

MAKE | BUILD | HACK | CREATE

HackSpace

TECHNOLOGY IN YOUR HANDS

hsmag.cc

May 2019

Issue #18



TRACK THE
**INTERNATIONAL
SPACE
STATION**

BUILD A
**SATELLITE
GROUND
STATION**

DISCOVER THE
**MAKERS
BUILDING
SPACECRAFT**

SPACE

JOIN THE MAKERS
EXPLORING THE
COSMOS

ALSO INSIDE

HEARING HELPER
A DIY solution to age-related
hearing problems

STEVE SHERWIN
Creating mechanical
art that moves

WEARABLE FASHION
Design your own 3D-printed,
light-up clothes

**REPAIR
CAFE**

Saving money
and the
environment



May 2019
Issue #18 £6



PYTHON ROPE OP-AMPS SOAP-MAKING

Over 10,000 Different Boards and Modules In Stock

**NO ONE
KNOWS
BOARDS LIKE
DIGI-KEY!**

DIGIKEY.COM/BOARDS




HDIGIKI

Digi-Key is a franchised distributor for all supplier partners. New
U.S. and other countries. © 2019 Digi-Key Electronics, 701 Bro

Electronics are registered trademarks of Digi-Key Elec




Right  These are so good, we had to include some work in progress shots to prove they aren't real tiles


Laser-cut jewellery

By Sophy Wong

 sophywong.com

Jewellery should be shiny by default – but what if you could make it light up? That's exactly what Sophy Wong has done here, with this bijou collection of NeoPixel costume jewellery. Each of these pieces is controlled by an Adafruit Gemma M0, and coded in CircuitPython, with the jewellery itself laser-cut out of thin maple and acrylic – or whatever else you have hanging around in your local makerspace. 



Left 
Like this? Tune in
next month for an
in-depth how-to

PLA spring motor

By Greg Zumwalt

hsmag.cc/ldfD0o

A

fter spending years designing and programming video games for Tandy, Atari, Electronic Arts, and many others, plus even more years designing flight control systems, I retired, purchased a 3D printer, downloaded a CAD program, and started designing models

for 3D printing.

In my early 3D printing days, most of my design and 3D printing time was spent trying to determine what I could and could not 3D-print in PLA on a 3D printer. In early 2014, I began testing whether a 3D printer could print a PLA spring motor, and the PLA Spring Motor Demonstrator was the result.

I was impressed enough with the PLA Spring Motor Demonstrator test to attempt to design, 3D-print, and assemble a variety of wind-up vehicles, including a wind-up car, wind-up bunnies, wind-up helicopters, and more.

I recently decided to revisit the original PLA Spring Motor Demonstrator design, using Autodesk Fusion 360, and the PLA Spring Motor Demonstrator II is the result.

The design incorporates the use of Fusion 360 'parameters', allowing me to alter sizes and clearances between the gears, axles, and axle mounting holes, with ease. As a result of this update, it will be much easier to design and print PLA Spring Motor-based models. ▣

Right ▣

Greg's done some fantastic work with springs, motors, and 3D printing: hsmag.cc/jxsiFI





Solar kitchen

By Luana Andrade

hsmag.cc/QNanZF

The idea of this project was born in the midst of an agitated discussion on a delicate topic in my country [Peru] – inequality, both in economic situations and in regards to services.

We found astonishing figures that made us open our minds to different alternatives to one of the most rudimentary and basic needs, the need for food and, therefore, cooking. In particular, 25.8% of the population has no access to the most rudimentary services, such as water or electricity. When you add to that the fact that 60% of the country is at risk of malnutrition, there's a clear need for a system that enables people to cook food cheaply and safely.

Taking different factors into consideration, we managed to build a simple – although a bit time-consuming – and cheap solar kitchen using an umbrella, cardboard, PVC pipes, and aluminium tape, among other materials, so it would be economically accessible to most, and an efficient solution to both the lack of cooking artefacts and the lethal side-effects that coal-powered ovens can bring. □





Left ♦ This solar oven can heat food up to 60°C, the minimum temperature needed to kill bacteria



Voronoi heart lamp

By Taner Ödzil

hsmag.cc/UBMkPY

In mathematics, a Voronoi diagram is a partitioning of a plane into regions, based on the distance between points in a specific subset of the plane. That set of points (called seeds, sites, or generators) is specified beforehand, and for each seed, there is a corresponding region consisting of all points closer to that seed than to any other. These regions are called Voronoi cells. The Voronoi diagram of a set of points is dual to its Delaunay triangulation.

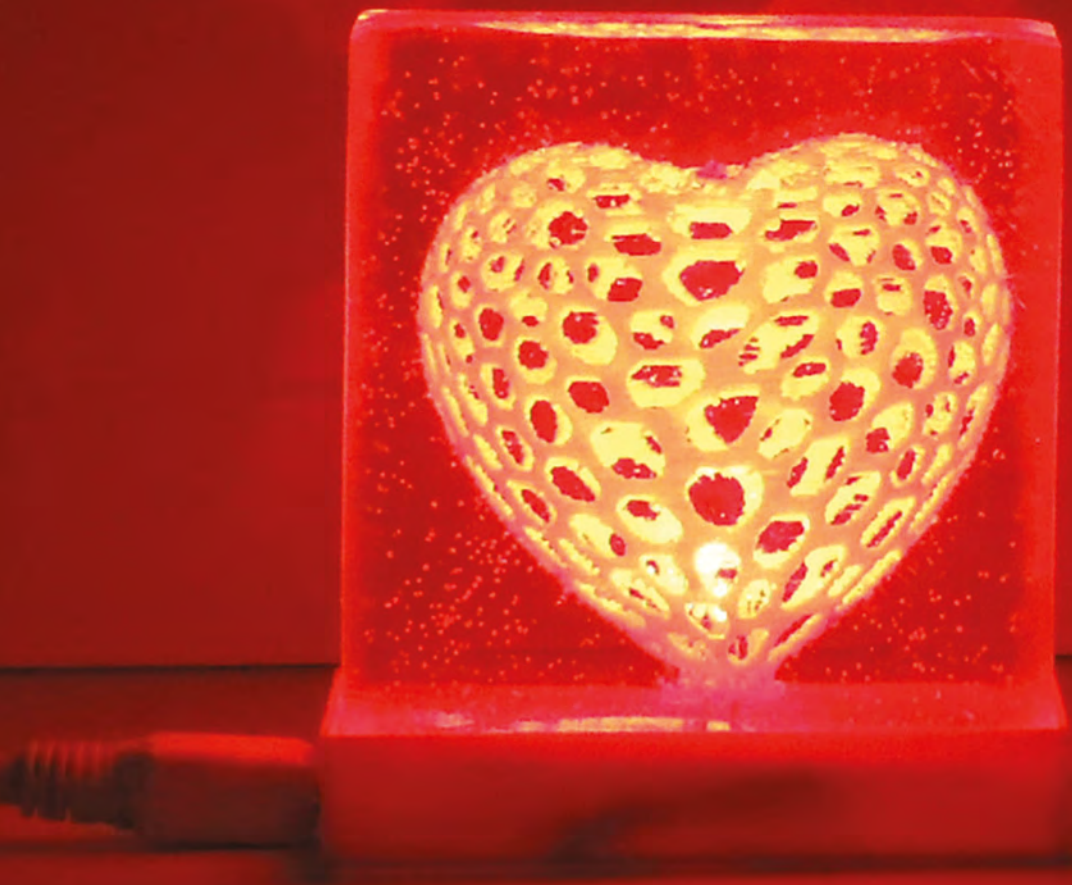
Epoxy resin offers resistance to heat and chemical applications, making it an ideal product for anyone needing a strong hold under pressure. Epoxy resin is also a durable product which can be used with various materials including: wood, fabric, glass, china, or metal.

So, we thought about a Voronoi heart. We designed it as a lamp. In this project, we are using just an Arduino Nano, a resistor, and a red LED. That's it. After you have a Voronoi heart lamp, use it yourself, or give it to your loved ones. We bet they'll love it! ▣



Right ▣

You can alter the inputs into your Voronoi calculation to get more or fewer triangles



Objet 3d'art

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

Fans of 1980s motorbike crime-fighting nonsense, *Street Hawk*, will love this: it's a (almost) completely 3D-printed electric motorbike designed by NOWlab, and printed by Berlin-based additive manufacturing company, BigRep.

Obviously, if you were to print a full-scale Triumph Bonneville and expect it to work, you'd be sorely disappointed the first time you fired up the engine and the frame collapsed. So, rather than being a conventional motorbike that just happens to be made of 3D-printed parts, the NERA is full of newly engineered parts, optimised to work with additive manufacturing.


First up, there are the airless tyres – these use a hexagonal lattice to provide a compromise between flexibility and strength, and to stand up to the forces of braking and acceleration.

Instead of forks to steer the front wheel, there are eight pivot joints, increasing contact between the front wheel attachment and the rest of the frame. And, instead of conventional suspension with springs and dampers, there's a flexible bumper to absorb vibration caused by uneven road surfaces.

Including an electric motor, battery, and all other components such as lights, the

bike weighs just 60 kg – it's also completely customisable, so can be adapted to fit smaller riders.

We don't have any information on performance or range, but really, if you had one of these you'd probably just hang about in front of the Reichstag at dusk: lead designers Marco Mattia Cristofori and Maximilian Sedlak have done a cracking job making this look the stuff of cyberpunk dreams.

Sadly, the NERA isn't available to paying customers just yet, but a functional vehicle made from 100% printed parts shows the amazing potential of the technology. We can't wait! 

 bigrep.com/nera-e-motorbike/



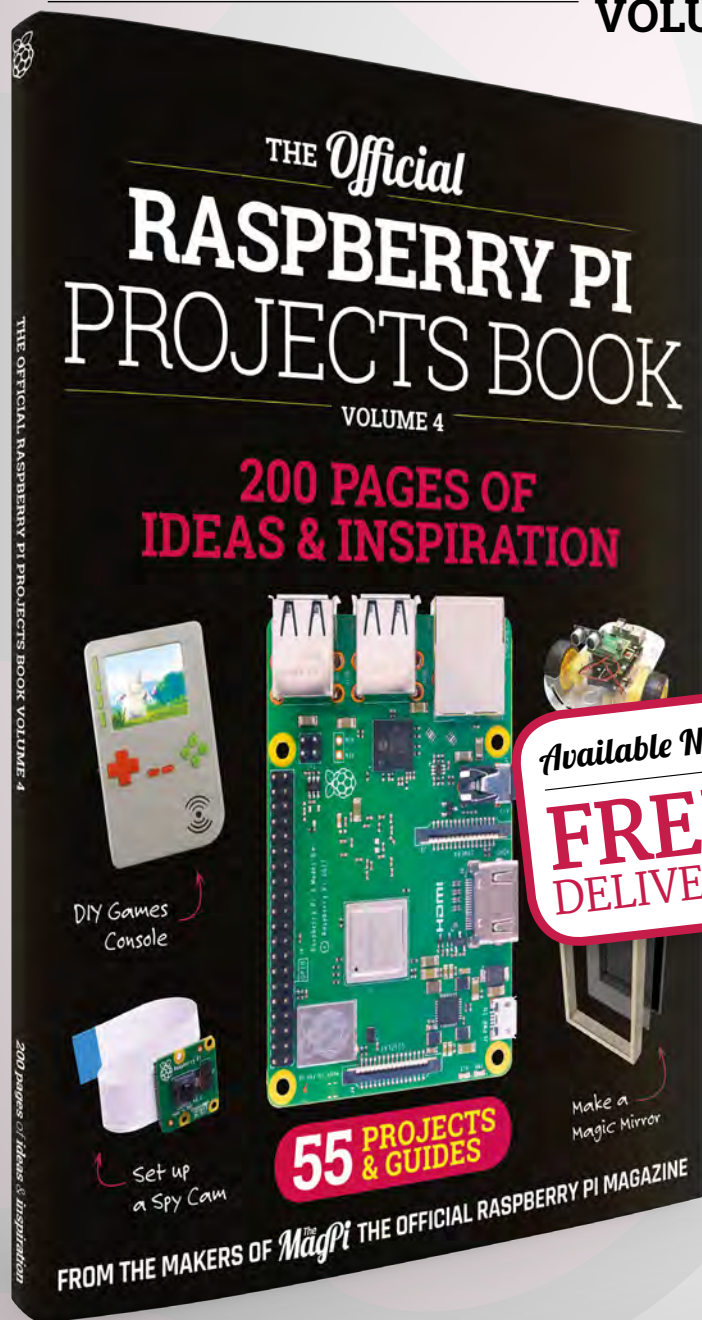


£12.99
200 pages of
Raspberry Pi

THE *Official*

RASPBERRY PI PROJECTS BOOK

VOLUME 4



**Amazing hacking
& making projects**
from the creators of
The MagPi magazine

Inside:

- How to get started coding on Raspberry Pi
- The most inspirational community projects
- Essential tutorials, guides, and ideas
- Expert reviews and buying advice

Available Now
**FREE
DELIVERY**

store.rpiexpress.cc

plus all good newsagents and:

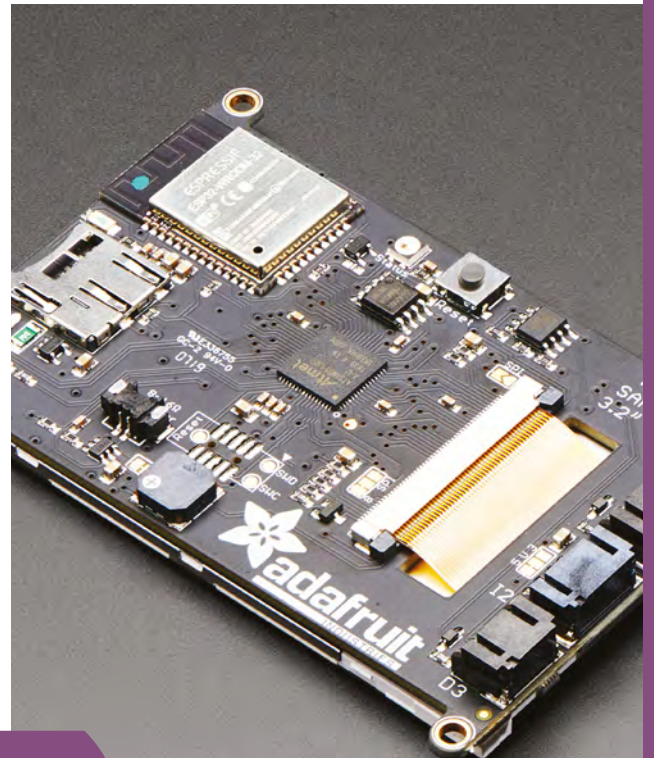
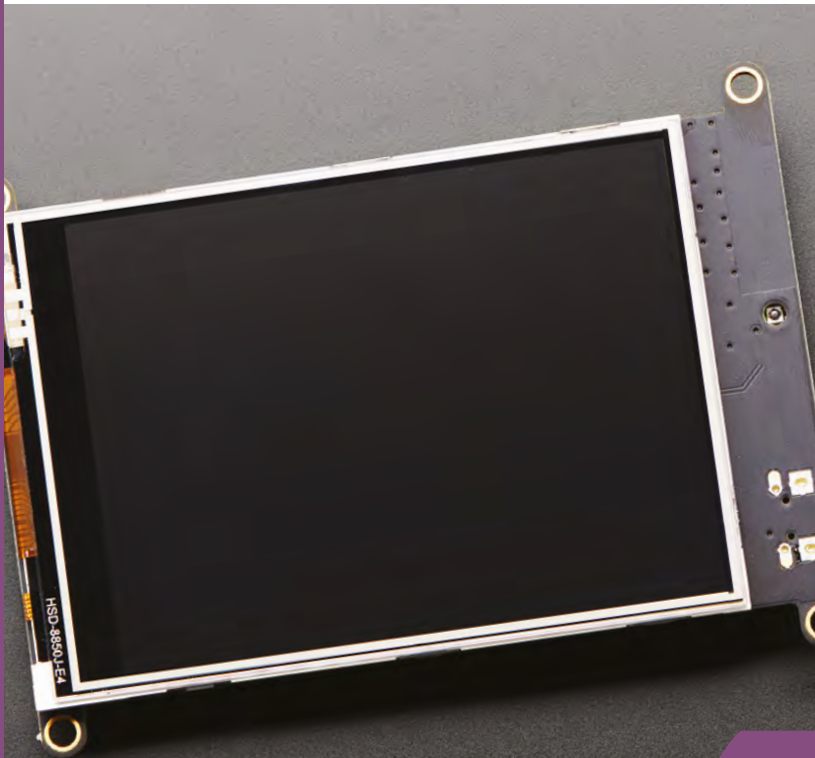
WHSmith **BARNES & NOBLE**



WIN!

1 of 5

Adafruit PyPortals



Go to
hsmag.cc/win

WORTH
\$54.95

Enter by 1 June
for your chance to win!

Terms & Conditions

Competition opens on 17th April 2019 and closes on 1st June 2019. Prize is offered to participants worldwide aged 13 or over, except employees of the Raspberry Pi Foundation, the prize supplier, their families or friends. Winners will be notified by email no more than 30 days after the competition closes. By entering the competition, the winner consents to any publicity generated from the competition, in print and online. We don't like spam: participants' details will remain strictly confidential and won't be shared with third parties. Prizes are non-negotiable and no cash alternative will be offered. Winners will be contacted by email to arrange delivery. Any winners who have not responded 60 days after the initial email is sent will have their prize revoked. This promotion is in no way sponsored, endorsed, or administered by, or associated with, Twitter or Facebook.

Meet The Maker: Steve Sherwin

What's better than shiny stuff? Shiny stuff that moves!



We met Steve at the Guild of Makers event in March, where he was showing off his wares.

The incredible precision of the steel and brass rods and gears working in concert captivated us,

so here they are presented for your enjoyment.

When we caught up with Steve for a phone call to find out about his processes and inspiration, he was on top of a roof, working under a blue sky in the sunshine. We were in an office next to a dual carriageway...

Below ♦
Steve learned how to use a lathe in his own spare time from videos on YouTube. So can you!



"I'm totally self-taught in everything that I do now. I've always messed about with things, all my life. I've always had motorbikes, taken them to bits, and put them back together. I made things out of wood when I was younger; I've made cabinets, wardrobes, bookshelves... I used to make those little fighting robots, a bit of everything.

"And then, about four years ago, I got a brazing kit and started getting into metalwork. I'd never been into metalwork, but I started to learn to braze things together.

"I'd just come up with this idea to make a beer bottle opener, because I like a bottle of beer. I made it with my brazing torch and a cheap pillar drill from Wickes. That's where it all started.

"You get inspiration from all over the place. In Stratford upon Avon, there's a museum called the MAD Museum (Mechanical Art and Design – hsmag.cc/gnOJXU). I visited there, and was inspired. I made this mechanical set of wings, showed them a video, took it there, and they wanted it, and it's on display there now. >

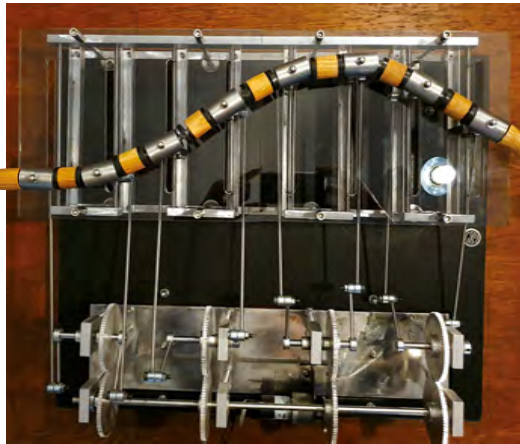




Left
The petals of this mechanical flower open and close once per minute. It's slow and mesmerising to watch



Above ♦ ■
Much hilarity was had when someone on the internet told Steve he'd done this snake automaton wrong. Ah, the internet...



LEVEL UP

"As I progressed, I ended up buying myself a lathe. I'd never been on one in my life. I plugged myself into YouTube, and taught myself how to use it. I was finding that I didn't have the equipment to do the things I wanted to do, so I bought myself a mill, and I've taught myself to use that as well. It's escalated from there.

"I make all sorts of things, whatever comes into my head. The other day, I made a big metal shelving unit for the kitchen, and then I made bespoke metal legs for a kitchen island, then I'll make a clock, then I'll make a robot, if there's some stuff hanging around the workshop. I'll mess about with anything, to be honest.

"I'm a bricklayer by trade... I build and develop houses for a living. I've worked for myself for 30 years. It's all I've ever done. But sometimes I think to myself: 'I don't want to go to work, I want to go in my workshop'. When you go in that garage, I can go in

//

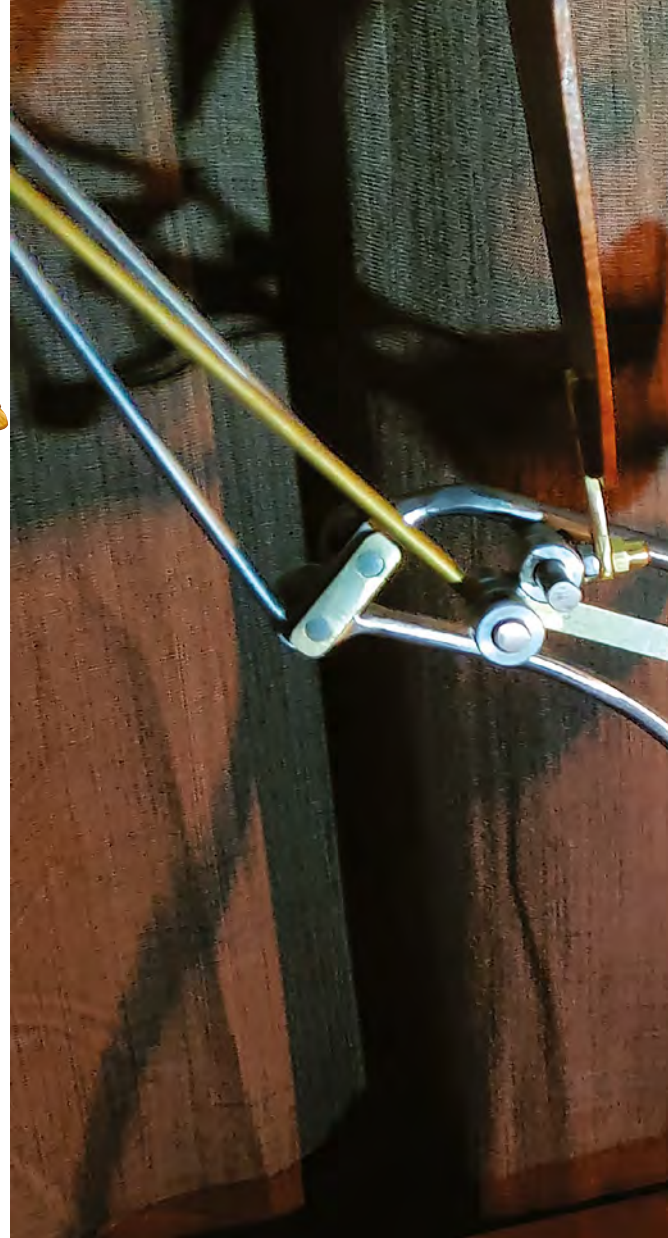
Going to the Guild of Makers is good because it makes me feel normal

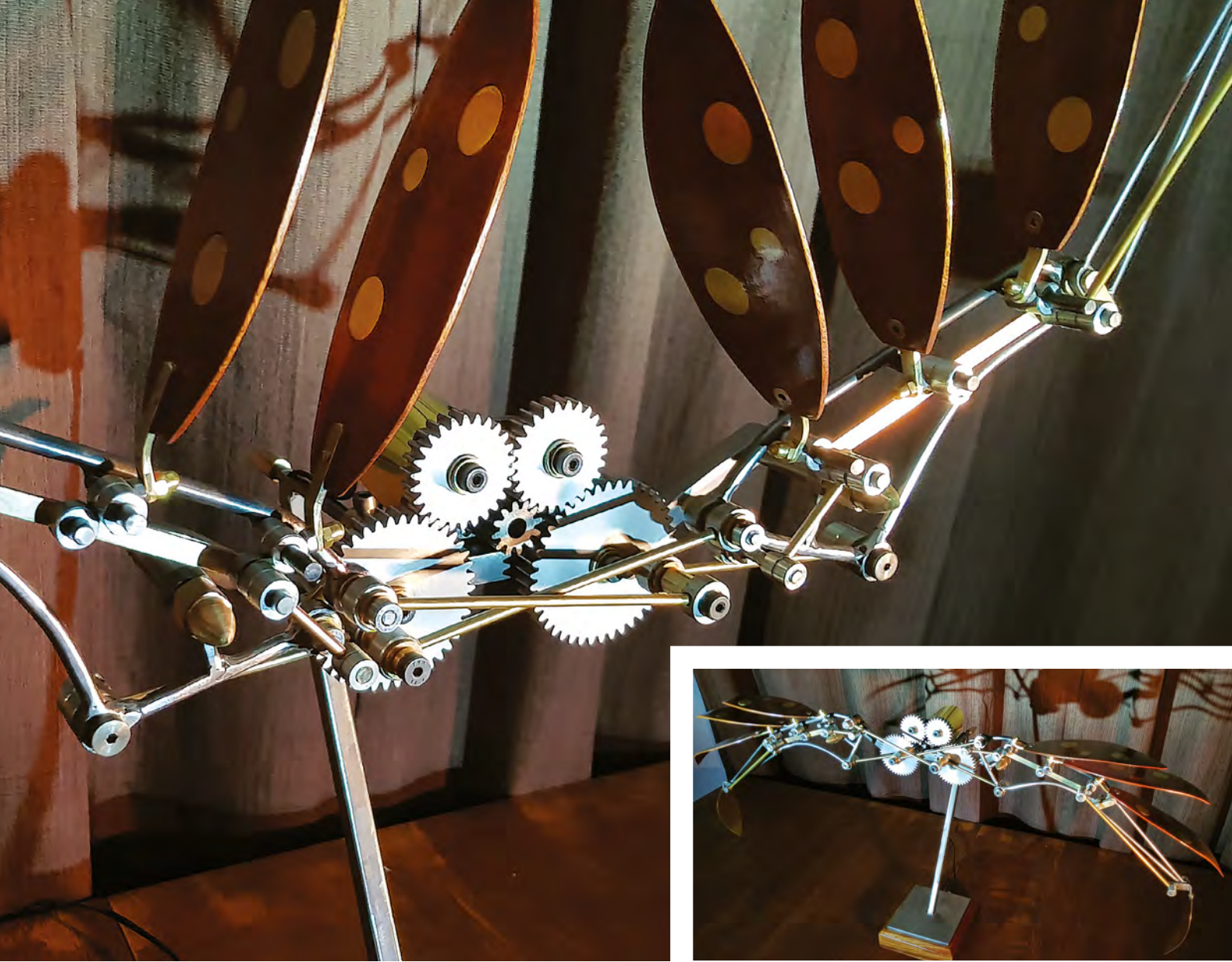
//

there at 7pm and, before I know it, it's 10 o'clock. My head's just in another world. I put Marc Riley on on 6 Music, listen to him chuntering rubbish, and that's it. You have to be a bit odd in the head I think.

"But going to the Guild of Makers is good because it makes me feel normal. There are lots of other people like me out there.

"I've got to know Paul Parry, who only lives up the road from me, through the Guild. I walked in his workshop and said: 'Thank God it's not just me that's an idiot.' It is what it is though; you either get it, or you don't. I do quite a lot of mountain biking, and if I don't go they say 'what you doing, messing about with your Meccano?' They don't get it. You either do, or you don't. ■





Above ♦
This is Maranti, the sculpture inspired by a visit to the MAD Museum in Stratford

Left ♦
There's a ton of examples of Steve's lighting work at hsmag.cc/JCqEaT



Not just handmade, person-made

When you make something, you imbue that object with a little of yourself



Lucy Rogers

🐦 @DrLucyRogers

Lucy is a maker, an engineer, and a problem-solver. She is adept at bringing ideas to life. She is one of the cheerleaders for the maker industry, and is Maker-in-Chief for the Guild of Makers: guildofmakers.org

My mum and I were in a charity shop recently, and saw a spinning wheel. I have fond memories of spinning wheels. When I was six or seven, my grandfather made spinning wheels for his daughters (my mum and aunts). I remember sitting in his garden on a beautiful summer's afternoon, the smell of lavender wafting past, the buzz of bees giving a background hum, and big flouncy hydrangea bopping their heads in time to my brother and I pedalling away. We'd challenged each other as to who could pedal the longest. We had no wool, we were just making the wheels spin.

So, even though my mum and I don't need another spinning wheel, we went and had a look. It was very familiar – but most spinning wheels are of a similar design. However, just as a dog owner knows their own Labrador retriever, you recognise your own. This was one of Grandad's makes. He was a clock-maker by training, but would turn his hand to anything – from rebuilding veteran cars, to making a steam boat. Everything seemed to be made on his tiny Boxford metal lathe

that was in the shed at the bottom of the garden – including the ash wood spindles of the spinning wheel.

My mum showed me the brass, old penny-sized disc that Grandad had used to balance the drive-wheel. I also saw the knot in the leather 'frontman tie' which, for some reason, I used to be fascinated with. We were sad to see the spinning wheel sitting there unloved in that shop. My aunts are not as crafty as my mum, so I suspect one of them had

passed theirs along a while back – it's not the smallest of ornaments. Grandad died about 20 years ago, but it is wonderful to realise that the things he made are still going strong. It has made me look again at some of the older handmade things around me.

So, even though my mum and I don't need another spinning wheel, we went and had a look. It was very familiar – but most spinning wheels are of a similar design

A floor mosaic in a shop, a gargoyle on a building, or a carving on a church pew. These were all made by someone – someone who had their own life, their own families. We probably don't know much about these craftspeople any more – but what they left still brings pleasure all these years later. I hope someone buys that spinning wheel, and thinks about the man who made it. He wasn't just a craftsman, he was my grandad too. □

Soldering tip: Use the Force (for feedback)

Going big to go small



Bunnie Huang

🐦 @bunniestudios

Andrew 'Bunnie' Huang is a hacker by night, entrepreneur by day, and writer by procrastination. He's a co-founder of Chibitronics, troublemaker-at-large for the MIT Media Lab, and a mentor for HAX in Shenzhen.

Fingertips can manipulate and sense objects that are smaller than the eye can resolve. Eyes fail to resolve pixels at densities above a couple of hundred pixels per inch, but fingers can feel the difference between thread counts approaching a thousand threads per inch. My first job in college, as a rework technician, introduced me to this amazing fact when I started using a microscope to aid with fine soldering work. I was astonished to find that I could easily scratch my name into the solder fillet of a 1206 resistor, a device about the size of a grain of rice, with the aid of a microscope.

I became obsessed with finding irons with finer and finer soldering tips. This continued until a friend of mine, Akiba, told me that he only uses large soldering tips, even for fine work. Initially incredulous, I then noted that the most skilled solder technicians in China also preferred a wide, flat 'knife'-style tip. The problem with small conical tips is that they lack the surface area to effectively transfer heat into the solder. Fat tips are much more effective at heating the workpiece,

and once the solder is hot, a bit of flux will allow the surface tension of the solder to do the hard work of getting the solder to the right places (and off the wrong places). However, a large soldering iron tip often obscures your view of the component. The trick, I found out, is to not rely on visual cues, but instead to rely on the feel of the iron as the solder underneath the tip transitions from solid to liquid.

I then noted that the most skilled solder technicians in China also preferred a wide, flat 'knife'-style tip

Thus, when I was presented with the 'SMD Challenge', I was actually quite pleased to find out that the soldering irons provided had large, rounded tips. However, there were no microscopes provided, so I had to rely on force feedback: first, I brushed liquid solder

across the pads to prime them with a tiny solder mound. Then, I placed a component by feeling when the tweezers bumped into the solder droplet, and then again feeling when the component sank into the solder as it liquefied. By using force feedback alone, I was able to solder the tiniest components using a large soldering iron, and no microscope, with just a couple of attempts. Soldering is not just a matter of seeing what you're doing – it's also a matter of feeling it. ■

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

LORA

I have a water butt at the bottom of my garden, controlled by an Adafruit Feather M0 over LoRa [Best of Breed, issue 17]. It's a simple solution, and I know I could automate it if I wanted to, but it's good enough for me, and it works.

David Robertson

Gloucestershire

Ben says: Sometimes the cleverest thing about a technology isn't that it's the biggest or fastest, but something more mundane. LoRa uses very little power, compared with Bluetooth, so it's great for applications that you can't plug into a mains socket.



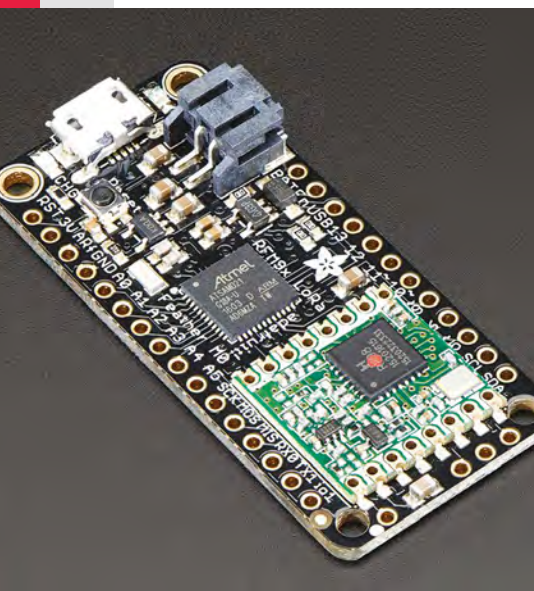
CARBON CAPTURE

I'd always felt a little guilty throwing failed 3D prints away. I always heard David Attenborough in my mind's ear, telling me off for polluting the environment that my grandchildren will inherit. Thanks for talking to Dr Adrian Bowyer last month, and setting the record straight – I will now be printing 24/7, in my own personal bid to requisition as much carbon as possible.

Paul Johnson

Maine

Ben says: That's just for PLA, mind, as it's plant-derived. Otherwise, have fun!



WHISK[E]Y

Sir, I must protest. Contrary to your how-to in issue 17, turning whiskey into vinegar is completely acceptable, and in the case of some, is the best use for it. It's whisky that should be kept pure in its natural state, and drunk as quickly as possible after opening the bottle, so that it doesn't go off.

Robin Campbell

Scotland

Ben says: You've given us a great idea for an export product: single-malt vinegars, made from finest Islay whisky, to imbue your fish and chips with the tang of peat smoke and iodine. It can't fail!



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.

CROWDFUNDING NOW

Nixie Tap

Glowing numbers of time

From \$99 | crowdsupply.com | Delivery: July 2019

The Nixie Tap is a minimalistic Nixie clock. There are no buttons, but a touch sensor that you can tap to change the display type – by default, it will cycle through four settings: time, date, temperature, and cryptocurrency prices. These details are collected from the internet via the internal ESP8266 microcontroller (tap the clock five times to enter the WiFi setup mode). The clock has been developed through the Microchip Get Launched competition, which aims to help makers launch products.

The raw parts are available without a case for \$99 (3D-printable STL files are included) if you'd rather build your own. Alternatively,

you can get the complete clock with a case (either dark walnut or anodised aluminium) for \$325. We're big fans of the walnut option (pictured). It looks glorious, but that does come at a price.

Perhaps the best bit about this clock isn't that it's a great looking clock, but that it's a hackable, web-connected display. It's only four digits, but those digits can be whatever you like (and you can switch between modes with a tap of the box). You can take a look at the source code at hsmag.cc/0SPR0T to see how this works. It's programmed in Arduino C++, and so it shouldn't be too difficult for an experienced Arduino user to convert this to display other data. ▣



Right ♦
The dark walnut case works really well with the red glow of the Nixies

Space of the month: TOG



tog.ie

Tog (or Tóg if you're going to get things right) is the Irish word for 'build', which makes it a brilliant name for a makerspace in the heart of Dublin.

Tog has been in existence for over ten years now, having celebrated its birthday in January 2019. And, it's doing well:

"We have over 90 members and are growing at a rate of around one new member a month," says Jeffrey Roe, Tog's CEO. "We've just had a very packed March programme of events. Members are building a lot of projects – a lot of projects – and we've been taking part in several outreach events.

"We're on our third building: we got hit with a large rent increase when we signed a new lease in December. There's a boom and bust cycle [in Dublin property], and the boom is definitely back."


Tog has two membership rates: €45 standard rate, and €20 reduced, so it's up to members to determine what they pay. In terms of equipment, they've got the normal things like 3D printers and a laser cutter, a lathe, a forge, CNC machines, sewing machines, pipe bender, welder, and all sorts of fabrication tools. Wait – a forge? "The forge is mainly a summer activity – we don't have an extraction system, so it's for outside use only. People use it for casting props and cosplay material", says Jeffrey. "We also have an extensive brewing community – I couldn't name what kind of brewing equipment we have, but every time I walk past it, things seem to be bubbling."

Like most successful makerspaces, Tog has built up a rapport with local businesses. It's recently had a donation of equipment from an electronics testing company, including oscilloscopes, function generators, and spectrum analysers. "We've had stuff from schools and various other organisations – it's up to us to put the word out there that we can put old equipment to use. And, of course, we get offered loads of ten-year-old PCs."

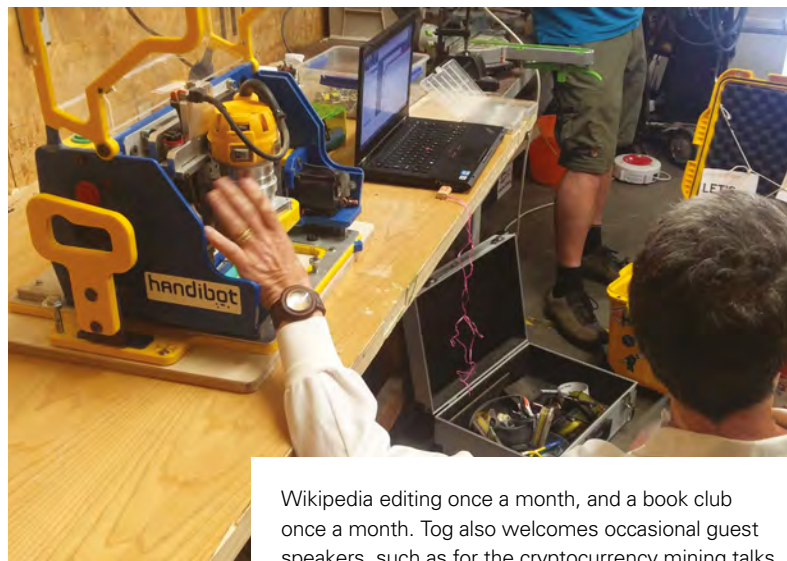
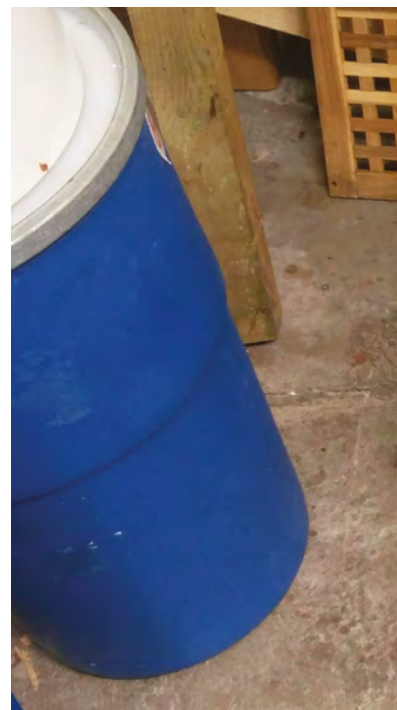
EVENTS

Regular events include the electronics night every two weeks, lock-picking every two weeks, →



Below  Jeffrey, Tog's CEO, also organises Dublin Maker, taking place this year on 20 July in Merrion Square





Wikipedia editing once a month, and a book club once a month. Tog also welcomes occasional guest speakers, such as for the cryptocurrency mining talks, which happen every other month. Regular events might get 10-15 attendees, with bigger, hackathon-type events getting in the region of 70 people.

And there's an even bigger event that Tog's involved with: Dublin Maker.

"Dublin Maker used to be a Maker Faire, but we decided to part ways with Make. It's been running for eight years now", says Jeffrey.

"We get about 10,000 on the day. Last year we had 59 maker exhibits. We try to keep it as non-commercial as possible; we get funding through a government agency called Science Foundation Ireland, which is an organisation that supports STEM outreach education.

That allows us to avoid the situation where everything has to be branded up and we have to keep pushing corporate products at people; we can focus on demonstrating cool stuff. "

"For the outreach that we like to do, we've found that the space isn't particularly child-friendly – and child protection legislation adds a lot of red tape. So, what we found the best thing to do, was to go out to other people's events and showcase what makerspace culture is like. We did one recently over the St Patrick's Day weekend, with science experiments and showing off projects.

