MAKE | BUILD | HACK | CREATE

HackSpace

TECHNOLOGY IN YOUR HANDS

hsmag.CC | October 2020 | Issue #35

ANIMATED EYES

Add spooky effects to your Halloween build

SOIL SENSOR

A WiFi module to keep your plants watered

Oct. 2020
Issue #35 £6

DIY ARCADE

BUILD THIS FULL-SIZED ARCADE CABINET

ROBIN BAUMGARTEN

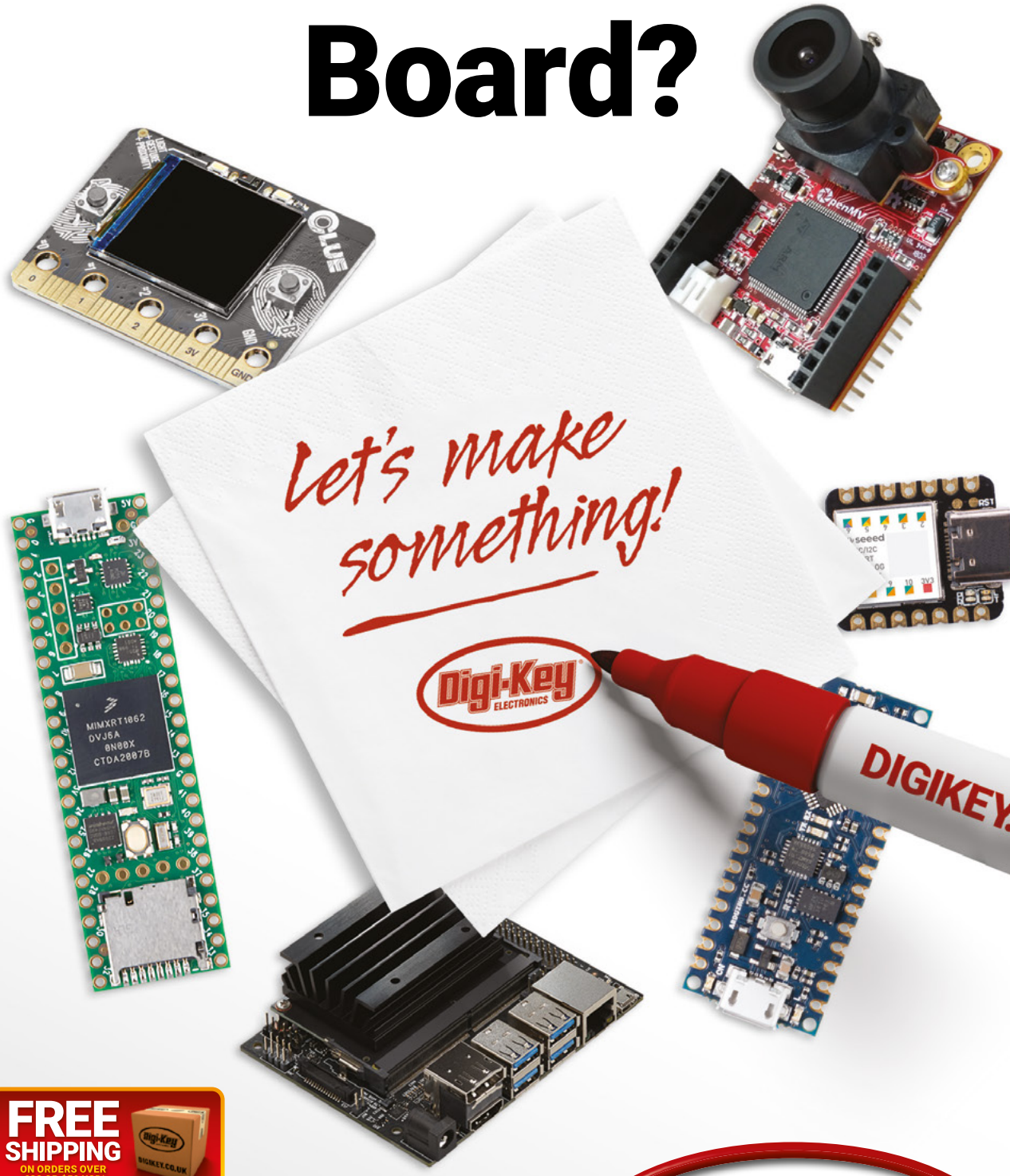
Interactive quantum BLINKENLIGHTS

SILICON MOULD-MAKING

3D PRINTING IN ASA

PLYWOOD INTERNET OF THINGS **CONTAINERS** ESP32

Board?



