presented with a standard interface regardless of what hardware you're using. You can do the standard things that you'd expect with a debugger such as watch variables, set breakpoints, view memory, and more.

The unit testing framework lets you easily and automatically test parts of your code on actual hardware. It works in a broadly similar way to other common unit testing frameworks such as JUnit and PyUnit. You set tests that define code that should run, and then create assertions that define things that should or shouldn't be true after they're run. For example, the code may use a temperature sensor and check that it received back a valid temperature.



PlatformIO will then run some or all of the tests and check the assertions after each one. As such, you can use it to quickly check that any changes you make to either software or hardware don't accidentally break any functionality.

PlatformIO is a really useful resource. It's a little more complex than some beginner-focused development environments, but in return, you get a huge amount of flexibility. You can select the development environment you prefer, and PlatformIO will do the rest. It's this flexibility of using the powerful core that stands PlatformIO above its competition. Some people prefer really stripped down environments like Vim, some prefer more fully featured tools such as Eclipse, and some prefer something in the middle like VS Code. For some people, it makes sense to use browser-based tools such as Cloud9. All of these people are catered for. Sure, it's possible to set up these as embedded development environments without PlatformIO, but each time you change the framework of the board you're working with, you'll have to manually configure your toolchain and that – in our view at least – is the least enjoyable part of development (other than perhaps spending hours tracking down a bug only to find that it was a missing semicolon).

> You can select the development environment you prefer, and PlatformIO **will do the rest**

Even if you're working across different frameworks and toolchains, with PlatformIO, your setup is just a few clicks away.

The more powerful features such as remote access and unit testing are perhaps a bit less relevant to hobbyists, but if you do need them, they're there to make your life easier. All in all, PlatformIO is a fantastic option for C and C++ development unless, perhaps, you're a complete beginner.

VERDICT

Easy embedded development in almost any development environment.

9/10

REVIEW

Tenma 72-10480

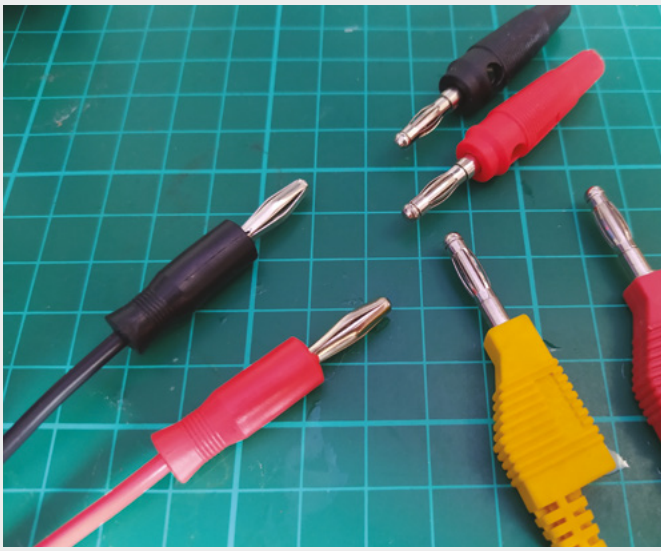
Get electrons into your circuits safely and reliably

TENMA ◈ **£62.21** | **hsmag.cc/Ih2bJe**

By **Jo Hinchliffe** @concreted0g

This Tenma benchtop power supply has a variable voltage output up to 30 V DC and can supply up to 3 amps, which is a capable range in terms of average electronics circuit work, only excluding those working with high-current consuming devices, such as larger DC motors. It arrived well-packed and very quickly from CPC Farnell. Opening the box, we first noticed a European plug/lead and thought we might be off to a tricky start, but delving further into the box, we were relieved to see a UK lead as well. Apart from the power leads, it's supplied without any connecting cables. The output connections on the front of the unit – positive, negative, and earth – all accept standard 4 mm banana jacks, so many people will have some cables already. Indeed, we had a collection of various leads, including a pair terminated with some crocodile clips which were perfect for testing. With a small instruction manual overviewing the unit, that's pretty much all there is in the box.