"One coming up is an event called Coolest Projects – it's where kids, who go to CoderDojos, come together to showcase the things they've made, kind of like a maker event, but much more coding oriented. What makes that good for us is that, when we're showing off at that event... yes, CoderDojo is great, but we want to show the kids that when you turn 18, and you grow out of CoderDojo, that there's somewhere else to go. That's Tog." □



CONTACT US

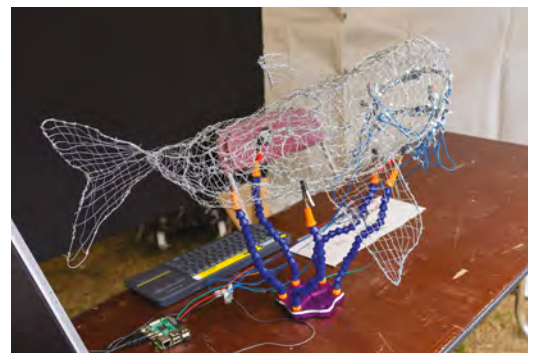
We'd love you to get in touch to showcase your makerspace and the things you're making. Drop us a line on Twitter @HackSpaceMag, or email us at hackspace@raspberrypi.org with an outline of what makes your hackerspace special, and we'll take it from there.



Above ♦
Cider pressing, maker-style. We approve

Left ♦
Tog's duck logo pops up in various guises around the space

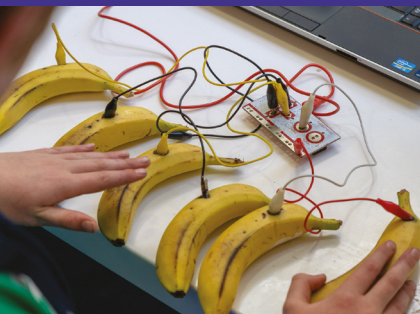
Below ♦
This free-soldered shark is just waiting for someone to stick a laser beam on its head





Liverpool

MakeFest



Saturday June 29th 2019

A free day of fun for all the family!

lpoolmakefest.org

LENS

HACK | MAKE | BUILD | CREATE

Uncover the technology that's powering the future

PG
50

HOW I MADE: TV AUDIO BOOSTER

Sick and tired of mumbling on the telly? Here's how one maker fixed it

PG
54

INSIDE THE REPAIR CAFE

How volunteers are fixing things in a little corner of Manchester

PG
60

INTERVIEW: PAUL BEECH

Pimoroni's master of all trades takes us inside the maker emporium of dreams



PG
34

THE FINAL FRONTIER

Build a satellite and track the cosmos with open maker hardware

PG
68

IMPROVISER'S TOOLBOX

Cool things to do with rope. Ahoy there, me sea dogs!

SPACE

FEATURE

SPACE

**BUILDING
PROJECTS THAT
EXPLORE THE
COSMOS**

By Jo Hinchliffe

Open-source projects are democratising space for makers and hackers – let's get involved!

Space is an amazingly tantalising subject for **hackers and makers**. While for many of us, the dream of escaping the confines of this planet will remain just that, a dream, it's increasingly possible for our projects to head outside the atmosphere into the cold void beyond. There, they can live vicariously and through their electrical sensors, we can get a brief glimmer into the cosmos.

More and more amateurs are building and launching small satellites, often in Low Earth Orbit (defined as any orbit below 2000km, but more often between 300km and 600km). At these heights, satellites orbit quickly and may travel around the planet in around 90 minutes. Here, you can run experiments in zero gravity, take photos of the world below, broadcast messages to the whole planet, or simply say 'Hello World'.

At this altitude, the environment is unkind for electronics: temperatures vary from -100°C to 120°C, and there's no airflow to help remove heat. You can't just pop up with a soldering iron if things go wrong, or unplug it and plug it back in, so you have to get it right first time.

Let's take a look at some of the makers who have gone before us and already sent their projects up, and see what we can learn from them for our own extra-terrestrial builds. ➤

SMALL SATELLITES

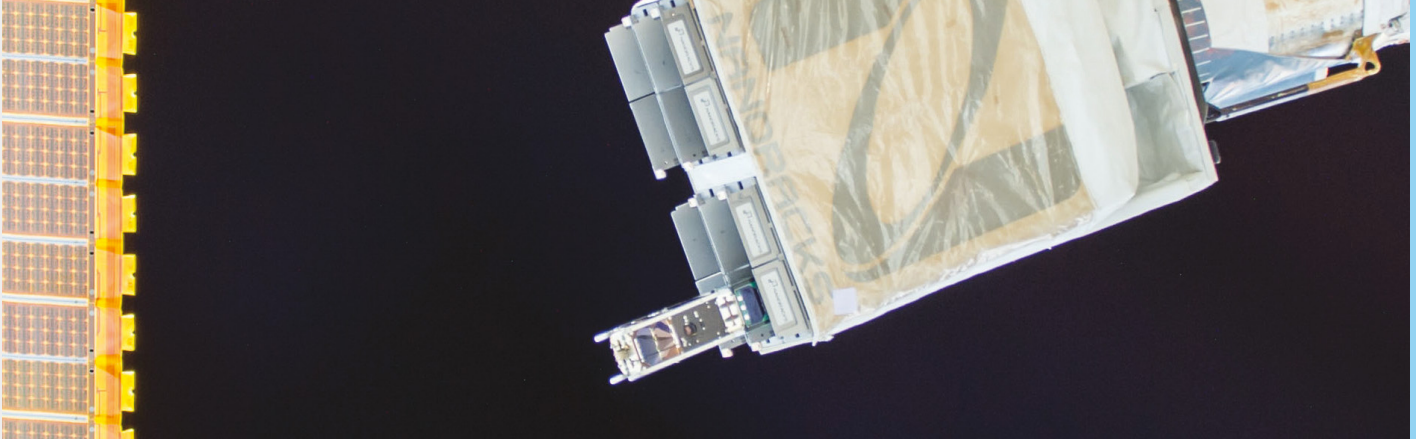
To keep launch costs down,
makers have to miniaturise

Small satellites come in a range of form factors, but over the last decade, one type – the CubeSat – has become established and popular. Over 1000 have been launched to date, and many more are in development

globally. A CubeSat is a 10×10×10cm satellite which can be built in multiples of this unit; for example, you can have a three-unit CubeSat measuring 10×10×30cm. The standard was created by Professor Bob Twiggs and Professor Jordi Puig-Suari in 1999.

The original concept was to have a satellite class that could be built by students completely within the time frame of an undergraduate degree. Professor Twiggs was fed up with students being demotivated by working on one small subassembly of a massive satellite project that they wouldn't see fly for many years, and certainly not during their degree.

The CubeSat specification deals with the mechanical, electrical, and testing standards that it must meet to be eligible to fly. For those with deep pockets, there are off-the-shelf circuit boards enabling you



to purchase different subsystems from CubeSat vendors. This can get expensive, and many people, and certainly amateur teams, often choose to design and build their own.


Whilst building a CubeSat might be affordable, launching a CubeSat costs significant amounts of money. However, there are often affordable options for non-commercial operations. Some CubeSats are launched from a service of the International Space Station (ISS), whilst some are deployed directly off payload sections of rocket missions. Either route can mean that your satellite is integrated (put in the delivery vehicle) months before it's deployed into orbit, providing another challenge for the designer. Other interesting problems to solve are that the satellite must be inert until a defined


period after launch (allowing it to be clear of other spacecraft), then it usually deploys antennae and begins operations.

SATELLITES IN MINIATURE

The following projects are inspiring, non-commercial missions that have some parts or all their design open for us to study, replicate, and build on. UPSat was a two-unit CubeSat that was a project between Libre Space Foundation (LSF) and the University of Patras. It was designed and delivered by LSF and flew successfully on orbit for 18 months. It is the first fully open-source satellite with every detail of the satellite design being open: every element of the design and development of UPSat with all the source code – from mechanical structure, to PCB fabrication, to the software design files – is available on the UPSat GitLab repository here:

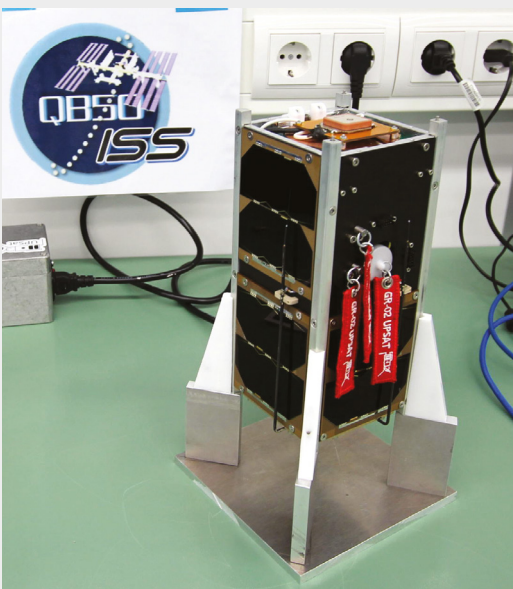
hsmag.cc/iBWNzO.

Above  UPSat, an open-source CubeSat satellite, being deployed from the NanoRacks service on the International Space Station

Left  UPSat, the flagship CubeSat mission of the Libre Space Foundation

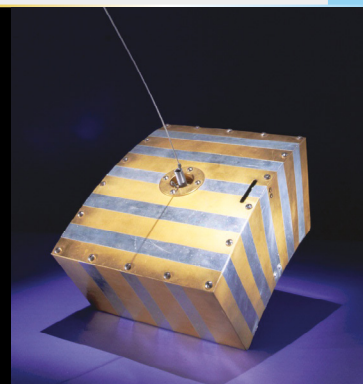
QUICK TIP

Legends say that the dimensions for a one-unit CubeSat were inspired by a tissue box!



PAVING THE WAY

The first amateur satellite, OSCAR I, was built and launched in 1961 by AMSAT, a mere four years after the first ever satellite, Sputnik I, successfully operated from orbit. This amazing global community of radio amateurs have pioneered amateur ways into space over many missions and are still extremely active. In the last few years, however, we are beginning to see that open-source communities are beginning to form and develop around the themes of space, enabling and encouraging new people and new ideas to emerge.



UPSat was built from the ground up, rather than relying on off-the-shelf subsystems – it was designed, as far as possible, to be reproducible by other teams using the types of tools and electronic equipment found in most hackspaces. With seven solar panels charging three cells, UPSat also had an on-board imaging system, a science unit with apparatus for measuring plasma, and its physical chassis was an interesting hybrid of aluminium and carbon-fibre-reinforced polymer panels.

Decreasing in size from CubeSats, there is another emergent form factor for satellites called 'PocketQube' (see **Figure 1**). With a lot of similarity to CubeSat, but packed into a 50 mm cube for a single unit (or 1P in PocketQube parlance), there are many PocketQube teams developing on the platform around the world. To date, only four PocketQube have ever been launched, but one in particular, '\$50sat', caught the imagination of many, as it was successfully built and operated by an amateur team and survived on orbit for 18 months – although called \$50sat, it was actually realised for around £250, which is still an astonishing achievement.

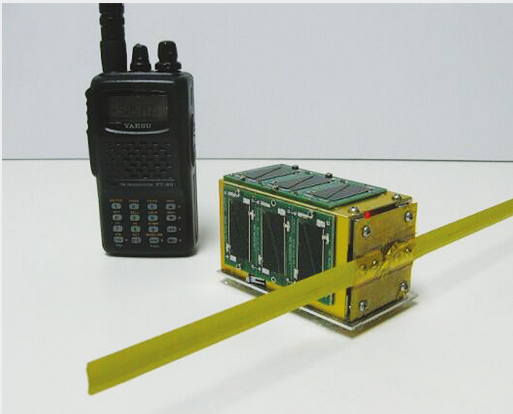
" Decreasing in size from CubeSats, there is a form factor called 'PocketQube' "

\$50sat used no specialist components, and combined a PICAXE 40X2 microcontroller with the common Hope RFM22B transceiver module, a commonly used radio module in many maker projects. The chassis of the satellite had no high-precision parts and was simply fabricated from sheet metal. \$50sat also used an off-the-shelf lithium-ion battery intended for a digital camera. Its success on orbit, combined with its price, demonstrated amazingly that with some work and ingenuity, small satellites can use standard components and be extremely cost-effective.

The \$50sat team brilliantly published all the schematics, code, and project notes on their website: 50dollarsat.info.

There's a large number of PocketQubes in development all over the world. Libre Space Foundation, creators of the aforementioned UPSat, has some designs and development around the PocketQube form factor – these are available here: hsmag.cc/igxFDx.

Another interesting current project on the PocketQube platform is FossaSat by Fossa Systems (**Figure 2**, overleaf). Fossa is using open-source and decentralised working styles to build and launch FossaSat, which has a LoRa payload. Hoping that interested parties will be able to use a tiny and affordable ground station to receive and interact with



Above ♦
\$50sat really set the bar high for PocketQube in terms of operating well and at low cost

FossaSat, it has a fabulous deployable solar array, and all the designs are available at: hsmag.cc/DelQSL.

The FossaSat team are hoping to launch towards the end of this year – the satellite will test LoRa spread spectrum modulation, which they hope will really democratise telecommunications. It's interesting to note that in many hackspaces around the world, people are working on similar LoRa/IoT devices, but they aren't considering putting them on orbit!

IS THAT A SATELLITE IN YOUR POCKET?

Going smaller still, we begin to see satellites being built on a single board – leading the way in this respect is the KickSat mission, which comprises two parts and is the long-term brainchild of Zac Manchester. The mission consists of a three-unit CubeSat mothership densely packed inside with 100

GLOBAL COVERAGE

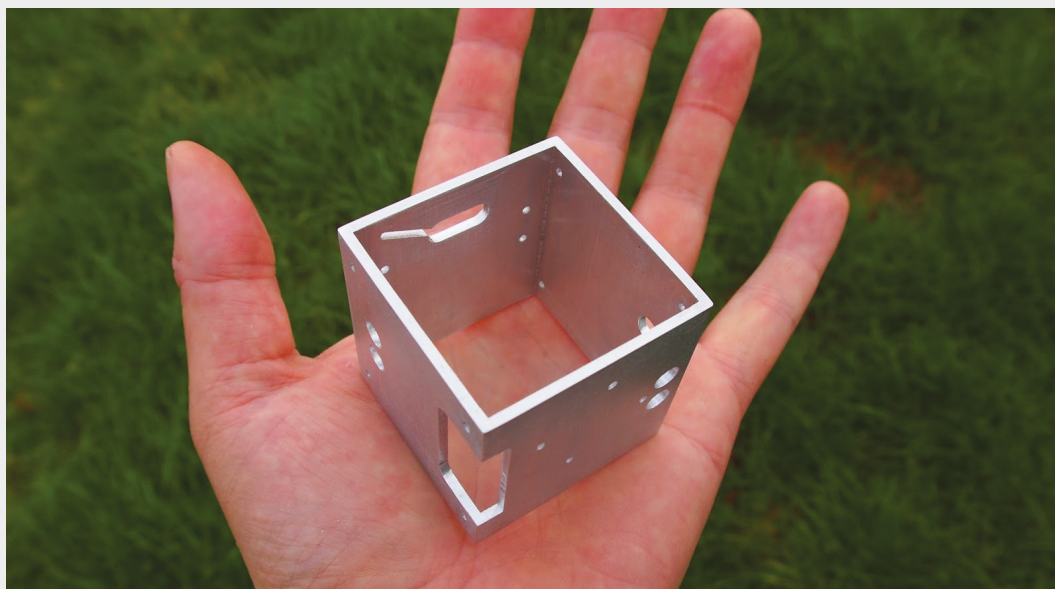
Most small satellites will want to communicate with 'home', and most teams want to get telemetry or sensor data back from their satellite, which means some kind of radio system must be used. For the ground side, a challenge is that a single ground station may only be able to receive a signal a few times per week for short periods (a few minutes) as the satellite passes overhead. You may think that we would definitely need something like the amazing radio telescope dish in the picture, but don't worry. Although it would be amazing to have an enormous dish on our house, smaller, affordable options are achievable. Later in this feature, we will look at an amazing open-source project providing solutions for this: the SatNOGS project, a network of ground stations providing near-global coverage.



tiny 'Sprite' satellites. Currently, the latest iteration of this mission is on orbit. The KickSat mother ship is deployed, and we await news of the Sprites being released.

Designing the Sprite to be so small and cheap to produce enables hundreds to be released simultaneously, which means that although they may return only a small amount of data each, the cumulative amount of work they can do is impressive. The Sprite (**Figure 3**, overleaf) is →

Above ♦
Attribution By
Uberprutser – Own
work, CC BY-SA 3.0 nl,
hsmag.cc/UPhfmv



QUICK TIP

In the six to seven years the author has been following CubeSat development, he has seen the price for 1u launch tumble from £200,000 to often less than £50,000.

Figure 1 ♦
An experimental prototype PocketQube chassis (made by the author), showing just how small they are

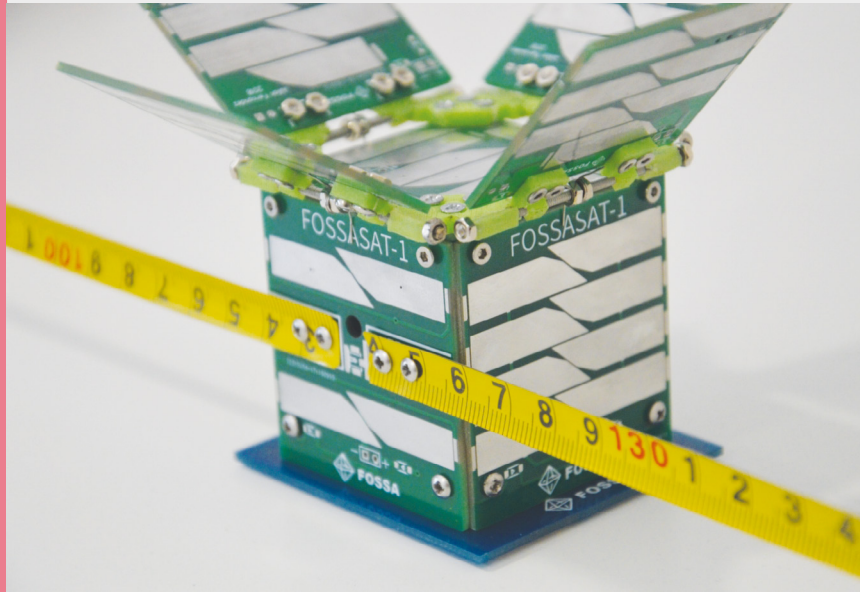


Figure 2 With its hinged deployable solar panels, Fossasat is looking to maximise its power budget and increase the chances of a successful mission

Figure 4 Stuart McAndrew, a PocketQube developer, has redesigned the open-source Sprite board, as he wanted to fit a different solar cell onto it

equipped with single-chip sensors as magnetometers and a gyroscope, but it could be redesigned to carry many other types of sensor. Due, again, to the Sprite being open-source and all the design files being available, people are already playing with the form factor and creating new versions of the Sprite satellite PCB (**Figure 4**).

SO YOU WANT TO BUILD A SATELLITE?

So, hopefully, you feel inspired to build a satellite, let's have a look at what you will need. Many satellite missions divide the bits they need in a satellite up into subsystems, which often may

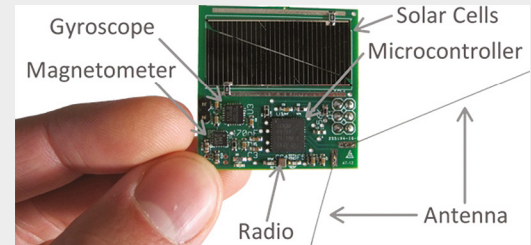


Figure 3 The Sprite – for a 35 mm square board, it has a considerable amount of technology and functionality built in

include the following. We can see in **Figure 5** that UPSat had many subsystems; let's look at a few of these that are common to most satellites...

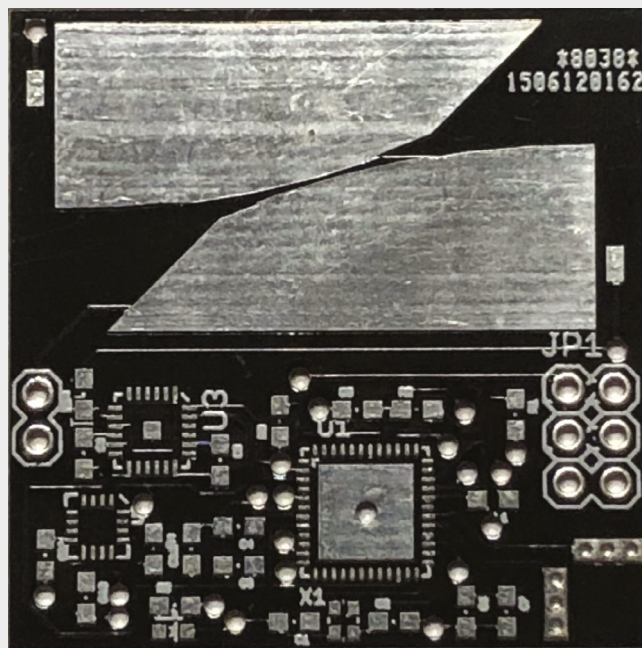
Structure – The structural components of a satellite are essential. Satellites undergo huge forces and vibration during their delivery to space and, as such, a resilient and tested structure must be produced. In Low Earth Orbit, the temperature range can be -100°C to 120°C and is also a near vacuum – the structure must be able to cope with this and more! Often, launch services and other specifications have weight limits and also have approved lists of materials that may be used to build a chassis.

WE NEED MORE POWER, CAPTAIN!

EPS – The electric power subsystem is going to contain some sort of power storage (usually batteries) and some kind of charging system, the most common being solar panels. The power system will, of course, supply power to the other subsystems, and as such, the designer needs a clear idea of the power budget it will be able to create. It's common

and good practice to then look at every other subsystem's power needs as items are added, with checks put in place to make sure that the power system budget is never exceeded.

COMMS subsystem – most small satellites will feature some type of communications system. Usually, this will consist of a transceiver (a radio enabled to receive and transmit). It's a complex area – and a key aspect is that satellites must coordinate their frequency allocation so that they don't interfere with other terrestrial or space systems. For amateur missions, the IARU is the governing body which allocates frequency and works with teams to go through the allocation procedure. The radio system will often return telemetry and perhaps payload/experiment data down to ground stations on Earth. The data can be packaged in numerous ways, from



QUICK TIP

The structure (and subsystems) have to survive a strict series of ground tests, vibrations testing, thermal cycling, vacuum, and more!

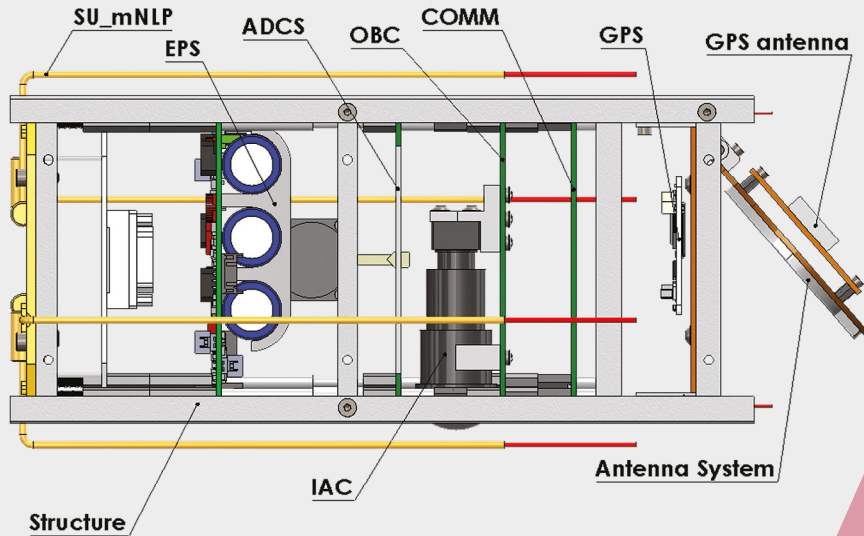



Figure 5  UPSat had some common subsystems that most satellites will have, as well as some specialised subsystems for its own mission requirements

Morse code to RTTY to AX.25 packet systems.

Other decisions that need to be made are about antenna design and deployment – a lot of the time you have to have a mechanism that releases the coiled-up antenna some time after deploying to make sure it is clear of the delivery/launch vehicle. A good place to start learning about these types of systems may be your local amateur radio club.

Payload – many amateur satellites may not carry a specific payload. Sometimes as an engineering experiment, they may just return telemetry about their own health and systems. However, many will fly a specific payload, perhaps an imaging system/camera, a science payload, or a particular component they want to test on orbit.

ADCS – the Attitude Determination and Control System is anything that helps stabilise the position of the satellite. These can be as simple as passive magnetic rods that make the satellite align with the Earth's magnetic field, through to the active version of this: 'Magnetorquers', which are controlled coils that can be powered in sequence to create a stabilising magnetic force in combination with the Earth's magnetic field. Some small satellites have included reaction wheels which spin to change the angular momentum and cause it to stabilise on a known axis.

OBC – the on-board computer subsystem; self-explanatory but with a wide range of complexity. Satellites have flown with incredibly complex computer systems through to small microcontrollers. ATmegs, PICAXE, STM, and many other common and familiar microcontrollers have flown in space. →

" The Attitude Determination and Control System is anything that helps stabilise the position of the satellite "

BUILD A SATNOGS SATELLITE GROUND STATION

Listen in on satellites as they orbit the Earth

YOU'LL NEED

- ◆ Raspberry Pi 3B+
- ◆ MicroSD card
- ◆ USB SDR dongle
- ◆ SMA to BNC adapter
- ◆ Antenna
(see article for options)

Once you've built and launched your small satellite, you'll want to be able to listen to all the glorious telemetry and data it's sending back to us as it hurtles around the Earth. Or perhaps you aspire to have a satellite up there, but in the meantime you want to listen to some other objects? What you need is a ground station, but a single ground station has one slight flaw. Most of the time a satellite will not be overhead of a single ground station; in fact, it may only pass over a ground station once every few days, massively reducing the amount of information or data we can receive. So, we need a network of ground stations. The SatNOGS network solves this by creating a global network of stations that can work together to increase coverage.

SatNOGS is an open-source project that has numerous designs for satellite ground stations, but whichever design you pick, you can join the network that links them all via the web.

Figure 6 ◆ This image shows the coverage of the SatNOGS network from a recent campaign to observe and capture slow-scan television (SSTV) images from the International Space Station (ISS)



A station owner can use the website to browse for future passes of a satellite, and then click a button to schedule for their station to turn on, tune to frequency, and record the pass, sometimes even rotating the antenna on the station to track the satellite. Not only can a station owner schedule an observation on their own station, but they can schedule observations on any station on the global network.

As we can see from this map (**Figure 6**) of data being collected of a recent SSTV broadcast from the ISS (sends single-frame



images transmitted via audio from the ISS), the SatNOGS network has near-global coverage, rivalling most professional institutions in the world.

SIMPLE SETUP

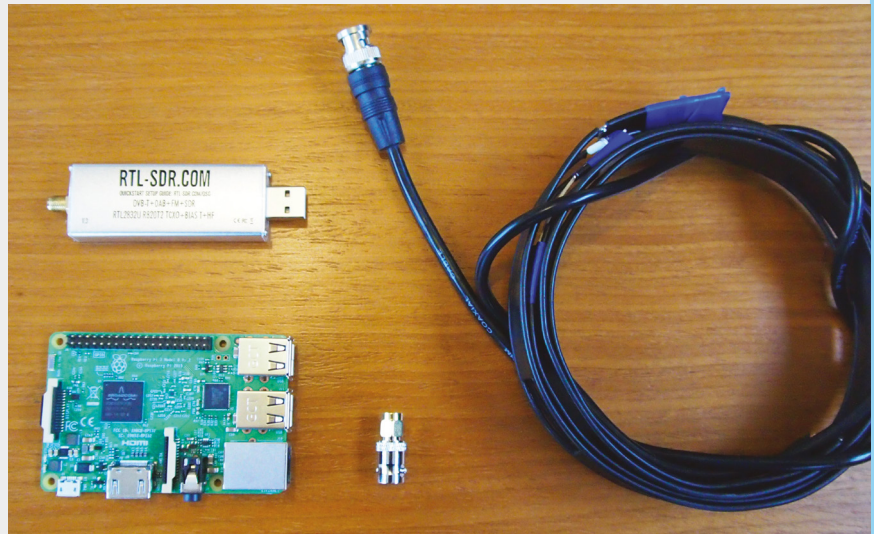
The simplest form of a SatNOGS station is one that doesn't move or track and is made from a static antenna, a Raspberry Pi, and a cheap software-defined radio (SDR) dongle. The SDR dongle has become ubiquitous in maker circles as it is an affordable entry item into the world of receiving signals via SDR. Looking at our ingredients (as shown in **Figure 7**), let's explore them a little more before we get started.

While a permanent station may do better connected by Ethernet cable, using the Raspberry Pi's built-in WiFi means we can run this simply with only a power cable. While many have used the cheapest Realtek SDR dongles with success, some people have found the slightly more refined versions can be more stable – a current recommendation is the RTL-SDR V3 which has a better casing for thermal dissipation, and slightly upgraded components. The RTL-SDR V3 is available here:

hsmag.cc/jbye0t.

The classic antenna recommended for a static SatNOGS setup (see **Figure 8**) is a 'turnstile' antenna; commercial models are available, such as the Wimo TA-1, but people have designed and built lots of different static antennas for different frequencies and with small budgets – check out the tutorial 'Make a Slim Jim antenna' on page 112. In order to set up a ground station, one of the first tasks we need to do is set up an account on network.satnogs.org. Registering on the site then gives us a dashboard where we can begin to set up a station. Click to add a station – we then need to supply it with some basic details (see **Figure 9** overleaf): a name for the station, a location in latitude and longitude (Google is your friend here!), and the elevation of the station above sea-level.

You need to decide what frequency your station is going to cover; the most common ranges are UHF and VHF, which would require different antennas, but either range has a huge number of objects you can schedule to observe. Many people opt for VHF, as this includes the frequency range for a lot of the different transmissions from the ISS, so we are going to choose VHF as well. You also need to add a minimum elevation value – this is the minimum angle that a satellite must be in terms of height for your station to see it – if you aren't sure, either ask for help on the forums, or leave it for now at the default 10 degrees.



Having filled in the boxes to create the station (leave the 'this is in testing' box ticked for now), you should now see a ground station entry has been made on your account (see **Figure 10**). You will see (even though it isn't set up yet) a list populating underneath the entry with 'Pass Predictions', which are things you could schedule to observe once you are up and running. Before we leave the website, we need to make a note of the number assigned to the ground station, and also our own personal API key – which we can find in our dashboard by clicking the API key button. These two pieces of information are what will ultimately connect our ground station hardware to the website account.

The next task is to sort out the Raspberry Pi. You can find the current custom SatNOGS image here: hsmag.cc/lnwYxT.

Flash this to your microSD card as you would for a regular Raspberry Pi setup – the free app Etcher, for example, is a simple tool that allows you to flash an image to a card.

Once done, boot the Raspberry Pi, and you can either SSH in to the Pi, or connect a keyboard and monitor and interact with the setup that way. The →

Figure 7 Parts for a simple SatNOGS station. Set up in a roof space with a cheap DIY Slim Jim antenna, this setup has received signals from space for less than £75

Figure 8 A permanent outdoor static station with a turnstile antenna and waterproof enclosure. Power and network connection are both supplied via a Power over Ethernet (PoE) cable



Figure 9 The website side of setting up a SatNOGS ground station is very simple

SatNOGS NETWORK

Home About Observations Ground Stations Community Wiki

10:49 UTC

Edit: 538 - MW6CYK_HS_TESTING

General Info

Name: MW6CYK_HS_TESTING

Description: Test station for an antenna experiment

Location

Latitude: 53.177 Longitude: -4.05 Altitude (ASL): 98

QTH Locator:

Settings

Minimum Horizon: 10

Antennas: VHF Vertical 135.000 MHz - 148.000 MHz

Target Utilization: 10

Testing? ☒

Submit

FINAL SETUP

Having rebooted, we need to run another setup tool to update packages and set up the SatNOGS client that will run on the Raspberry Pi. SSH back in to the Raspberry Pi and run the command `sudo satnogs-setup`. We should see a menu like the image here. You will need the station number, your API key, and the location details you added into your dashboard on the SatNOGS website. We will add 'rtlsdr' to 'define our RX device' and will select 'NO' as the answer to 'Enable_Hamlib_Rotctld' – as we aren't

building a rotating station, we don't need to add any details to the final line of the setup menu. There is a more detailed walk through this menu and setup, if you need it, on the SatNOGS wiki page here: hsmag.cc/RNurVq.

We won't make any changes to the advanced setup, so let's apply the changes we have made and the Raspberry Pi will configure itself; this may take some time, so it's a good time to make a cup of tea! Once it has applied the configuration changes, click back to exit the setup tool.

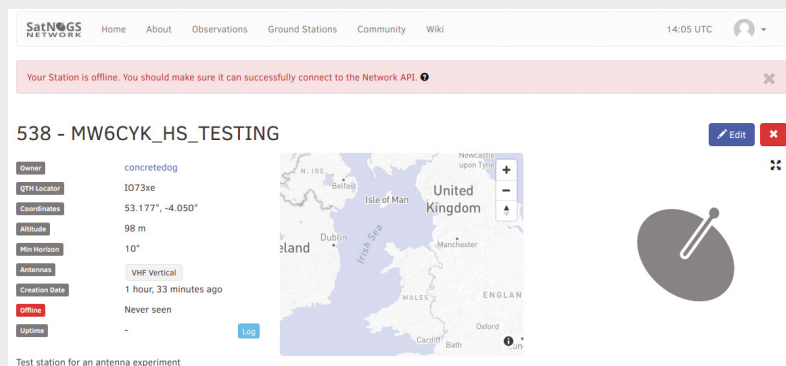
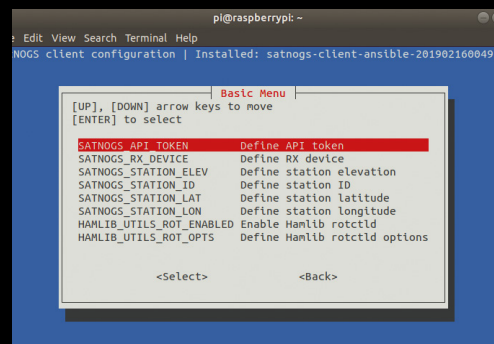
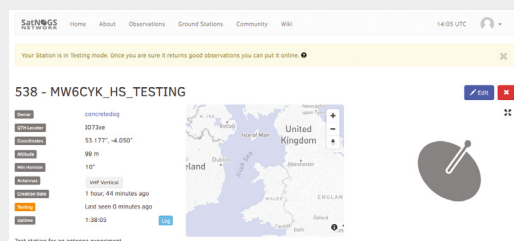


Figure 10 When you first set up the website side of the station, it will appear as offline – don't worry, this is correct until your hardware connects to the website

Figure 11 If all has gone well, you should now see your station is online!



first things we need to do are not SatNOGS-specific, but are the usual things we do when setting up a Raspberry Pi. We need to set up a different password by running the `sudo raspi-config` command. Once you've set a password and expanded the file system, it's also useful to set the time zone to UTC, as this is used throughout the SatNOGS network. If you want to run this test station wirelessly, then you need to configure your network connection at this point. If you are connecting via an Ethernet cable, then you

don't need to do anything else. Apply the changes and reboot (then see 'Final setup' box above).

Now, if we go back to our dashboard on the SatNOGS website (perhaps wait a few minutes and click Refresh), we should see that the station is now online (**Figure 11**). We should see an orange spot on the network map showing our proud station in testing. Being in testing means that only you can schedule observations on the station, but when you are ready, you can change settings to take it out of testing and then it is fully on the network.

ON THE HUNT

Power down one last time and connect the RTL-SDR dongle and the antenna, then reboot – you are now ready to hunt satellites! Scheduling observations is as simple as selecting passes from the list and clicking Schedule. There may be drop-down choices for different transmitters to listen for on the same satellite, and other choices, but essentially you click Calculate to create the observation and then Schedule for the job to be created and sent to the queue for your station. There are hundreds of satellites to try to observe, so don't worry if you don't understand what any of them are – in the pass predictions list, if you click the name of a satellite you will get a pop-up with information about it. For a more detailed walkthrough of scheduling an observation on the SatNOGS network, check out this blog post: hsmag.cc/TtgIDG.

After the time of the pass, return to the observation page and, hopefully, you should see some signals. Don't worry if your first few

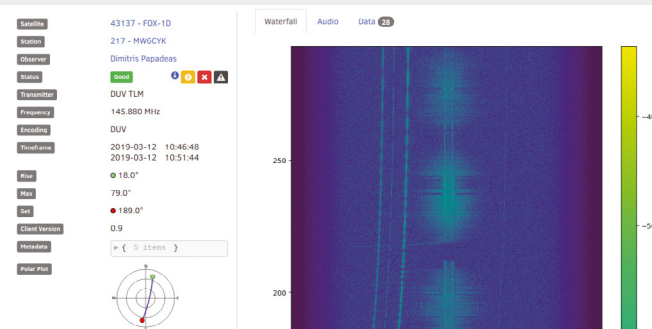


Figure 15

A returned observation: we can view the main waterfall tab, listen to the audio tab and some satellites may have a decoder built into the SatNOGS system providing data automatically in the data tab

observations aren't successful: try at least a dozen observations before making any changes, as there are many reasons that a signal may not be picked up; indeed, the satellite may not even have been transmitting. If you have received a signal, you should 'vet' the observation as good; this is particularly important if you have scheduled on someone else's station – etiquette says we should check and vet our own observations. Check out the Slim Jim antenna (page 112) for a link to a successful observation you can listen to.

HAPPY SATELLITE HUNTING!

Finally, it's worth repeating that it's a great idea to join the Libre Space Foundation community forum (or IRC), as it hosts the SatNOGS community channels, and there is a wealth of expertise and help available there from a very welcoming community. If you build a station, go and share your achievement on the forum – everyone will be pleased to see it. →

GOING FURTHER

For more information on the standard parts of setting up a Raspberry Pi, here are some useful links...

For more information on standard RPi setup, see here:
hsmag.cc/ndbEeE

How to SSH in to a Raspberry Pi (note – when using the SatNOGS image, SSH is already enabled on the Raspberry Pi):
hsmag.cc/ROOnEa

Setting up wireless connection on a Raspberry Pi:
hsmag.cc/illLOU

" If you click the name of a satellite, you will get a pop-up with information about it "



LIBRE SPACE

The team of hackers making space more open-source



Throughout this feature, we have looked at various projects of the Libre Space Foundation, and we thought we would end with a little overview of their story, and the amazing work on all fronts they are doing to democratise space.

Born out of the SatNOGS project, Libre Space Foundation is a not-for-profit foundation set up in Greece. It was originally formed around the Athens Hackerspace (Hackspace.gr) but now works globally with a decentralised team and community to democratise space through various projects. Passionate and committed to open-source

methodologies, all projects are publicly held and licensed clearly so that people can use, modify, and contribute back into the designs. We have spoken about its flagship projects elsewhere in this article – namely, SatNOGS and the UPSat CubeSat that flew on Low Earth Orbit – but it also has projects and developments around other space themes.

The Libre Space Foundation back story is of interest as well, as it all stemmed from the development of the SatNOGS project. SatNOGS was conceived and early development took place at a 'Space Apps' challenge event in 2014 in Hackspace

WORKING PARTNERS

Currently, Libre Space Foundation has some funding via the European Space Agency's ARTES programme to deliver the SDR makerspace project – a programme developing, again using open-source methods, various solutions to identified themes around software-defined radios for space applications. They also partner on delivering the incredibly popular Open Source CubeSat Workshop (OSCW), which is now in its successful third year bringing open-source communities working on space projects together to celebrate and share their work.





Left Libre Space Foundation has a selection of boards designed for use in high-power rocketry, and in project areas in which it is keen to expand

Greece. The International Space Apps Challenge is a global hackathon that takes place concurrently over a weekend, with the same space-related themes and challenges being tackled at every location. During and after this initial intense weekend, the SatNOGS originators worked hard on the SatNOGS project, and later that year they won the Hackaday Prize.

JUST DO IT!

After winning some funding from the Hackaday Prize, and exploring their vision of democratising space through open-source methodologies, the core team quickly realised that SatNOGS was just one of many space-related projects that could be undertaken. They needed an umbrella organisation to cluster their various projects, and so the Libre Space Foundation was born. It's now four years later, and Libre Space has a huge pool of contributors from all over the planet working on many and varied themes. Current projects include CubeSat development, PocketQube satellite development, high-power rocketry, software-defined radio development for space applications, developing the SatNOGS network, and more. 

BUILDING THE COMMUNITY

So you want to get involved? Great! The first port of call is to go to the Libre Space Foundation community forum page, introduce yourself and say a little about your skills. Usually, someone will then indicate some area of Libre Space work that might be of interest, or where your skills may be utilised. With an organisation that has so many outputs, a huge range of skills and knowledge is needed, so don't be put off if you aren't a software developer, for example. We see lots of contributions from all different types of people: wiki editing, CAD, graphic design, technical writing, PCB layout, amateur radio, and of course, code in many different languages. Above all, be prepared to ask questions and learn, as the community are always happy to share.