FREE SHIPPING
ON ORDERS OVER £33 OR $50 USD*

0800 587 0991
DIGIKEY.CO.UK

Digi-Key® ELECTRONICS

9.2 MILLION+ PRODUCTS ONLINE | 1,100+ INDUSTRY-LEADING SUPPLIERS | 100% FRANCHISED DISTRIBUTOR

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

ECIA MEMBER
Supporting The Authorized Channel

Welcome to HackSpace magazine

Computer games give us a chance to play with technology in more than one sense. Yes, we're playing a game – often against an AI opponent – but also, they give us creative freedom to try out techniques and methods that we might not find space to experiment with otherwise. As such, they can be an important part of our maker practice. In this issue, we're looking at building an arcade cabinet. Yes, this is a great, fun way of reliving your youth (and perhaps relieving your loved ones of their 10p pieces). It's also a chance to have a go at furniture making. It's not the most complex furniture making, but you can have a go at seeing how plans come together in three dimensions. It's also a good chance to have a go at different finishing techniques. Want to try your hand with spray paints? Now's your chance. How about stencils? Go for it. The more outlandish the better, and if it goes wrong, just paint over it and start again. So, let's get playing, and then once you've finished playing with the design, it's time to start playing with the games.

It's not the most complex furniture making, but you can have a go at seeing how plans **come together in three dimensions**

BEN EVERARD
Editor ben.everard@raspberrypi.org

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

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

GET IN TOUCH

hackspace@raspberrypi.org
hackspacemag
hackspacemag

ONLINE

hsmag.cc

Available on the App Store

GET IT ON Google Play

EDITORIAL

Editor
Ben Everard
ben.everard@raspberrypi.org

Features Editor
Andrew Gregory
andrew.gregory@raspberrypi.org

Sub-Editors
David Higgs, Nicola King

DESIGN

Critical Media
criticalmedia.co.uk

Head of Design
Lee Allen

Designers
Sam Ribbits, James Legg, Ty Logan

Photography
Brian O'Halloran

CONTRIBUTORS

Lucy Rogers, Drew Fustini, Gareth Branwyn, Jo Hinchliffe, Mayank Sharma, Andrew Lewis, Andrew Robinson, Ben Hardwidge, Marc de Vinck, Matt Bradshaw

PUBLISHING

Publishing Director
Russell Barnes
russell@raspberrypi.org

Advertising
Charlie Milligan
charlotte.milligan@raspberrypi.org

DISTRIBUTION

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

SUBSCRIPTIONS

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

To subscribe
01293 312189
hsmag.cc/subscribe

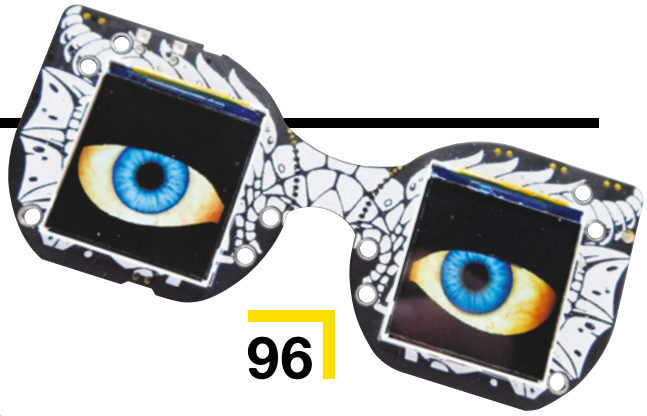
Subscription queries
hackspace@subscriptionhelpline.co.uk

recycle

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

 ISSN: 2515-5148.