> On startup, it defaults to the first memory slot and **doesn't supply any power to the outputs**

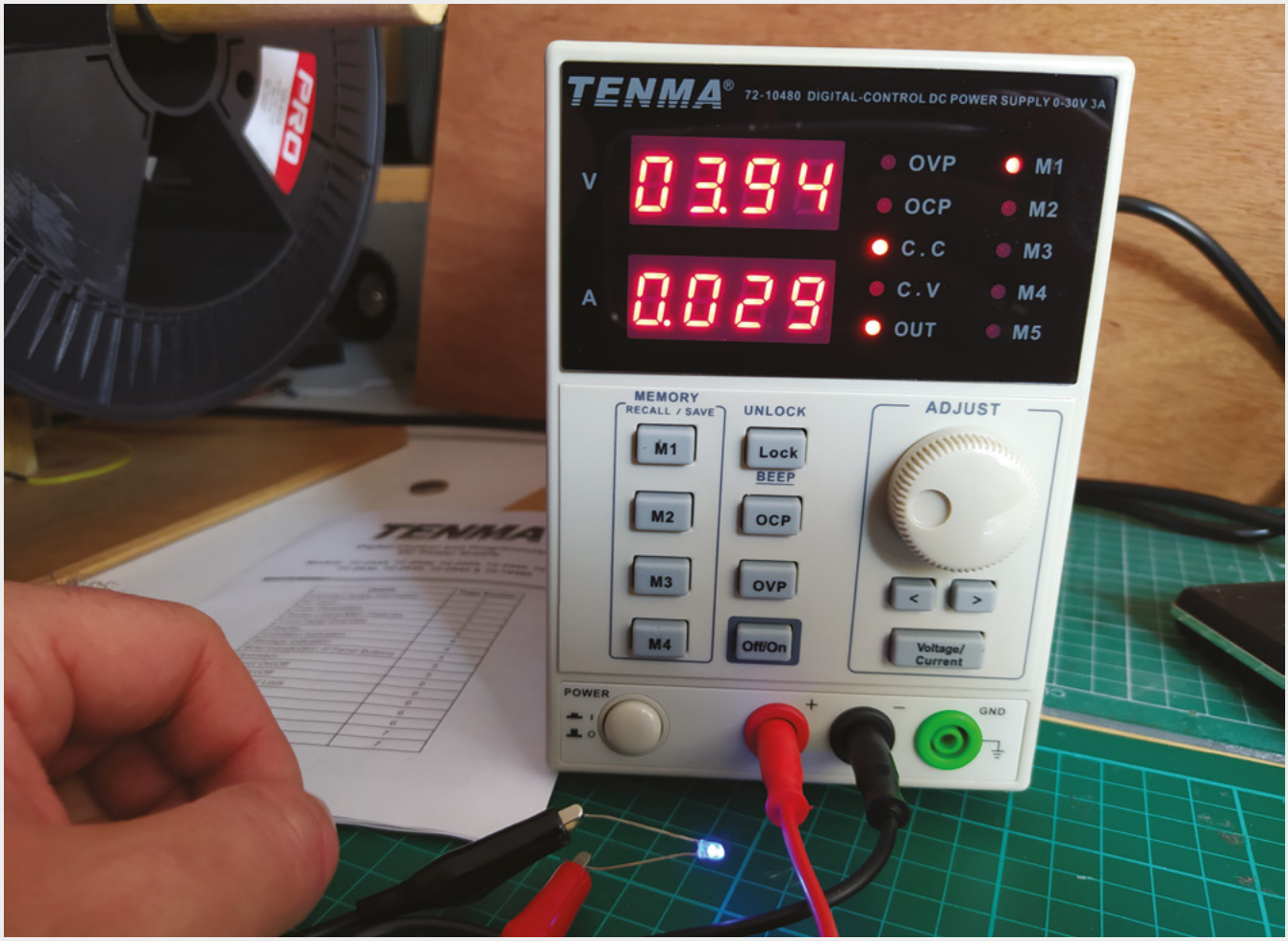


Above ◈
The unit isn't supplied with any connectors, but different styles of 4 mm banana jack cables are widely available

Setting it on the desk, it feels like a solid unit but remains compact and doesn't eat up much desk space. The buttons and dial feel like good-quality items, with a nice positive feel to them. A press of the power button on the front and it kicks into life, with a small click and, whilst there is a little fan noise, it's not a lot and is tolerable even for longer periods of work. Whilst we didn't have a need to use the earth connection on the front panel, we did check it for continuity. With the unit powered down, we used a multimeter set to continuity mode to check that the unit was earthed, and we can confirm that the chassis and case, and the earth connector on the front panel, are all contiguous to the earth pin on the plug.