Category	Topics	Latest
SatNOGS Category to discuss all topics related to SatNOGS . Hardware: 1 new, 1 unread, 2 new Software: 1 unread, 1 new	28 / month 1 unread 2 new	V3.1 - sources for North American builder Hardware rotator 5 6h
Gpredict This category should be used for topics related to support and development of Gpredict software.	3 / month	Where are the STLs? Hardware rotator 4 7h
Satellites & Observations Discussions about satellites details and their observations.	24 / month 1 unread	SSTV ISS - ARISS/NOTA Slow Scan TV Event Feb 8 - Feb 10 Satellites & Observations 1 13h

“ The International Space Apps Challenge is a global hackathon that takes place over a weekend ”



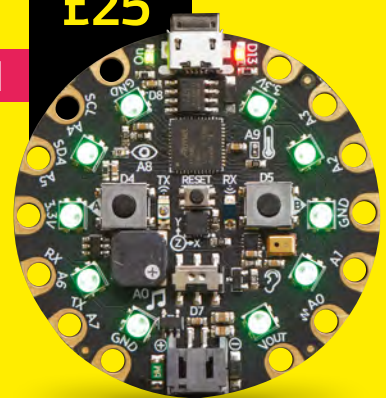
CIRCUIT PLAYGROUND EXPRESS

WITH 12-MONTH PRINT SUBSCRIPTION



FROM JUST
£55

WORTH
£25



**12-month
subscription
from £55:**

- UK: £55 per year
- EU: £80 per year
- US: £90 per year
- RoW: £95 per year

Offers and prices are subject
to change at any time

Visit: hsmag.cc/subscribe

SUBSCRIBER BENEFITS ↓

SAVE UP TO 35% ON THE PRICE
FREE DELIVERY TO YOUR DOOR
EXCLUSIVE OFFERS AND GIFTS

OTHER WAYS TO SUBSCRIBE

Quarterly subscription

**Get your first three
issues from £5:**

- Use the code HS-SAVE at the checkout
- Spread the cost of your subscription
- Try out HackSpace magazine with no commitment

Rolling subscription from just £5 a month:

- Quick and easy to set up
- Cancel any time
- No long-term commitment
- No large up-front cost

DIGITAL SUBSCRIPTIONS ALSO AVAILABLE



Visit: hsmag.cc/subscribe

How I Made AN AUDIO EQUALISER TO HELP ME HEAR

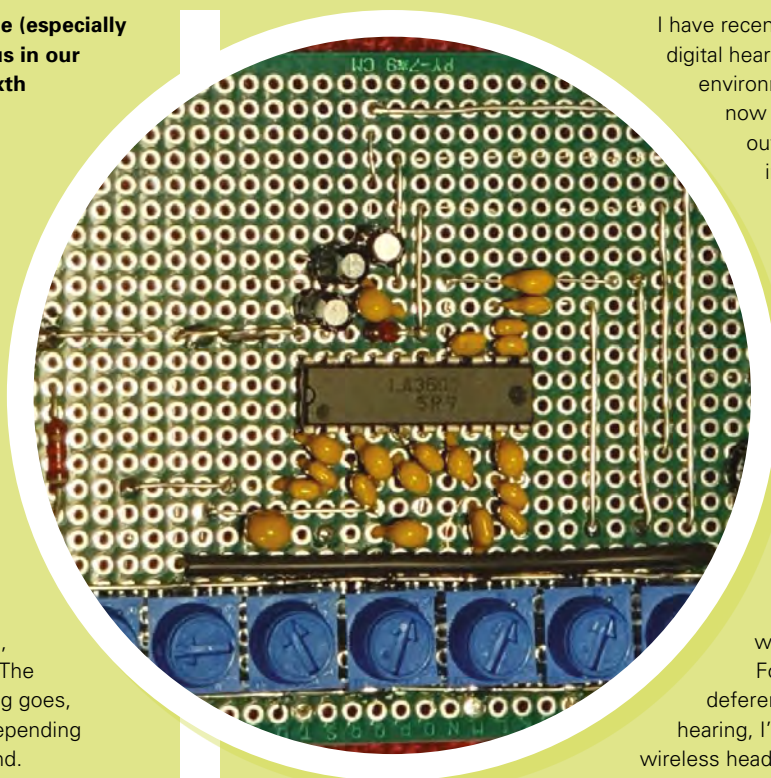
Fighting back against the ravages of time

By Dermot Dobson

Many people (especially those of us in our fifth or sixth decade) struggle to hear

voices clearly from TV programmes. You may have read in the press about the many complaints from viewers who were unable to pick out dialogue from some high-profile dramas last year. Voices are lost in bass-heavy music. This is caused by our ears losing frequency response as we age. Technically known as presbycusis (from the Greek words for 'old' and 'hearing'), the results can be dramatic. The problem isn't just that hearing goes, but that it goes differently depending on the frequency of the sound.

My hearing (as measured by an audiometrist when I got my hearing aids), is not too far off **Figure 1**, though there is a slight difference between left and right ears.



Above ♦
Prototype layout of a single board.
Frequencies increase from L to R

I have recently started using NHS digital hearing aids; they work well in environments with no echoes (I can now hear bird-song clearly while out walking – for the first time in many years), though they don't work well for TV use.

While these hearing aids correct the age-related loss of middle and high frequencies needed to clearly hear voices 'naturally', most of the affordable types (including my NHS ones) have a 'behind-the-ear' microphone, which accentuates echoes from around the room, particularly when there is a wall close behind, with a TV directly in front.

For a few years now, in deference to my wife's excellent hearing, I've been using a pair of wireless headphones fed from the TV – these have the disadvantage of not correcting the frequency response, though boosting overall sound levels. Of course, they also boost sound that doesn't

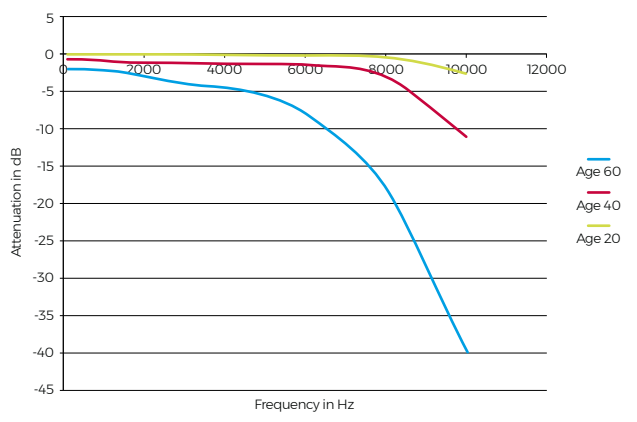


Figure 1 ♦ Common frequency-related hearing loss with age. X axis is frequency in Hz, Y axis is attenuation in dB for three different ages

contribute to understanding dialogue clearly! In addition, by keeping sound levels high, I'm possibly causing further degradation in my hearing, due to the overall high volume sometimes required. While there is an option to use the 'T' setting on the hearing aids with an inductive loop hard-wired to the TV, I would then only hear both channels mixed down to mono.

Clearly, what is needed is a pair of wireless headphones with a way of adjusting the frequency response to more or less emulate hearing aids – but after a good look around, I find these are extremely expensive (and, looking somewhat flimsy, prone to expensive breakage too) or have no more than a bass boost – about the last thing that's useful to us sufferers.

GET MAKING

For my first attempt at an equaliser, I hacked up a simple analogue three-band (bass, middle and treble) Baxandall-type tone control using a pair of op-amps, and inserted it into the 3.5mm cable from TV to headphone transmitter. The result was significantly better, but I still had trouble with higher-pitched voices and louder music.

I clearly needed more specific control than this type of circuit

could offer. Forty years ago I built a ten-band stereo graphic equaliser, that used inductors and capacitors for filters, but it was the size of an average domestic amplifier. Some of the inductors were quite large, and changing frequency bands wasn't easy. I needed something like this, but more compact and easier to tinker with.

I still had trouble with higher-pitched voices



Right ♦ The next iteration of the author is unaffected by time

Far Right ♦ The next iteration of the equaliser with full-sized controls

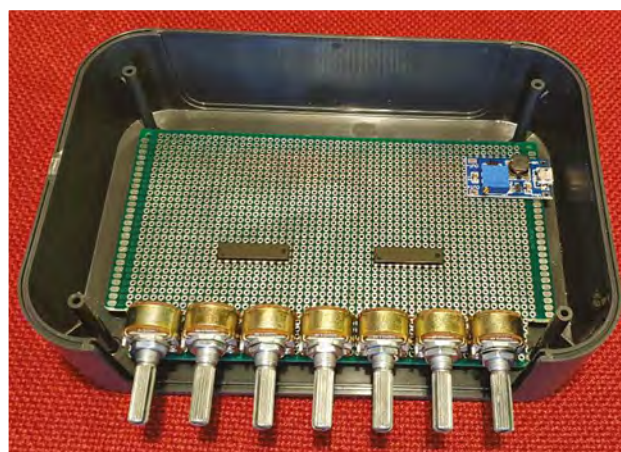
What would I do differently in the future?

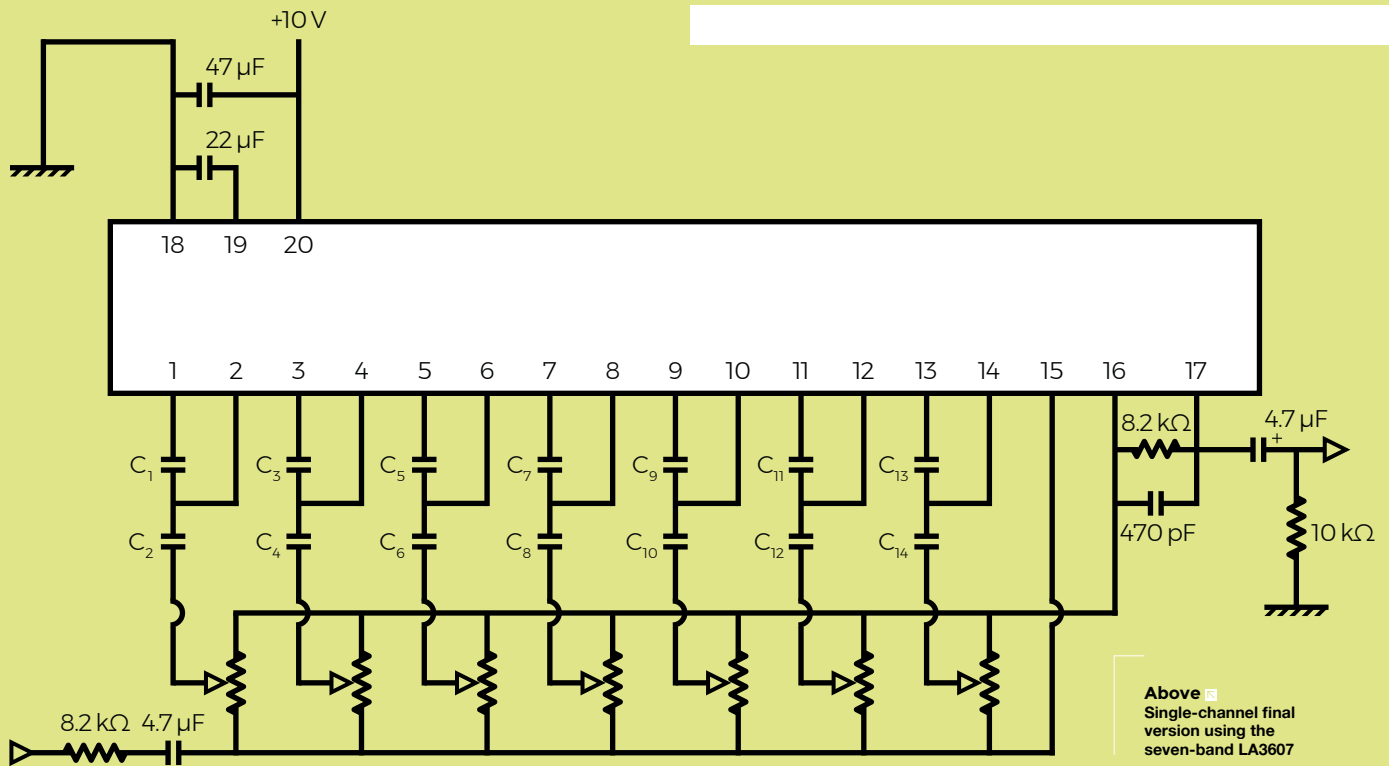
A few friends have expressed interest in getting one of these, having seen the prototype. For those, I will use full-sized stereo potentiometers, so that frequency response can be adjusted without diving into the box to adjust tiny presets. Initially, I opted for individual controls for left and right, as I could see that my ears are of slightly different frequency response. In practice, this proved to be insignificant, at least in my case.

I would also fit a changeover two-pole switch so the equaliser can be switched into and out of circuit instantly. I've found it very instructive for people with normal hearing to listen with the controls set to the *inverse* of the settings I use – they get an understanding of how those with hearing loss hear the world.

If making a number of them, a stereo PCB layout with a balance control would be the way to go. I'd also make provision for an optional buffer amplifier so it could drive wired headphones directly. I'd like to make this easier to construct so as many people as possible can benefit.

Enter the LA3600 series of chips; a five-band 'graphic equaliser' linear chip. Essentially, it has input and output buffers and five stages of 'gyrators' (a rather neat idea that uses op-amps and capacitors to emulate the bulky inductors that would usually be needed for the series of band-pass/band-stop audio filters that I used long ago). →





Safety

One important thing to consider: unexplained sudden hearing loss – particularly on one side only – may be a potential symptom of serious illness, including neurological disorders. You should not rely on this device unless you are confident that you are suffering only from formally diagnosed age-related hearing loss. If not, you should seek medical advice.

Electrical safety: ensure the USB wall wart is undamaged and from a reputable manufacturer (I have a number of them from the bigger cellphone manufacturers); in use, it should get no more than slightly warm.

Because this is intended to operate with wireless headphones, there is no connection between the user and the mains supply. If planning to use with wired headphones (for which you would likely need a buffer amp), I would use a fully isolated boost converter – PCB mount ones are available for a few pounds. These would take the 5V input from a USB charger and boost it to 9V while providing 1000V of isolation. I didn't feel this was necessary for wireless headphones.

The default values from the LA3600 datasheet provide filters at 108Hz, 343Hz, 1.08kHz, 3.43kHz, and 10.80kHz – these values are appropriate for the intended domestic or car audio system use. This improved matters still further, but there wasn't enough selectivity to fully boost the voices, without increasing the other sounds. Fortunately, you can change this by switching in different capacitor values. I got better results by recalculating values to work only in the 800Hz to 6kHz range,

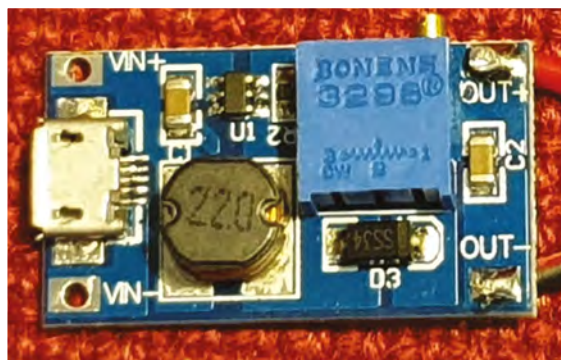
but there was no control over music in the lower registers.

I then started over with the LA3607 chip; similar to the 3600, but supports seven bands instead of five. This made a substantial difference, and I can now hear dialogue clearly, even on programmes where the sound is rather 'muddy'.

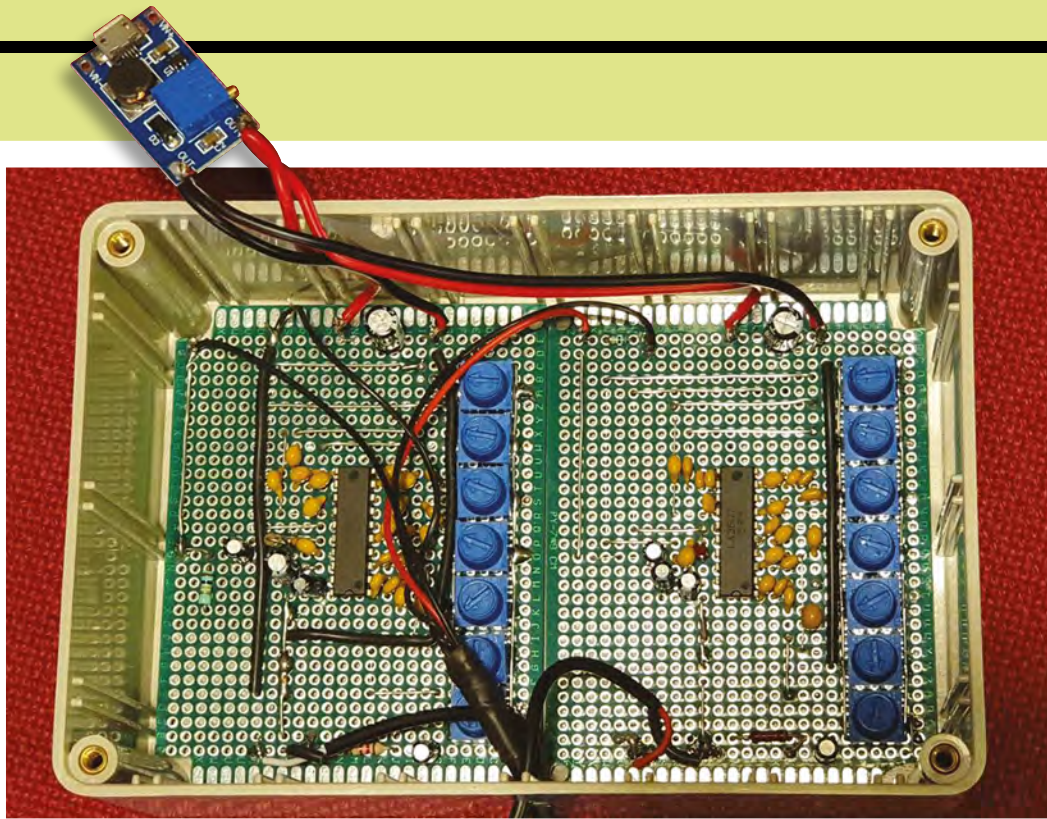
THE FINAL CIRCUIT

The default values from the LA3607 datasheet provide filters at 60Hz, 150Hz,

400Hz, 1kHz, 2.5kHz, 6kHz, and 15kHz. I found the top frequency of little value in emphasising voices, so I recalculated the values to bring the top frequency down to a



Left  Go beyond 5V from USB with a boost converter



Left Both boards mounted, ready to mount the USB boost board on the lid

more useful point. I've found the best results (for my hearing) by selecting the following frequencies (which are approximate, bearing in mind the limited range of standard value capacitors that are easily available): 160Hz, 345Hz, 500Hz, 915Hz, 1.6kHz, 3.4kHz, and 6.25kHz. These are a good balance between boosting voices while still being able to hear a reasonably full range of music.

You can experiment here: just keep the even-numbered capacitors of each filter to be 18 times (approx.) the value of the associated odd ones, and you can change the channel values/spacing. You can see that the frequency scales linearly with the capacitor values.

Filter component values I used are:

160Hz -	C1: 27nf,	C2: 470nF
345Hz -	C3: 12nF,	C4: 220nF
500Hz -	C5: 8.2nF,	C6: 150nF
915Hz -	C7: 4.7nF,	C8: 82nF
1.6KHz -	C9: 2.7nF,	C10: 47nF
3.4KHz -	C11: 1.2nF,	C12: 22nF
6.25KHz -	C13: 680pF,	C14: 12nf

Depending on your requirements, you may care to experiment with some other values, perhaps skipping one or more of 160Hz,

345Hz, or 500Hz and adding compensation for higher or lower frequencies:

75Hz -	56nF,	1000nF
5KHz -	820pF,	15nF
7.5KHz -	560pF,	10nF
11KHz -	390pF,	6.8nF
16KHz -	270pF,	4.7nF

Note that you will see a degree of interaction with filters when set to be closely spaced.

I also use an equaliser between my PC and speakers

For ease of use, I chose a standard USB charger for power, with a boost converter to get the correct voltage. If possible, I prefer using otherwise discarded stuff, and almost everyone has one or more old wall warts kicking around. The draw at 5V is only around 50mA, measured by a USB power checker, so any type should do the trick.

My build came in at around £14 of parts (plus from my stock of components). The only potential pitfall in the build is making

sure you set the voltage correctly (I settled on 10V) *before* connecting to the load, because the boost converter modules come already set to some random value up to 24V! I also use an equaliser between my PC and speakers; now I hear podcasts and other audio content to be far better.

Of course, the audio does sound rather strange to anyone with normal hearing; a bypass switch could make this a bit more useful for other people.

READY TO LISTEN

For flexibility, I chose to leave flying leads with in-line 3.5mm plugs and sockets, as all of the cheap wireless headphones I have use those connectors. Follow the colours to ensure that

you don't swap over left and right channels along the way.

The easiest way to set up the equaliser is by putting one earphone on one ear only, then adjusting the filters to suit. Most users will probably need a little 'cut' at the lower frequencies and a progressively increasing 'boost' at higher frequencies. Once complete, copying settings to the other channel is a good start, before doing any final small adjustments.



Inside the repair café

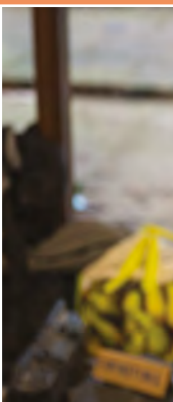
Dead appliances?

Give them a shot at resurrection
at your local repair café

The ethos behind the Repair Café movement isn't new at all. It's there in the 'make do and mend' wartime posters. It's there in the environmental movement. It's present in all of us, whenever we have to buy a new appliance for the sake of a broken bit of plastic that would probably cost a fraction of

a penny to squeeze out of an injection moulding machine. What's changed in the last couple of years is that this feeling now has a name. We now know that it's not just us; we know that, in every large town around the UK, there are like-minded people giving up their skills and time to repair things for free.

We paid a visit to Levenshulme Repair Cafe in Manchester to see what we could see – here's what we found. →





Above ♦ There are other organisations promoting repair culture – one such is Restart, which also lobbies for more repair-friendly laws

Left ♦ Spring is the time of year to replace brake pads and give loose gear cables a bit of a once-over



Even if you don't feel like you have the necessary skills to fix people's broken appliances, you might have the skills to bring other people on board:

"I don't have any repair skills at all," says Jake Lloyd, Levenshulme Repair Café's organiser. "I can change the fuse on a plug, and that's about it. But I'd heard about repair cafés, and thought they sounded really cool. I put an advert up to try and recruit some volunteers, had a meeting; there seemed to be enough interest.

"I put ads on social media, I put a poster up here (in the centre). I went to the first repair café in Chorlton, to find out how they did it. There, I met Alfred Chow and Sue Archer. They're amazing. If they left Manchester, it would fall apart. They keep it running. They're involved in three or four different repair cafés, and so many other community projects."

LOCATION, LOCATION, LOCATION

The Levenshulme Repair Café is in a community centre on a main road. It's easy to find, and easy to give directions to. The location also gets a lot of footfall, and you can see in the big glass windows, so anyone walking by straightaway knows that there's something happening in there that's open to the public. It's pretty much a perfect location.

"The biggest thing is getting the space," says Jake. "We get this place for free. This café is open on a Saturday, but it's their quietest day of the week, so for them, they get more customers coming in. It's the space and the volunteers."

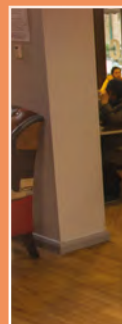
The amount and the type of work that goes on in a typical repair café will vary according to the season. "Bikes are very seasonal," says Jake. "In December, we didn't get a single one; in summer, the bike guys were outside on the street, and there were several of them working constantly.

"We get a lot of electrical items. I reckon the most regular thing to come in would be a toaster. ... We've had a few record players, a few computer problems, people coming in with elbow patches that need sewing onto jackets, and a whole array of household appliances. We try to avoid doing really big things. And microwaves we won't touch."

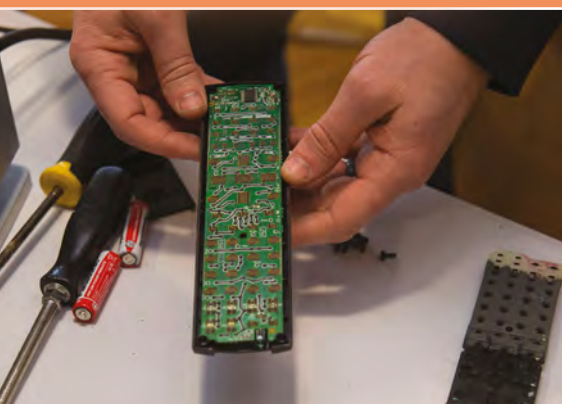
JOIN IN

You don't have to be an elite-level hardware hacker to lend a hand at a repair café. Sometimes the most useful fixes are the simplest.

"This hair-dryer is a great example," explains Jake. "It was brought in in a non-functional state, with a trivial fault – over the years of use, the power lead had flexed and bent so many times that it had just snapped. To many people reading this, that's an obvious fix, but unless you know that you can do that, it would never occur to you.



**It's easy to find,
and easy to give
directions to. The
location also gets
a lot of footfall**



Above ♦ This remote control circuit board was covered in rust – a lot of sandpaper and it might just live to ride again

Left ♦ Broken router – fixed!

Below ♦ Even if the repair hasn't been possible, if the owner has learned something, that's a successful visit



"A lot of the time you've got to go back to basics, to do root cause analysis of what the problem is. We saw a lamp that had been brought in – rather than doing the standard electrical tests to make sure that it was getting good current, the volunteer just removed a load of fluff from the bulb, and it was the fluff that had isolated the bulb from the electricity supply. It worked just fine after that."

Over the course of the two-hour session, 26 people came in through the doors, with bikes, clothing (broken zippers and torn fabric), a couple of laptops, furniture, and lots of electronic appliances. Fourteen of them left with a successful repair, which Jake reckons is below average for the Levenshulme café. "However, people often go home with the knowledge of how they can fix it on their own after, for example, buying a spare part. So, I think that these stats ought to be taken with a pinch of salt."

DIAGNOSIS

With the best will in the world, it's never going to be possible to fix everything. But sometimes it can be just as useful – or very nearly as useful – to give the owner an idea of what they should be looking for, so they can try to fix the issue themselves.

Alfred Chow (@Maker_of_Things on Twitter) tells us: "Just now, we've diagnosed a record player. She took a record player and recorded sound from it, explained the symptoms, and we gave her advice on what she can do. She's going to change the belt on it, clean it all up, and then see how it goes. If there are still issues, she can bring it in."

One man came in with a surround sound stereo that wasn't working properly. After a bit of investigation, the volunteers found the problem: the battery had been left in the remote control for a year, and had leaked and corroded the connections.

"All I can do with it is sand the connectors down. I've done all the cleaning that I can do. It could be that it's corroded a component on there."

"There should be a Dolby surround sound symbol on the screen, and it didn't have it; it just had left and right. So we knew that it wasn't sending out a signal to the speaker."

"We've only got 20 minutes – that's not enough time to be stepping through component by component →

Reasons for fixing things

1/ The planet

From an environmental perspective, repair cafés are a huge blessing. According to the Health and Safety Executive, over two million tonnes of e-waste is generated in the UK each year, comprising six million individual items. Worse, up to half of these items are either in working order, or are easily repairable.

2/ Money

If you're faced with the choice of repairing a big item or throwing it away and getting a new one, you might as well get someone to have a look at it for you. One customer at Levenshulme brought a digital television recorder: "I'm not sure what's wrong with it or even whether it can be fixed, but before I scrap it, I'd rather get it looked at."

3/ Learning

According to Jake, the most satisfied customers are often not the people whose repairs have been successful, but the people who've come in, had a good chat, and learned a lot. We spoke to one woman who came in with a set of fairy lights that had a broken connector. This would have cost at most £2; so, economically, getting it fixed was crazy – when you factor in the bus fare, the time spent waiting in the repair café, and the time it took to diagnose and fix the fault, she was easily out of pocket. But she'd brought three children with her, and forced them to sit and watch as various faults were tried, the break in the circuit found, and the wire twisted together, soldered, and patched up.

"I knew exactly what I'd done as soon as I did it: I knocked it with the Hoover," she says. "I wanted these lot to see that you don't have to throw stuff away."

with a multimeter. There are three other people queueing up, and there's only so much we can do here. Ideally, I'd like the time to get this working and solve every single one. On quieter days we might spend up to 40 minutes, with a bunch of us brainstorming things to try.

"On busier days, the best we can do is recommend that there's nothing wrong with the speakers, but you need a new remote control. Or, you could go home and see if you can get the remote working, because that's where we think the problem is.

FIXING THE UNFIXABLE

As the owner of one broken item tells us, "If you took it somewhere to get it repaired, it would cost £20. It's only £30 for a new one, so there's no point. That's what they're doing, isn't it? That's what the manufacturers want."

And it's true: a lot of simple fixes are complicated by 'what manufacturers want'. In the case of the hair-dryer, the casing was held together by a removable screw, as well as a load of glue. Take the screw out and the things hold together; you then have to try to prise it apart using just the right amount of force, taking care not to crack the plastic.

We've seen circuit boards of commodity hardware potted in resin to stop the user exchanging components. There may be a legitimate business reason for this: potting components can reduce hum in audio equipment, or protect the IP of high-grade circuitry that the manufacturer wants to remain secret. Call us cynical, but we think most of the time this technique is used just to make items unrepairable.

Most insidious was a toaster held together by non-standard screws. Rather than use a Phillips or Pozidriv screw to hold together a cheap bit of kitchen kit, the manufacturer had chosen to use a two-pronged screw, accessible only if you've got a tool that almost nobody owns. A set of interchangeable screwdriver heads costs £20 from RS Components, but for most people, it's more economical to throw the toaster away and pay £5 for a new one. This is an example of an easily fixable appliance being made deliberately harder to fix, for no good reason at all. But overwhelmingly, even if you can get close to a result, you're winning: "In the case of the stereo, the customer had redecorated his living room around the speakers – replacing them would mean not just buying a new stereo, but also ripping wires out of the walls and replastering. To find out that the only thing wrong was the remote control represented a potential saving of hundreds of pounds and a load of effort." ▣

A lot of simple fixes are complicated by 'what manufacturers want'



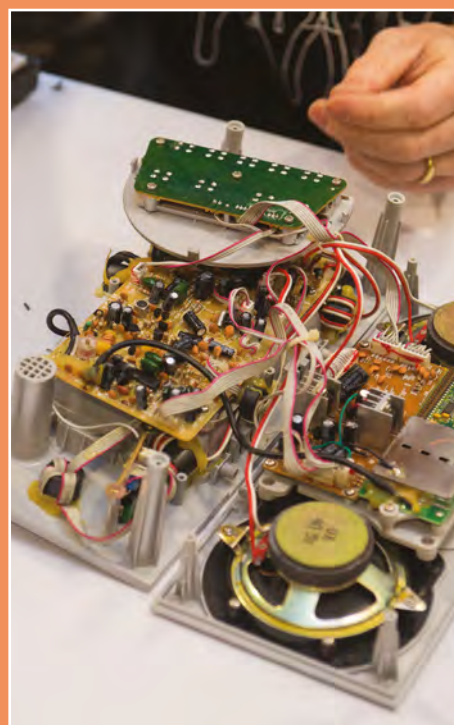


Get fixing!

Want to set up a Repair Café near you? Hold on! See if there already is one first, and offer to lend a hand. If you know how to strip wire, sew, or use a drill, you're already qualified.

If there isn't one near you, get in touch with Repair Café international (repaircafe.org). For a voluntary €49 fee, they'll give you a starter pack that includes various branding, signage, and the all-important disclaimer forms protecting you and your volunteers from being sued in the event that you can't fix a problem (or inadvertently make things worse).

Sharing your skills doesn't have to take over your life either: just two hours a month can make a massive difference. You might even get a few free cups of tea out of it.



Above ♦
Gas-fired soldering irons are essential in a community space without many power leads

Left ♦
One surround sound stereo – not fixed, but in 20 minutes the owner knows what he needs to do next to get it fixed

Right ♦
All this radio needed to make it good as new was a new switch costing pennies



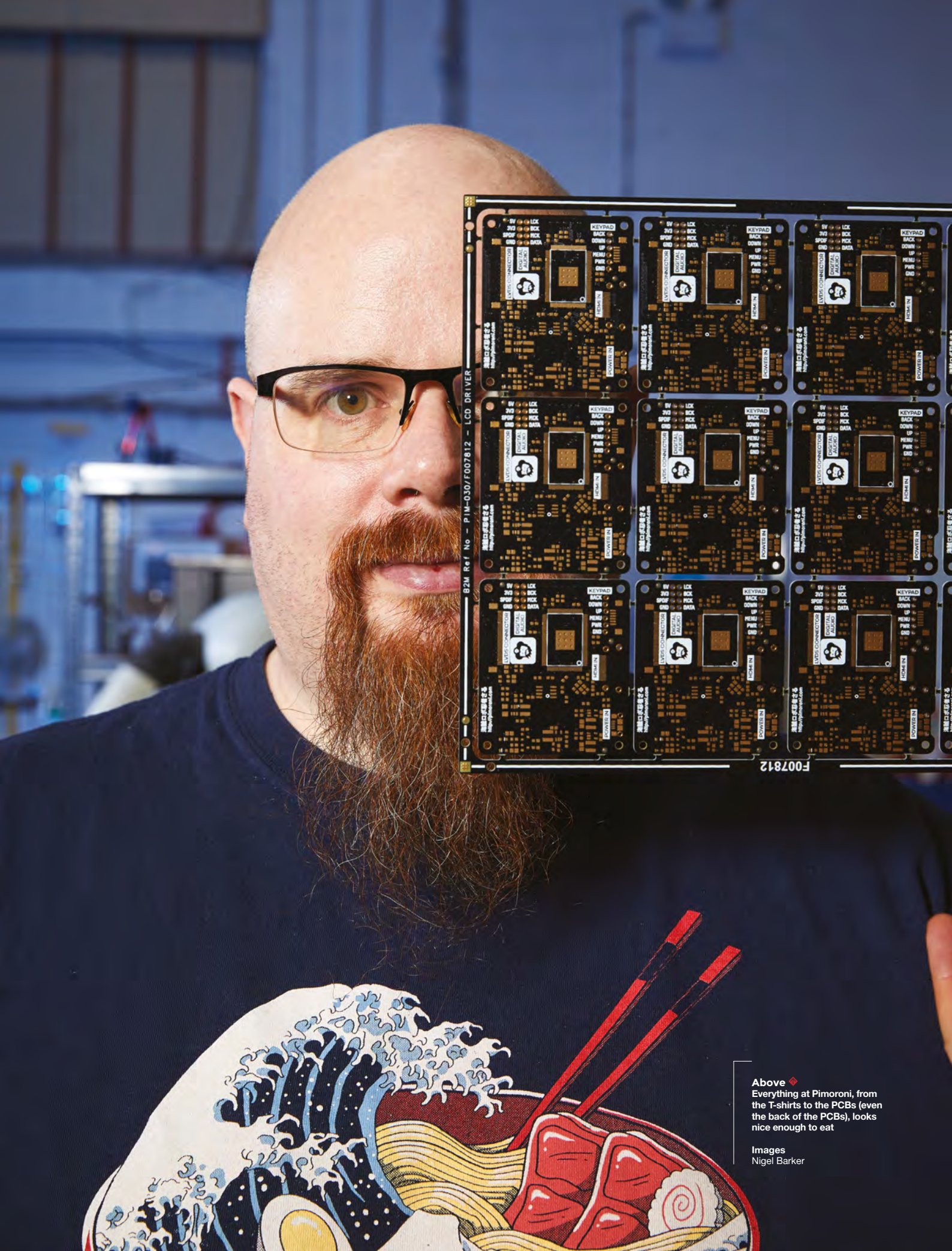
HackSpace magazine meets...

Paul Beech and Pimoroni

On a mission to make electronics accessible and fun

If you've been making things with a Raspberry Pi, it's virtually inconceivable that you've not come across the wares of Pimoroni. This company, nestled in the seven hills of Sheffield, creates electronics for makers and tinkerers, making it more accessible for the average curious person to make something interesting.

We arranged to visit Paul Beech, designer, guru, and Pimoroni's joint boss, got to watch the hypnotic pick-and-place machine, and had a bonus trip to the pub for a chat with a few of the Pimoroni crew: Tanya Fish, Phil Howard, and Sandy Macdonald. →



Above ♦ Everything at Pimoroni, from the T-shirts to the PCBs (even the back of the PCBs), looks nice enough to eat

Images
Nigel Barker

HackSpace Magazine So, Pimoroni: who are you, and what do you do?

Paul Beech We're a bunch of geeks and we think of, design, manufacture, sell and distribute, fun, beautiful electronics. We spark joy. Marie Kondo has given a lexicon for that now.

HS The name suggests Raspberry Pi-based stuff.

PB It does, but it precedes the Raspberry Pi. The name came a good year or two before the Raspberry Pi.

Most of the stuff we make and ship is for the Raspberry Pi. We've got a lot of loyalty and a lot of history with the Raspberry Pi.

The only reason we diversified is because micro:bit was out there and did the microcontroller end of things, which was important. We are geeks; we like other things. We like general electronics, we like making quadcopters, we like making music. We like maths. Anything around that, we quite happily get into as well, as long as it makes sense for us to do it.

It's very easy to say, "We do just Raspberry Pi," rather than making things for the whole gamut of single-board computers that are out there, and offering support for a load of different platforms.

It's the only single-board computer we do. We've kind of had other companies bribe us and say, "Will you please make a case for this," or add-ons for this, but no. It's going to be here today, gone tomorrow. The Raspberry Pi abides. It's good having that single platform.

HS Does that mean you don't do anything for Arduino?

PB Not yet. As of last year, we took the approach that there are now four core platforms in the makerverse: Raspberry Pi, micro:bit, Arduino, and Feather/Adafruit. Feather is now a platform

because Adafruit has done it so well and backed it up with CircuitPython.

They are the big four; anything else is a little bit fringe. The other one you might include is Teensy, but that's very focused.

HS What does your typical customer look like?

Tanya They've got a smile on their face; they're enjoying what they do.

PB I think the customers we have now are the customers we want to have. We've only had to fire a customer on a handful of occasions. Mostly, the experience people have is when something goes wrong, or you don't understand, you immediately go back and rant at them, saying that this is

“

As of last year, we took the approach that there are now four core platforms in the makerverse: Raspberry Pi, micro:bit, Arduino, and Feather/Adafruit

”

somehow wrong, or you've ripped me off, or this is faulty. You come out with anger, because you expect to be basically given short shrift by the company; you expect to be lost in the customer support maze. We don't do that.

When people come to us angry, we respond really pleasantly, with empathy. Mostly it only takes one email for the grumpy people to come around; maybe two. But if after that they still continue to be grumpy, after we've tried to make them happy, then maybe we'll consider firing the customer.

But almost 100%, we have the nicest customers. There's one fellow who kind of rants at us, and every minor thing we got wrong he'll pick it up and say, "This is not right". And he can seem grumpy, but actually, he cares. He's not ranting to

other companies, because he doesn't care about them. It's just his personality.

HS So it's just an ongoing relationship of criticism?

PB Yes. On some tiny, nitpicking stuff. And we love it.

Tanya The day that he doesn't complain is the day that he stops being interested in what we do.

HS Or, you send someone round to check up on him.

PB It's happened. There was somebody who went strangely quiet after interacting with us loads as a company, and they stopped interacting with us. But they live overseas and we weren't able to actually go and check on them, so we asked around, the community rallied, and it turned out that they had just been really busy. There have been a few instances of that.

Phil Our best customer is friendly, understanding, and doesn't mind fixing code here or there. Someone who raises a pull request. Which means instead of going, "This is broken, can you fix it please?" Going "This is broken, I've fixed it".

HS Do you get a lot of that? With it being open?

Phil Not as much as I'd like. Which is still pretty decent. I had a recent one where a guy had gone through the entire repository of code for the Display-O-Tron HAT I think it was, and fixed everything. He'd gone through the whole of a slightly neglected repository and fixed everything.

HS That's lovely. You wouldn't do that for Apple.

PB We appreciate it. A lot of us have come from the background of freelancing, where you don't want to take the →



Above 
"[Circuit boards]
should look pretty,
they should be a joy
to use."



Left
"Pimoroni really took off with the HAT format. That's when we realised: we're really doing this"



community, or people doing free work, for granted.

Although we have people on salary, we don't have an overtime rate, so if they do out-of-hours work, they do get time in lieu, or get paid for it, even if they're salaried. We've been through that.

Work-life balance means that you disincentivise too much overworking. So I think, for that reason, we're reluctant to lean on the community too much.