Contents



96

06 SPARK

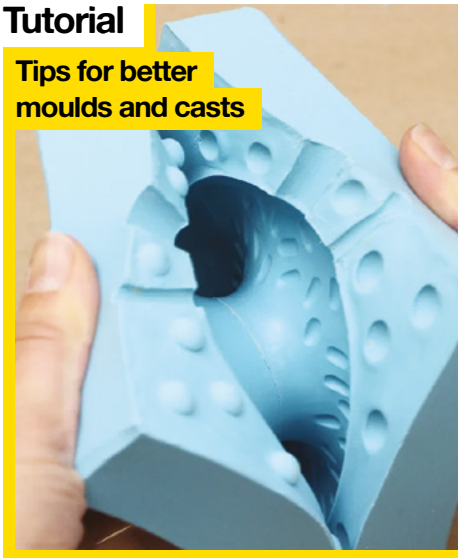
33 LENS

Cover Feature

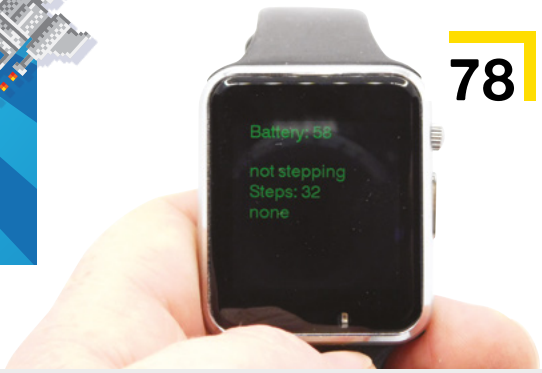
34

Tutorial

Tips for better moulds and casts

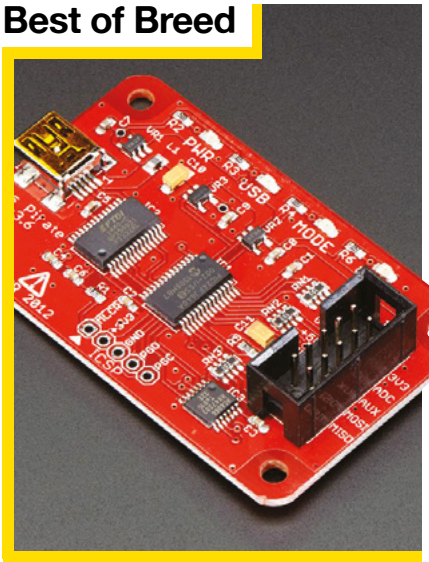


86 Enjoy the satisfaction of a perfectly smooth cast



78

Best of Breed



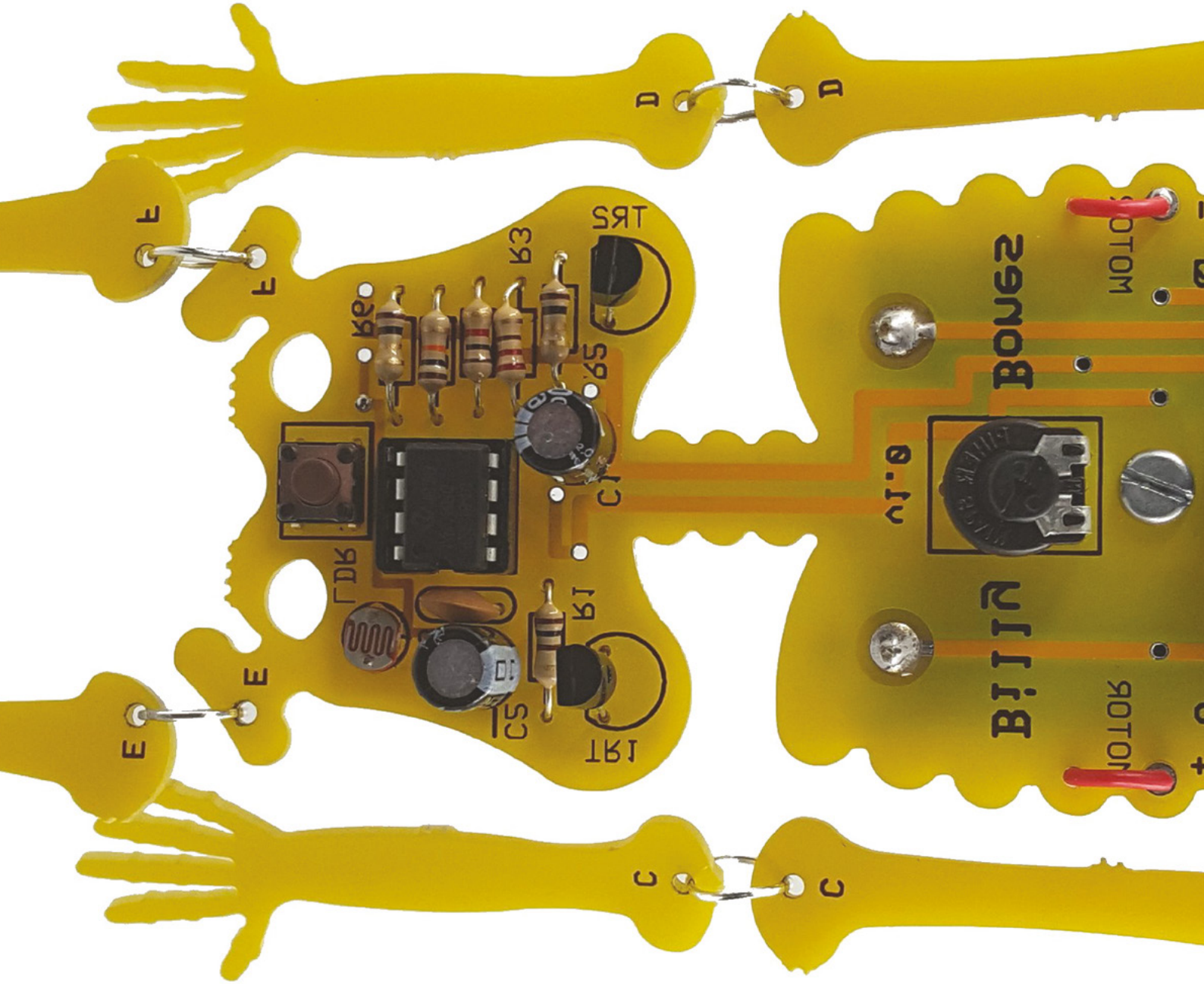
Interview

Robin Baumgarten

69 FORGE

101 FIELD TEST

REGULAR

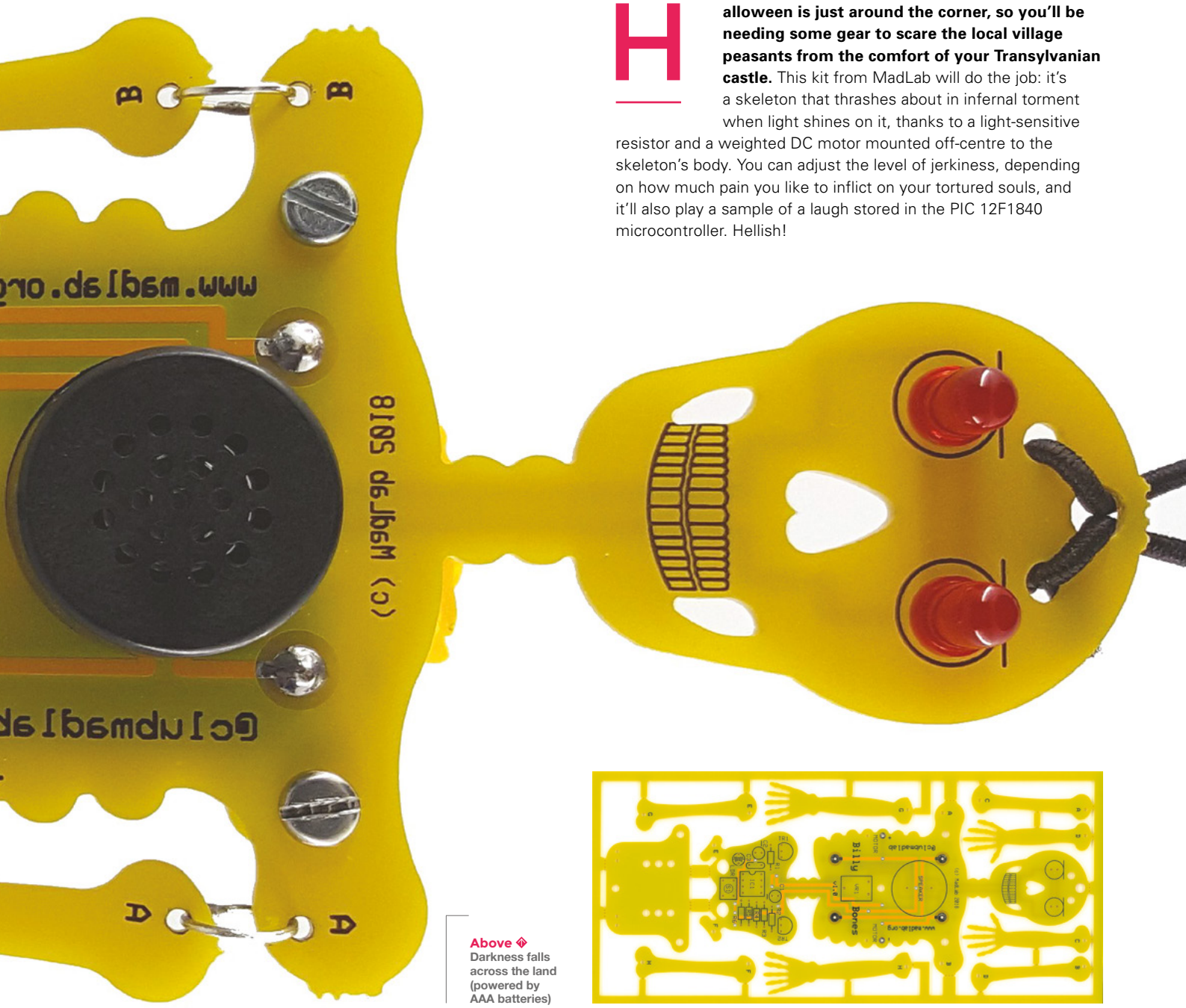


Billy Bones

By MadLab

hsmag.cc/5oSsgK

Halloween is just around the corner, so you'll be needing some gear to scare the local village peasants from the comfort of your Transylvanian castle. This kit from MadLab will do the job: it's a skeleton that thrashes about in infernal torment when light shines on it, thanks to a light-sensitive resistor and a weighted DC motor mounted off-centre to the skeleton's body. You can adjust the level of jerkiness, depending on how much pain you like to inflict on your tortured souls, and it'll also play a sample of a laugh stored in the PIC 12F1840 microcontroller. Hellish!