On startup, it defaults to the first memory slot and doesn't supply any power to the outputs. The output is enabled and disabled by the output switch. There are five memory slots, with the first four being accessed by pressing a corresponding hardware button labelled M1–M4. The fifth memory slot is only accessible by first selecting the M4 memory slot and then turning

Right ◈
The display is clear and reasonably quick to update, with everything labelled well



the dial to switch to M5. When any change is made in any memory slot, it's automatically saved. Switching between any memory slot automatically turns off the output, so you can't accidentally fry something by switching to a different supply.

With a memory slot connected, clicking the voltage/current button underneath the dial allows you to adjust the voltage or the maximum current, and there are left and right cursor arrow keys so you can adjust the different units. It's capable of dialling in 1/1000ths of an amp and 1/100ths of a volt. The unit has a 'beep' that confirms button presses, and this can be turned on or off with a long press of the 'OCP' button. We've left it on, as it isn't overbearingly loud and is useful to confirm button presses etc.

The unit has overvoltage protection (OVP) and overcurrent protection (OCP) built in, which means that if the circuit tries to draw more than it is set to, it will disable the output. We checked the OCP system by trying to run a small DC motor we knew drew around 800 mA of peak current on startup, before settling to around 500 mA. Setting the current to 600 mA meant that it would automatically cut the supply on startup, as the current draw exceeded the threshold. Bumping up the supply to 900 mA let the motor start up, and the display quickly settled to around 500 mA. It's handy to be able to look at the current consumption of a device, and so this feature can be useful. We spent some time tinkering with how few mA a particular LED needed to actually be bright enough for the task in hand at a certain voltage, meaning we could, using the power supply only, optimise a circuit we were looking at.

We found the display simple but effective. It's pretty quick to refresh and becomes intuitive to use and read after a short time playing with the supply. The manufacturer's instructions are reasonably clear, but, as often is the case, they suffer in terms of readability at some points. It's noteworthy that CPC Farnell has produced a data sheet for the product with the specifications, and for an extra fee the firm also offers a calibration service for power supplies.

We found this a great unit to use and it has some nice features for its price point. However, better instructions and including a few leads in the box would be nice additions. □

Above
We used the power supply to experiment with slightly different supply voltages and constant current limits for an LED

VERDICT

For the price, this offers a good control of the power for your projects.

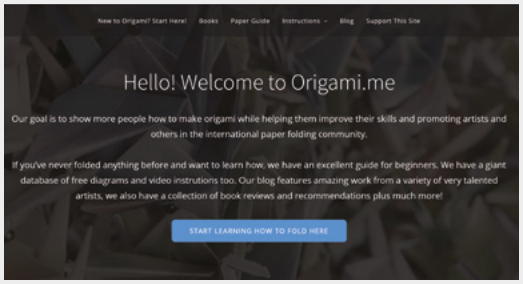
8/10

Origami.me

Paper-folding for beginners

Free | **origami.me**

By **Ben Everard** @ben_everard

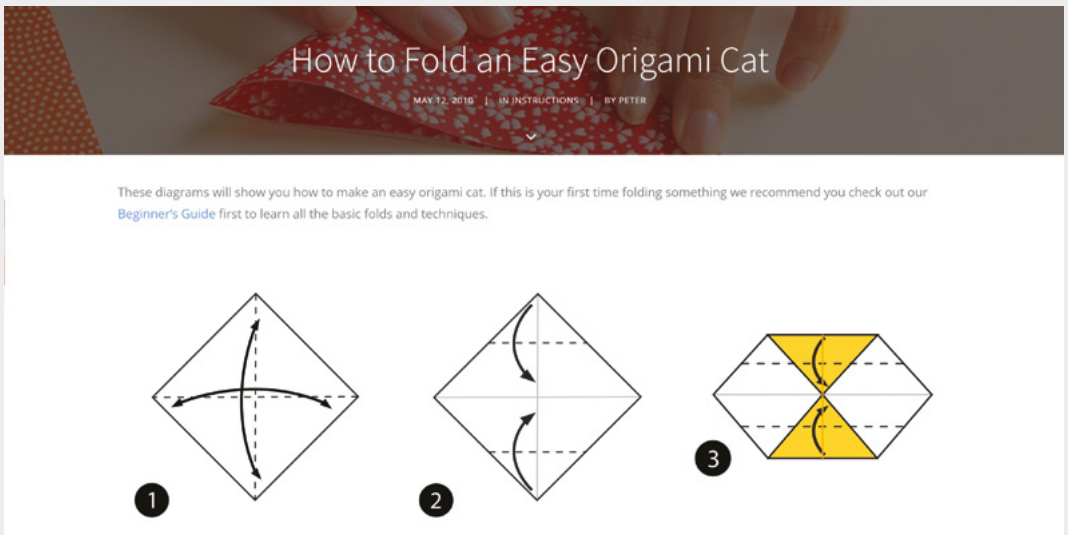


Origami is one of those hobbies that you can pick up easily and cheaply. In fact, there's a pretty good chance that you already know one or two patterns. To get started, you'll need a sheet of paper. Some types of paper are better than others, but at least for simple things, any paper should give you a route into the pastime. Couple that sheet of paper with some instructions – such as those at **origami.me** – and you've got a way into the hobby.