The question was who is the ideal customer. Someone who enjoys what they got, and uses it. Doesn't let it gather dust.

HS And what do they use it for?

PB [Much laughter.] You can't narrow that down! I think Keybows are the most interesting thing recently. Have you seen Keybow? It's a faintly ridiculous project. You can get a miniature numeric keypad from Amazon for about £15. You can get mechanical keyboards for probably twice as much as that.

The Keybow is a keypad with a whole computer in it. There's no good reason for it. It's the footprint of a Raspberry Pi. I was into mechanical keyboards at the time and I looked at the Pi and thought, "There's just enough room on that to fit a 4x3 matrix of keys."

And so I went at it, and did it, and had a few people shouting at me the ways in which it was electrically wrong, until I had something that fitted in that footprint. Then I chucked it at Sandy, Phil, and Jamie, and they took it and ran. It was ridiculous but beautiful. Then Sandy, who was the driving force, said, "Right, I'm going to put Lua on this and make it programmable," and did a ton of work to make it a fantastic experience for people.

On the back of that, people have started making crazy customisable keyboards.

Sandy The best one was done by a woman who's a blogger in the States. She was getting loads of creepy guys who were DMing her on Twitter. So she hacked the keyboard to be a kind of auto-reply device for all these creepy guys.

PB About a year and a half ago, there were these two people who were sailing their boat around the Pacific. They'd just completed it and got back to Japan. But, while they were on the journey, they'd picked up a Raspberry Pi and some of our stuff. They were using the pHAT BEAT as their home entertainment system. They had a coding platform, and they started writing what looks like a text editor — like nano — a text editor, command-line input, but for music.

"The question was who is the ideal customer. Someone who enjoys what they got, and uses it. Doesn't let it gather dust"

Sandy The unique thing about it is that every letter of the alphabet is a function, so A is add, F is if, C is clock, D is delay, J

is jump. You set it going and it shows the cursor going across the text editor, 80x20 columns or whatever, just updating things, moving around, and making weird music. They've attached a Keybow, but they've also made it a chording keyboard, so you hold it in one hand and press multiple keys and you can type with one hand. They're coming from Japan to Sheffield for the Algorave in May, which is part of the AlgoMech Festival.

Tanya I think that's one of the nice things that we do: we enable people to see through their weird ideas. I find everything that everyone is doing fascinating. I go out and meet customers. I don't see the support side of it so →

INTERVIEW

much, I see them in person, and people are always asking me, "What's new, what can I get next? What can I justify this time?" It's a very tight-knit community, very friendly.

HS You have open-source software; is the hardware open source as well?

Phil Generally no; we have about four or five bits of hardware. But if you give people the hardware and the software so they can flat-out clone it, they will. We don't mind people reverse-engineering it. We want to put the schematics out there, but we don't like how ugly they look.

Not much of what we do is really that complicated. We'd love people to look at what we use, but we'd prefer them to go through the learning process and make their own boards.

I think our stance is that that doesn't enable the downloaded clone things. The reason for that is kind of nebulous; I want people in Sheffield to have jobs. And what we've found when we've released some things in the past – I think the Pibow case we originally released as CC non-commercial – it just got downloaded and cloned.

I like the progress you get with open hardware, but I don't like that it makes it harder to employ people.

Phil Do what you want. Include this licence file so everyone else knows that they can do what they want. With the code that you've done what you want with. It's pretty much what Adafruit does as well.

PB I think in some ways... we build things with other people's chips. Making that chip was the really hard, innovative bit. Putting that chip into a product is a creative act, is an act of interfacing and arranging things in a way that people can use them. We're not making the fabs, we don't have an 8 nanometre fab or whatever it is. Where are we now?

40 nanometres is a bit old-school now on chips.

HS Right, you've lost me now with that terminology. When you talk about a factory in terms of nanometres, what do you mean?

Phil It's the scale of a transistor on a chip die. So as transistors get smaller and smaller, the measurement that measures them gets smaller and smaller... you can pack more of them into a given area.

But that scale is limited by technical factors such as the processes used to make chips – and obviously, the smaller it gets, the fewer atoms you have making up the transistors. You reach a point where you can't get any smaller; it's physically impossible.

The most interesting things
are created by people
who learn programming
by happenstance, rather
than by professional
programmers

When you make something smaller, you reduce the amount of physical space with a transistor, which reduces the amount of electricity it consumes, which enables you to put more stuff in a smaller space, using less power and producing less heat, which makes the processor better and more efficient from a consumer point of view, but a lot harder from a manufacturing point of view.

PB The bulk of the business [of chip manufacturing] is processors for mobile phones and computers. They are the cutting edge, and they are constantly making new multibillion point fabs for each new process that Intel or ARM licensees

are using. Raspberry Pi is clever in that all the fabs that use the older process, the less accurate process, the less efficient process, the cheaper process, they suddenly have no work, because all the business has moved on. Raspberry Pi was using 40 nm fabs when 28 nm was the cutting edge. That's the essence of it.

But the whole design process, that's the hard, heavy investment bit.


And then a company will release a chip that does accelerometers, or does distance sensing, or does optical sensing... we take that and package it in a way that's useful. Us, Adafruit, and SparkFun. None of us are making our own integrated circuits. We're not at the cutting edge of human progress. We're there to make people aware of what's out there.

Phil We're at the cutting edge of making it available to people who don't know what the hell it is. The most interesting things are created by people who learn programming by happenstance, rather than by professional programmers. If your profession is horses and you learn to code, you'll do something that benefits the horse industry.

You've done a lot more than most programmers who just sit around and make frameworks that benefit other programmers, who make more frameworks that benefit programmers who make more frameworks...

PB And that's a beautiful thing, because if people understand what's inside stuff and what they need, then they're not bugging the people making, say, really trivial devices that don't push us forward, don't get us on to other planets, don't get us to Mars. And if their time's taken up by making a shinier phone, they're not doing the stuff that's going to save the planet, or get us to new planets. So by raising the water level of knowledge, by spreading it to more people, we're raising standards all round. ■



Left  Look out for the new version of the Picade with extra flashy buttons, coming soon



ROPE



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.

Use the twisted strands of fibres to string your way to the maker's den

The rope is one of the oldest pieces of tech – it even predates the wheel and the axe. It's the one thing that was pivotal to the construction of the pyramids of Giza, the Wright brothers' maiden flight, and even Hillary and

Norgay's ascent of Mount Everest. Construction-wise, a rope is a bundle of flexible fibres that are braided and twisted together in order to increase its tensile strength. The twists help keep the rope together, but more importantly enable it to more evenly distribute tension among the individual strands.

Ropes were originally made by hand using natural fibres, and it was the Egyptians who first developed special tools to make rope. They used a wide range of fibres: water reed, date palms, flax, grass, papyrus, leather, and even camel hair. The craft of making rope spread across Europe and Asia over the next several thousand years. Leonardo da Vinci conceptualised a rope-making machine and, by the early 1800s, several machines had been built and patented, inspired by his sketches.

Natural fibres were used to make rope until the 1950s when synthetic materials, such as nylon, became the popular replacement. While the

exact process of making rope has undergone dramatic changes, the overall concept remains the same. Strands of fibres are first formed into yarn, which is then twisted, braided, or plaited according to the type of rope being made. The diameter of the rope is determined by the diameter of this yarn, the number of yarns per strand, and the number of strands or braids in the finished rope. Modern ropes utilise many newer synthetic materials to improve their strength and make them lighter. Rope can also be made from iron and steel. Instead of rope though, they are referred to as cables, and are used in elevators, cranes, and bridges.

A rope is broadly divided into four categories based on its diameter. Ropes under 5mm in diameter, such as clothesline and sash cord, aren't really considered ropes. Ropes with a diameter between 5mm and 13mm are deemed fit for light-duty work. True ropes are the ones with a diameter between 13mm and 38mm. Thicker ropes, with diameters in excess of 38mm, are used for heavy-duty work such as mooring ships. In addition to diameter, general purpose ropes are sold by their tensile strength. Different ropes go through different types and amounts of testing, based on their intended purpose. For instance, ropes designed for critical applications, like rescue work and rappelling, are tested more thoroughly and have a finite service life.

But ropes can be used for a lot more than just pulling and tying knots. Let our dauntless makers show you the ropes!

BOOK-SHELF

Project Maker
ED LEWIS

Project Link
hsmag.cc/cwRqsu

Right

Unlike all his other projects, things didn't go sideways during the build of the bookshelf. What happened afterwards is another story



Simple suspension rope bridges are overpasses that have no piers or towers and whose deck stoops in a downward arc. Ed Lewis has used the flexibility of a lightweight nylon paracord to replicate the same in a bookshelf. He chopped a wooden board into several 2" pieces and a couple into 2.5" ones. He drilled a pair of holes into each, and fed paracords through them to connect them

"THE KNOTS AT BOTH ENDS WERE SINGED WITH A LIGHTER INTO A GOOEY MASS"

all, using three steel washers as spacers. The knots at both ends were singed with a lighter into a gooey mass to make them extra secure. The deck of the shelf was hung to the wall using a pair of shelf holder brackets. Ed cautions us that, while the bookshelf looks cool, it is also very wobbly: "Grabbing any one book out is sure to make all the others fall on their sides. Pretty much the same with adding one in. I found it to be pretty satisfying after a while, because it was fun to get the balance just right by rearranging with the heavier books in the middle. This work would go out the window whenever anyone wanted to check it out and make it all tumble, but guests always found it fun to put back together again." ▣

WINDPROOF LIGHTER

Project Maker

MARCUS DUNN

Project Link

hsmag.cc/OllEyl



To light a cigarette or a pipe in the strong winds on deck, sailors often use what's known as a 'Shepherd's lighter'. It works in the wind since it doesn't use conventional fuel and doesn't ignite a flame. Instead, it sparks an ember on a piece of cotton rope. They're great for hikers who need to start a fire in the outdoors. Lighter aficionado Marcus shows you how to build one. Marcus tells us that all it takes is a, "couple of common copper plumbing parts and some natural fibre rope" to create a rope lighter, and he has illustrated easy-to-follow instructions in his Instructables page. The

" IT WORKS IN THE WIND SINCE IT DOESN'T USE CONVENTIONAL FUEL "

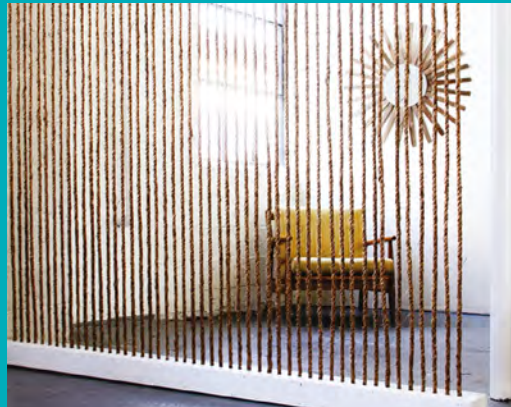
only issue that Marcus had was the diameter of the rope he used was smaller than the copper tube that acted as the barrel of the lighter: "Back when sailors used to use these lighters, they would stick a fish hook through the top of the rope and then connect that to the top of the lighter. I didn't want to take the chance of getting hooked myself, so I went down a different route." He instead added a thumb screw into the body of the lighter: "If you want to ignite the rope, you just loosen the screw, push the rope up, and then tighten it again." ▣

Left ♦

Sourcing a pure cotton rope turns out to be the biggest hurdle for many attempting to replicate the build

ROPE WALL

Morgan is a designer who runs a decor business. She was roped in by her friend and fellow designer Shelly Leer, who was setting up a new workshop. Shelly needed help with defining areas for various different workspaces: “Functionally, this one big room had to be used for many, many purposes and still be open, accessible, and of course look awesome.” Morgan’s solution was to build vertical jute ropes division walls, instead of traditional partitions, primarily because the place was a rental: “A plus when using this type of jute rope is the added benefit of bringing in a chunky, graphic, warm texture to an otherwise cold and hard industrial room.” The simple construction process involved custom wooden boxes with holes drilled in them for the ropes, and Morgan has close-up images of the process on her blog. ▣



Project Maker
MORGAN SATTERFIELD

Project Link
hsmag.cc/TCyOaB

Above ♦ Although the build is simple and economical, the real cost of the project, according to Morgan, is “labour and time”

TRENDY TABLE

Project Maker
SARAH GOLDBERG

Project Link
hsmag.cc/Bnlfbg

Sarah is an avid DIYer and blogs her builds and crafts at *WhileTheySnooze*, where she literally scribbles when her kids are in bed. As a participant in *Creating with the Stars* show back in 2013 where the theme was upcycling, she turned an old discarded tyre into a trendy table that was good enough to shoot her into the next round. She used circles cut from plywood to form the table top and the base for the tyre to sit on. She then glued 350 feet of light-coloured sisal rope, along with the darker natural manila rope, to the tyre: “Turn the table over, start around the base, and work your way to the top of the table.” Sarah’s detailed the process in her blog, and offers suggestions on how to avoid the pitfalls she encountered during the build. ▣

Right ♦ Sarah created wooden legs for the table, but suggests you could use an upside down flower pot, or buy pre-made legs from IKEA



HackSpace

TECHNOLOGY IN YOUR HANDS

Download the app

Out now for smartphones & tablets



£2.29

rolling subscription

or

£26.99

subscribe for a year



Available on the
App Store



GET IT ON
Google Play

FORGE

HACK | MAKE | BUILD | CREATE

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

PG
90



MAKE SOAP

Keep yourself clean with personalised cosmetics

PG
94

MAKER'S TOOLBOX: HEAT GUN

Bend, shrink, and melt your way to better makes

PG
96

ISS TRACKER

Watch humanity's extra-terrestrial home

PG
100

LIGHT UP JACKET

Fashion that you can't find on the high street

PG
74

SCHOOL OF MAKING

Start your journey to craftsmanship with these essential skills

74 Electronics: Op-amps

80 Make a PCB

84 CircuitPython



PG
106

BINARY KEYBOARD

Who needs 96 keys when two will do?

PG
110

DIY ANTENNA

Catch radio waves as they try to sneak past

Electronics 101.10: Op-amps

The arithmetic engines of analogue electronics



Dave Astels

daveastels.com

Dave's career started in the 8-bit days, with the Z80 and 6502, and he's been working with computers ever since. Check him out at: daveastels.com and learn.adafruit.com

Up until now, we've been learning about discrete components: resistors, capacitors, inductors, diodes, transistors. We've looked at some things we can do with them: filters, amplifiers, and such. In the last issue, we had a fairly deep look at integrated circuits. In this issue, we'll mash that all together by looking at a category of integrated circuit devices that let us do what we were doing before, but more easily and more effectively: operational amplifiers, or op-amps for short.

An op-amp is essentially a sophisticated differential amplifier – they have two inputs, and amplify the difference between the inputs. A key feature of an op-amp is that it has little impact on whatever is connected to its inputs, due to having a very high input impedance. Conversely, it has a very low output impedance, and is able to supply significant current to its load. It also has an incredibly high gain (typically 10^5 – 10^6 : a fraction of a millivolt difference between its inputs can cause the output to go from one extreme to the other.

OP-AMPS ARE HIGH GAIN DIFFERENTIAL AMPLIFIERS

Figure 1 shows the functional block diagram of one of the four op-amps in the LM324N, which we'll be using. Since the input stage is a differential amplifier, an op-amp has - and + inputs. These are better thought of as inverting and non-inverting inputs, respectively, and not negative and positive.

Figure 2 shows the pinout of the LM324N, and **Figure 3** shows the standard symbol for an op-amp.

While many op-amps often require dual voltage power supplies (e.g. +12V and -12V), there are some that work with only a positive supply voltage. For the hands-on circuits, we'll be using the LM324N, which you can power with from +3V to +32V. Unless specifically noted, the described circuits assume dual (i.e. + and -) supply op-amps.

GOLDEN RULES

As stated in *The Art of Electronics*, there are two rules that an op-amp adheres to:

1. As given above, an op-amp has a high gain: the output jumps to one extreme or the other based on the relative values of the two inputs. It does this as it attempts to make the voltage difference. Due to this, the key to op-amp circuits is to feed this output back to one of the inputs to either enforce or reduce that input difference.
2. The inputs are high impedance, which means they draw no current, and so have a negligible impact on the connected circuit.

We'll see how these two features can be made use of later.

BASIC CIRCUITS

We'll start with some basic circuits that show the essence of op-amp use.

Figure 4 (overleaf) shows a basic inverting amplifier circuit. As you can see, the input signal is applied (through R1) to the inverting input, and the output is fed back to the same input through R2. The non-inverting input is connected to ground. With negative feedback, an op-amp will attempt to make the voltage difference between the two inputs zero. As we've tied the non-inverting input to ground, that's 0V. R2 has Vout across it and R1 has Vin across it. Now, since the input isn't drawing any current, the current is only flowing through R1 and R2, so $V_{out}/R2 = -V_{in}/R1$. Given that, the gain (V_{out}/V_{in}) will be $-R2/R1$.

Figure 5 (overleaf) shows a simple non-inverting amplifier configuration. Notice that here, the input signal goes into the + (non-inverting) input and the feedback uses the - input. By rule one above, $V_a = V_{in}$, but V_a is at the midpoint of a →

YOU'LL NEED

- ◆ Solderless breadboard
- ◆ Adjustable 10 kΩ resistor/potentiometer
- ◆ Several 10 kΩ and 100 kΩ resistors
- ◆ Capacitors in the 0.1 μF–100 μF range
- ◆ CdS photocell
- ◆ LED
- ◆ LM324N quad op-amp

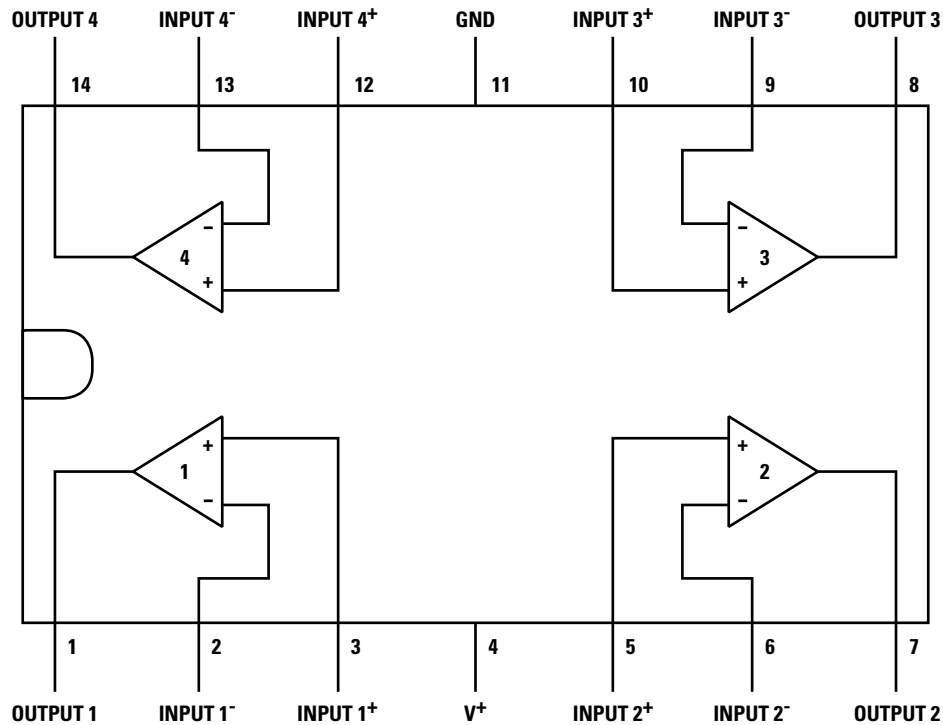
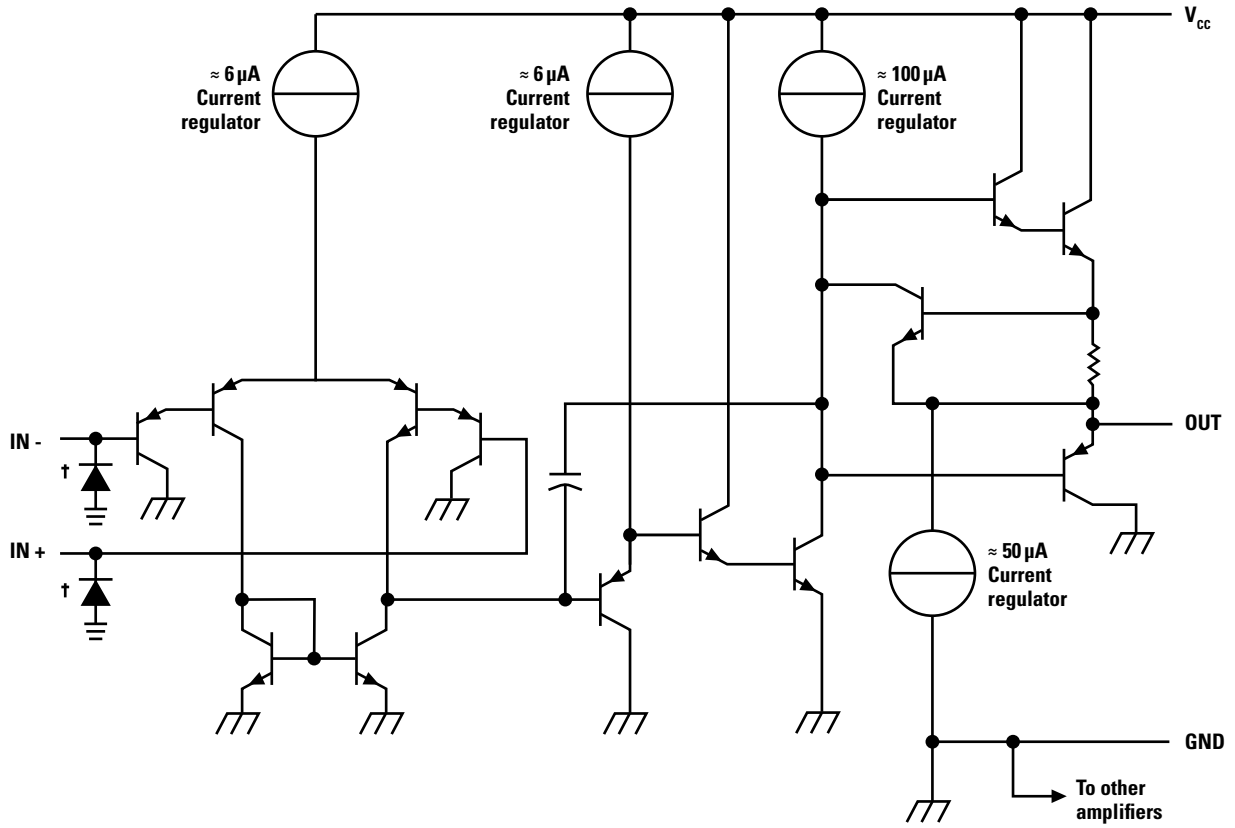


Figure 1 ♦
Functional diagram
of one of the
LM324N's op-amps

Figure 2 ♦
Pinout of the LM324N

Figure 3 ♦
The standard
schematic symbol for
an op-amp

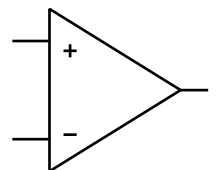
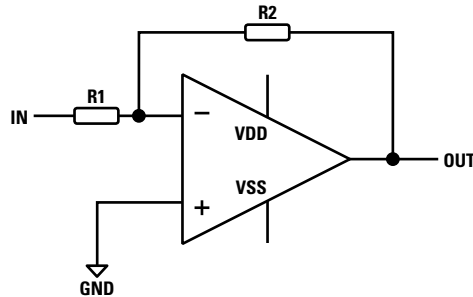


Figure 4 ♦
An inverting amplifier circuit

Figure 5 ♦
A non-inverting amplifier circuit



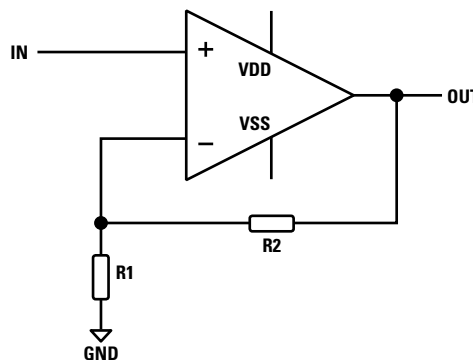
voltage divider using R1 and R2 between Vout and ground. This means that $V_a = V_{out} R1 / (R1 + R2)$. Since V_a and V_{in} are by definition equal, the gain V_{out}/V_{in} becomes:

$$\begin{aligned} \text{gain} &= V_{out} / (V_{out} R1 / (R1 + R2)) \\ \text{gain} &= V_{out} (R1 + R2) / (V_{out} R1) \\ \text{gain} &= (V_{out} R1) / (V_{out} R1) + (V_{out} R2) / (V_{out} R1) \\ \text{gain} &= 1 + (R2/R1) \end{aligned}$$

As expected, the gain is a positive number since the amplifier is non-inverting. Note that the gain will always be ≥ 1 .

QUICK TIP

The op-amp's high input impedance and low output impedance make it an exceptionally good buffer.



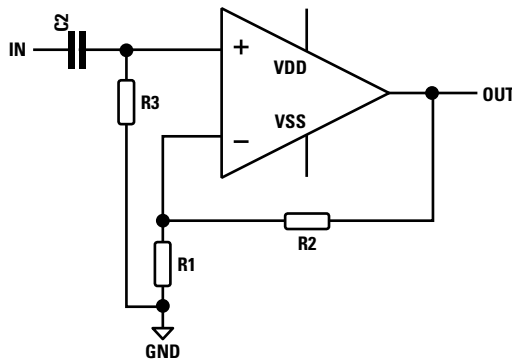
AC AMPLIFIERS

So far, these have been DC amplifiers. By AC-coupling the input using a capacitor, we can make this into an AC amplifier. See **Figure 6** for the circuit. This will need to use a dual supply op-amp as the DC bias in the input is removed by the capacitive coupling. Since the op-amp's input is of such high impedance that it doesn't allow the capacitor to discharge on its own, R3 provides a path to ground for that.

NON-AMPLIFYING AMPLIFIERS

A follower (aka buffer) is an amplifier that has a gain of 1. So it's a stretch to call it an amplifier. What's the point then? Well, it takes advantage of the op-amp's high input impedance and low output impedance to isolate (buffer) a signal so that it is unaffected by the circuit it is driving. **Figure 7** shows the circuit. Comparing this to the non-inverting amplifier in **Figure 5**, you can see that R1 has infinite resistance, and R2 has 0. The resistor-related term in the gain equation is removed by the 0 numerator, and we are left with a gain of 1. So it doesn't change the signal, it isolates whatever is producing the input from whatever is connected to the output.

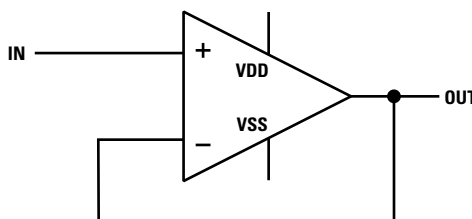
Figure 6 ♦
An AC signal amplifier circuit



Since it's a capacitor,
the current is the
interesting quantity here



Figure 7 ♦
A buffer circuit



INTEGRATOR

An integrator does, well... integration. Specifically integration of the input signal. **Figure 8** shows the circuit. This one is a bit different in that the feedback is through a capacitor rather than a resistor. Since it's a capacitor, the current is the interesting quantity here. Remember that capacitors deal in the storage and movement of electrons. Also, recall from issue 9 that current $I = V/R$. Since the op-amp input draws no current, all the current to charge/discharge the capacitor is flowing through R, and will be V_{in}/R , and we get $V_{in}/R = -C(dV_{out}/dt)$. Moving things around and integrating to get rid of the derivative, we get an equation for Vout at time t:

$$V_{out}(t) = -\frac{1}{RC} \int V_{in}(t) dt + \text{const}$$

What this means is that the V_{out} will change at a rate proportional to the V_{in} , in the opposite direction (because the input is connected to the op-amp's inverting input). So if V_{in} is positive, V_{out} will decrease. Conversely, if V_{in} is negative, V_{out} will increase. How fast V_{out} changes is determined by how negative or positive V_{in} is. Finally, if V_{in} is 0V, V_{out} doesn't change.

This all assumed a perfect op-amp, which is unlikely. However, by carefully selecting the op-amp to fit the requirements, you can get pretty close.

DIFFERENTIAL AMPLIFIER

Figure 9 shows a classic differential amplifier. The gain is set by R_2/R_1 – it is critical to correct the operation so that the two R_1 s and the two R_2 s are as identical as possible in order to remove any signal that is common to the two inputs. After all, the whole purpose of this circuit is to isolate and amplify the difference in the two input signals. Resistors with a tolerance (i.e. how much they vary from the stated resistance) of 0.01% are desirable. It's worth noting that the gain (i.e. R_2/R_1) is applied to the difference in the inputs, so $V_{out} = (V_2 - V_1)(R_2/R_1)$

MIXER

Since we use a series resistor on the input to an inverting amplifier configuration, why not do that with multiple inputs? This would give us what is called a summing amplifier. See **Figure 10** for an example circuit.

Since the non-inverting input is tied to ground, the inverting input is pushed to stay at 0V. We know that the op-amp's input doesn't impact current flow, so all the current due to the input signals has to flow through the feedback resistor, R_f . Since $I = V/R$, $V_{out}/R_f = V_1/R_1 + V_2/R_2 + V_3/R_3$. If we take the simplest case where all resistors have the same value (R), we get: $V_{out} = R(V_1/R + V_2/R + V_3/R) = V_1 + V_2 + V_3$. If we change the values of R_1 - R_3 (the circuit can be extended to more inputs) we get a weighted sum: the portion of each input is determined by the corresponding resistor. If the resistors are adjustable, we have a mixer.

COMPARATORS

The op-amp circuits we've looked at so far have been linear: the output varies in proportion to the input. In contrast, a comparator is a non-linear circuit.

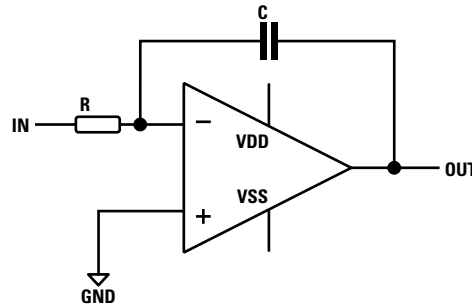


Figure 8 A signal integrator circuit

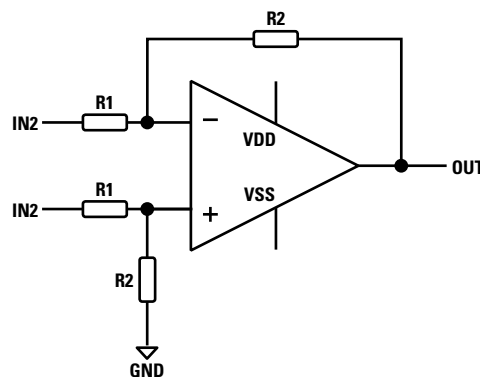
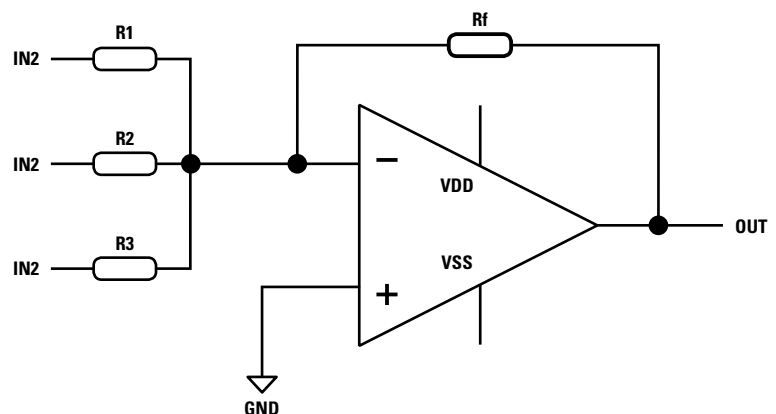


Figure 9 A differential amplifier circuit

Recall that the basic operation of an op-amp is that the output slams hard in the direction that, through feedback, will force the inputs to be equal. Without that feedback, the output snaps all the way in one direction or the other and stays there until the relative value of the inputs changes.

Figure 11 (overleaf) shows the basic comparator circuit. Since the input is connected to the op-amp's inverting input, whenever the input is greater than the voltage set by the R_1 - R_2 voltage divider, i.e. $V(R_2/(R_1+R_2))$ which we'll call the threshold, the output will be at ground. When the input is less than the threshold, the output will be at V . →

Figure 10 A signal mixer circuit



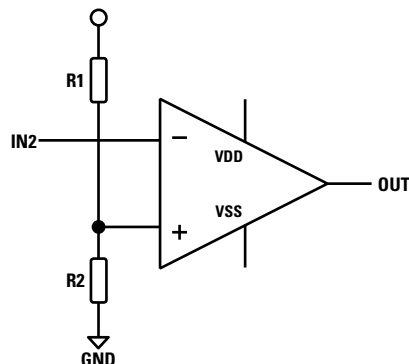


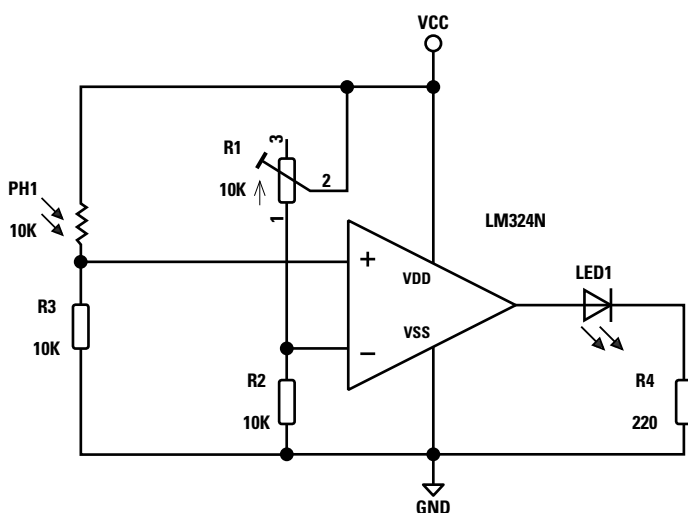
Figure 11 ♦
A basic
comparator circuit

QUICK TIP

At its heart, an op-amp is a differential amplifier.

Grab your breadboard and some parts. **Figure 12** shows a simple comparator circuit that controls an LED based on the amount of light shining on the photocell. Bright light shining on it will lower its resistance, causing the voltage at the inverting input to be higher. When it gets higher than the non-inverting input, the output switches to 0V and the LED turns off. If it's dark, the resistance of PH1 is higher, so the voltage at the inverting input will be lower. When it's below the voltage at the non-inverting input, the output switches to 5V and the LED lights up. R1 is an adjustable resistor (a potentiometer) and is used to set the voltage at the non-inverting input, and thus the point at which the output and the LED changes state. This translates directly to the amount of light shining on PH1 at that point of change. This is the basics of a night light.

Figure 12 ♦
A night light circuit



A problem with this simple comparator design shows itself when the input signal is jittering back and forth across the threshold: the output will switch back and forth between its extremes to match. For the photo comparator, this isn't an issue, but if we want to avoid that jitteriness, we need to add something.

To clean up the jittering, we'll call on the op-amp's best friend: feedback. **Figure 13** shows how to add a feedback resistor to turn the op-amp comparator into what is called a Schmitt trigger.

In this case, we use positive feedback (feedback to the non-inverting input) to affect the threshold depending on the output voltage. The output in the circuit will be at 0V or 5V, so effectively R3 (the feedback resistor) will be in parallel with either R2 or R1. This will change the voltage divider, and hence the threshold. When the output is 0V, the threshold will be slightly lower than what it would otherwise be, and when the output is 5V, the threshold will be slightly higher. The result is that the output state doesn't change when the input would cross the threshold (i.e. what it would be without the feedback resistor in place); to switch the output to 5V, the input has to drop below the new slightly lower threshold, and to switch the output to 0V, the input has to go above the slightly higher threshold. This results in a dead zone between the two new thresholds: if the input is in that zone, the output won't be affected. This is commonly referred to as hysteresis.

This is a lot like debouncing a switch, in that you want to ignore the noise from bouncing mechanical contacts that cause a series of quick presses and releases before settling into one or the other states. The big difference is that with switch debouncing, the hysteresis is in terms of time rather than voltage.

OSCILLATORS

Figure 14 shows a square wave oscillator built around a single op-amp from the LM324N chip. This is configured as a comparator with the non-inverting input setting the threshold. Like the Schmitt comparator, this uses feedback to (assuming we use a 5V supply) set the threshold to 1.68V when the output is at 0V, and 2.77V when the output is at 5V. Here, R5 and R6 form the voltage divider providing 2.5V, and R7 provides the feedback to create the hysteresis. The difference with this circuit is how the inverting input is used. R8 connects it to the output and C1 can use all the current flowing through that to charge and discharge (since the input draws no current). Grab your breadboard and wire this up. Play

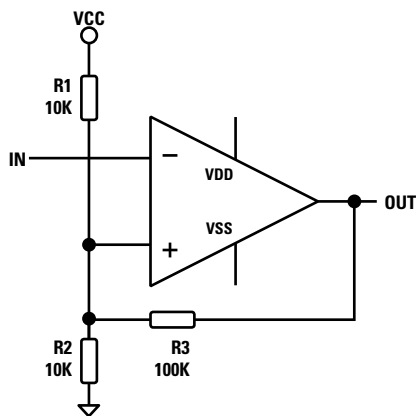


Figure 13 ♦
A Schmitt trigger comparator circuit

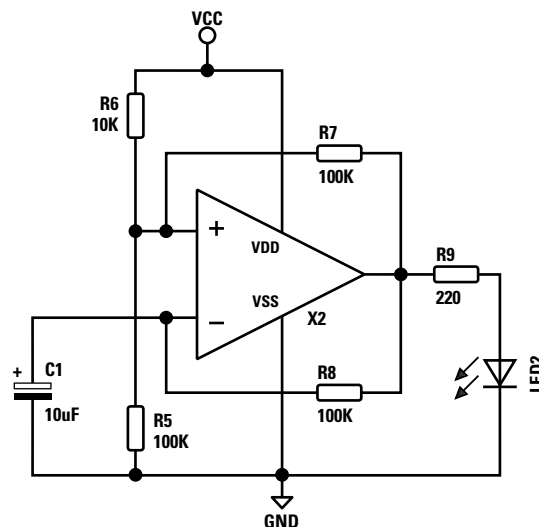


Figure 14 ♦
A square wave oscillator circuit

around with different capacitor values to generate different frequencies.

Assume C1 is empty as a starting point (as it would be when the circuit is powered on). That means that the inverting input is at 0V. Ignoring the feedback, the voltage divider holds the non-inverting input at 2.5V. With the non-inverting input at a higher voltage, the output will be 5V (which means that the non-inverting input is actually at 2.77V). This causes current to flow through R8, charging C1. When the voltage on C1 exceeds 2.77V, the inverting input will be at the higher voltage and the output will switch to 0V. That does two things: changes the voltage at the non-inverting input to 1.68V, and lets C1 discharge through R8.

As that happens, the voltage at the inverting input drops until it's below 1.68V, which is the voltage at the non-inverting input. That causes the output to switch to 5V, and the cycle repeats. So the voltage at the non-inverting input switches between 1.68V and 2.77V as the output switches between 0V and 5V. At the same time, the output causes C1 to charge and discharge. The frequency of the oscillation is dependent on the RC time constant of R8 and C1.

Figure 15 shows the voltages at the inputs and output over time.

Op-amps are an extremely popular analogue circuit building block, and one of the most widely used types of component. At its heart, it is a very high gain differential amplifier: the output is an amplified version of the difference in voltage between its two inputs. You would be hard-pressed to find any significantly complex analogue device that doesn't

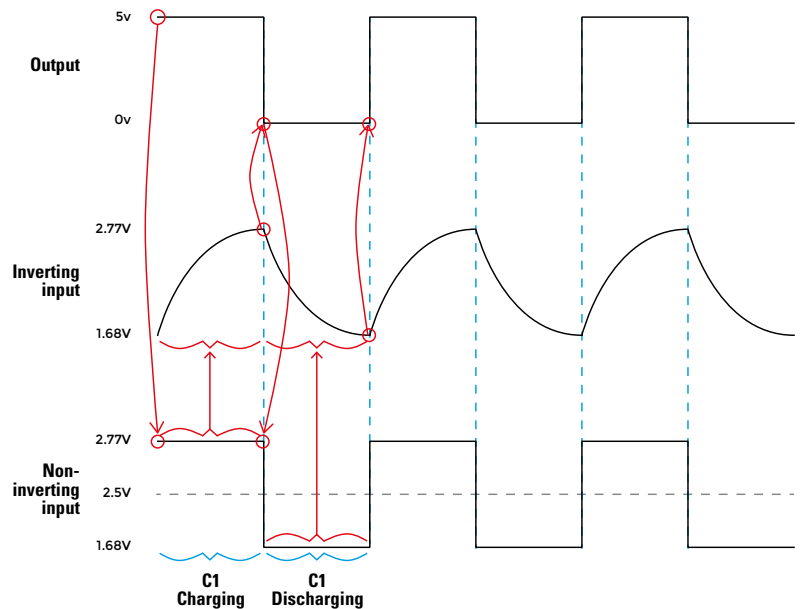


Figure 15 ♦
Input and output signals in the oscillator circuit

contain a few op-amps. Hopefully, this has piqued your interest. If so, there are many places online and in print for learning more.