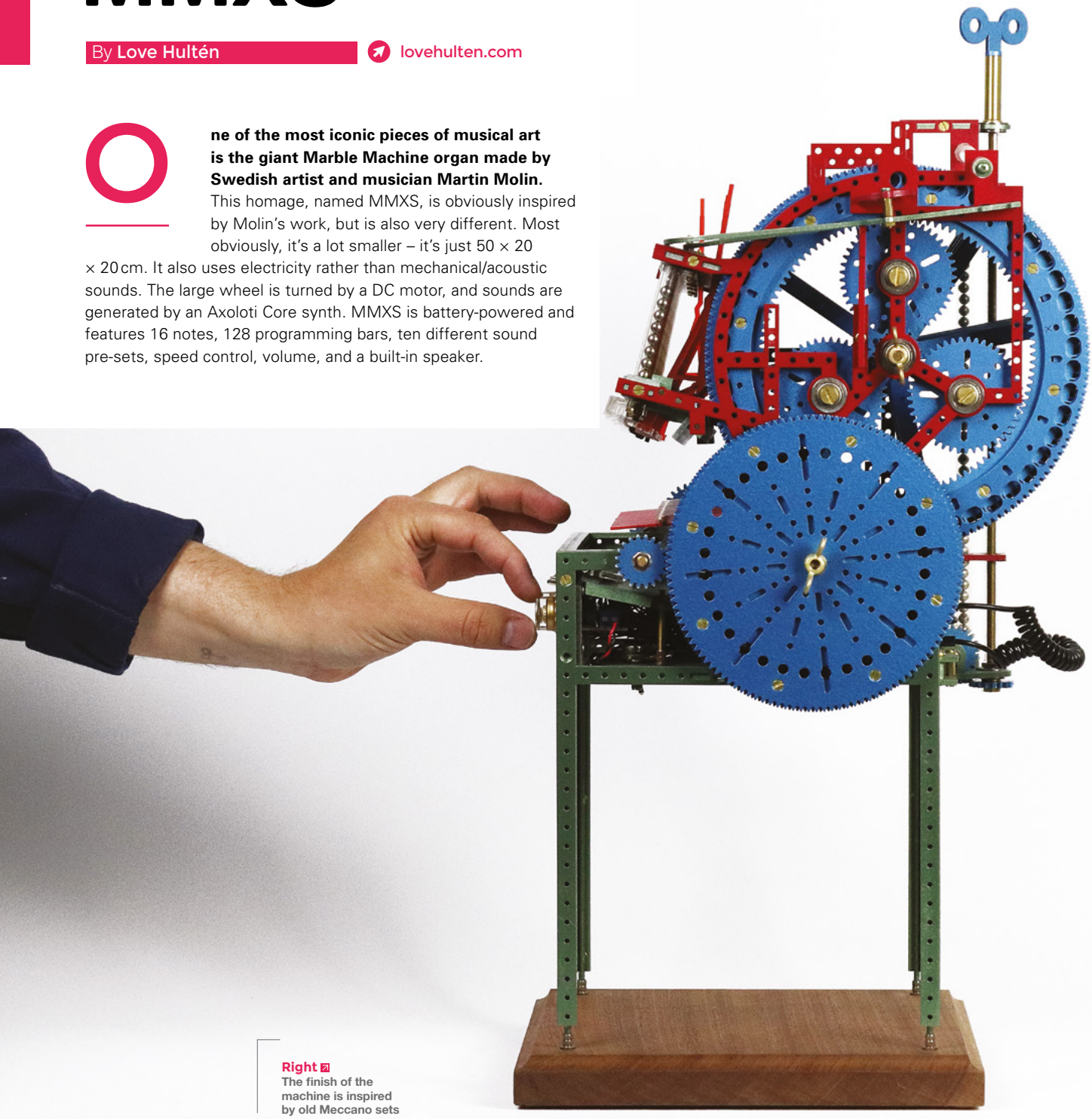


Above
Darkness falls across the land (powered by AAA batteries)