There's a beginner's section which takes you through some of the basic folding techniques you'll need, then a selection of diagrams. Perhaps the most impressive part of the site is the crease pattern database which collates patterns from all over the internet and categorises them by difficulty and design. Are you looking for an easy toad pattern? Just go to the Amphibians category, and see what you can find. Looking for an advanced peregrine falcon? There's one in the Birds and Bats category.

There is also a video section of this curated list, but since these are on YouTube, it might be easier to search on there. We also find it easier to follow the word-and-diagram instructions rather than the videos, but perhaps that's just us.

This reviewer is certainly no origami master, and he probably never will be, but he had fun trying out a few of the designs. Whether you're looking to entertain yourself for an afternoon, or embark on a new hobby, you should find **origami.me** useful. □



Above
There's plenty here to help you learn how to fold paper

Left
An easy model of a cat for helping you learn origami

VERDICT

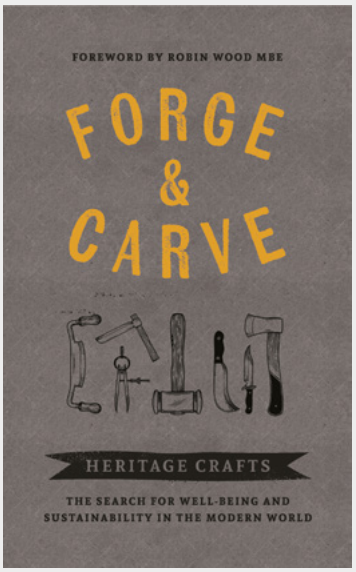
'Getting started' guides and a categorised guide to online origami.

9/10

Forge & Carve

VARIOUS £24.99 | Canopy Press

By **Ben Everard** @ben_everard



***Forge & Carve* is a book about heritage crafts by the people that do them.** Each chapter is written by a practitioner of a different craft, such as woodwork, pottery, or blacksmithing. We learn what the craft is, how it's performed, and how that practitioner is surviving in the modern world.

Perhaps the most interesting part for this reviewer was learning how social media is changing the market for traditional crafts. Instagram, particularly, has created a new market whereby sellers can reach potential customers directly without having to carry their products around craft fairs.

It isn't easy to make a living in the modern world using traditional methods honed through generations of craftsmen and women, but it is possible. This book is a message of hope that shows there is still a place for skilled artisans to make a living, and that technology can help preserve crafts.

VERDICT

A look behind the scenes at what it means to be a professional maker using traditional skills.