Get in touch and show us your op amp (or other electronic) creations on twitter at [@HackSpacemag](https://twitter.com/HackSpacemag) or email hackspace@raspberrypi.org.

QUICK TIP

If one input is fixed, the differential amplifier becomes a comparator.

Make a PCB in KiCad: Part 2

Let's learn how to lay out a simple PCB in KiCad, ready to be professionally fabricated



Jo Hinchliffe

@concreted0g

Jo Hinchliffe is a contributor to the Libre Space Foundation, 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, milling machines, and CNC kit.

QUICK TIP

Refer back to Part 1 of this tutorial to remind yourself of the keyboard shortcuts, if needed.

YOU'LL NEED

A computer with KiCad installed

Last month we created a schematic and associated files that we needed to begin this project. This month we will finish off laying out a small PCB, ready for professional fabrication.

Welcome back to this two-part tutorial. If you haven't been working on anything else in KiCad, you should be able to open the main project page, and the project we started in the last issue will be automatically loaded. In this final part, we are going to be working with the Pcbnew application. There are two ways to open this: we can either open Eeschema first, and then click the 'Run Pcbnew to layout printed circuit board icon' (second from right on top toolbar); or, on reopening KiCad, you can open the Pcbnew program directly from the project page.

You will be met with a blank page in Pcbnew and, before we import the footprints, let's change a few settings. Firstly, we can set the grid to any resolution we like. You can change it at any time, but it's useful to have it set to 0.254 mm, as often components have pins spaced to this pitch. Also, by default, the track width is set to 0.5 mm, which is fine for this project, but just so we know how to, let's click setup-design rules, and then select the global design rules tab, then add some extra track widths. Add 0.8 mm and 1 mm, then click OK. Now, in the drop-down track widths dialog, these options should be available in the main Pcbnew page.

You will notice an icon on the top toolbar that looks the same as the 'create netlist' icon we used in Eeschema in the first part. However, in Pcbnew

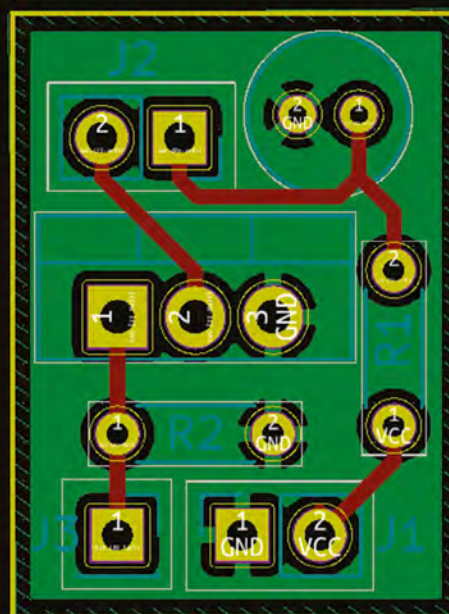


Figure 1 What we are aiming for! The completed PCB, laid out in Pcbnew

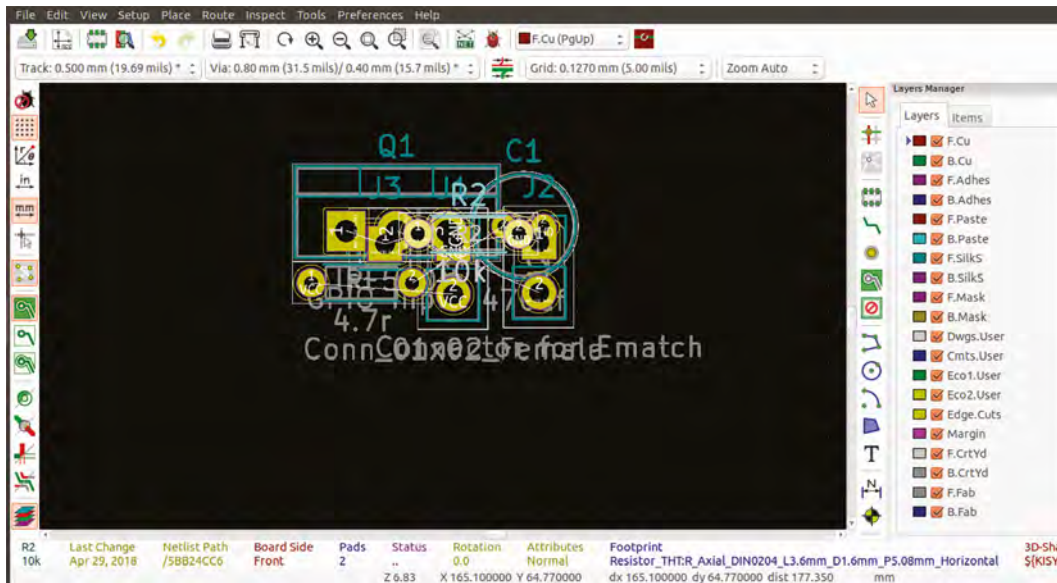


Figure 2 Importing the netlist usually results in a big messy pile of stuff getting added. Don't worry – this is correct!

Press M to move each component to spread them out, and see everything a little more clearly

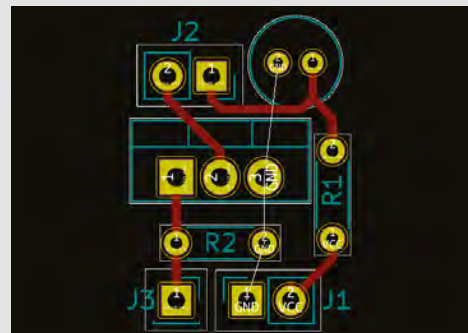
this icon means 'read the netlist'. Click this and it should automatically have the location of the netlist we made in the previous issue. If not, navigate to it using the browser. Then click 'read current netlist' and you should get some messages in the output messages box. Then click Close. You should now see a messy pile of component footprints all overlapping, with a collection of white lines or 'rat's nest' showing the connections as seen in **Figure 2**.

Hovering the cursor over each component and aiming for the centre of the component, in turn, press **M** to move each component out to spread them out and see everything a little more clearly. Then comes the game of attempting to arrange all the components in a sensible order, to then simplify where we lay the traces. Once spread out, you will note that there is a lot of text surrounding the components, and this can help you identify which component is which, but also can make the display very cluttered. To remove this text, untick the 'F.Fab' layer in the right-hand window. Optionally, you can also turn off the silkscreen layer 'F.Silks' to →

UNTANGLE THE RAT'S NEST

Once you have settled on a layout, you can begin to wire the physical connections between the connected components. When connected with a trace, the white line of the rat's nest should disappear. Don't place any tracks to any of the gnd connected pads, however, as we will flood the PCB with a copper ground plane that will connect them all in one go later on.

So, using the 'Route tracks' tool, which is the fifth down on the right-hand vertical toolbar, let's begin to lay tracks. It's similar to laying a wire in Eeschema, in that a left-click begins the trace. A single left-click can be used to create a point about which to turn the track, and a double-click will end the track. Make sure to place tracks to the centre of pads; if you need to move a component, you will need to redo the track, hover the track in question, and press the **DELETE** key. Again, aim to get the tracks routed in a similar fashion to the image.



QUICK TIP

You'll notice that some pads on component footprints are square instead of round. This is often to indicate orientation of the part for assembly. However, should you wish to change the pad to round, or change the dimensions, you can select the individual pad and press **E** to edit.

QUICK TIP

When selecting items, Pcbnew will sometimes pop up a dialog asking you to confirm the item you are trying to select from a list. It's useful, as often PCB designs are incredibly densely packed with items.

further declutter whilst you arrange the component footprints. Aim to get the footprints arranged similarly to in **Figure 3**.

Turn the 'F.SilkS' layer back on and you will see that the silkscreen references annotating the components are not in great places. Clicking on the references enables them to be moved and rotated like any component, so they can be moved to better positions. They can also be made to disappear by editing: mouse over and press **E**, select 'edit' next to the reference (the text you wish to remove), and then, in the pop-up box, change the setting to 'Invisible'. For example, as there is only one capacitor and one transistor on this board, it is unlikely to get them mixed up, so deleting those references to save a little space may be an option. Silkscreen references are often useful when populating the board with components, but may not need to be seen afterwards, so it makes sense to place the R1 and R2 silkscreen references inside the footprint of where the resistors are to be placed.

BECOMING GROUNDED

We need to connect the 'gnd' ground pads to a ground plane now; before we do so, however, we need to define the edge of the board. This is done by drawing the outline shape of the board we want. We are going to draw the board edges on a layer called 'edge cuts', by selecting it from the layers drop-down menu on the top toolbar. Once selected, click the 'add graphic lines' icon (ninth down on the right-hand vertical toolbar). Proceed to draw a rectangle around

Figure 4 ♦
The filled zone tool dialog box

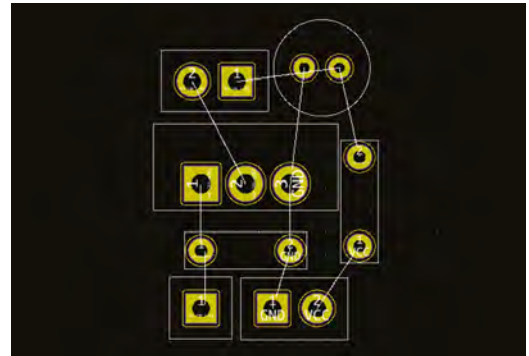
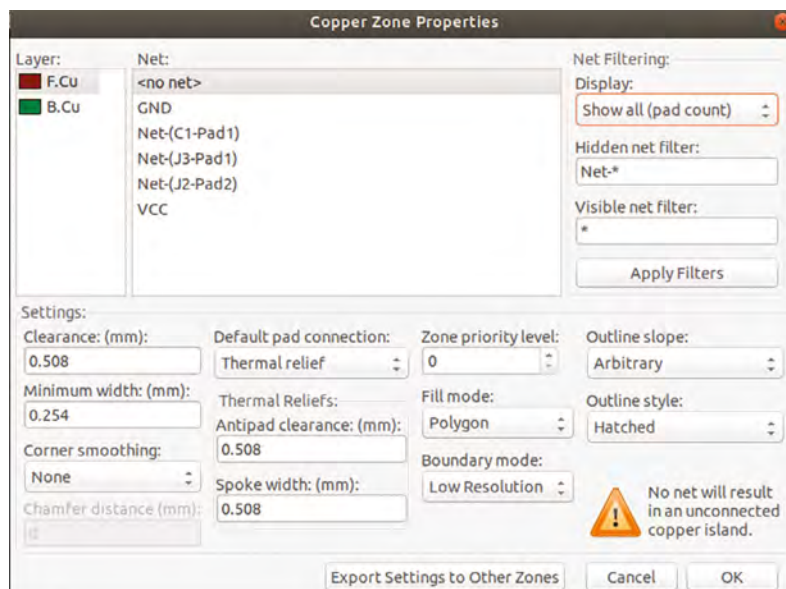


Figure 3 ♦
The components are decluttered and arranged to try to simplify the placement of traces. Turning off some of the text and silkscreen layers may make this process easier

the components you have laid out. In **Figure 1**, this is the yellow outer line we can see. Whilst complex geometries of edge cuts are difficult to make in KiCad, you aren't limited to squares – you can use the circle and arc tools in combination with the 'add graphic line' tool to create many different shapes.

Finally, to finish the connections on this board, we are going to create a copper flood area connected to the ground net. Common practice (although opinions differ) on two-layer boards is to place the ground copper flood on the lower layer (which then connects to the lower side of the through-hole pads). However, many manufacturers place copper floods on both sides of the board (not necessarily connected to ground or other power planes) in an effort to reduce the amount of etchant used, as they have to remove less copper.

To add a ground plane copper flood, first select the 'B.Cu' layer from the drop-down list. Then click the 'add filled zone' tool icon (seventh down on the right-hand vertical toolbar). We are going to use this to draw another rectangle, tracing over

MULTI LAYER

We have currently only worked on the topside layer of the PCB and, due to the way we have laid it out, we don't need to draw any tracks on the underside. However, it is extremely valuable to be able to draw a track and then continue the track on the lower side to avoid crossing another track. To do this, we would need to add a via, which connects the top and bottom layers. Experiment with trying this whilst drawing the track: right-click and select 'place a through via'. Left-click to place the via and then continue to lay a track. You will notice that the track has turned green, signifying that this is being placed on the bottom layer. Of course, these tracks are then allowed to cross red tracks, as they physically aren't in contact. Delete your experiments, though, as they aren't needed!

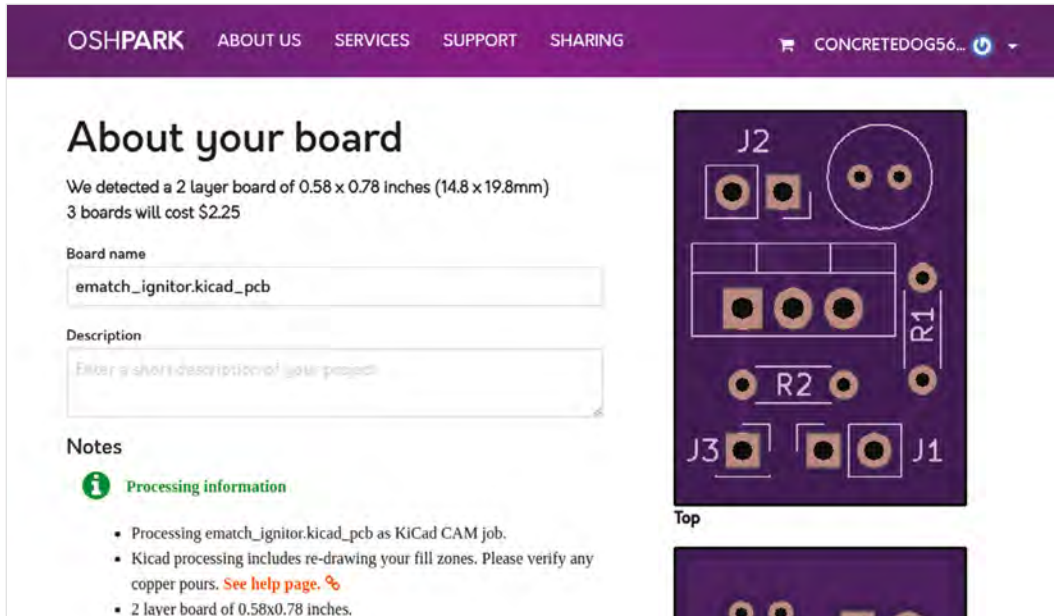


Figure 5 A popular choice for PCB fabrication is OSH Park, which makes it simple to order. You can upload KiCad project files directly

the edge-cut rectangle we created earlier. When we click to start, however, a dialog box appears (Figure 4); this contains various settings around how much clearance the flood will leave, the minimum distances it will tolerate, flood between objects, and more. We are going to leave everything as it is, but we are going to click on 'GND' from the list of possible nets, then click OK to close the box. Next, proceed to click around the rectangle until you get to the third corner, where you should see a green preview box filling the entire PCB. Double-click and it should produce a large green area which has connections to the pads that are connected to ground. You can still click back to the top layer to see your work in all its glory. We should now have a PCB that looks similar to that in Figure 1.

GERBER GATHERING

Now that we have our PCB design finished, we probably want to get it manufactured. There are lots of options and companies, and most will require that you export Gerber files, with one for each layer of the PCB. These layers include things like where the copper is going to be exposed and where it is masked/covered, where the silkscreen items are, etc. They also require a drill file which tells them where the drilled holes are, what diameter they are, and so on. KiCad is capable of creating all industry-standard types of Gerbers, and often PCB fabrication companies will have details online as to what type of Gerber file they will need.

To finish the connections on this board, we are going to create a copper flood area connected to the ground net

For a simpler option, there is a great PCB fabrication service, OSH Park, that is well-known in hardware hacker circles for great-quality PCBs and has a very simple way of ordering. The OSH Park site allows you to upload a KiCad PCB file (also compatible with a lot of other PCB software), which is what is created by the Pcbnew program we have used, and can be found in the project folder. Uploading the file to the website also gives you a really handy preview image of both sides of your board in the browser, and allows you to check over and make sure that your board is correct. As you can see in Figure 5, to get three of these boards made and delivered from OSH Park, at the time of writing, would cost \$2.25, offering excellent value for money. The only thing is, if you don't like purple, you can't choose another colour!

There are many aspects of KiCad we haven't explored (it's worth looking at editing/creating your own footprints, and also exploring how to use the design rules check tools), but hopefully, these articles have given you the confidence to explore KiCad.

Now it's over to you. Set forth and create your own circuits. Let us know how you get on and send a tweet to @Hackspacemag

QUICK TIP

In Pcbnew, it's often useful to work out dimensions to be able to place things accurately. A useful feature is that pressing the **SPACE** bar creates a datum/zero point, and then the coordinates from this zero point are displayed on the lower toolbar.

First steps in CircuitPython

Ditch the compiler and get coding the interactive way



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.

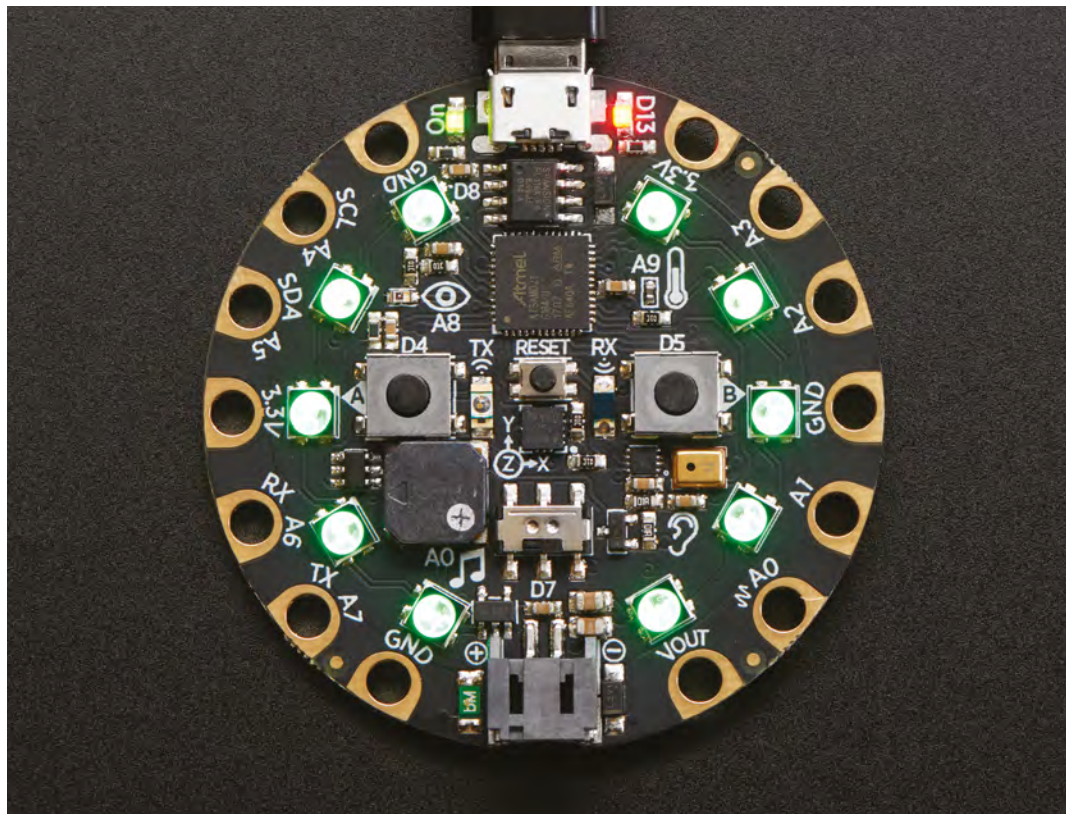
When you have a hammer, the saying goes, every problem looks like a nail. The same is true for our programming languages. Particular languages make us think about problems

in particular ways and, therefore, learning new languages help us think in new ways. These new ways of thinking can help us be better programmers, regardless of what language we're using. In this article, we're going to take a look at one of the hottest languages in the microcontroller world: CircuitPython.

For a long time, compiled languages, like C and C++, have been the languages of choice for microcontrollers, while interpreted languages such

as Python, JavaScript, and PHP have become more and more popular on the desktop and server. The difference between the two is that compiled languages are written in an English-like language, then go through a process called compilation, where they're converted wholesale into instructions that the CPU understands. Interpreted languages, on the other hand, are read by the computer line-by-line and executed directly from this human-readable language.

Each language is different, and there are a lot of arguments for each style that we won't go into here, but if we had to generalise, compiled languages run more quickly, while interpreted languages are easier to develop. This explains why microcontrollers favour compiled languages – with few computing



Right

Subscribe to HackSpace magazine for twelve months and you get a free Circuit Playground Express – a great board for getting started with CircuitPython


```

1 import board
2 import random
3 import neopixel
4 import time
5
6 pixels = neopixel.NeoPixel(board.NEOPIXEL,1)
7
8 colours = {'red':(10,0,0), 'orange':(10,3, 0), 'yellow':(10,7,0),
9           'green':(0,10,0), 'blue':(0,5,10), 'indigo':(0,0,10), 'violet':(2,0,10)}
10
11 rainbow = ['red', 'orange', 'yellow', 'green', 'blue', 'indigo', 'violet' ]
12 while True:
13     for colour in rainbow:
14         print(colour)
15         pixels[0] = colours[colour]
16         time.sleep(1)

```

Adafruit CircuitPython REPL

```

yellow
green
blue
indigo
violet

```

resources, we need compiled languages to take best advantage of that. However, many modern microcontrollers aren't all that constrained. Take the ATSAM51 microcontroller in the PyPortal that we review this month. It runs at 120MHz and has 256kB RAM. To young whipper-snappers, this may sound pretty limited, but some of us remember playing 3D games on systems with less power. In other words, with systems this powerful, we don't have to focus only on programming in the most efficient way – we can allow ourselves a few comforts, even if these come at the expense of performance. Let's take a bit more of a look at what this means in practice.

When programming in Arduino, you need the Arduino IDE on your desktop. This is a front-end to a whole toolchain that compiles, links, and uploads your code to a board. On the board itself, there is just a very minimal bootloader that enables the board to easily accept the new program. With CircuitPython, the situation is almost reversed. There's code running on the board that allows your operating system to recognise the board as a USB mass storage device. It then monitors the contents of this storage and, when new code is uploaded (that is, the actual text of the programming language, not compiled 'machine code'), it executes this code. This is done by software running on the board, and all that's needed on the main machine is a lightweight text editor.

In principle, any text editor should work, but in practice, many don't. As the code is triggered by →

VARIABLES

Technically speaking, Python doesn't have variables. When we create something like:

```
variable = 1
```

...we're creating a name and pointing it to the value 1. The important difference between this and C is that it does memory assignment in a very different way. In C, when you run the following:

```
int variable = 1
```

...you're reserving a block of memory the right size to store an integer and that memory will be held until you kill that variable. Every variable has a type and that type cannot change.

In Python, when we create a name, we don't reserve any memory. We use some memory (we have to do so to store the contents), but a name doesn't have a type attached to it, so the following is all perfectly valid:

```

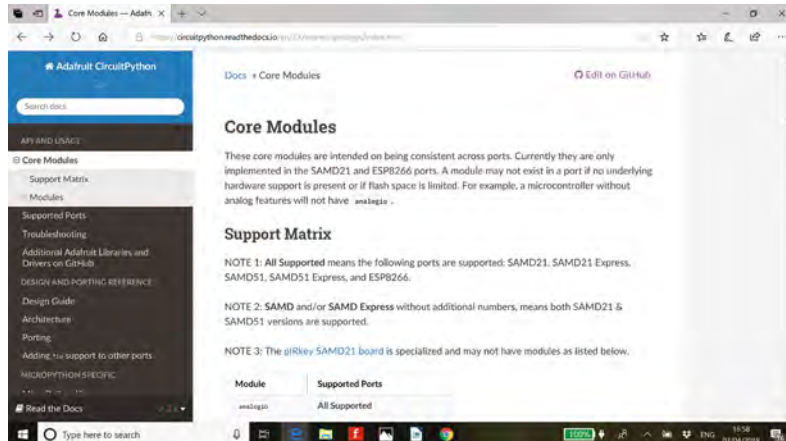
variable = 1
variable = variable +1
variable = "hello"

```

This weak typing can seem a complete anathema to people coming from languages like C that have a strong type system. At its best, it means that we don't have to worry about what a bit of data is; we just have to worry about what it can do.

Above

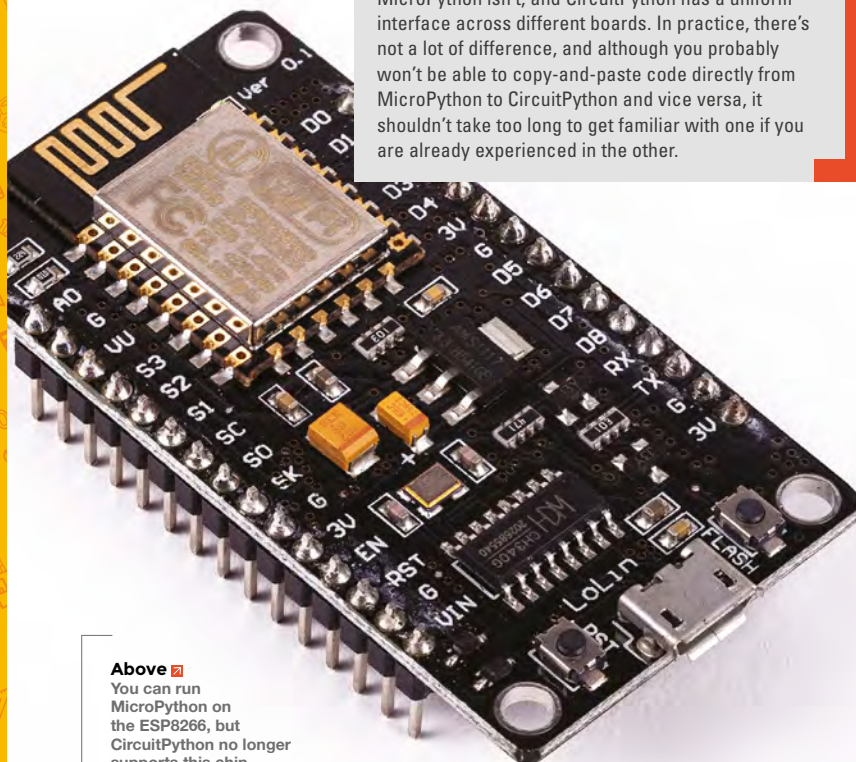
We recommend starting with the Mu text editor – you can always try other ones once you know that your setup is working properly



Above You can get a detailed look at the CircuitPython API at hsmag.cc/ppuJfT

MICRO AND CIRCUIT

There are two types of Python available for microcontroller boards: MicroPython and CircuitPython. In general, a particular platform only supports one of the two Python forms. The most obvious difference between the two is that in MicroPython, the main file is called **main.py**, while in CircuitPython, it's **code.py**. This superficial difference hides philosophical differences – CircuitPython is a strict subset of CPython (i.e. regular Python) while MicroPython isn't, and CircuitPython has a uniform interface across different boards. In practice, there's not a lot of difference, and although you probably won't be able to copy-and-paste code directly from MicroPython to CircuitPython and vice versa, it shouldn't take too long to get familiar with one if you are already experienced in the other.



Above You can run MicroPython on the ESP8266, but CircuitPython no longer supports this chip

Before getting into the code, you'll need to plug in your CircuitPython board and select CircuitPython

a save, any editor that does something clever with autosave will cause problems. We recommend using Mu, a text editor designed specifically for Python.

Before getting into the code, you'll need to plug in your CircuitPython board and select CircuitPython from the button in the top-left corner of the text editor.

WHITESPACE

Let's take a look at a really simple bit of code that will show off a few of the basic language features. This should work on any CircuitPython board with a built-in NeoPixel, but we tested it on a Circuit Playground Express. It will cause one of the NeoPixels to flicker on and off.

```
import board
import random
import neopixel
import time

pixels = neopixel.NeoPixel(board.NEOPIXEL, 1)

pixels[0] = (10, 10, 10)

while True:
    flicker = random.randint(0, 10)
    print(flicker)
    if flicker == 1:
        pixels[0] = (10, 10, 10)
    else:
        pixels[0] = (0, 0, 0)
    time.sleep(0.5)
```

One of the most basic aspects of programming is how code is grouped. In the above, there are **if** statements and loops that run blocks of the language in particular ways. In Arduino, for example, these blocks of code are enclosed in curly braces: {}. In Python, we use the indent level to show this. As far as Python is concerned, this indent can be any number of spaces or tabs as long as it is consistent, and increases or decreases as necessary. However, in order to avoid chaos, the Python community has agreed that the standard indent is four spaces. Stick with this and you should be able to copy-paste

IMPORTING MODULES

There's relatively little built in to the Python language – almost all of the useful features are imported from modules, and it's the sheer range of modules that makes Python so useful.

All CircuitPython hardware should have the **board** module, which includes details of what hardware is attached where. In this case, we've used this to locate the on-board NeoPixel, but we could also use it to locate any of the pins (with, for example, **board.A0**).

You can find details of many of the available modules, including hardware drivers, at: hsmag.cc/OzQCC0.

code from other sources and it'll all just work. In an editor designed for Python (such as Mu), the **TAB** key should insert four spaces, but double-check this before relying on it, as ending up with code that's half tabs and half spaces is difficult to work with.

As you can see opposite, code blocks are introduced with a colon.

BEING OBJECTIVE

Python is multi-paradigm: you can use objects, but you don't have to. In general, CircuitPython scripts are procedural, in that they start at the top and execute code line-by-line until they get to the bottom, where they end. In the example opposite, we use the **NeoPixel** object that we get from the imported **neopixel** module. When we create the **pixels** variable, we have to instantiate this with the details it needs to create the object – in this case, the pin that the NeoPixels are connected to and the number of pixels. Notice that we don't have to declare the variable in any way; we just start using it and Python will create it for us (as we do with the **flicker** variable as well).

Unlike Arduino, there aren't specific setup and loop functions. In many cases (like opposite), you'll have a **while True** loop that runs forever, but this isn't necessary.

We can send data to the serial console using the **print()** function. Anything printed (as we have done with the **flicker** variable) will appear in a new line on the serial monitor.

Python doesn't have arrays in the same way that a language like Arduino C++ does; instead, it has two different, but similar, array-like data structures: lists and tuples. Actually, we're not using a true list in this example, but a **NeoPixel** object that behaves in the same way.

Tuples are enclosed by brackets (...) and also contain an ordered group of data, but unlike lists,



they're not changeable. Once you've created a tuple, you can't reassign its elements. You can overwrite the whole thing (as we do when we set **pixel[0]** to either (10,10,10) or (0,0,0), but we couldn't change a single value with something like **pixel[0][1] = 10**, which we would have been able to do if they were lists rather than tuples.

FOR ITEM IN LIST

Let's take another look at a list. Replace everything under the **pixel=** line in the above with:

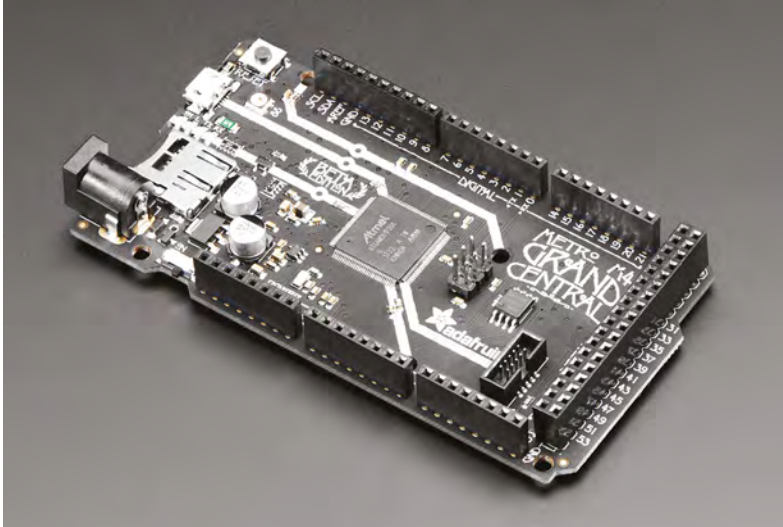
```
rainbow = [(10,0,0), (10,3, 0), (10,7,0),
(0,10,0), (0,5,10), (0,0,10), (2,0,10)]
while True:
    for colour in rainbow:
        pixels[0] = colour
        time.sleep(1)
```

Above ♦ Looking for CircuitPython with a screen? Then turn to page 126 where we review the Adafruit PyPortal

SETTING UP A BOARD

CircuitPython development is progressing rapidly, so it's a good idea to make sure you're running the latest version of the firmware. This is especially important on CircuitPython, as this includes the interpreter. This varies a bit between devices, but often involves double-tapping the reset button to create a different USB drive on your computer, and copying a UTF file (which you should find on your hardware manufacturer's website) onto this drive.

As well as the main firmware, you'll need libraries. These appear in the **lib** folder on the USB device. You can get the latest version of these from hsmag.cc/dayPqg. Download the zip file, and unpack it into the **lib** folder. You can just include all of them, but if you're short on space, you may prefer to include only the ones that you actually need.



Above ♦
Want CircuitPython with a bucket-load of GPIOs and a fast chip? Take a look at the Adafruit Grand Central

INTERACTIVE SESSION

Because CircuitPython is evaluated line-by-line (rather than being compiled), we can program interactively. To do this, open the serial monitor in Mu. If there's code already running (i.e. if your code has a loop that isn't finished), you'll need to hit **CTRL+C** to end this. You should see the line:

```
Press any key to enter the REPL. Use  
CTRL-D to reload.
```

REPL stands for Read, Evaluate, Print, Loop – this is the interactive session that we want to enter, so press any key and you should see a prompt with `>>>`. You can type code here and it'll be executed when you press **ENTER**. So, for example:

```
>>> print("Hello World")  
Hello World
```

You can import modules and run any bits of code you like. If you enter code that should start an indented code block, the prompt will change to `...` and it won't be executed until everything's been entered and you've put in an empty line with no indent.

You can execute your main **code.py** file from the interactive session with:

```
import code
```

Obviously, if you've got an infinite loop, you'll never get control back again unless you kill it with **CTRL+C**. However, if you remove the infinite loop, this is a good way to test out your code, as you can interrogate hardware and try different things out without having to re-run your code from the start each time you want to try something new.

Here we have a list of pixels called **rainbow**. You can address this like a two-dimensional list. For example, **print(rainbow[0][0])** would return 10 (even though we use different braces to define lists and tuples, we use square braces when retrieving data from either).

In Python, **for** loops are usually done in this format: **for <item> in <list>**, rather than in the numeric sequence common in C and other languages. If you want to do a numeric sequence, you can use the **range()** built-in method to return a list of numbers, such as:

```
for number in range(0,10):
```

There's an optional third argument that is the step; for example, the following will iterate over the even numbers:

```
for number in range(0,10,2):
```

//

In Python, **for** loops
are usually done
in this format:
for <item> in <list>

//

Let's now take a look at one final example which does the same thing as previously, but introduces a new data structure, the dictionary:

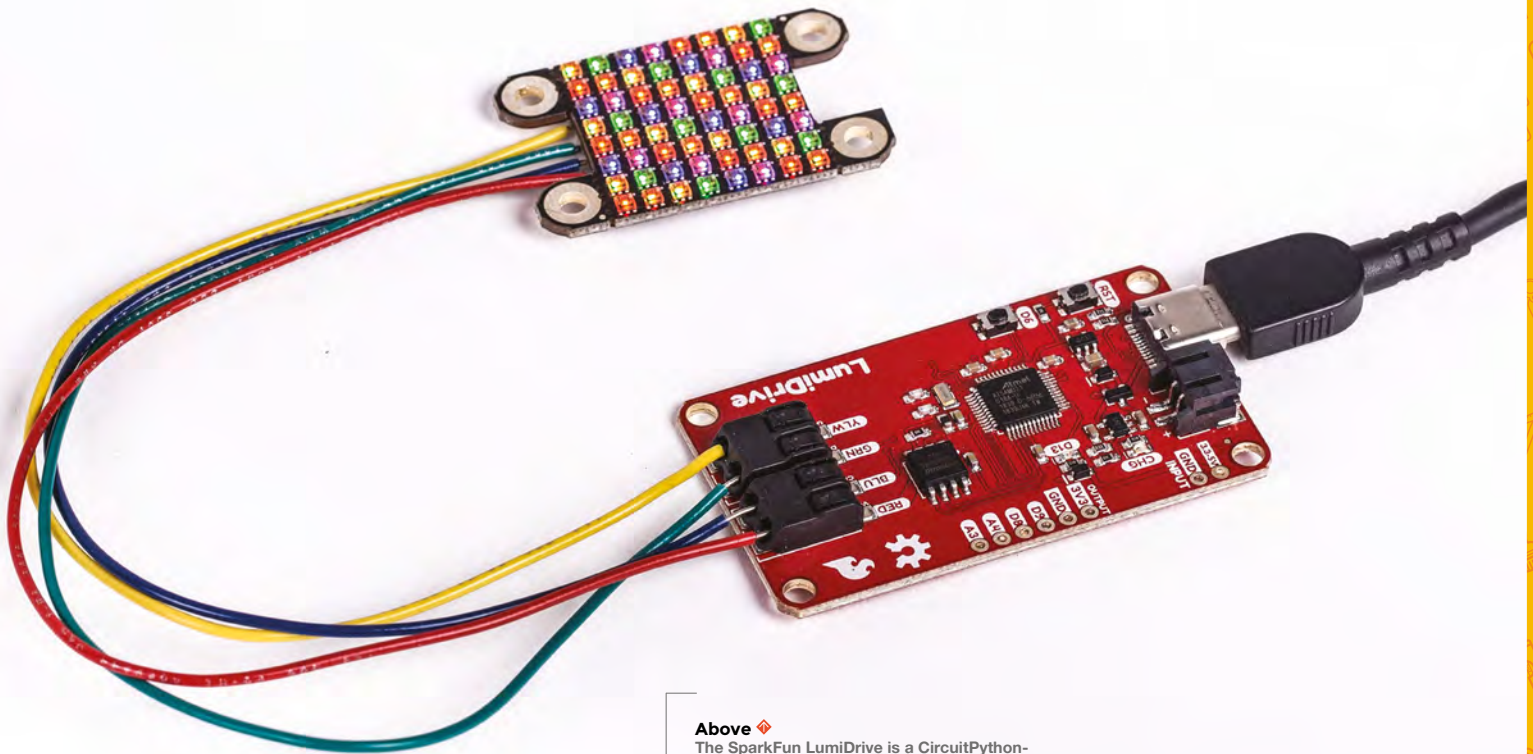
```
colours = {'red':(10,0,0), 'orange':(10,3, 0),  
'yellow':(10,7,0), 'green':(0,10,0),  
'blue':(0,5,10), 'indigo':(0,0,10),  
'violet':(2,0,10)}
```

```
rainbow = ['red', 'orange', 'yellow', 'green',  
'blue', 'indigo', 'violet' ]
```

```
while True:
```

```
    for colour in rainbow:  
        pixels[0] = colours[colour]  
        time.sleep(1)
```

Here, we create a dictionary called **colours** that links each colour name with RGB values for that colour. As you can see, the whole structure is enclosed in curly braces, and items are defined using a colon. Dictionaries don't have a particular order when used in a **for** loop (well, they do as of version 3.7, but it's best not to rely on this now unless you know for sure which version of Python you'll be using it in), so as well as declaring the dictionary of colours, we've



Above ♦
The SparkFun LumiDrive is a CircuitPython-powered board for driving APA102 (DotStar) LEDs

created a list of the order we want them in. While this is a bit over the top in this example, it would be useful if we were using the colours in different ways – we could define the palette in one place and call the colour by name later.

Dictionaries are addressed by strings. In this case, we're looking up the strings in a list, but we could equally use strings directly. The code `colours['red']` would return (10,0,0).

There's much more to Python and CircuitPython than we've got space for here, but hopefully, this should have given you an overview of the basics and given you a place to start. There are some fantastic examples around – particularly on the Adafruit website – that you can explore to learn more about how it works.

There's no point in kidding ourselves: even as performance of CircuitPython gets better, it's never going to reach the performance of a compiled language. If you need this performance, then CircuitPython isn't for you – and that's OK. No language is perfect for everyone; it's about finding the languages that are right for you and your projects. For us, CircuitPython helps us get started quickly and use the same programming language across many different devices and projects. For that, we think it's a great language to learn for both new programmers and old. □

WHY?