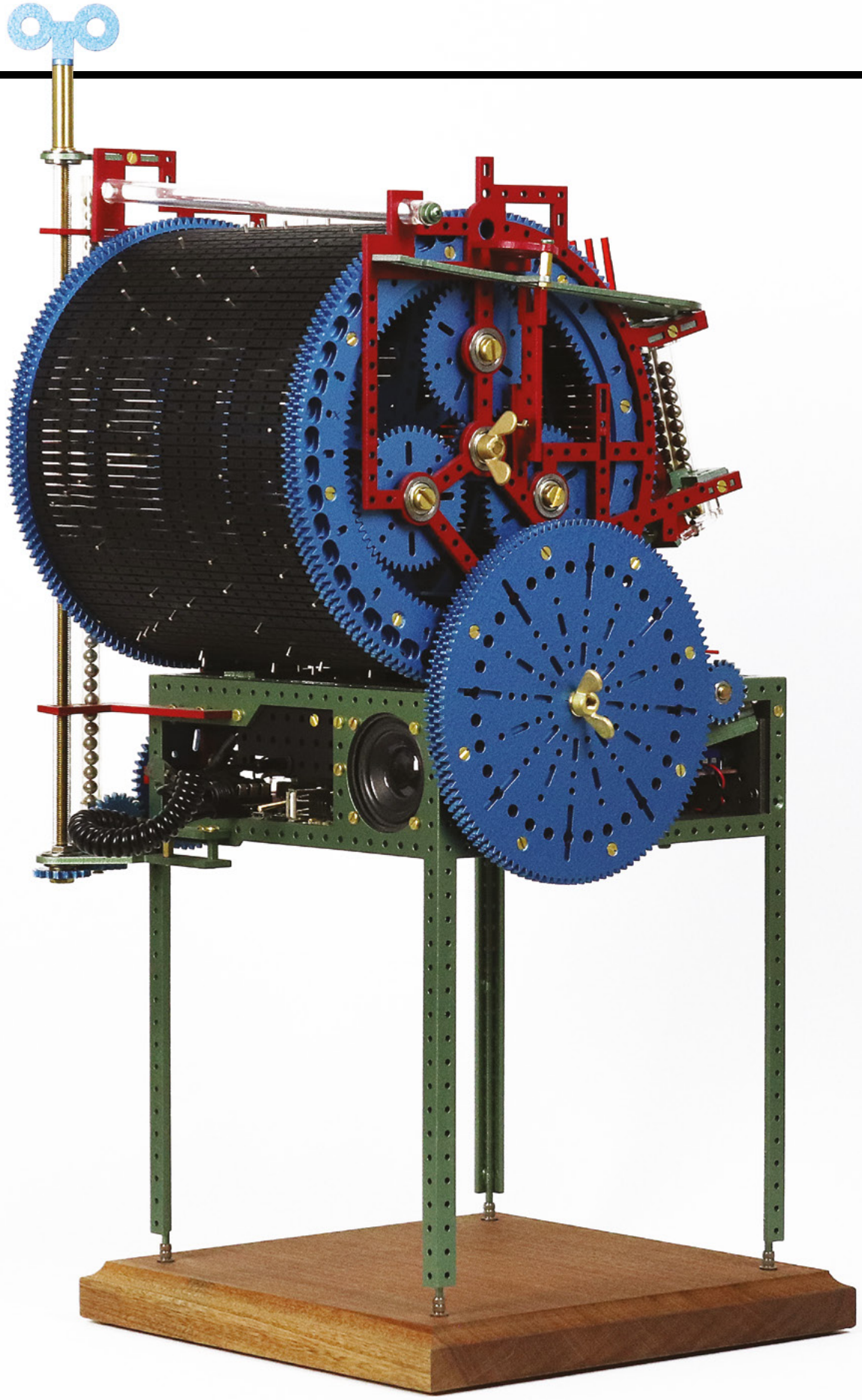
MMXS

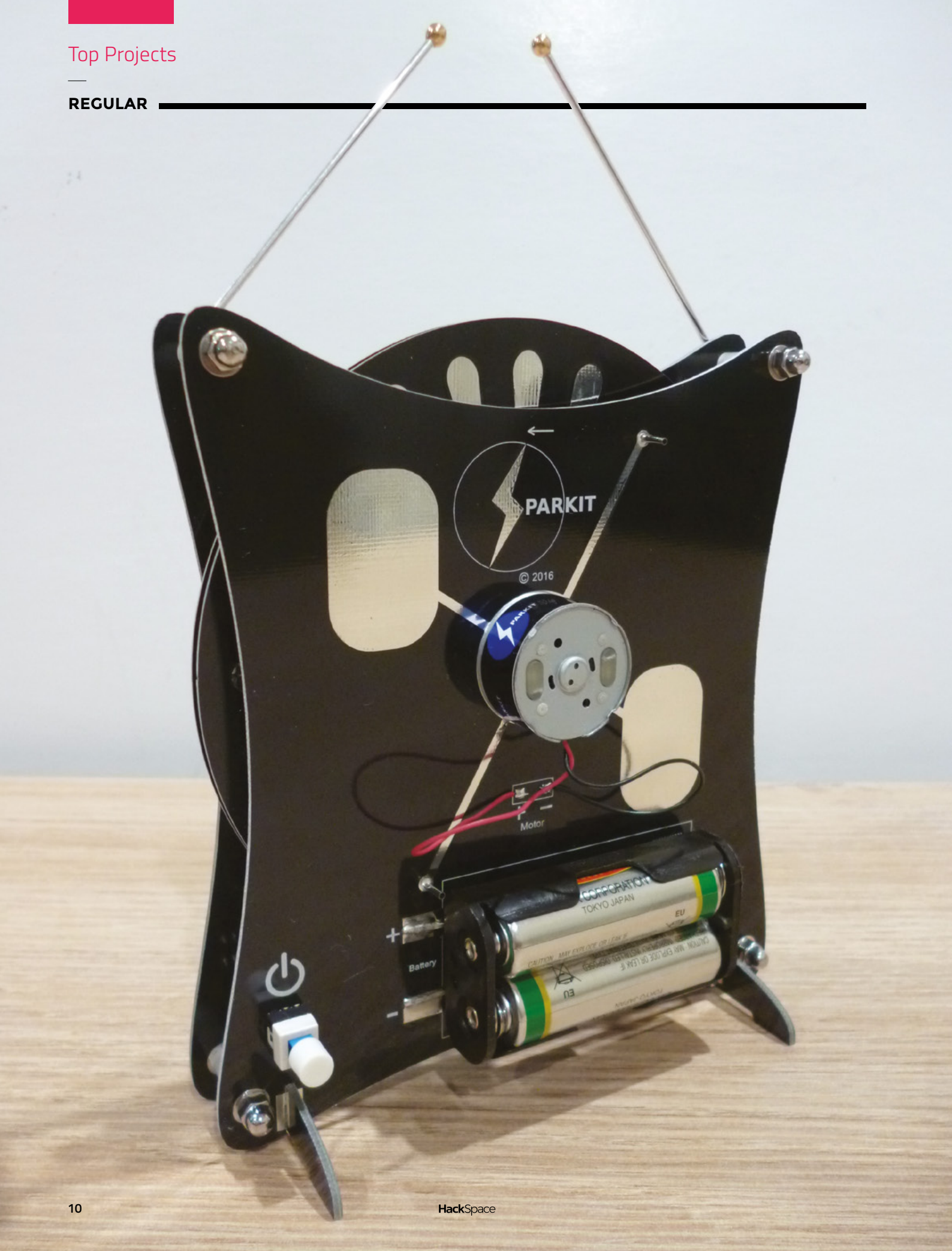
By **Love Hultén** lovehulten.com

One of the most iconic pieces of musical art is the giant Marble Machine organ made by Swedish artist and musician Martin Molin. This homage, named MMXS, is obviously inspired by Molin's work, but is also very different. Most obviously, it's a lot smaller – it's just 50 × 20 × 20 cm. It also uses electricity rather than mechanical/acoustic sounds. The large wheel is turned by a DC motor, and sounds are generated by an Axoloti Core synth. MMXS is battery-powered and features 16 notes, 128 programming bars, ten different sound pre-sets, speed control, volume, and a built-in speaker.



Right
The finish of the machine is inspired by old Meccano sets




PARKIT
© 2016
Motor
TOKYO JAPAN
Battery

SparKIT

By **Luka Phillips**

hsmag.cc/Z92Q6w

The SparKIT is a Wimshurst machine built mostly from PCBs. It is the result of a science fair project back in 2016, when I was 12.

"The idea for a PCB Wimshurst machine was not mine, but my father's. Many years ago, he attempted to make a Wimshurst machine from PCBs but was not successful.