9/10

issue

#35

ON SALE
17 SEPTEMBER

MAKE AN ARCADE CABINET

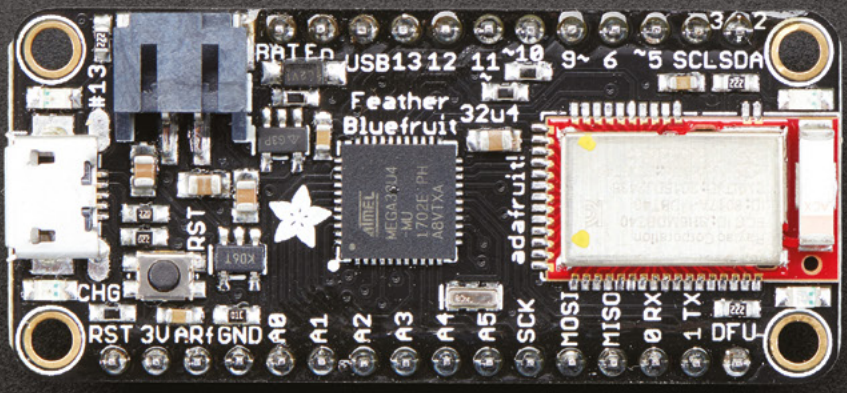
ALSO

- MUSIC
- PRINTING IN ASA
- RASPBERRY PI
- THE BEST PROJECTS IN THE MAKERSPHERE
- AND MUCH MORE

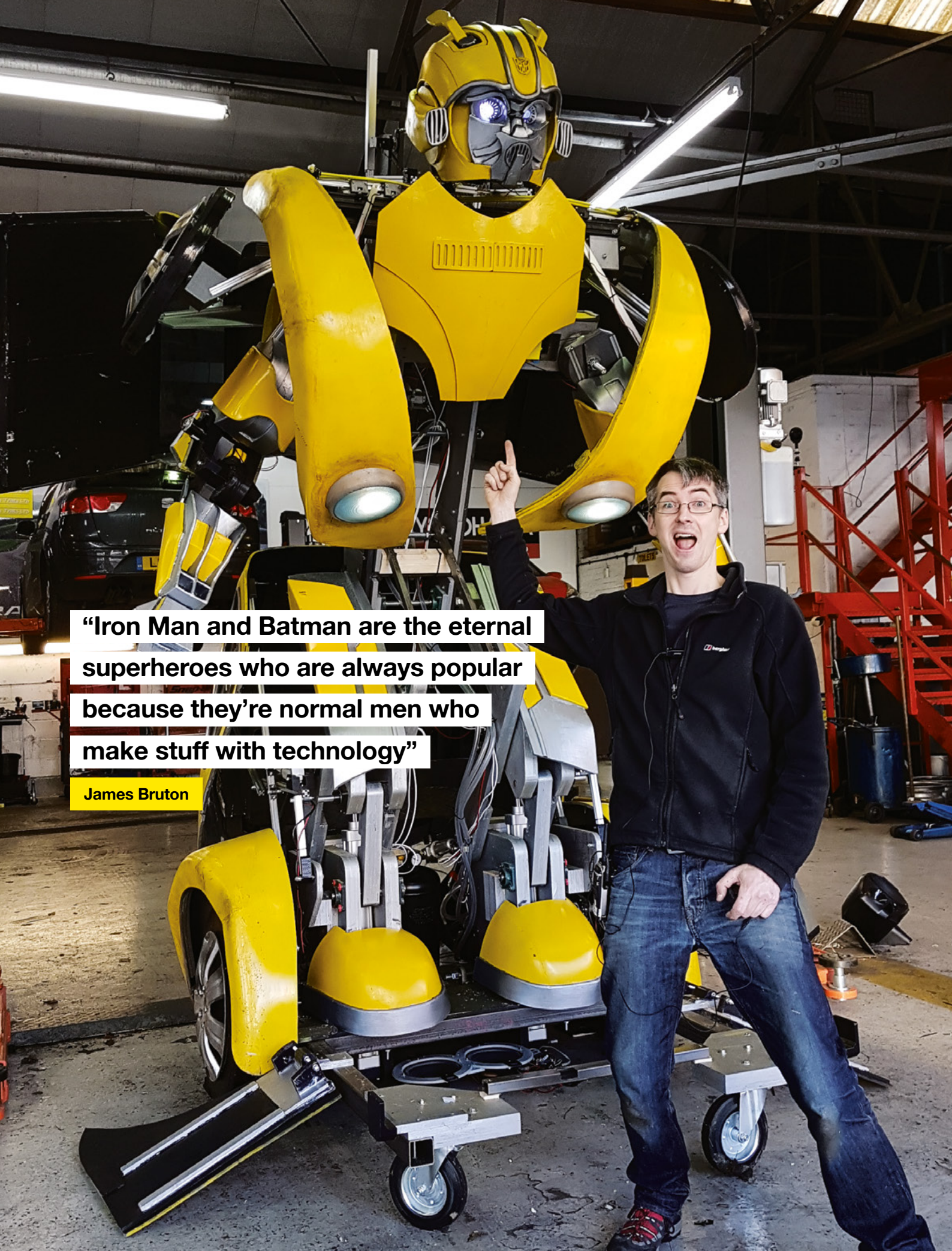
DON'T MISS OUT

HSMAG.CC/SUBSCRIBE

First Feather



Adafruit's Feather 32u4 Bluefruit was the very first Feather created. The board is wide enough to accommodate a Bluetooth module, yet still narrow enough to fit comfortably on a breadboard. It has space for plenty of IO and a LiPo charger. This format proved so useful that many more followed, both from Adafruit and others, and it's now one of the dominant ecosystems for makers. This month we're celebrating Feather in all their forms.



“Iron Man and Batman are the eternal superheroes who are always popular because they’re normal men who make stuff with technology”

James Bruton