Given the performance limitations of CircuitPython and the sheer amount of code already available for Arduino, why should we care about CircuitPython?

There are a few advantages. CircuitPython is higher-level than Arduino. You don't need to worry as much about types of variables and the minutiae of how things are done. This can mean that it's quicker to develop software using Python – of course, this depends on your experience. If you've spent years working with Arduino, it'll probably take a while to build up to the speed you already have, but if you're just getting started, Python can be much quicker.

Python (and CircuitPython is just a subset of Python) is also more cross-platform than Arduino. With the Blinka library, CircuitPython can run on some general-purpose computers, as well as microcontrollers

There's a joke that says that Python is the second-best language for anything. While Python's not perfect for many tasks, it's very good at a wide range. It's widely used in machine learning, data analysis, web development, desktop apps, and now embedded tools. It's used by beginners and professionals alike. The website itjobswatch.co.uk ranks it as the fifth most in-demand programming language in industry.

By learning Python, what you lose in performance, you'll make up for in development speed, compatibility, and opportunities for both geekery and employment.

Make your own soap

How to 3D-print moulds and create personalised soap



Poppy Mosbacher

[@PoppyMosbacher](#)

Poppy started making soap to raise money for a dolphin charity aged twelve. She later started a natural skincare range and taught groups to make their own cosmetics. poppymosbacher.com

Made in small batches and scented with natural essential oils, the ingredients for each bar can cost less than £1, even if you choose to go organic. If you have access to a 3D printer or laser cutter, it's

easy to create your own moulds and put names, logos, or patterns on your design.

Tinkercad provides geometric shapes, customisable gems, and even a butterfly in solid and hole formats which can be paired up to create a mould. We're going to combine its preset boxes to create personalised moulds and a separate alphabet mould to make it easy to add raised text in different colours.

Making soap from scratch involves neutralising an acid with an alkali in a chemical reaction, which can seem scary for beginners. There are safety

precautions to follow and it takes time to 'cure', so many soap makers prefer to start with a natural glycerine soap base. This tutorial uses an easy melt-and-pour recipe, with suggestions for adding fragrances and colours to make unique bars of soap.

FIRST YOU NEED A MOULD

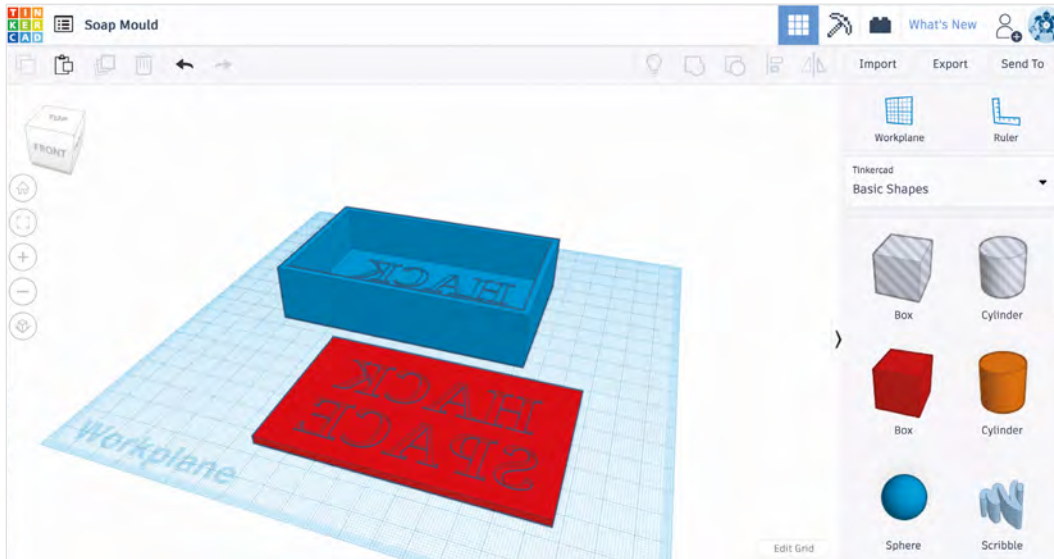
Go to tinkercad.com to get started and open a free account, if you haven't already got one.

Click 'Create New Design' and drag a box from the Basic Shapes toolbar and drop it onto the workplane. You can make soap any size you like, but for this exercise let's make the outside of the mould 105×65×24 mm. Just click on one of the little grey squares at the corners of the box, and alter the dimensions that appear. Add a hollow box anywhere on the workplane and resize it to 100×60×22 mm.

BUYING SOAP SUPPLIES

As well as online stores and eBay, craft shops such as Hobbycraft in the UK and Jo-Ann in the US stock soap basics, and health food shops sell essential oils. To be kind to the environment, look for soap bases with natural or organic ingredients and avoid palm oil unless it has an RSPO certificate from the Roundtable on Sustainable Palm Oil.





Left ♦ Reverse any asymmetrical design, as your mould is a mirror image of the soap

YOU'LL NEED

MOULD MAKING:

- ♦ TPU flexible filament
- ♦ 3D printer capable of printing TPU
- ♦ A few drops of light cooking oil (if 3D printer has a Bowden tube)
- ♦ SD card or lead for transferring design to 3D printer

SOAP MAKING:

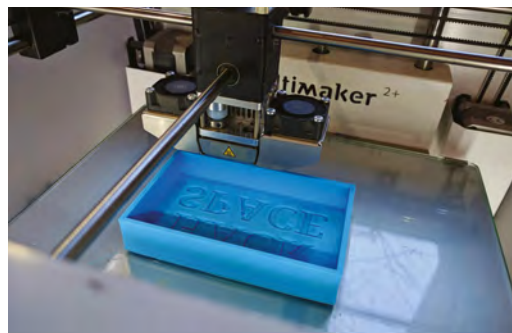
- ♦ Tweezers
- ♦ 500 g White glycerine soap base to make five bars
- ♦ 5 ml essential oil or fragrance
- ♦ A few drops of liquid soap colouring
- ♦ 1 × medium and 1 × small heatproof bowl
- ♦ Saucepan of boiling water or microwave oven
- ♦ Hob (if not using microwave)
- ♦ Oven gloves for removing bowl from heat and pouring
- ♦ Knife and chopping board
- ♦ Spoon for stirring
- ♦ Measuring spoons

Click on the hollow box, and use the tiny black triangle above it to move the hollow box upwards by 3 mm.

Select all, then click on the Align icon near the top-right of the screen. A grid appears with buttons around the edges. Choose the middle ones on both bottom edges, and the hollow box will be almost swallowed up inside the solid box. Click away from the objects to cancel the Align tool.



Instead of letters,
you can import a
logo or drawing



Above ♦ Use flexible filament, such as TPU, to make it easier to get the soap out of the mould

Select both boxes and click on the Group icon. After a moment the parts will merge, and you've created the basic mould. If you want to edit the individual parts, use the Ungroup button to separate the boxes and then group them again when you're done.

MAKE IT PERSONAL

Drag the word 'TEXT' from the Basic Shapes area and drop it anywhere on the workplane. Set the dimensions to be 2 mm thick and 2 mm above the workplane, then click on the Flip icon. Choose the horizontal arrow that appears below the word to reverse the letters. Select both the text and the mould to align and use the two central buttons that appear on the bottom plane to centre it as before. The text will be half submerged inside the base of the mould, but still visible.

Whenever the Text object is selected, a sidebar appears with options to type in words and change the font. With the text in its current position you get a clear idea of how it will look, so now's a good time to write your message and resize to fit. Also, change the text from a solid to a hole.

Instead of letters, you can import a logo or drawing that's been created in different software. →

SAFETY PRECAUTIONS

Check the safety precautions on bottles of essential oils before using and avoid touching the neat oils, except lavender, as they may burn the skin. Some essential oils are contraindicated during pregnancy or are too intense for soap.



Figure 1 ♦
Balancing the bowl on a metal cookie cutter or other heatproof object stops the bowl from bobbing about in the boiling water

// Once the soap base has melted, stir in the essential oils; when you add them to heated soap mix, they fill the room with a heady aroma **//**



Above ♦
Wispy trails of filament may appear in letter moulds. Use tweezers to remove them, otherwise they will anchor the soap in the mould

It needs to be saved as an SVG file, which stands for Scalable Vector Graphics. If your design is a JPEG or PNG, there are free, online conversion tools which will turn it into an SVG. Once you've imported a logo or drawing, you can resize and move it around in the same way as other shapes.

When you're happy with the design on your mould, select all and click Group. Export your design as an STL file and use slicing software such as Cura to prepare a file for 3D printing.

When setting up the 3D printer, choose an infill density of 40% to make the mould strong and durable. If your printer has a Bowden tube, flexible filament will kink slightly and could get stuck. Wipe a few drops of light cooking oil, or sewing machine oil, along the first half metre of TPU before inserting it in the tube to reduce friction.

After printing, your mould is ready to use for one colour-embossed soap. If you want your soap to have a contrasting colour for your lettering or logo, the most effective way is to make a separate mould for the design or an alphabet mould, if you plan to create soaps with different messages. Just repeat the process of sinking indented letters or logos,

but this time in a box 100×60×3 mm. Also, make the design 2 mm deep instead of 1 mm, so that the design will end up half embedded and half raised in the finished bar of soap. If the letters were just on the surface, they might fall off when the soap is used.

MELT AND POUR

Melting soap is similar to melting chocolate. First, cut the soap base into small chunks, and place about 100 g in the medium heatproof bowl. Either melt in a microwave using 30-second bursts, or create a double boiler by placing the bowl of soap base over a pan of boiling water on the hob (**Figure 1**). Once the soap base has melted, stir in the essential oils. These are highly concentrated scents extracted from plants, and when you add them to heated soap mix, they fill the room with a heady aroma. You may think you've added too much, but as long as you've

QUICK TIP

No 3D printer or laser cutter? Soap moulds come in all shapes and sizes and you can make use of household items such as bread tins and plastic food boxes.



QUICK TIP

When looking for inspiration online, the American spelling, 'mold', brings up many more relevant results than the British spelling, 'mould'.

Left

To create a simple rustic look, wrap soap in greaseproof paper. You can attach labels and list all the ingredients in case someone has allergies

measured correctly, and not used peppermint or pepper which really are too strong, the smell will become more pleasant as it dissipates and will be even more appealing in the finished bar.

ADDING COLOUR

Pour a little of the melted soap mixture into the small bowl. Add a few drops of soap colouring and stir, adding more to get the desired shade. If the soap mixture starts to harden before you're ready to pour, put it back in the microwave or double boiler to melt again. Carefully pour the soap mixture into the alphabet mould. Remove any excess with a knife on its side. This is easiest when the soap has been left for a few minutes and is starting to set.

When the soap has set, remove the letters, and press into position in the main soap mould. Top up with a different colour of soap base, and leave to set firmly before removing the soap. Avoid too much contrast between colours because a tiny amount of bleeding between colours may occur.

THAT'S A WRAP

Making your own soap helps cut down on unnecessary packaging, but when it's for gifts, you

may want to wrap bars individually. Templatemaker.nl has free customisable designs for little boxes, including pillow packs which are ideal for soap. Simply type in the measurements, click 'Create', and download your template. Then laser-cut, or print on card and cut out.

Glycerine soap with essential oils keeps for at least two years if wrapped well and stored in a cool, dry place. When other natural ingredients are added, such as dried flowers, the soap needs to be used much sooner. Avoid storing the soap in the bathroom before use, because glycerine attracts water from the air, and little droplets can form on the surface of the soap, which could soak the packaging. □

GOING ORGANIC

Organic soap is usually a natural yellow colour, which remains translucent when soap colours are added. It can be coloured using pink or white powdered clay, found in face masks. To prevent lumps, mix the desired amount of clay with a little liquid glycerine, available from supermarkets, before adding to the melted soap base. Avoid red clay because it may stain towels.

Maker's Toolbox: Heat gun

Ready, aim, melt!



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.

Heat guns are like supercharged hair-dryers: they blast out hot air. There's a wide range available, and the main difference is the amount of heat they can put out. Typically, tools aren't considered hot-air guns unless they can spit out at least 300°C, and powerful ones can go up over 700°C. These guns are often designed for stripping paint, but we can use the heat for a wide range of things.

Heat shrink tubing is plastic tubing that shrinks dramatically when it's heated. The exact temperature needed for shrinking depends on the particular formulation of plastic used, but it's typically around 150°C to 200°C. This is great for protecting exposed wires. For example, if you need to solder two wires together, you can cut a length of heat shrink that's wide enough to slot over the soldered joint. Feed one of the wires through it before joining them together, then solder them together. You can now reposition the heat shrink over the soldered joint, and heat until it shrinks and fits snugly over the joint.

Heat shrink comes in a range of sizes, and larger tubes can be used to cover small PCBs or components that have been soldered in-line with wires.

Shrinking plastic doesn't only come in tubes. You can get a couple of different types of sheets: vinyl wrap is used to apply a new surface to cars and other vehicles, and shrink wrap can be used to bundle things together securely.

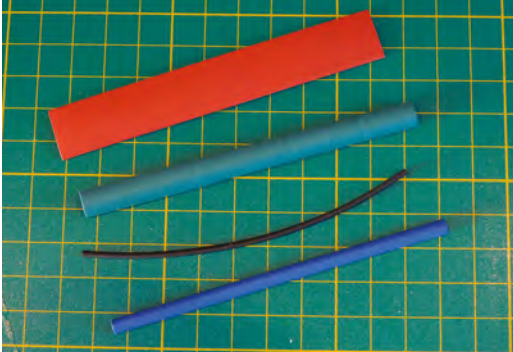
PLASTIC FANTASTIC

Most – though not all – plastics soften when heated, and many soften enough for us to change their shape. Acrylic sheet can be cut to size, then heated and bent into almost any shape. Most plastic boxes can also be modified with the application of a little heat.

When it comes to plastic, though, there's one area of making that's all about heating and manipulating: 3D printing. A bit of heat can help sort out printing problems, remove stubborn prints from unheated beds, and tweak design errors without reprinting. For obvious reasons, 3D printer plastic tends to melt at quite a low temperature, so care is needed, and heat



Right ♦
The SparkFun Heaterizer needs 110V, so if you're in a country with a different mains voltage, you'll need a transformer



guns with temperature controls can be more useful than the 'on/off' types.

Plastic is, of course, not the only thing that melts with heat. If you get metal hot enough, it'll melt. The only common metal that melts at a low enough temperature for a heat gun is solder. Take a look at the box below for heat guns in electronics soldering. You can also use a heat gun to solder pipes together, if the pipes are small enough and the heat gun powerful enough. A blow torch is a better tool for the job, but a heat gun might get you through at a pinch.

// Plastic is, of course,
not the only
thing that melts
with heat

IN THE WORKSHOP

Let's take a look at two heat guns: the rather grandly named SparkFun Heaterizer XL-3000, and the more mundanely titled Black and Decker KX1650-GB. The SparkFun Heaterizer, despite its 'XL' claim, is quite modestly powered, sucking in just 300 watts and spitting out air at about 250°C. This is plenty for heat shrink and plastic work. The nozzle is quite narrow, making it possible to direct the hot air with reasonable precision. It's just about hot enough to reflow solder for PCB work. For light workshop use, it's a great tool.

The Black and Decker is a more industrial tool. Its two heat settings are 460°C and 600°C. At these temperatures, you've got to be a bit more careful to avoid burning (both the thing you're heating and yourself). It's a bit of a beast for workshop use with heat shrink, etc., but if you want one tool for DIY use (paint stripping, etc.) and workshop, then this will do it. ▢



HOT-AIR REWORK STATION

Most people learn to solder using a soldering iron – a pen-shaped device with a heated metal tip. These are great for basic soldering as you can control the position of the heat very precisely. For through-hole and larger surface-mount components, these are the best choice. However, when working with tiny components (or components with connections underneath them), soldering irons are much harder to use.

These components aren't really designed to be hand-soldered – in production, the whole PCB (components, solder, and all) is placed in an oven that's heated until the solder is melted, then cooled, so it solidifies in the right place. However, this isn't practical if you need to fix something.

Hot-air rework stations are a special type of hot-air gun designed specifically for fixing circuit boards. They're desk mounted with a hand-held tool that's more accurate than a gun-style tool, they usually have more control over heat and airflow, and have a smaller nozzle for more accurate direction of the hot air.

You can use a regular heat gun for solder-work, but the wide blast of heat can make it tricky to work with, and if they're too hot, they can damage components.

Above left ▣ Heat shrink tubing comes in a range of sizes, and you need to match the size with the object you want to cover

Above ♦ In principle, this gets hot enough to braze with, but we were unable to get an object up to brazing temperature as too much heat was conducted away from the joint to be brazed

Build an ISS count-down timer

Know when the International Space Station is passing overhead with this timer



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.

Below

You can use any code editor you like. We've used Mu, but most programmers' editors work well with Python

```
1 import urllib.request
2 import json
3 import time
4 import board
5 import neopixel
6 from math import floor
7
8
9 pixels = neopixel.NeoPixel(board.D18, 6)
10
11
12 lat=51.45
13 long=2.89
14
15 colour_days = (50,0,0)
16 colour_hours = (0,50,0)
17 colour_tenmins = (0,0,50)
18 colour_mins = (15,15,15)
19 colour_now = (50,50,50)
20
21 ISS_url = "http://api.open-notify.org/iss-pass.json?lat="+str(lat)+"&lon="+str(long)
22
23 wait_time = 1
```

The International Space Station (ISS) is one of the greatest achievements of mankind. Ever. It's certainly one of the most expensive with an estimated cost of around \$150 billion, and it's continuously housed people in space since November 2000. One of the most impressive things about it is that it can be seen from Earth. Every 90 minutes, it completes an orbit of the Earth, but the Earth is rotating underneath it, so each orbit goes over a slightly different section of the Earth.

Depending on where you are, and the particular week and month it is, you will probably find that you can see the ISS a few times a month, but when? We'll build a notifier that lets you know when the ISS will next be overhead.

Fortunately, the hard work of working out when the ISS will be overhead has been done for us, and there's a web API we can call to get the information. For this to work, you need to know your latitude and longitude. For example, in Bristol we can use the following URL with the relevant 'lat' and 'lon' values:

api.open-notify.org/iss-pass.json?lat=51.45&lon=2.89

You can enter this in a browser, and you'll get a response something like this:

```
{
  "message": "success",
  "request": { "altitude": 100, "datetime":
1552572620, "latitude": 51.45, "longitude":
2.89, "passes": 5 },
  "response": [
    {
      "duration": 551,
      "risetime": 1552601375
    }
  ]
...}
```

This encoding is called JavaScript Object Notation (JSON), and it's pretty common on web APIs, as it allows structured data to be passed via text. The important information for us is in the response section. This gives us a list of the next five times the ISS passes over us, with a duration of the pass (which we won't use), and the time it'll first be visible (which we will).

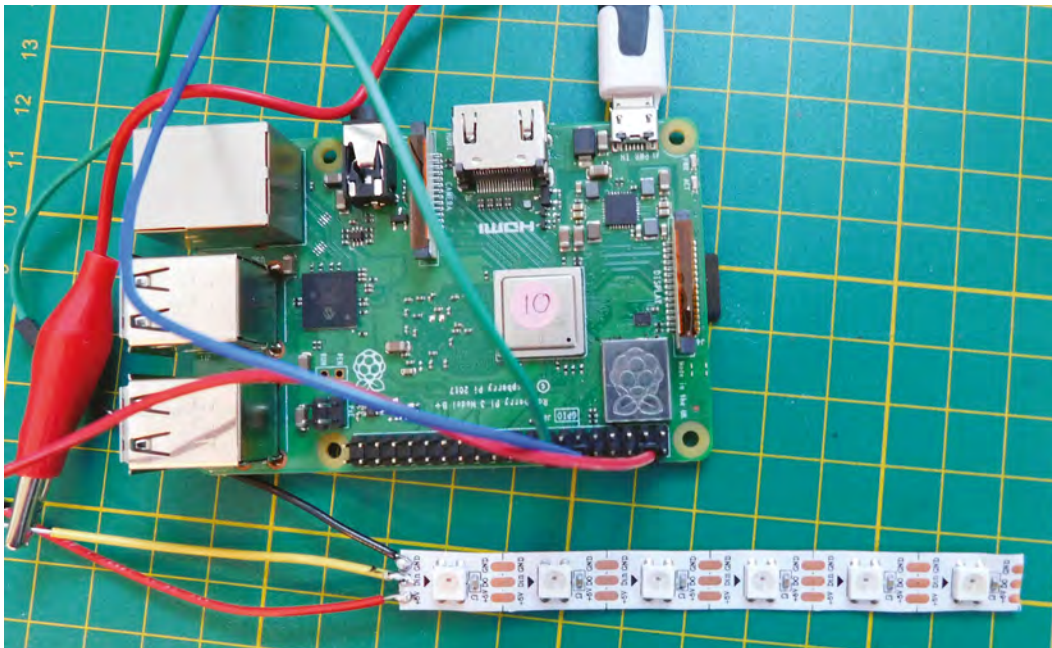
You can get this from any internet-connected device, and you could build this notifier on a microcontroller, but we'll be using a Raspberry Pi. Any of the WiFi-enabled models should work, but we'll use a 3B+.

We'll be using Python 3 for the brains of our project. The very basic code that grabs the next rise time is:

```
import urllib.request
import json

lat=51.45
long=2.89

ISS_url = "http://api.open-notify.org/iss-pass.js
```

Left ♦
All wired up and ready to run

```
n?lat="+str(lat)+"&lon="+str(long)

with urllib.request.urlopen(ISS_url) as url:
    data = json.loads(url.read().decode())
    print(data)
    print(data['response'][0]['risetime'])
```

// **JavaScript Object Notation (JSON) is pretty common on web APIs as it allows structured data to be passed via text**

As you can see, this uses the JSON module to parse the response from the API and convert it into a Python data structure that is a combination of lists and dictionaries. We can then grab the bit of data we want out of this data structure.

GETTING PHYSICAL

The next step is to create the data display. You can obviously use whatever data output device you want, but we're going to go with a row of six NeoPixels. The display will be:

- If there's more than a day until the next time the ISS passes overhead, all will be red
- If there are between one and 24 hours, the LEDs will show a binary count-down of hours in green
- If there are between ten and 60 minutes, the LEDs will show a binary count-down of ten-minute periods in blue
- If there are between one and ten minutes, the LEDs will show a count-down of minutes in white
- If there's less than one minute to go, all the LEDs will glow bright white

There's obviously quite a lot of binary count-downs in that list, so we need a way of converting a time period into data that we can send to our NeoPixels.

The NeoPixel module takes a list of tuples, where each tuple contains three digits: one for red, green, and blue. We can generate this list with the following function:

```
def countdown_leds(seconds, period, colour):
    led_array = []
    binary_string = "{0:b}".format(floor(seconds/
period)).zfill(6)
    for digit in binary_string:
        if digit == '1':
            led_array.append(colour)
        else:
            led_array.append((0,0,0))
    return led_array
```

The key part of this function is the line:

```
binary_string = "{0:b}".format(floor(seconds/
period)).zfill(6)
```

This uses a slightly obscure part of the **format** string function to convert the number into a binary string. The **format** function takes some data and puts it in a placeholder in a string. The standard placeholders are {0} .. {n}. In our case, our string is only a placeholder, but this doesn't have to be the case, →

YOU'LL NEED

- ♦ **Raspberry Pi**
- ♦ **Strip of 6 NeoPixels**
- ♦ **Connecting wire**

it could have been: “the binary digit is {0:b}”. The placeholders can also take specifiers which, in our case, is a b which specifies that we want to insert the number as a binary string. The last part uses **zfill** to fill out the left-hand part of the string with zeroes until it's six digits long.

The main part of our code needs a few bits to work, including some modules and variable definitions:

```
import urllib.request
import json
import time
from math import floor

lat=51.45
long=2.89

colour_days = (50,0,0)
colour_hours = (0,50,0)
colour_tenmins = (0,0,50)
colour_mins = (15,15,15)
colour_now = (50,50,50)

ISS_url = "http://api.open-notify.org/iss-pass.json?lat="+str(lat)+"&lon="+str(long)

wait_time = 1
```

This pulls in a few modules, defines the URL (you'll need to change the latitude and longitude for your location), and defines the colours we'll use. The only unusual bit there is the **wait_time** variable. This holds the number of seconds our code will wait between calls to the API. We keep calling the API to get the latest information, but we don't want to hammer the API, so we pause for a given number of seconds between calls. The API is rate-limited to one request a second, so we shouldn't exceed that, but really,

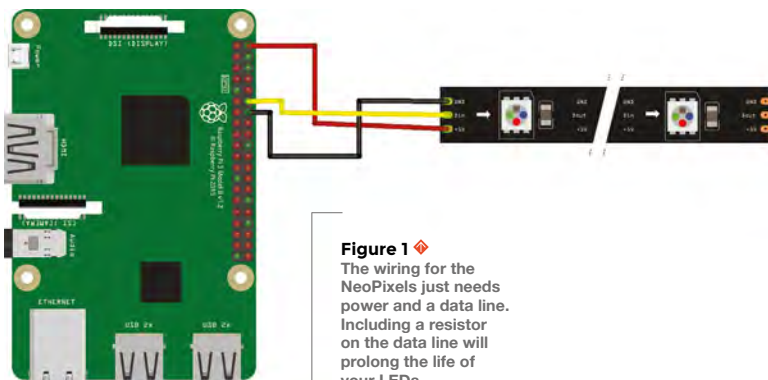


Figure 1 ♦
The wiring for the NeoPixels just needs power and a data line. Including a resistor on the data line will prolong the life of your LEDs

VOLTAGE PROBLEMS

Technically, NeoPixels require the data signal to be at least 0.7 times the drive voltage. We'll be driving them off the 5V pin (as the 3V pin isn't enough to drive them). This means that we should have a data signal of at least 3.5V. The GPIO pins on the Raspberry Pi are only 3.3V. Usually, we find this doesn't cause a problem, but if your NeoPixels are behaving badly, you may need to do something to either decrease the drive voltage, or increase the data voltage. There are some tips for doing this here: hsmag.cc/eYvUWK.

we don't need to even be close to that. We'll use a sliding wait period, where the longer until the next predicted passover, the longer we'll wait.

The code for the main loop is then:

```
while True:
    time.sleep(wait_time)
    with urllib.request.urlopen(ISS_url) as url:
        data = json.loads(url.read().decode())
        next_vis=data['response'][0]['risetime']
        seconds_to_next_vis=data['response'][0]
        ['risetime'] - time.time()
        if seconds_to_next_vis > 86400:
            neopixel_array = [colour_days]*6
            wait_time = 600
        elif seconds_to_next_vis > 3600:
            neopixel_array = countdown_
            leds(seconds_to_next_vis, 3600, colour_hours)
            wait_time = 60
        elif seconds_to_next_vis > 600:
            neopixel_array = countdown_
            leds(seconds_to_next_vis, 600, colour_tenmins)
            wait_time = 10
        elif seconds_to_next_vis > 60:
            neopixel_array = countdown_
            leds(seconds_to_next_vis, 60, colour_mins)
            wait_time = 2
        else:
            neopixel_array = [colour_now] * 6
            wait_time = 2
    #write LEDs to neopixels
    print(neopixel_array)
```

As you can see, this has one big if-else-if-else statement that calculates the correct lighting for the particular time. If you bring this all together, you'll get the code for turning the NeoPixels on and off. Now, let's look at the hardware.

// You can cut the LED strip to any length; just cut through the exposed pads **//**

CONNECT THE LEDS

First, we'll need six NeoPixels. These come in a wide range of forms, but we'll go with a common LED strip cut so there's six. You can cut the LED strip to any length; just cut through the exposed solderable pads, making sure that there's enough to solder onto on each side. Make sure you solder to the correct end of the strip: there are data-in and data-out sides. Solder to the data-in side (there may be an arrow indicating the direction of data flow). You should have a power wire, a ground wire, and a data wire. It's good practice to include a 330Ω resistor on the data wire before the first pixel, but this isn't necessary for it to work (it will mean the pixel lasts longer).

The 5V wire can be connected to a 5V pin on the Raspberry Pi (see the Voltage Problems box on previous page), and the ground to a ground pin. The data connection should go to pin 18 (see **Figure 1**).

There's a bit of software you need to install to make it all work. Type the following into a terminal session on the Raspberry Pi:

```
sudo apt install python3-pip
sudo pip3 install rpi_ws281x adafruit-
circuitpython-neopixel
```

You can check that this is all working by opening a Python 3 prompt with root permissions. Root is needed to control the NeoPixels; enter **sudo python3**. Now, enter the following code:

```
>>> import board
>>> import neopixel
>>> pixels = neopixel.NeoPixel(board.D18, 6)
>>> pixels[0]=(0,10,0)
```

This should turn the first NeoPixel on green. You can use this to turn on any pattern you like. Once you've finished playing, let's get back to our main code.

You need to include the lines to set up the NeoPixels directly under the other **import** blocks:

```
import board
import neopixel

pixels = neopixel.NeoPixel(board.D18, 6)
```

```
25 def countdown_leds(seconds, period, colour):
26     led_array = []
27     binary_string = "{0:b}".format(floor(seconds/period)).zfill(6)
28     for digit in binary_string:
29         if digit == '1':
30             led_array.append(colour)
31         else:
32             led_array.append((0,0,0))
33     return led_array
34
35 while True:
36     time.sleep(wait_time)
37     with urllib.request.urlopen(ISS_url) as url:
38         data = json.loads(url.read().decode())
```

Now, you just need to write the values from the **neopixel_array** list to the **pixels** list. Add the following directly under **print(neopixel_array)** at the end of the main loop.

```
print(neopixel_array)
for i in range(0,6):
    pixels[i] = neopixel_array[i]
```

That's all there is to the main code. You can run this by entering:

```
sudo python3 iss.py
```

You'll need to change **iss.py** to whatever you've called the file. Running like this works, but you'd need to re-run this command every time you turned on the Pi. Obviously, not ideal for something that's designed to be embedded. There are a few ways of configuring your Pi to run a script every time you turn it on, but one of the easiest is to add a line to the **/etc/rc.local** file. You'll need to edit this as root, so enter the following in a terminal:

```
sudo nano /etc/rc.local
```

This opens the nano text editor and lets you edit the file. Immediately above the line **exit 0**, add:

```
python3 /home/pi/iss.py &
```

Here, the code is saved as **iss.py** in the Pi user's home directory, but you can change this if you prefer. The **&** at the end tells the script to continue running and run our command in the background – essential as **rc.local** will block the boot process if it doesn't finish. Hit **CTRL+X** to save and exit.

That in place, you can restart your Pi, and it should run the script automatically. All you have to decide is how to mount and show off your ISS notifier! □


Above ♦
One function takes care of the conversion from seconds to a binary clock

3D-printed jacket mod with LEDs

3D-print a design directly onto fabric for a flexible 3D appliqué



Sophy Wong

 @sophywong

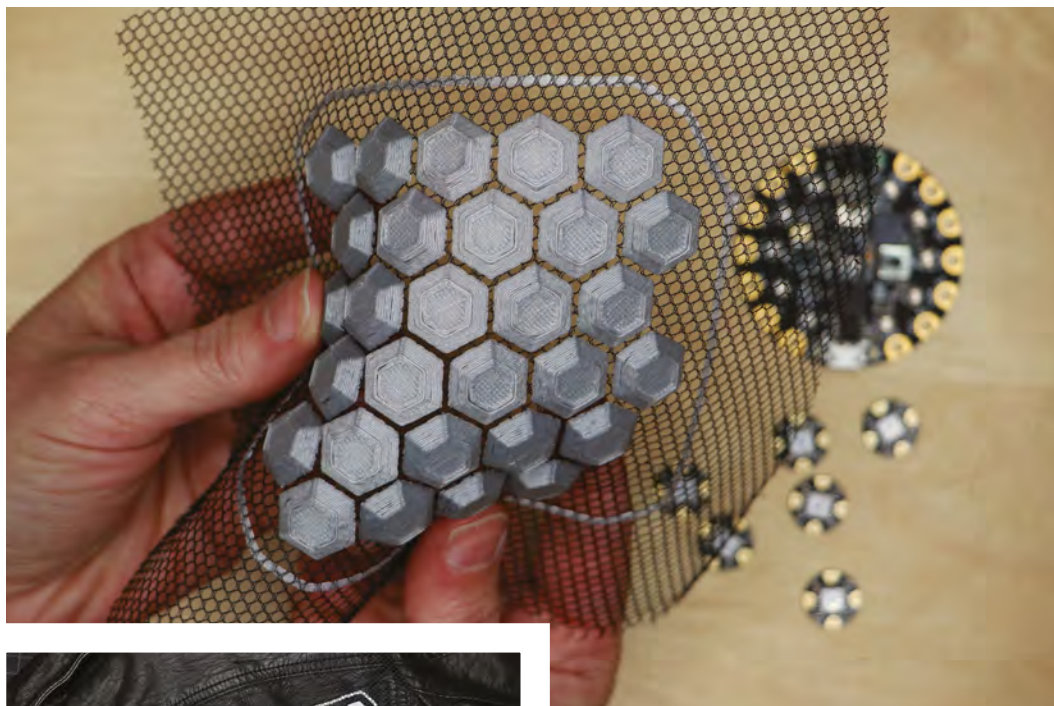
Sophy Wong is a designer, maker, and avid creator. Her projects range from period costumes to Arduino-driven wearable tech. She can be found on her YouTube channel and at sophywong.com

3D printing is a great way to make custom parts for projects, and it's perfect for wearables! Making your own models in 3D takes time to master, but sites like Thingiverse.com and MyMiniFactory.com can get you started with models designed by others in the 3D printing community. There are lots of ways to incorporate 3D printing into wearables, from custom enclosures for your electronics, to sewable diffusers for LEDs, and more. In this project, we'll print directly onto fabric for a flexible 3D appliqué that can be sewn to any garment.

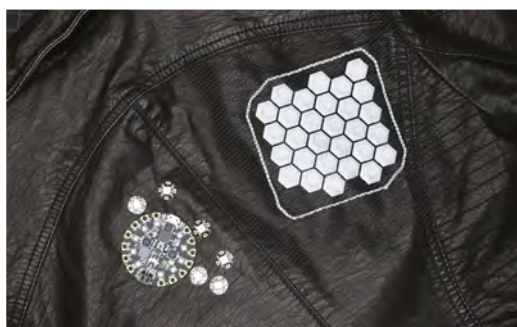
If you haven't already seen the technique of 3D printing onto net or tulle fabric, seek out the work of David Shorey, Penelopy Bulnick, Amie Dansby, and others – and prepare to be amazed! Printing a repeating pattern on fabric can create fluid, flexible panels that mimic feathers, chain-mail, or even dragon scales.

In our project, we'll use a simple pattern of extruded hexagons, and make two 3D-printed panels that will be sewn to the back of a jacket. We'll pattern our pieces directly off the jacket so that they fit perfectly. Translucent filament is a fantastic diffuser for LEDs, so we'll add some lights to the back of the jacket. We'll control our LEDs with a Circuit Playground Express, using its on-board buttons to switch between animations. If you've never programmed a microcontroller before, the Circuit Playground Express (CPX) is the perfect place to start. Coding the CPX is simple with MakeCode, an online block-based visual programming tool from Microsoft. The finished piece will be a programmable, 3D-printed LED jacket, fit for your next cyberpunk adventure!

Printing on fabric is definitely an experimental technique, so expect to do some tests to dial in the settings on your printer first. Use synthetic netting, mesh, or tulle fabric for this. These fabrics have large



Left ♦
Our 3D patch,
ready to sew
onto the jacket



Above ♦
The main components for the project

openings between their yarns that allow the 3D-printed layers to fuse together around them. In a nutshell, the process is as follows. Print the first few layers directly on the printer bed, as you'd normally do. Then, pause your print and lay the fabric down over the already printed layers. With the fabric held in place using tape or clips, resume the print and let it finish. When complete, the fabric will be sandwiched into the base of your print, connecting the separate shapes of your pattern together into a flexible panel.

For this project, we recommend laying the fabric down after just one layer, so that the finished panel will be as flat on the underside as possible. You can start with our hexagon model, but you may need to rearrange the hexagons in a 3D modelling program to fit your jacket. You can create your own model, or search on **Thingiverse.com** or **MyMiniFactory.com** for other models designed for printing onto fabric. It is well worth doing a web search before starting, to

get a better idea of the process. If you're new to 3D printing, practise by printing regular models (not on fabric), to familiarise yourself with your printer and the troubleshooting required for great results with this medium.

Choose a jacket with a lining so that the wiring can be completely hidden and protected during wear. You'll also need a pocket for housing the CPX and the battery pack. Black faux leather is easy to cut holes in, won't fray, and looks positively Daft Punk!

EXPERIMENT WITH PRINTING ON FABRIC

Start by doing some test pieces and dialling in your settings for 3D printing on the net fabric. Use a slicer program to convert our **jacket-panel-3D.stl** file (from hsmag.cc/issue18) into G-code for your printer. We used Cura, which provides an option to automatically pause the print after a specified layer (in Extensions > Post Processing > Modify G-code > Pause At Height). The script also moves the printer head out of the way for easy access. Set the standby and resume temperatures to match your print temperature, so that the printer can resume quickly without needing to reheat. Note: modifying G-code is an advanced technique, and can cause damage to the printer if done incorrectly. Proceed with caution, and do not attempt without full understanding of the modifications you are making.

Whether using a G-code script or pausing manually, do some research on how to pause a print with your specific slicer and printer. The 3D-printing community is full of wonderful contributors generously sharing →

YOU'LL NEED

- ♦ **Jacket with lining and pockets**
- ♦ **Synthetic net or tulle fabric**
- ♦ **Blue painter's tape**
- ♦ **3D printer**
- ♦ **Translucent filament** (shown: MatterHackers clear PLA)
- ♦ **Circuit Playground Express**
- ♦ **Flora sewable NeoPixels**
- ♦ **AAA × 3 battery pack with on/off switch**
- ♦ **Set of three-pin connectors such as JST PH**
- ♦ **Silicone-coated stranded wire**
- ♦ **Soldering tools and supplies**
- ♦ **Heat-shrink tubing**
- ♦ **Hand-sewing supplies**



Synthetic fabrics are sensitive to heat. It's a balancing act between good adhesion and protecting your fabric from damage

Above ♦
Measure the panel on your jacket so you can make your 3D print the correct size

their experiences and methods, and that information is extremely helpful when attempting this project. Also, consider helping the community grow by sharing your experience and results so others can learn from you too!

Other settings you'll want to tune include:

Bed Temperature – Synthetic fabrics are sensitive to heat (they're basically plastic too!) and in some cases, you may not want to expose them to a heated printer bed. It's a balancing act between good adhesion and protecting your fabric from damage. In our project, the synthetic netting held up fine with our regular bed temperature for PLA (60°C).

Skirt – Printing with a skirt (a solid line around the model, but not touching it, a few layers high) can be useful for holding the tulle in place during printing. We were able to get satisfactory results without one, but if the tape is not holding your fabric down well enough, you might try printing with a skirt. After printing, you can cut the skirt off, but keep in mind that you will need 1 cm or so of seam allowance

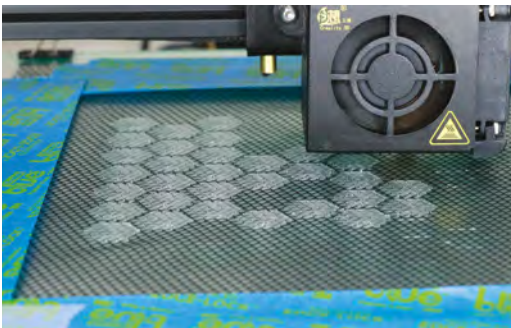
for sewing the panel to your jacket. You could also choose to incorporate it into your design as an outline. Run through the entire procedure several times:

1. Start the print.
2. Pause your printer after the first layer. (Or, wait for the automatic pause if you inserted a pause command in your G-code.)
3. Tape your fabric in place on the printer bed, then resume the print. When placing the fabric, be very careful not to let it touch the hot printer nozzle or it will melt. Tape it down on all sides so that the fabric is held flat and taut directly on top of the print. Place the tape far enough outside the print so as not to be in the way while printing. You'll want to keep the pause as brief as possible, for good adhesion of the two layers separated by the pause. This is the trickiest part of the process, and doing a few practice runs before your final pieces will help.
4. When finished, let the printer bed cool completely before trying to remove the print. Do not pull on the fabric to release the print, or it may tear.
5. When you've got satisfactory results, you're ready to move on!



MAKE A PATTERN

If you're using our hexagon model as is, you can skip right to the printing. But if you're making a design



Above ♦
3D-printing the panels on fabric

from scratch, you'll want to create a custom pattern based on your jacket. Here's how to do it.