"My science fair was an investigation into why his design didn't work, and followed my journey of experimentation and testing to work out the problem and find a solution.

"Unfortunately, the project did not win any prizes in the science fair, but it left me with a design for a Wimshurst machine that could be easily manufactured, and brought to market.

"The following year, I ran a successful Kickstarter and, since then, I have been selling the SparKIT Wimshurst machines all over the world."

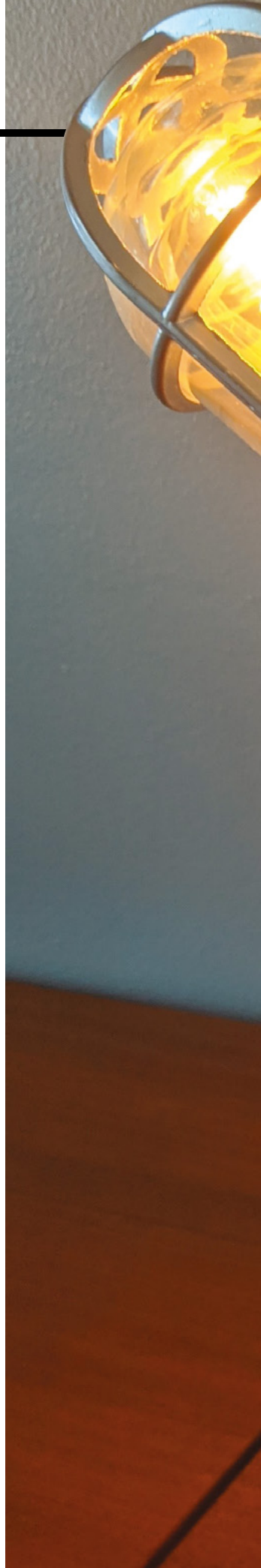
Left
If Luka has seen further, it is because he was standing on his dad's shoulders

Rocket lamp

By dberkel1 hsmag.cc/9ARv37

This rocket ship lamp is so good it must have taken countless hours in CAD, needed a resin printer, and years of metalwork skills, right? Wrong! Every single component of this build is available off the shelf from your local DIY store. The main fuselage of the rocket is a security light, the mount comprises a few standard plumbing components, and the flames are flickering candle bulbs. The cleverest bit of this build is the stroke of genius it took to turn the candle bulbs upside down, which transforms them from cheesy, festive decorations to powerful, interplanetary exploration engines. For a pragmatic, practical afternoon build, this is brilliant.