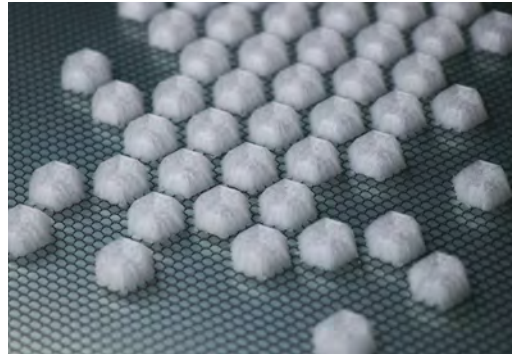
Use blue painter's tape to make a pattern of the upper back of the jacket. The panels will be symmetrical, so you only need to do one side. Place overlapping pieces of tape on the jacket on an area larger than your design will be. Then, use a marker to draw the shape you want your fabric panel to be. Pull all the tape off in one piece, and stick it down to a piece of paper. There's your pattern!

Now you can scan this pattern into your computer and trace it in a vector program, like Adobe Illustrator, to digitise it. Then you can create your 3D design using the traced vector shape as a guide, and know that it will be the correct size and proportions for your jacket. Remember to leave 1 cm or so around the edges plain, for sewing. If your design is symmetrical, like ours, you can mirror the design to create the second side.

To check your design, print a 2D version on paper and cut it out. Lay it over your traced pattern to make sure the size is still correct. Then, lay the printed pattern over your jacket to make sure that you like the placement of everything.

PRINT YOUR FABRIC

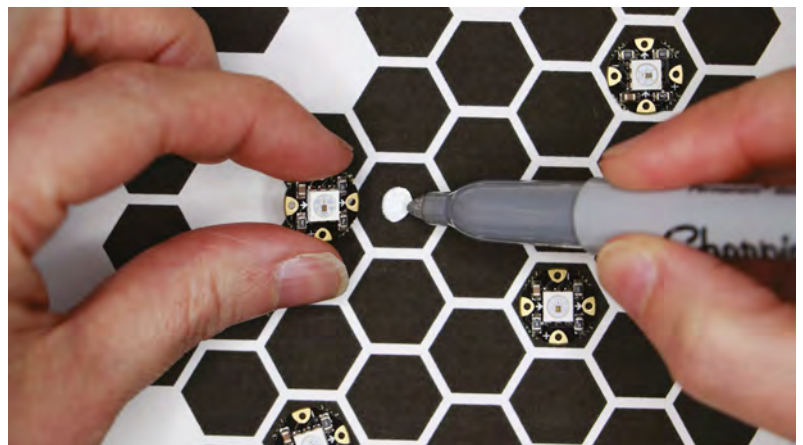
Using the settings you determined with your test prints, start printing your first panel. Use a piece of your net fabric large enough so that the painter's tape will be placed close to the edges of your printer



Far Left ♦
The 2D print, ready for the circuit design

Left ♦
Little lumps of 3D-printed loveliness

Below ♦
Marking out the position of the LEDs on the jacket

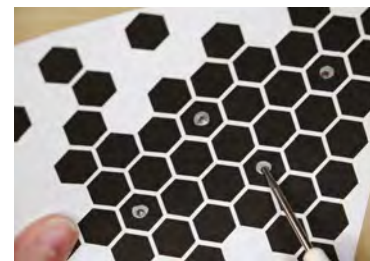


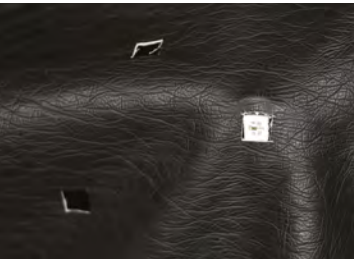
bed. You'll cut the panel down to size after printing. After the first layer, pause, tape your fabric in place, and resume. This is where all that practice pays off. Watch the print for a couple of layers to make sure the insertion was successful.

BUILD CIRCUIT

While your fabric is being printed (each of our panels took about 2.5 hours), work on your circuit! Print and cut our 2D design file, making sure to print at 100% scale (or use the printed pattern piece you made for checking your design). Lay out your NeoPixels where you want them to go in the design. You'll get great diffusion by placing them directly under the 3D-printed hexagons. Mark the centre of each hexagon that will have a NeoPixel, and use an awl to make a hole at each mark. Then, lay the pattern in place on your jacket and mark the placement of each hole. Flip the pattern over to mark the second side.

On the inside of your jacket, use a seam ripper to open the centre seam of the jacket lining, exposing the inner workings of your jacket. Make the opening large enough so that you can move the lining out of the way when cutting the holes for the NeoPixels. →

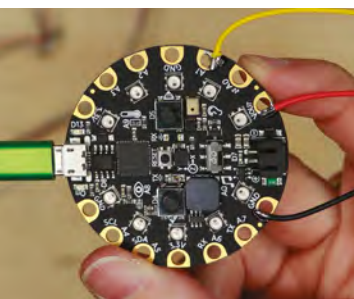
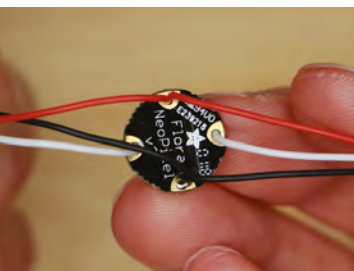




Above ♦
Using blue tape as a guide for cutting a hole for the NeoPixel

Below ♦
Wiring a Flora NeoPixel

Bottom ♦
The first NeoPixel in the chain is connected to the CPX



Use a craft knife to cut a square hole at each mark for the NeoPixel to show through, making sure not to cut into the jacket lining. A small square of blue tape works well as a cutting guide.

Use your template to measure the distances between each NeoPixel in your design and cut wires to length for your NeoPixel chain. For the first NeoPixel in the chain, cut longer wires that will reach the pocket on the front of the jacket where your Circuit Playground Express will live. Add a few centimetres to all your wire lengths for insurance. Note the small white arrow on each NeoPixel that indicates the direction that data flows, from in to out. When chaining the NeoPixels together, all the arrows must point the same way.

To allow the NeoPixels to lay flat against the jacket fabric, feed wires into the back of each NeoPixel and solder on the front. Start by soldering a long wire to the data-in pin on the first NeoPixel. The power and ground pins will have two wires each: one long wire (for connecting to the Circuit Playground Express), and one shorter wire (for connecting to the next NeoPixel). Lightly twist the two wires together before feeding them through the hole and soldering in place on the first NeoPixel. Then, solder a short wire to the data-out pin, and move on to the next NeoPixel.

Repeat the above process to keep adding NeoPixels to the chain, continuing to connect power to power, ground to ground, and data-out to data-in on each subsequent NeoPixel. The last pixel will have only one wire for power, ground, and data-in.

// We'll use two NeoPixels on the CPX to tell us which animation is currently showing on the rear

For easy installation, add a three-pin connector between the first NeoPixel (at the beginning of the long wires) and the Circuit Playground Express. This will make it possible to attach and detach the Circuit Playground Express, and make installation easier. To connect the Circuit Playground Express to the first NeoPixel, solder the connector wires so that Vout connects to power, GND connects to ground, and A1 connects to data-in.

PROGRAM CPX

Your print is probably still going, so let's do some coding while we wait. We're going to program the

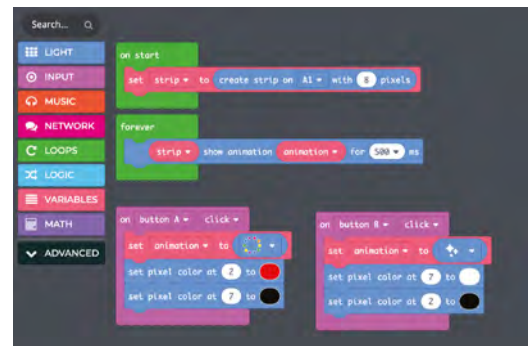


Figure 1 ♦
The MakeCode program for our jacket

Circuit Playground Express with MakeCode, and use the on-board buttons to choose between two animations. Since you won't be able to see the lights on your back, we'll use two NeoPixels on the CPX to tell us which animation is currently showing.

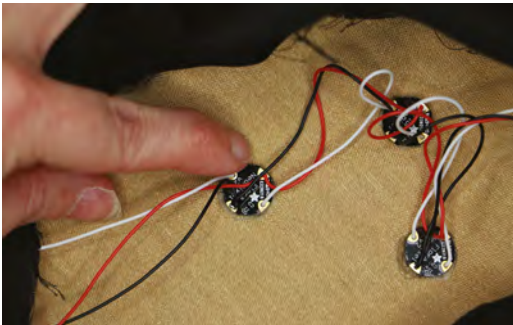
Before running any power through your circuit, lay the chain of NeoPixels out flat on your table so that no uninsulated connections are touching each other. A short circuit can cause damage and be dangerous, so don't skip this step. Plug the CPX into your computer with a micro USB cable, and navigate to makecode.adafruit.com to get started!

If this is your first time working with MakeCode, Adafruit has a handy guide that will get you up and running in no time: learn.adafruit.com/makecode. The guide also contains helpful information on downloading and flashing your code to the Circuit Playground Express when you're all done. Here's a look at how the MakeCode program (**Figure 1**) for this project works...

In the 'on start' block, we use a NeoPixel block from the Light menu to create our NeoPixel chain as an object called strip. In this block, we also tell the CPX that the chain is connected to the A1 pin, and contains eight NeoPixels.

In the 'forever' loop, we simply tell the Circuit Playground Express to play an animation for 500 milliseconds. Because it is a loop, it will play this animation over and over seamlessly. We could choose just one animation to play forever, but we want to use the two buttons on board the Circuit Playground Express to show two different animations. To do this, we've also created a variable in this line called **pattern**.

Finally, we've created two blocks, one for button A and one for button B. In each block, we've chosen to set the variable **pattern** to a different animation



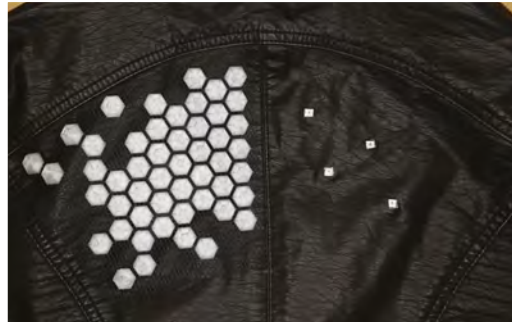
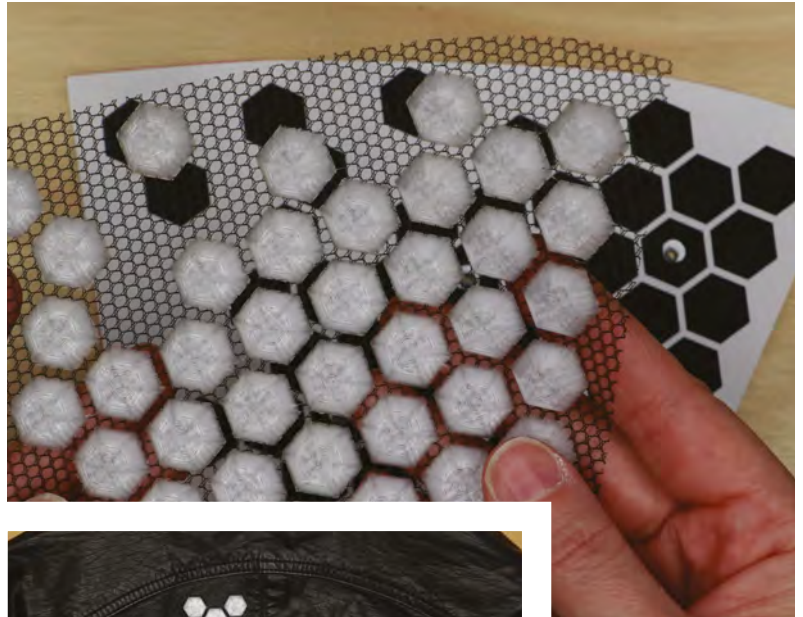
pattern – button A sets it to a rainbow, and button B sets it to a white twinkle. We’ve also selected a nearby pixel from the on-board ring on the Circuit Playground Express to illuminate depending on which button has been pressed. This will make it easy to keep track of which button you’ve just pressed, and which animation pattern is currently showing. Button A will turn the pixel next to it red, and button B will turn the pixel next to it white.

When your code is complete, click Download and follow the on-screen instructions to install the code on your board. Press the buttons on your board and watch the animations change. When everything is working as you want it to, you’re ready to sew it in!

INSTALL CIRCUIT

Unplug the Circuit Playground Express from the NeoPixel chain at the three-pin connector. Behind the lining on the inside of your jacket, locate the back of the front pocket that will hold the Circuit Playground Express. Use a seam ripper to open a bit of the seam so you can feed the three-pin connector end of the NeoPixels into the pocket. Pull the connector through, and out of the pocket a little way. On the inside of the lining, before the wires enter the pocket, make a loop in the wires and anchor it with a few hand-stitches to prevent pulling on the NeoPixel chain. Sew the hole in the pocket closed around the wires.

Then, start gluing the NeoPixels in place on the wrong side of the fabric at the back of the jacket. Faux leather should take hot glue well, but it’s safest to use a low temperature setting. Test in a small inconspicuous area first. Apply glue to the front of each



Above ♦
Everything coming together in the final assembly

NeoPixel, then place them so that the raised square LEDs show through the square holes you made. On the back of each NeoPixel, cover each solder point with a dab of hot glue for strain relief and added protection.

ATTACH PANELS

When both of your 3D-printed fabric panels are done printing, rejoice! Using your printed paper patterns, cut each panel into the correct shape. Then lay the panel on top of your LEDs and make sure everything lines up nicely, and the LEDs will be covered correctly.

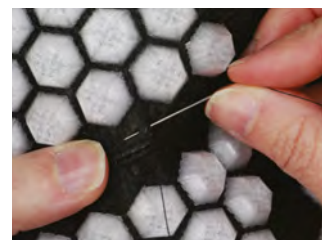
You can use a tiny misting of spray glue on the back of each panel to hold it in place while sewing. Hand-sew around the edges of each panel with a long straight stitch to attach securely. Be deliberate about each needle hole you make, as all holes are permanent in leather, pleather, and vinyl.

FINISHING

Close up the lining at the centre seam by whip-stitching over the section you opened up. Plug the battery pack into your CPX, and flip the switch on to light up your 3D-printed fabric panels!

We’ve touched on a lot of different processes in this project, and now you’re primed to take them further! Share your experiments with us at hackspace@raspberrypi.org! ♦

Below ♦
A few stitches hold the panel in place



A binary keyboard for programmers

Type with just two keys and good knowledge of ASCII



Andy Clark

workshopshed

For the last ten years, Andy has been making and repairing in a shed at the bottom of the garden. You can see more of his exploits at workshopshed.com

It is said that there are 10 types of people in the world; those who understand binary and those who don't. So if you do, why have all those keys on your keyboard: all you need is a one and a zero. By using a microcontroller that supports a HID (Human Interface Device), we can have a keyboard that lets you type the ASCII codes for the letters using just two keys.

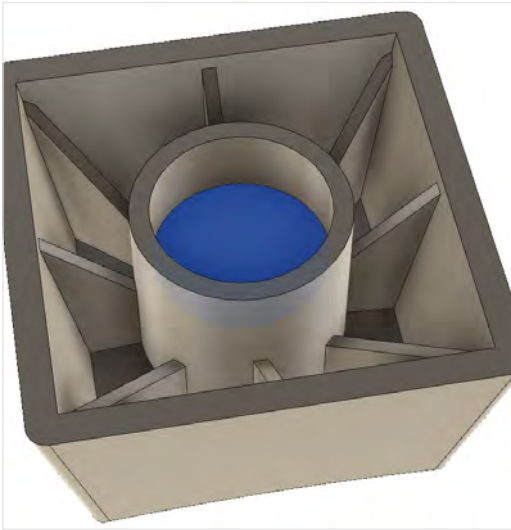
The main design principles used for this project are to create a strong, but light, item and to avoid the need for supports when printing. We can use the same key design for both keys and add the numbers before exporting them as an STL file. The key starts as a square sketch on a plane in Fusion360. From this, we can extrude up with a 10° taper to give the key its basic shape.

To give the distinctive rounded top to the key, we subtract a small amount of a large sphere from the top. This is done with the Combine Objects command. Select the key as the target body and the sphere as the tool body and cut away the required piece. The outside of the key is finished by filleting the edges to give them a rounded shape.

To remove the middle of the key, flip it over and select the base face. Choose the shell command and set the wall size to 2mm. To allow this hollow box to print without supports, we add ribs. Select two opposite faces and create a midplane. Sketch a diagonal line on this plane from the middle of one wall to the centre of key. Choose the rib command, and create a symmetrical rib 2mm across. Repeat this step to have ribs on the other four faces. Now choose



Right
The ultimate Geek Chic
keyboard



YOU'LL NEED

- ◆ Adafruit ItsyBitsy, or an Arduino Micro
- ◆ 2 × microswitch
- ◆ Filament for your 3D printer
- ◆ 2 × 30 mm compression spring
- ◆ Block of wood

adjacent faces to create a midplane and add ribs on the diagonals.

To form the plunger, use the base to create an offset plane with zero offset. Sketch a 33 mm diameter circle on the plane, and extrude up till it meets the top of the key. Return to the plane and sketch a second concentric circle 22 mm and extrude that up 15 mm, cutting a hole in the middle. Filleting all of the internal top edges will mean the 3D printer has less spanning to do.

To form the letters on the top, sketch the outline of the numbers and extrude to give it depth. It is a good idea to fillet the edges of your numbers. Next, move the number to the top of the key and use the 'combine objects' command to cut the number from the top of the key.

The last step is to add some clips to the bottom of the cylinder, to stop the key from rising too far. Sketch lines on the bottom of the cylinder and then extrude up to remove the material. Add cylinders to the end of the clips by drawing a circle on the side and extruding.

Once your key is complete, use the STL export to produce a file to print; alternatively, you can print directly from Fusion360.

NEED A LITTLE SUPPORT


The keys slide over a 21 mm cylinder which is mounted on a simple square base. Start by creating a cylinder, then removing the middle to make it into a tube. Cut across with a rectangle.

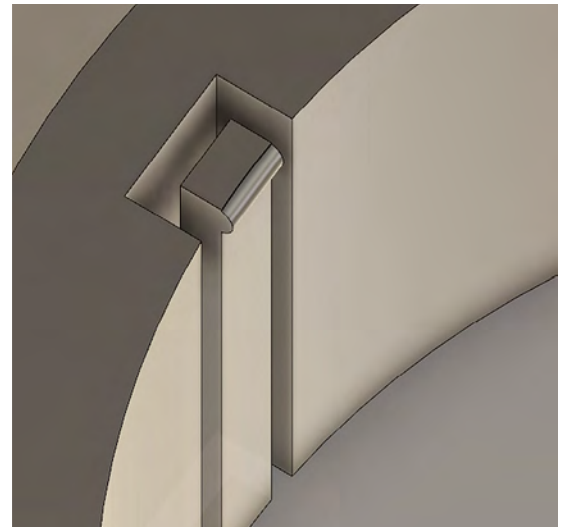
Sketch and extrude a rectangle next to one side to form a semicircular shape. Don't join this to the cylinder, but create a separate body.

Use the flat face to cut 3 mm holes through this block for the bolts. Add countersinks for the bolts, if required. Repeat this step with a 6 mm hole; when setting the parameters, deselect the block so this larger hole just cuts the outer cylinder.

Above Left 
The final inner form of the keycap

Above 
The flanges add strength to the key

Right 
You can 3D-print your clips in place to hold everything together



The body of the keyboard is made from two parts: a 3D-printed top case and a sturdy wooden base. As with the keys, this starts with a sketch and a tapered extrusion. Import the key by dragging it in from the data panel. Duplicate the key and move along so the two are side by side. Sketch a line around the outside of the project where the key meets the →

USB DEVICES

The secret to creating a keyboard controller is to use a microcontroller such as the ATmega32U4 featuring built-in USB capability. There are different classes of USB devices such as disk drives, keyboards, and sound devices. To act as a keyboard, the microcontroller shares the classes it supports with the host computer – it can then communicate using the standard drivers built into the operating system.

Below 
The support tube helps ensure smooth running of the switch



QUICK TIP

Using the drawing option, you can print a template for drilling holes and cutting the base.

bottom surface. Use the offset command to add a gap between the key and the case.

Next, extrude with a taper, using a 10° taper angle, cut to make a hole for the keys. Sketch a rectangle on the base and push in to make a hole for the ItsyBitsy. We modelled a simple version of this in the form of a block to check the spacing. Chamfer or fillet the inside edges so there is less of a span when it comes to print. Use sketch, cut, and chamfer to make cut-outs for the cables to run through. Add holes in the base so that it can be screwed down. Flip the case over and fillet the top and side edges to give a smooth appearance.

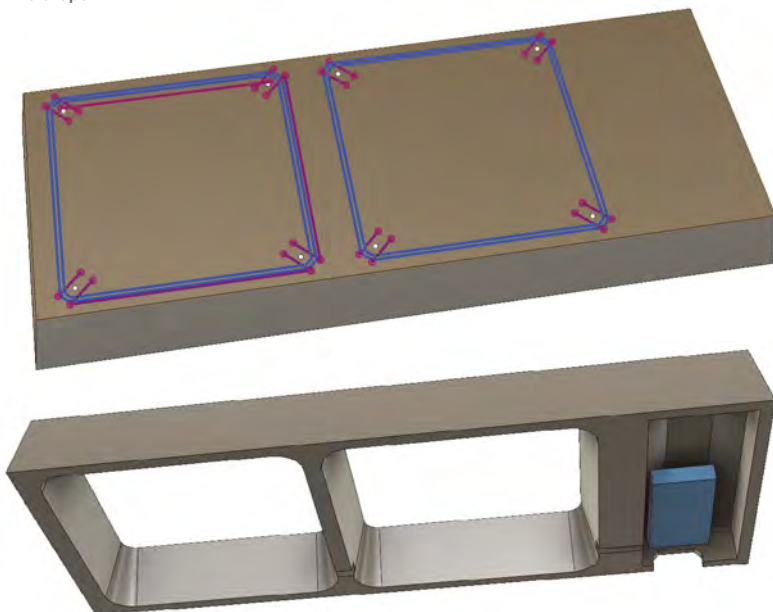
A piece of 11 mm-thick MDF or plywood is a good size to make a heavy base. Cut this out and drill the holes for the mounting screws. Countersink the holes so that the screws do not scratch your work surfaces.

SPRING TIME

Finding springs the right size is a challenge. They need to have a relatively large diameter of 35 mm and a length of 50 mm. The thickness of the wire and material will determine the force needed to compress them. We chose to make our own springs from some 2.4 mm copper-coated steel TIG welding rods. If you had stainless steel, then a thinner wire would be advised; again, if welding wire was chosen, then 1.2 mm is a good thickness for stainless. If you are buying springs, a longer spring can always be cut to length. Buying springs is also a better solution if you are getting lots of use from your spring as these are heat-treated so that they keep their shape and spring.

Below ♦
Cut-outs for the wires and USB connector

Bottom ♦
Offset adds a small gap inside or outside a shape



Now that the mechanical aspects are complete, we move to the electronics. Microswitches typically have three connections which are marked 'Common', 'NO' (which stands for Normally Open), and 'NC' (Normally Closed). Ensure you have enough length of wire to reach from the end of the case to the middle of the keys, passing through the cut-outs. Solder wire from the two common terminals to the ground pins of the ItsyBitsy. The NO pins are wired to Pin 2 for the zero key, and Pin 3 for the one key.

BITS AND BYTES

To use the Adafruit ItsyBitsy, you need to configure the Arduino IDE to support that board using the instructions at hsmag.cc/nYiYCP. The Arduino Micro does not require additional configuration.

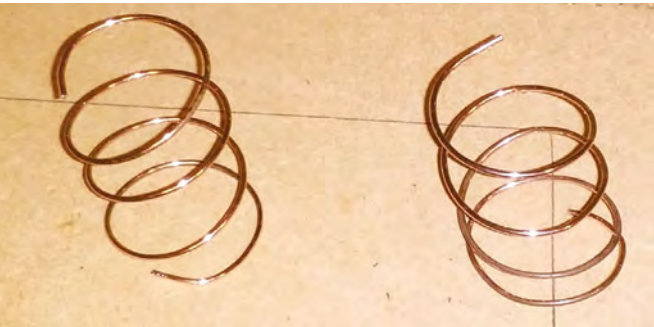
It's possible to code using low-level USB commands, but there is a keyboard library that provides all the features we need. The software uses just three variables. The first (**keyCount**) stores the number of key presses, up to eight. **keyPressed** is the last key pressed, either 0 or 1; this variable is marked as 'volatile', meaning that it might be changed by code running in an interrupt. The last (**keyCharacter**) is the character represented by the binary number you have typed.

```
#include <Keyboard.h>
```

```
int keyCount;  
volatile char keyPressed;  
char keyCharacter;
```

The pins used for the switches are configured so they have a high state when nothing is connected. This is done with an internal 'pull up' and saves us having to use an external resistor to provide this signal. The code tells the USB port to act as a keyboard with **Keyboard.begin()**. Finally, we attach our two pins to functions we wish to run when the keys are pressed. The **FALLING** refers to the falling edge signal, i.e. the transition from high to low when the button is pressed.

```
void setup()  
{  
  pinMode(2, INPUT_PULLUP);  
  pinMode(3, INPUT_PULLUP);  
  Keyboard.begin();  
  attachInterrupt(digitalPinToInterrupt(2), key0, FALLING);  
  attachInterrupt(digitalPinToInterrupt(3), key1, FALLING);  
}
```

Above ♦
Springs from welding rod

Our two interrupt handlers are almost identical and store the last key that was pressed.

```
void key1()
{
  keyPressed = '1';
}
void key0()
{
  keyPressed = '0';
}
```

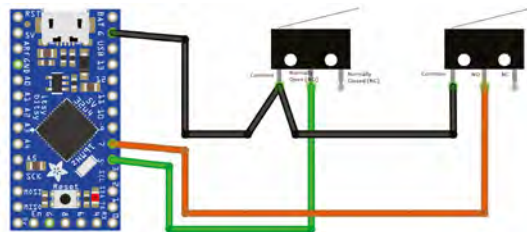
The main loop is responsible for writing the keys to the USB. Each time it sees a key press, it adds the bit to the **keyCharacter** variable, shifting the existing bits one position to the left to make room. Once we have eight bits, the keyboard will delete those characters and replace them with the ASCII character that they represent.

```
void loop()
{
  if (keyPressed == '0') {
```

```
    Keyboard.print('0');
    keyCount += 1;
    keyCharacter = keyCharacter << 1;
  }
  if (keyPressed == '1') {
    Keyboard.print('1');
    keyCount += 1;
    keyCharacter = keyCharacter << 1;
    keyCharacter = keyCharacter | 1;
  }
  if (keyCount == 8) {
    for(int i=0;i<8;i++){
      Keyboard.write(KEY_BACKSPACE);
    }
    Keyboard.print(keyCharacter);
    keyCount = 0;
    keyCharacter = 0;
  }
```

INTERRUPTS

Usually, a microcontroller will just keep running the same code repeatedly. For Arduino projects, this is the code in the **loop()** function. This is fine until we want to respond to an external input from a user or another piece of hardware like a serial port. Interrupts are a mechanism that allows the microcontroller to jump away from that loop in response to this external event. The 'Interrupt Service Routine' or 'handler' is a short piece of code that responds to this event and typically stores some information to be used back in the main loop. Whilst handling an interrupt, the microcontroller cannot do other activities, so these routines need to run very quickly. Accessing the serial port or USB is often not possible whilst in these routines and there may be other limitations specific to your hardware.



Left ♦
Wire two interrupt pins to the microswitches

Below ♦
How to type digits and letters using binary code

```
}
keyPressed = ' ';
delay(10);
}
```

By combining simple shapes and operations, it is possible to create complex designs to be printed. By thinking about how they will be printed, strong and light parts can be printed without the need for supports.

That's all there is to it. Assemble everything and program your board, and you'll have your own binary keyboard. Now you just need to learn binary. □

0	00110000	a	01100001	n	01101110
1	00110001	b	01100010	o	01101111
2	00110010	c	01100011	p	01110000
3	00110011	d	01100100	q	01110001
4	00110100	e	01100101	r	01110010
5	00110101	f	01100110	s	01110011
6	00110110	g	01100111	t	01110100
7	00110111	h	01101000	u	01110101
8	00111000	i	01101001	v	01110110
9	00111001	j	01101010	w	01110111
		k	01101011	x	01111000
		l	01101100	y	01111001
		m	01101101	z	01111010

Make a Slim Jim antenna

Let's make a simple, and affordable, starter antenna for a SatNOGS ground station



Jo Hinchliffe

@concreted0g

Jo Hinchliffe is a contributor to the Libre Space Foundation, 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, milling machines, and CNC kit.

Right

The Slim Jim antenna, seen here coiled up with the rest of a simple SatNOGS ground station kit

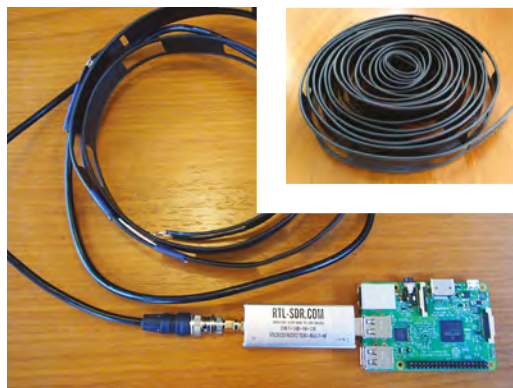
YOU'LL NEED

- ♦ **2 metres of 300 Ω ladder feeder cable** (available quite cheaply in short lengths online)
- ♦ **50 Ω RG58 BNC cable**
- ♦ **Cutters**
- ♦ **Soldering iron**
- ♦ **Electrical tape**

Earlier this issue, we walked through setting up a SatNOGS ground station and, whilst many people buy an antenna to complement their station, it's possible to begin hunting satellites by making a simple 'Slim Jim'-type antenna for less than £15.

There are numerous tutorials and calculators online for these antennas, and they are all worth checking out. The dimensions for the antenna we will build are based on the figures from a calculator at hsmag.cc/YnLbhd, which I found via an excellent Essex Ham tutorial here: hsmag.cc/muniTO.

We are going to aim to make an antenna that is sized and tuned so that it should be optimised



TUNE IN

There are many frequency bands that satellites operate on, and we have opted to build an antenna that suits part of the VHF (very high frequency) range. Another common band is UHF, which is even higher (ultra high frequency). Due to the actual physical length of the radio waves at these frequencies, they often get referred to as the two-metre band or the 70 cm band respectively. It's worth knowing these terms, as they get used a lot!

to receive signals around 145.800MHz. This is a common area of frequency for many, many satellites, and also the International Space Station periodically transmits different signals on this frequency. So, setting the frequency to 145.800 in the calculator on the M0UKD website above, we are given the collection of dimensions we need and the layout of the Slim Jim antenna. The first task is to cut a length of the ladder feeder wire so that it is a little over the total length required. The length required is 150.2 cm, so cut a length to around 156 cm.

We then strip and form one end, so that the wires meet, and solder them together, as shown in **Figure 1**. Then, we measure to try to bring the antenna to its total length from the joint we have just made, allowing a short section to be left over to form the loop at the other end. Cut and strip, and solder the other end in the same way as the



Above

The m0ukd.com website has a calculator which will give the dimensions of a Slim Jim-type antenna for any target frequency you submit. Go to this calculator (hsmag.cc/YnLbhd) and input 145.800 to get all the desired information

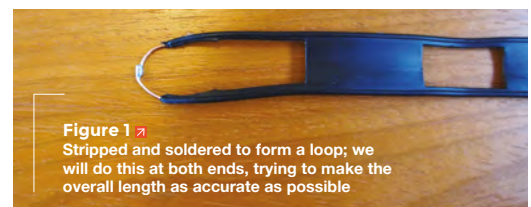


Figure 1 Stripped and soldered to form a loop; we will do this at both ends, trying to make the overall length as accurate as possible

first. Again, we are aiming for the overall length from the end of each soldered loop to be our target dimension of 150.2cm, so try to be as accurate as possible!

Next, we need to cut a small gap in one side of the loop only. We measure up the distance given from the calculator, which was in this case 49.4cm, from one end and then carefully cut out a 2.1cm notch in one side of the antenna only! On our ladder

// We decided to add some support, keenly engineered from a coffee stirrer and some tape

wire, this notch happened to land on an unsupported section of the wire, so we decided to add some support, keenly engineered from a coffee stirrer and some tape!

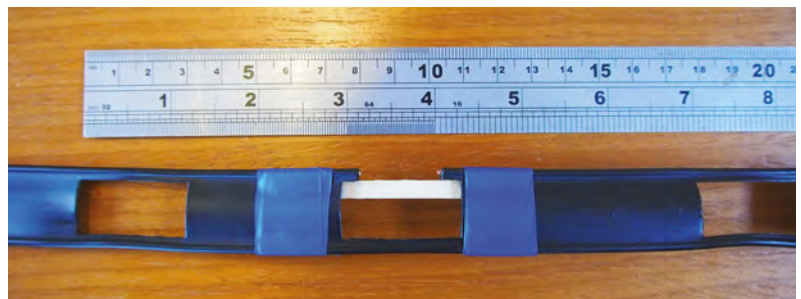
GETTING IT CONNECTED

The next task is to clear/strip a small section for RG58 coaxial cable to be connected. If you have experience in attaching BNC connectors to coaxial cables, you might opt to make up your own cable here. But, in order to keep it simple, it is as cheap to buy an RG58 coaxial cable with a BNC connector on each end and cut it in half. Either way, we need to carefully clear a 3–5mm section on both sides of the ladder feeder wire, so that we have just enough space to solder the RG58 cable on. This should be done so that the RG58 connections are as close to 4.9cm from the end of the antenna as possible (between the gap we cut and its closest end). Solder the shield of the coaxial cable to the side of the antenna that has the gap we cut into, and solder the central core of the RG58 cable to the other side. It's helpful to perhaps tape the RG58 cable to the ladder cable prior to soldering, to help keep things steady, as shown in **Figure 2**.

These antennas are designed to be hung off things and so should be positioned vertically, and many people have made them more permanent and weatherproof by installing them into plastic tubes, etc. They are also very portable, in that they can be rolled up and chunked in a bag for mobile use. A rolled-up Slim Jim and a key-chain carabiner can make a very quick-to-deploy ad hoc antenna. We can see in **Figure 3** that, even though this antenna has been hung inside, it can still receive a signal from space! If you click through to the Audio tab at **network**.

READY TO SEND

If we were wanting to use this antenna to transmit, it would be worthwhile trying to check and assess its standing wave ratio, and perhaps adjust it to make it extremely efficient and accurate. People also fit chokes and filters, and perhaps also a low noise amplifier (LNA) to further enhance the performance. However, for our 'receive only' duties, it is less important, and we can hook it up to our SatNOGS ground station and give it a go!



Above ♦ The gap, cut and reinforced

Figure 2 ♦ Using some tape, to physically attach the RG58 cable, helps with soldering this step

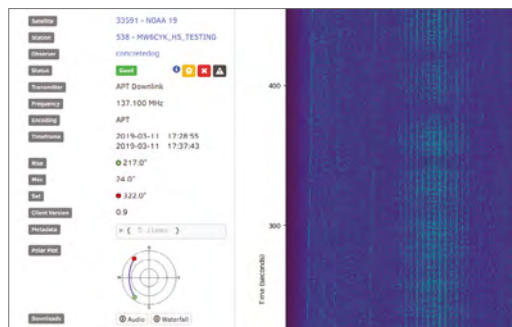
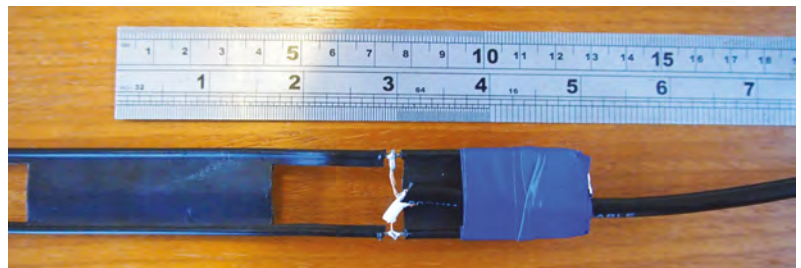


Figure 3 ♦ Here, we can see a successful observation using the simple SatNOGS test station we built earlier connected to the Slim Jim antenna

satnogs.org/observations/53010 and scroll to around three minutes, you will hear the Morse code telemetry beacon of a small satellite called XW-2F CAS_3F, captured on this antenna with the SatNOGS station built as part of the space cover feature in this issue. □

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



SUBSCRIBE FROM JUST £5

- **FREE!** 3 issues for the price of one
- **FREE!** Delivery to your door
- **NO OBLIGATION!** Leave any time



FREE PI 3 A+*
With your 12-month subscription to the print magazine
magpi.cc/12months

* While stocks last

**PLUS! FREE
CASE, THREE
COVERS &
CABLES**

Buy online: store.rpipress.cc

HackSpace
TECHNOLOGY IN YOUR HANDS

FIELD TEST

HACK | MAKE | BUILD | CREATE

Hacker gear poked, prodded, taken apart, and investigated

PG
114

DIRECT FROM SHENZHEN: BLUETOOTH RECEIVER

One board to grab and
amplify wireless audio

PG
122



CAN I HACK A WIFI RELAY

Remote control any
mains-powered device

PG
116



BEST OF BREED

Make your projects move –
our look at the best wheeled
robot kits around

REVIEWS

124 **BALENA FIN**
Industrial IoT



126 **PYPORTAL**
Connected display

129 **CRAFTIVISM**
Change the world

DIRECT FROM
SHENZHEN

Bluetooth amplifier

Receive and amplify music with a single circuit board

By Ben Everard

@ben_everard

This author once converted an old 1940s radio into a Bluetooth speaker. It looks marvellous and sounds good, but it needed separate Bluetooth and amplifier modules wired together. Not the most onerous task,

but when we came across all-in-one receiver and amplifier modules, we decided to find out if we could use them as a ready-made device for upcycling old speakers into wireless audio receivers. In principle, these modules should let you add wireless audio by just connecting up the wires. Let's see if they really work.

We bought a 'TDA7492P 2*25W Wireless Bluetooth 4.0 Audio Power Amplifier Board Digital Amplifier Module with AUX Diy Electronic PCB Board', from diymore Official Store on AliExpress,

for £5.40, plus £2.15 delivery to the UK. This board integrates a Bluetooth receiver and 2x25 watt stereo amplifier. The only additional hardware needed is an 8–24V power supply and speakers. There's a 3.5 mm audio output if you want to use it with an external amplifier.

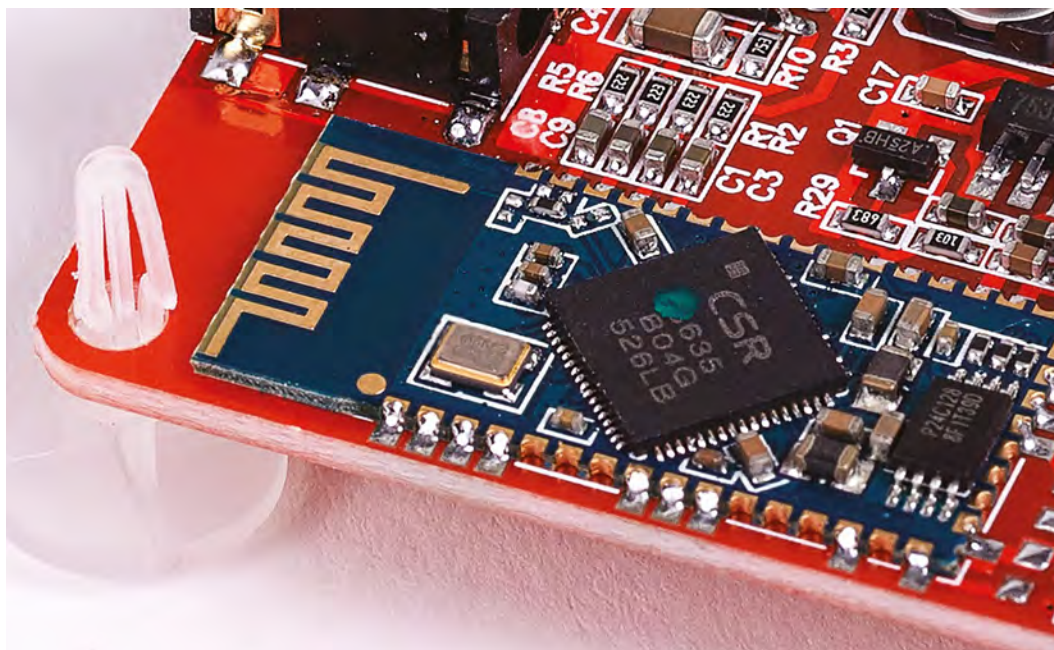
From a functional point of view, this module worked really well. The Bluetooth came online as soon as the unit was powered on, and paired easily with our phone. As soon as we piped music through it, it started playing. There are five control buttons (play, back, next, volume up, and volume down), and again, these worked well. These buttons are small, and lower down than some of the higher components, so they're not particularly easy to operate once the module is in a case. We've seen laser-cut cases with extenders for these buttons, or alternatively, it would be fairly easy to solder onto this board to have additional controls that could be mounted elsewhere.

TROUBLING FREQUENCIES

The first clue that it might not be the highest-quality product is the alignment of the components – the larger through-hole bits, like the screw terminals, were all wonky. Not a huge problem, but a sign that care hasn't been taken in the manufacturing. There is though, a much bigger issue: the sound quality was atrocious. When working with cheap audio hardware in the past, we've had problems with hums, clicks, and other bits of audio errata introduced, but we didn't have that problem with this receiver; instead, the frequency response was off completely. Low 'bass' sounds had almost completely disappeared from the music leaving a very tinny, high-pitched sound. Something has

Below ♦
Wonky components
are the least of the
problems on this
receiver-amplifier





Left ♦
The Qualcomm CSR8635 Bluetooth module should be capable of better audio than this

gone very wrong with this module to end up like this. It's so far off that it's not easily explainable by the components. Bluetooth comes via a CSR8635 Qualcomm module, and amplification is done with a TDA7492P. Both of these should be capable of producing reasonable quality audio, so either a/ one of these is a low-quality fake part, or b/ there's some gremlin in the way they're set up. It might be something as simple as the wrong value capacitor used somewhere along the line, and if it were, it would be a real tragedy that the module was rendered just another bit of e-waste by something so simple. Perhaps the manufacturers would benefit from reading Bunnie's advice on hardware testing in issue 17 (hsmag.cc/issue17).

AN UNSURE FUTURE

The sound problems are in both the speaker output and the audio-out jack. In principle, it might be possible to recover the bass using an equaliser, but it'll always be very distorted, and you're never going to get results as good as if the bass wasn't filtered out in the first place. Also, if you need extra hardware to correct the sound, we might as well use separate modules in the first place.

Unfortunately, it's hard to see any useful future for this module. Even traditionally low-fidelity audio uses, such as spoken word, would be distorted to the point that it's difficult to listen. There's certainly no need to turn the thing up to 25 watts, and we

“ Despite costing only £5.40, this still feels overpriced, but then it probably would at any price ”

wouldn't want to be anywhere near the full 25 watts of ear-splitting, high-pitched whine that would result if you did. Perhaps the one thing slightly redeeming the module is that the large capacitors and inductors add a certain aesthetic charm and could be used if you're looking for something that adds an 'electronics-y' look, without needing it to do any actual electronics.

Despite costing only £5.40, this still feels overpriced, but then it probably would at any price. If you ever find yourself in need of 25 watts of remotely controlled, high-pitched audio, then this is the module for you. Otherwise, keep looking. We've had much better success using a module that is just a Bluetooth receiver (such as the KRC-86B, or even a Raspberry Pi) which can then be paired with an amplifier of your choice – whether integrated into an existing sound system, or one of the many available from a range of sources. The end result will probably cost a little more, but you should be able to get a sound quality that's actually worth listening to. □

DIRECT FROM SHENZHEN



Best of Bots!

A few of my favourite wheeled robotic friends

By **Marc de Vinck**

 @devinck

Ah, robots. What engineer, maker, or DIYer doesn't love a good robot?

This author has been accused of having a problem when it comes to his regular acquisition of bots.

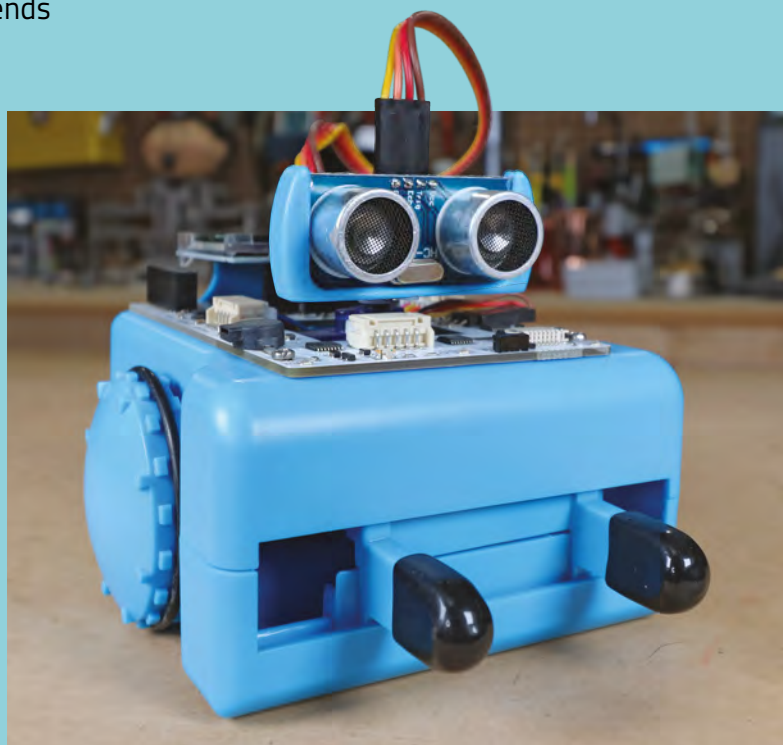
At least, that's what his wife says.

He doesn't see it as a problem – it's educational, and most of all, it's fun. Here at HackSpace, we're especially supportive of robotics programmes for education, as it's one of the best ways to get kids engaged in experiential learning, and the application of STEM principles.

The FIRST Robotics Competition, with over 3500 teams worldwide, is an interesting example of a programme that encourages kids to engineer amazing robots that battle it out in arenas around the globe. We've been to a few FIRST Robotics events, and we're always incredibly impressed.

But what if you, or your child, isn't in high school, or you don't have a FIRST Robotics team in your town? (start one!) That's where this roundup of bots comes into play. Anyone can enjoy learning, or competing, with robots. Prices of robots can range from just a few dollars to thousands, or more. In this Best of Breed, we're going to take a look at a few moderately priced robots, and only ones with wheels. In future articles, we'll take a look at other types of robotic platforms.

The world of bots is so diverse that we can't look at all the varieties – such as walking, flying, and slithering – in one go (yes, there are snakebots). The best place to start your adventure is to get rolling



with a wheeled bot. These are typically easy to use and really fun to race, or drive, around your home.

Narrowing the category down to just a handful of contenders was extremely difficult. It's a huge market! There are dozens, if not hundreds, of different wheeled robot kits on the market. We're featuring the ones we're more familiar with. If you are just getting started with robotics, we suggest you buy one from a named brand company, since many off-brand kits offer little support or, to be honest, don't even really work. Trust us, we've owned far too many bad bots.

Above ♦
Sparki has been living on our workbench for many years

mBot vs Sparki

Two educational robotics platforms go head-to-head

MBOT ♦ \$70 | makeblock.com

SPARKI ♦ \$149 | arcbotics.com

The mBot, from Makeblock, is a STEAM education robot aimed at beginner robot enthusiasts. We've had a chance to play with different Makeblock robots, and would say that they are fun for all ages. Based on the ATmega328, anyone familiar with Arduino will have fun programming this bot. The instructions are geared towards kids, but it's a powerful bot with lots of potential.

One of the benefits of the Makeblock ecosystem is the amount of add-on kits available, ranging from a dancing cat, yes cat, to an interactive light and sound kit, to our favourite one, the six-legged robot. The entire Makeblock ecosystem is based on high-quality design, with great projects. If you have a child that is interested in robotics, this kit should be on your list. You'll have as much fun as them building, programming, and hacking this bot. It's even more fun once you discover all the additional parts available. □



Above □
You can expand Sparki in lots of ways



Below □
Not many bots at this price have grippers and a screen

Sparki, from ArcBotics, is a complete educational platform that has a lot of interesting features, and an amazing number of examples and projects online. You can learn to

program Sparki's LCD display, grippers for grabbing objects, line-following sensors, remote control, speakers, light sensors, distance sensor, edge detection sensors, and more. That's a lot of features packed into this diminutive little bot. Best of all, it's based on the Arduino platform, so there are virtually limitless resources online for learning more.

Sparki comes fully preassembled, which is great for some, but others might have wished they could have built it themselves. The kit includes a remote control, line-following poster, USB cable, Bluetooth module, and more. It's a nicely thought-out robot kit, so you won't worry about missing a critical accessory or component. Just open the box and get programming. □

VERDICT

mBot

An easy-to-use bot with a good range of add-ons

9/10

Sparki

Lots of interactivity and motion with this little bot

9/10

Pololu 3pi Robot

An a-maze-ing little powerhouse

POLOLU ♦ \$100 | pololu.com

Right ♦
A fairly high-
performance and
speedy little bot



VERDICT

A fast little bot
that is good at
maze-solving

8/10

The 3pi Robot, from Pololu, was designed to excel in line-following and maze-solving competitions. We first played around with one many years ago, but that's not to say it's outdated technology.

It's based on the incredibly popular ATmega328 microcontroller, found in the Arduino Uno.

The 3pi is small in size, measuring in at only 3.7" in diameter, and only weighs 2.9oz, without batteries. With that size and weight ratio, along with some clever power management, the bot can run at 3ft per second or more, while making precision movements.

You can program the 3pi in several different ways, including Atmel Studio or the popular Arduino IDE, among others. And there are extensive libraries that make it easy to get up and running fast. Pololu also includes a number of sample programs for performing complex behaviours, like fast-line following and maze-solving applications. □

Arduino Engineering Kit

Learn some engineering fundamentals

ARDUINO ♦ \$299 | store.arduino.cc

The author wasn't familiar with this kit prior to doing some research for this Best of Breed article, and he's glad he found it. The Arduino Engineering Kit was designed for university students, providing a state-of-the-art, hands-on, robot building platform. The kit includes everything you need to build three different Arduino-based projects that help teach students the fundamentals of engineering and mechatronics principles.

Couple the included Arduino MKR1000 board, Motor Shield, and IMU Shield with MATLAB and Simulink to build and simulate complex robots. Of



Left ♦
Educational,
but also fun!

the three different models you learn to build, by far our favourite has to be the balancing motorcycle. It's a truly unique DIY kit that looks like a lot of fun to build. □

VERDICT

A unique
collection of
projects suited
to any STEM
programme

8/10

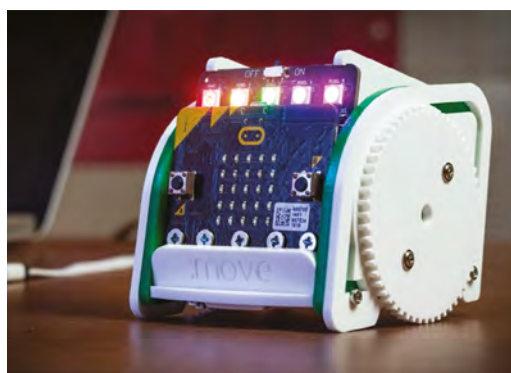
:MOVE mini buggy kit

A whole lot of fun in a mini bot

KITRONIK ♦ \$35 | kitronik.co.uk

The Kitronik :MOVE mini is a fun little robot, built exclusively for use with the micro:bit. The bot is powered by two continuous rotation servo motors. You can control the direction and speed easily via code in the Microsoft MakeCode editor. This lovable little bot also includes five RGB LEDs, and can be controlled wirelessly with

" This lovable little bot can be controlled wirelessly with your smartphone



Left ♦ There are plenty of add-ons available for this little bot

your smartphone. It's a great little kit – just remember to pick up a micro:bit, since it's sold separately.

There are plenty of examples and documentation to get you up and running fast. Once you get through learning how to program the :MOVE mini, you might also want to pick up one of the expansion packs, like the bulldozer kit, trailer, or bumper kit. These add-ons, along with the core kit and micro:bit, are all great value and lots of fun. □

VERDICT

The micro:bit's Bluetooth gives this bot wireless control

8/10

SumoBot Robot Competition Kit

Wrestling robots in a ring

PARALLAX ♦ \$249 | parallax.com

Right ♦ Who doesn't like a little robo-battle?

VERDICT

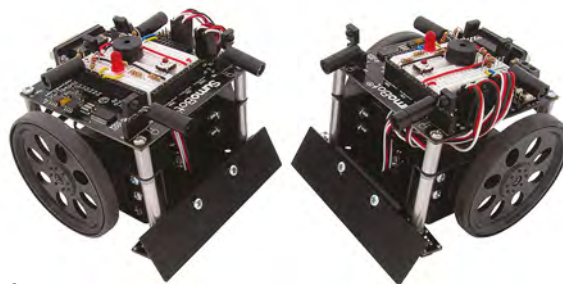
Two bots are always better than one

10/10

W

ith the SumoBot Robot Competition Kit, from Parallax, you will build and program two high-quality SumoBot robots, designed to wrestle in a mini-sumo

competition ring. The kit includes everything you will need to have some amazing robotic battles. Once you've had fun with some basic battles, you can continue to learn about applied robotics, including friction analysis, memory optimisation, self-calibrating sensors, and sensor-based navigation.



One feature we found particularly interesting is the ability to store the metrics in EEPROM. This allows you to add data-logging features, which can be displayed after a battle to gain insight into the inner workings of your robot.

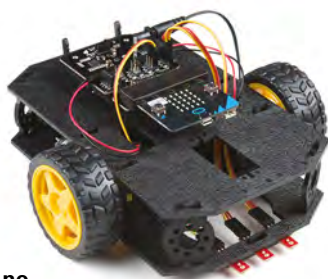
One bot is fun, but having two is definitely double fun! Parallax has a long history of making quality parts and components, including robots, so you can be sure that this kit is top-notch. □

SparkFun micro:bot Kit

Quick, simple robots

SPARKFUN ♦ \$62.95 | sparkfun.com

The growing popularity of the micro:bit has spawned a few interesting kits, and this kit is no exception. The SparkFun micro:bot kit leverages the micro:bit simplicity with the SparkFun moto:bit carrier board, along with a custom robot chassis. You'll be able to create simple robots quickly, without spending hours learning how to build a basic robotics platform. Especially interesting is that this kit doesn't require any soldering, making it really easy for any new robotics enthusiast to get started. □



Below ♦
A simple no-solder kit for building your micro:bit bot

VERDICT

A very welcome accessory to the micro:bit ecosystem

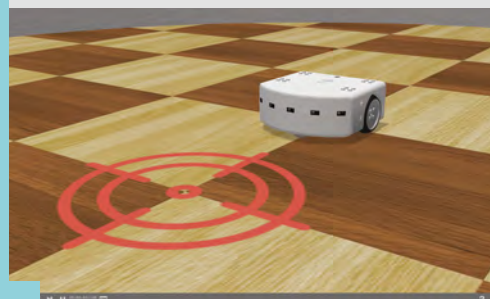
9/10

WEBOTS

You might not know that you can start learning about robotics, even without a robot.

Webots is an open-source robot simulator that was co-developed by the Swiss Federal Institute of Technology and Cyberbotics. Originally developed in 1996, which is a lifetime ago in robot years, it has been continually updated to include complex robotics movements, physics, autonomous car research, and many other validation simulations. You can even use it to develop your own flight simulator, connecting to common hardware and VR equipment.

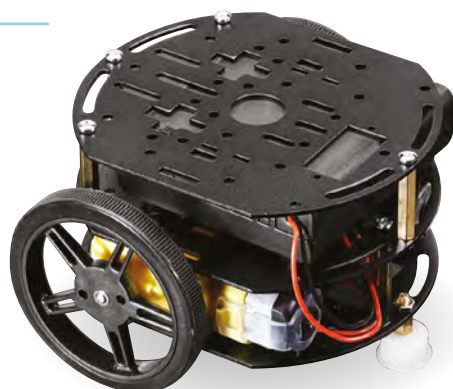
cyberbotics.com



Mini 3-Layer Round Robot Chassis Kit

Bot-building basics

ADAFRUIT ♦ \$25 | adafruit.com



Sometimes you just can't find a complete kit that will make your robot dreams come true, and that's where the Mini 3-Layer Round Robot Chassis Kit, from Adafruit, comes in. The kit includes all the core

components to build a two-wheel-drive robot. Just add your favourite microcontroller, like Arduino, or a single-board computer, like the popular Raspberry Pi, your favourite motor controller, and some power.

The plates are anodised aluminium, so there are no worries about stability, or wearing out any time soon. There are also three of the plates, giving you ample room for the accessories your robot needs. If you're looking for something a little smaller, Adafruit also offers a two-layer kit for a little less money. Either choice will be a good start for building your bot. □

Left ♦
A perfect platform for building your next bot

VERDICT

A great platform for building your own bot

9/10

Wireframe

Join us as we lift the lid
on video games



Visit wfmag.cc to learn more

Can I Hack It?

Sonoff Basic Smart Switch

Can we hack a Sonoff Basic Smart Switch to do more?



Les Pounder

@biglesp

Les Pounder loves taking things to pieces and seeing how they work. He teaches others how to be makers and tinkerers at events across the UK. He blogs at bigl.es

Smart homes and home automation are some of the biggest communities in the maker community. Could this cheap, simple smart switch from Sonoff enable anyone to make their home smart? Well, the only way to find out is to take it apart and take a look.

Made from a rigid workable plastic, the case for the Sonoff Basic can be easily worked with hand tools. But great care should be taken, as the case has been designed to reduce the chance of electrocution. It features cable strain relief that will grip the mains cable for the input and output terminals. These grippers should remain unaltered, as they are there for your safety.

INTERNAL SURGERY

The Sonoff Basic is really just an ESP8266 with a connected relay. The ESP8266 runs on 5V DC which is transformed from AC to DC and regulated via the board. The mainboard is split into two sections:

the low-voltage section, and the high. On the low-voltage section, we see the ESP8266 chip, antenna, and a series of unpopulated GPIO pins – more on that later. The high-voltage side of the board has two large tracks of solder. These tracks connect the live and neutral input and output together. Neutral is connected at all times, but the live track goes via a 250V relay which has been set to normally open (NO), and that means the default behaviour of the relay is to break the live connection. The high- and low-voltage sections are separated by gaps in the board, giving the two sides enough space to prevent issues. To input and output the AC voltage, we see two screw terminals. These terminals secure the wires in place, and while they are adequate for the task, we really need to be careful and ensure that our wires are carefully twisted and inserted cleanly into the correct terminal. Even a single strand of wire bridging the two terminals could prove dangerous, so check and double-check before securing!

YOU'LL NEED

Sonoff Basic
10 A Smart WiFi
Wireless Light
Switch

COST

£8.90

WHERE

hsmag.cc/AAOWYL

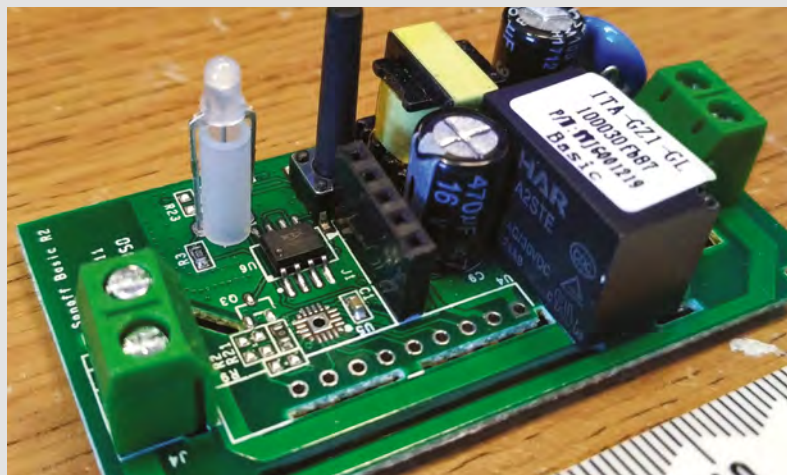
Right

Inside a boring white plastic case there is enough power to automate your home, one device at a time!



WARNING

The Sonoff Basic is powered by 90V–250V AC electricity. Do not attempt to open or hack the device while it is connected to AC mains power. There is a serious risk of electrocution or death. Make sure that any modifications you do to the case leave it in a safe condition, where users won't come into contact with electronics. If you're unsure about working with mains voltages, only do so under the supervision of someone who is competent in this.



HACKABILITY

We mentioned that the Sonoff Basic is just an ESP8266, an extremely cheap microcontroller that brings down the cost of creating Internet of Things (IoT) projects. The standard Sonoff software uses an app on your smartphone to enable control of the switch via the app. The app sends a message to the ESP8266 which triggers the GPIO pin responsible for the relay to trigger. This then connects the AC voltage input to the device connected to the AC output, turning said device on/off.

The standard software on the ESP8266 and the app is rather basic, but luckily for us, we can fix that! On the board, we can see five unpopulated GPIO pins between the push-button and a capacitor. These pins offer a direct connection to the ESP8266 and, using a CP2102 CH340G USB to serial adapter, we can load custom firmware onto the chip. But in order to do that, we need to solder a female header connection to those pins. It would be prudent to only use a female header, rather than a male header, otherwise we are adding unshielded metal pins into a high-voltage device. It might be OK 99% of the time, but never take the chance with high voltage. The pinout from the pin nearest the push-button is 3V, Rx, Tx, GND, GPIO14.

When flashing custom firmware onto the Sonoff Basic, DO NOT power the board using AC power – only use 3V power from the CP2102. Anything higher will damage the ESP8266, and could potentially kill you! Connect 3V and GND from the USB to serial device to the ESP8266 pins. Then, connect from the USB to serial Tx to Rx on the ESP8266, and then Rx from the USB to Tx on the ESP8266.

For this article, we chose to use Tasmota firmware, as it offered the most flexibility. We had to jump through a few hoops to make it work – chiefly, learn how to flash the firmware using the Arduino IDE – but when it did work, it was glorious! During the flashing

process, we can specify the SSID and password for our WiFi, or set the Sonoff to boot in AP mode and enable us to connect and configure the device remotely. Nice! Tasmota also works with Home Assistant (hsmag.cc/wwwAkW), which can be run from a Raspberry Pi. This means we can control groups of Sonoff Basics. For example, we can create a group for 'Lounge Lamps', and turn on every lamp at the push of a button. We can even use remote sensors to control our home lighting and heating.

For under £10, this is a great, relatively easy-to-hack smart switch. The ease of use enables us to flash custom firmware that elevates the product to a much more interesting and usable device. Via Tasmota we were able to control the switch using its own web interface and via MQTT, a lightweight message system that works across many devices and languages. You do need to be careful, as this is mains voltage-powered, and always remember to only flash the device when it is connected to 3V power only, not AC! But as long as you are careful, this device can truly automate your home for very little money. □

Above ♦
The high- and low-voltage sections of the board are clearly separated and provide adequate distance. We can also add our own header to flash the ESP8266 chip

LET THERE BE LIGHT

In this article, we used the Tasmota firmware, hsmag.cc/eSKeUY, as it offered the greatest amount of features versus ease of use. But there is other firmware available. For example, the ESPurna firmware, hsmag.cc/TiGJaJ, offers a similar feature set to Tasmota. It also offers integration with Google Assistant, IFTTT, and Alexa, so we can easily link up our smart assistants to control our home by voice alone. We also have access to ThingSpeak, an API that enables us to link real-world sensors to the cloud, collecting data, and then using it to power projects. For home users, we can use ThingSpeak to control our home lights, Christmas decorations, and heating based on keywords and sensor data. All of this is made possible by custom firmware for this £9 product.

But it doesn't stop there! If you just need a simple on/off switch, a timed switch via a PIR sensor, then we can use the GPIO of the ESP8266 and a little MicroPython code to write our own code to control the Sonoff Basic. All we need to do is use the `esptool.py` command-line tool and flash the latest MicroPython for the ESP8266. Then write the code, and you have your own bespoke smart switch.

balenaFin

Taking the Raspberry Pi to the enterprise

BALENA ♦ From \$154.80 | [Balena.io/fin](https://balena.io/fin)

By Ben Everard

🐦 @ben_everard

You may not have heard of Docker, unless you happen to be involved in Linux DevOps, in which case you're probably sick of hearing about Docker. It is, in essence, a way of controlling what's running on machines remotely, by managing

things called containers. To avoid going down a geek-rabbit-hole, we'll leave the technical description there. Docker's been pioneered by people who run large numbers of services, as this technology (along with a few similar alternatives) has made it easy to keep software running and up-to-date on a large number of machines.

It turns out that the basic problems faced by people managing data centres are very similar to the problems faced by people managing Internet of Things networks. If you've got tens or hundreds of machines scattered about the place, how do you manage them? What do you do if you update your software? Or want to run a new bit of software on each machine?

Docker is a bit of software, not a bit of hardware, but it does bring in a few hardware requirements. Firstly, you need to run Linux – Docker requires some features of the Linux kernel in order to work, and while there are some hacks that let it work on Windows and macOS, these aren't things to rely on in production. There's no real equivalent for it on microcontrollers.

PREPARE TO DOCK

So, if Docker is a bit of software, why have we spent so long talking about it in a review of a bit of hardware? Well, balena hosts infrastructure that's designed specifically to run

Docker on embedded Linux boards, and while the firm's software supports a wide range of hardware, the Fin is its board designed to bring the ultimate experience to the embedded Docker world.

At the heart of the Fin is a Raspberry Pi Compute Module. This brings the Pi 3B+ system, along with the software and hardware support (the Fin breaks out the GPIO pins so you can fit HATs as usual) that come with it. On top of this, it brings in robust eMMC storage (8–64GB), a



Above ♦
The Fin keeps the GPIO structure of other Raspberry Pi boards, letting you add on all the usual hardware

**Above** ♦

The Compute Module slots in a connector on the back of the Fin

power supply that can take a wider range of voltages (5–24 V), a mini-PCIe and SIM card slot for cellular connectivity, a real-time clock, and a microcontroller that can turn the Compute Module on and off, and do its own processing for low-power applications.

This all adds up to a very capable embedded board that – thanks to what it inherits from the Raspberry Pi – has a great community and ecosystem around it. All this comes at a price – from \$154.80 for the 8GB version to \$238.80 for the 64GB version. In both cases, you'll need to add on \$35 if you need the Compute Module as well. While this is an order of magnitude more than hobbyist boards, the Fin is designed, specced, and built to be industrial level.



**A very capable
embedded board that has
a great community and
ecosystem around it**



The balenaCloud web interface provides you with a place to manage all your boards using balena, (whether they're Fin or other Linux boards, such as Raspberry Pi). Your first ten devices are free, but beyond that, you'll need a monthly account which starts at \$99 + VAT for 20 devices. There's also openBalena that's free to use, but limited to command-line control (rather than the web-based interface of the balenaCloud).

ENDLESS UPGRADES

While the system does let you use the vast ecosystem of Raspberry Pi software, you will have to be familiar with Docker to wrap it up for this hardware. Docker isn't too complex to use –

especially by system administration standards. If you're running, or planning to run, a large fleet of embedded devices, then learning Docker is a small price to pay to help your software install, run, and update smoothly across a wide range of devices. If you're only ever planning on running one or two, it's an unnecessary extra hurdle.

The balenaFin is unashamedly an industrial-level product – the marketing bumpf calls it 'a board for fleet owners'. The hardware is robust, and there are commercial plans for the software with differing levels of support with guaranteed response-time service level agreements – exactly what you want if your business relies on your IoT network running.

This comes with a price tag that puts it out of reach of most hackers and makers – for most hobbyists, the cost is hard to justify over, say, a Raspberry Pi 3B+ (with perhaps an additional microcontroller for real-time hardware control like the coprocessor does on the Fin). That's OK: no one board will work for everyone, and the needs of hobbyists and professionals are quite different. What the Fin does mean is that, if you ever want to take your hobby projects and expand them into the professional world, there's an easy path through from the Pi in your workshop to enterprise-grade IoT.

There's no other Linux board we know of that manages to bridge the two worlds of hobbyist and professional in quite the same way: the hobbyist world brings in hardware and expansion modules that are easy to use, and the professional world brings in a design and infrastructure that's designed to be easy to run and maintain. □

Below ♦

There are two camera connectors for watching multiple things or stereoscopic vision

VERDICT

Brings the power of the Raspberry Pi ecosystem to a professional IoT setup.

8/10



PyPortal

A one-piece internet counter for almost any internet data

ADAFRUIT ♦ From \$54.95 | adafruit.com

By Ben Everard

🐦 @ben_everard

The PyPortal combines a 320×240 colour TFT display with a Cortex-M4 processor and an ESP32 WiFi module to create a web-based display programmable in CircuitPython or Arduino.

The most basic use of the PyPortal is as an information display. The CircuitPython library includes the ability to do this almost automatically. Feed in your WiFi credentials, the URL of the data (in JSON format), and the bit of

data you want to extract. Pair this with a background image and the location on the screen you want the text to appear, and you've got your Internet of Things display. Adafruit has guides using this basic formula to display stats for social media (such as YouTube subscriber counts or Reddit viewership), environmental data such as air quality, and text via a quote of the day.

Of course, you don't have to use the PyPortal like this. It's completely customisable – and compatible with the displayio CircuitPython library, which allows you to easily place text and graphics at different points on the screen. At the moment, performance isn't great – of the order of two or three frames per second, so animations aren't going to be smooth, but perfectly fine for static images and slowly changing text. The hardware is capable of much more than this, and it's currently slowed down by the CircuitPython libraries (which should hopefully see a speed boost in future versions). If you need animations, there's Arduino support, where we were able to get >20fps on animations (take a look at the graphics demos here for examples: hsmag.cc/jHheml).

If your main exposure to touchscreens is using phones, you need to be aware of the technology here. It's a 320×240 pixel 3.2-inch display with resistive touch. This has 125 pixels per inch (PPI). The iPhone X has 458 PPI,

Below ♦

The PyPortal is only slightly bigger than the screen, so it's easy to embed in a picture frame or other mount (there are several 3D-printable designs available from the Adafruit website)



and other modern phones are similar – of course, they're also a lot more expensive and a lot less hackable. For text and simple graphics, the PyPortal works well. For photographs and highly detailed images, you're probably going to struggle. Similarly, the resistive touchscreen on the PyPortal works well for simple finger taps, but doesn't have the sensitivity of high-end capacitive touch displays.

A built-in speaker and connections make it easy to give your projects audio, as well as visual, output. There's also an on-board temperature sensor (though the display can warm the whole board up a little), and an ambient light sensor which could be used to change the brightness of the display based on the conditions.

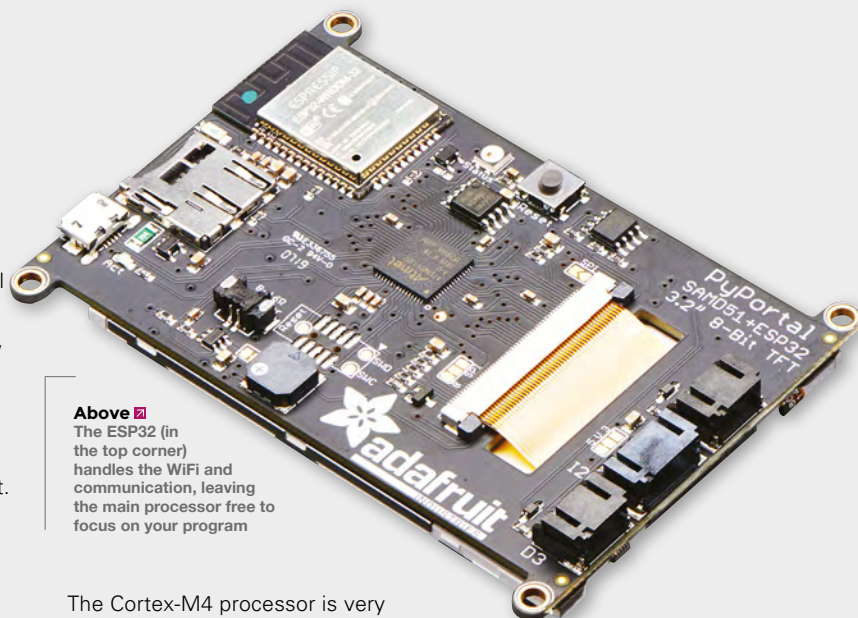
You can interact with extra hardware via an I²C bus (5V, but can be converted to 3V with a cuttable trace) and two digital IO/analog-in pins, all of which are in Grove-compatible connectors.

USER EXPERIENCE

For a web-based data display, the out-of-the-box experience with PyPortal is brilliant. Grab one of the examples (such as the Reddit subscriber count here: hsmag.cc/kToalq), change the JSON source, the background image and caption, and you've got an internet counter. The range of examples has grown significantly between when we started reviewing the PyPortal and when we went to press. If you're already moderately familiar with Python and JSON, you can probably get something working in under half an hour.

Going further in CircuitPython can be a bit tricky at the moment. The displayio module is new in CircuitPython 4, and the documentation is still catching up. That said, there are some examples out there that show how to use it – take a look at the weather display example, particularly the `openweather_graphics.py` file or the HalloWing Magic 9 Ball example (hsmag.cc/SISvKr), for details. The basic idea is that the displayio root contains a list of nodes. Each node can be either a thing to be drawn or another list. These things are then drawn in the order they are in the list (so things go over or under each other in this order). You can add text or graphics in this way, and manipulating them automatically updates the display. Of course, things are improving all the time, and when you read this, there may be more information available.

Using Arduino, the path is a bit more well-trodden. The ILI9341 chip that drives the display has a well-tested Arduino library that works with the classic GFX library.



Above ■ The ESP32 (in the top corner) handles the WiFi and communication, leaving the main processor free to focus on your program

The Cortex-M4 processor is very heavyweight for such a product, especially as the data connection is handled by the ESP32. CircuitPython is getting faster all the time, but it's still a bit of a resource hog, so having this power gives you the space to do a reasonable amount of processing. If you use Arduino, rather than CircuitPython, then you've really got a lot of processing power to play with.

You do have access to the programming pins on the ESP32, so in principle you could do more with the processor it has if you want to, whether this is more processing of the data flowing in and out, or activating some of its other features such as Bluetooth.

Overall, the PyPortal is a great one-piece solution to displaying internet data. It's got powerful hardware, is easily expandable, easy to program, and is great value for money. □

Below ◆ The weather station example code lets you know what's going on outside without having to open the curtains

VERDICT

The best hackable out-of-the-box IoT display available at the moment.

9/10



THE OFFICIAL Raspberry Pi Beginner's Guide

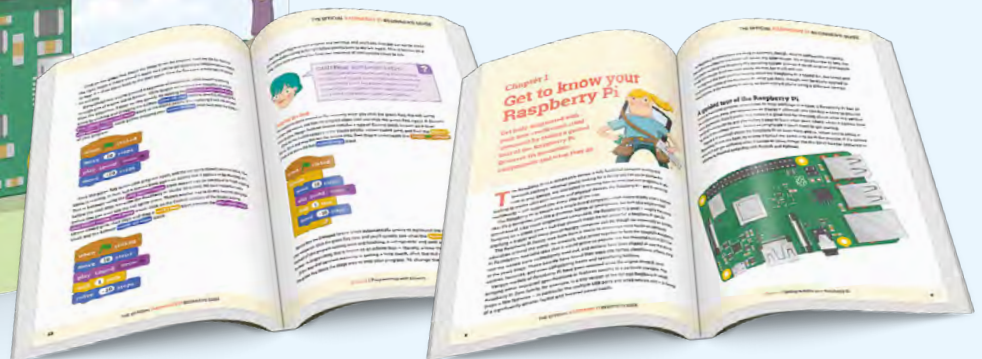
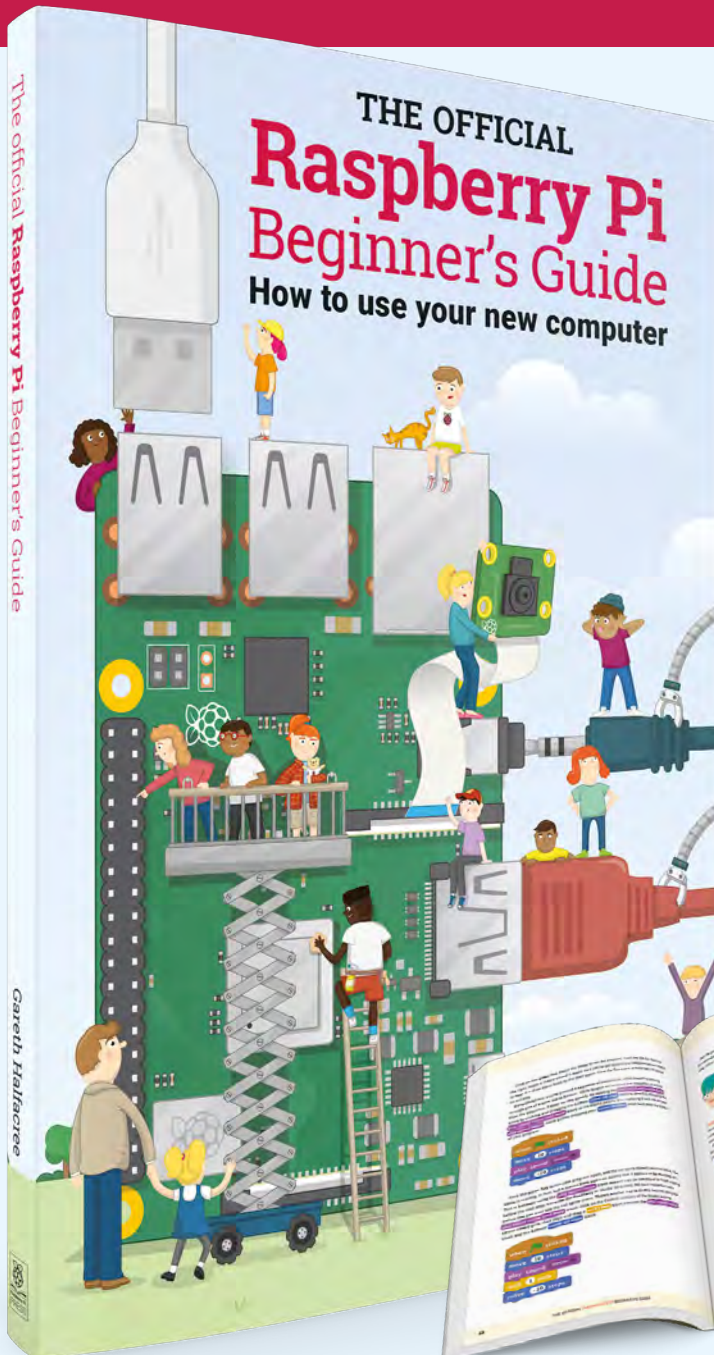
**The only guide you
need to get started
with Raspberry Pi**

Inside:

- Learn how to set up the 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 and Python languages
- Create amazing projects by connecting electronic components to the Pi's GPIO pins

Plus much, much more!

**£10 with FREE
worldwide delivery**



Buy online: hsmag.cc/BGbook

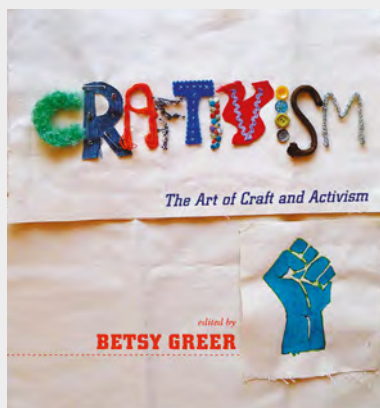
Craftivism: The Art of Craft and Activism

Betsy Greer ♦ £29.99 | craftivism.com

By Ben Everard

@ben_everard

We all make for different reasons. Some people make because they want things that they can't otherwise get, some people make because they enjoy the act of making, and some people make because it gives them a place to express their frustration with politics, consumerism, oppression, or some other force.



In *Craftivism: The Art of Craft and Activism*, Betsy Greer explores people who fall into that latter category.

Betsy explains that craft is actually a natural outlet for these frustrations: "The creation of things with our hands leads us to a better understanding of democracy because it reminds us that we have power".

The book is an anthology of articles and interviews by and with different crafters that Betsy has come into contact with through running craftivism.com. It's stories of how craft and activism have come together, rather than an instruction manual for how to do it yourself – though if you're anything like us, once you've read *Craftivism*, you'll be brimming with ideas for how to express your frustrations with politics through handmade items. Of course, the idea isn't just to express frustration, but to engage in the process of change.

Take, for example, Gabriel Craig, a jeweller who (amongst other things) took his craft to the streets. He works with people to create rings, then gives them to them for free. His goal was to encourage people to think about the work that goes into an object, and to value objects for that reason. Or, perhaps you'd rather consider Clare Thomas, a craftivist who wears red, superhero pants while crafting with rubbish found on beaches. These, and many more, are stories of people who are looking to improve the status quo.

This book focuses on more traditional crafts – sewing, knitting, mosaicking, etc. – but the message, that handmade things can be used as part of a conversation about improving society, could equally apply to more tech-y based makes. □

VERDICT

A enlightening overview of how people are using craft to instigate change.

9/10



issue
#19

ON SALE
23 MAY

FEATURING ROBOTS

ALSO

- GARDENING
- RESTORING OLD TOOLS
- BUILDING A CNC
- ARDUINO
- AND MUCH MORE

DON'T MISS OUT

hsmag.cc/subscribe

next month



The Monkey Head Nebula,
located in the head of Orion,
is about 6400 light years
away. You can view the cluster
with binoculars, but with a
telescope you can make out
more details.