Right
Build a rocket and get to space before Elon ruins it



REGULAR

Voodoo robots

By Geeekclub.com

geeekclub.com

Most PCB art projects limit themselves to two dimentions, using the solder mask layer to turn utilitatian electronics into something that's a pleasure to look at. These robots by Geeekclub go several steps further – not only are the PCBs themselves lovely to look at, they're functional construction sets, a lot like Airfix models. These voodoo-inspired creatures use vibrating motors to search for light, an each one presents a unique engineering challenge. According to Geeekclub co-founder Nico, they're inspired by Lego, Peruvian indigenous art, Hopi culture, and too much hanging around with engineers.

Right
A selection of Voodoo Bots lighting the way

Objet 3d'art

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

If there's something strange in your neighbourhood. Who you gonna call? Sean Charlesworth! That's who made these incredibly accurate (we wouldn't be surprised to learn they're fully functional) ghost traps from the 1984 classic *Ghostbusters*.

In Sean's own words: "I designed the trap to be completely 3D-printed, aside from [its]fasteners, but I also designed most of the parts based on real materials so [that] 'deluxe' parts could easily be swapped in. I teamed [up] with Jeremy Williams (Game Frame) to do the electronics – I designed the mechanics, and Jeremy figured out all the electronics and programming. The original setup was Arduino-based."

Sean has released the 3D files so that budding paranormal investigators can print their own traps, and catch their own ectoplasmic manifestations (**hsmag.cc/hSwt05**). Since tracking down the hardware for the traps can be tough, Sean has started putting together kits. □

charlesworthdynamics.etsy.com


DANGER
HIGH VOLTAGE 1KV
ELECTRIC SHOCK HAZARD
REMOVE POWER BEFORE
REPAIR OR ADJUSTMENT
SIDE MM PN
SIDE MM PN

Meet The Maker: Caroline Buttet

Giving new life to ordinary things

Finding new ways to interact with old objects. Taking dumb objects and making them smart. Combining Arduino and Raspberry Pi with junk from charity shops, and injecting some creativity to help others see the world in a new way. This is the mission of Caroline Buttet, artist and re-purposer of things. And when she's not doing that, she's bringing the world's art collections to the internet as part of her work with Google's Arts and Culture Lab. Here's how she does it:

> We collaborated with the industrial designers, **so they made an object, and we had to make the object 'smart'**

"I did a bachelor's degree at a design school in Switzerland; the branch I took was media and interactive design. I did some 3D, some websites, and also we did a little bit of programming, which is where I got my introduction to Arduino. We did some work along with a student in our school who studied industrial design. We collaborated with the industrial designers, so they made an object, and we had to make the object 'smart'. That was my introduction to Arduino and Raspberry Pi.

"What we came up with was a cloche, like they have at fancy restaurants when they hide your dish so that there's a magical reveal when they serve you your meal. Inside that cloche, we put a speaker, and the cloche would detect when anyone picked it up, and it would then play sound, like a kind of miracle occurred.

"This was the first project. It used a Raspberry Pi and a Bluetooth speaker – very simple, but that was my first introduction. This was maybe 2015. The first time I bought a Raspberry Pi I wanted to plug it into my computer because I didn't realise it was a computer – it was so small. I was blown away.

"When I was little I would always draw, and was thinking about going down the classical art route until I found out what you can do with programming. I brought technology into my work, which gave a whole new dimension to my practice. I went from drawing, which I loved, to making smart objects, installations, which are far more interesting to me than drawing. You can take an object that everyone knows, like a map or a globe or a telephone, and with the same interaction that people are used to, you add something new. That's what's powerful about design; you can recreate or hack objects in a meaningful way.

"One example is the peephole I made. I discovered this website – it's called **insecam.org**. It gathers all of the insecure webcams that are online; insecure because the owner of the camera didn't set up a password, so you can access it with the manufacturer's default credentials. There are so many webcams on that site, it's amazing. There are webcams on the outside of buildings, but also in interior spaces like offices. So I saw this website and →

Left
A Raspberry Pi and a Bluetooth speaker combined in the cloche of miracles

REGULAR


ARCTIC OCEAN
PACIFIC
OCEAN
ATLANTIC
OCEAN
PACIFIC
OCEAN
INDIAN
OCEAN
THE WORLD
MERCATOR'S PROJECTION


SOUTH AMERICA
AFRICA
EUROPE
ASIA
Montevideo
Porto Alegre
RIO DE JANEIRO
Lisbon
Madrid
Cairo
Khartoum
Addis Ababa
Tehran
Kabul
Karachi
Delhi
Bombay
Tashkent
MOSCOW
LENINGRAD
Principal Railways
Steamship Routes
Capitals of Countries
Lengths of Rivers in miles
The Dials indicate the Time on the different Meridians when Noon at Greenwich.
REFERENCE
Ocean Currents (Warm, Cold)
Heights in English Feet

Left
Capacitive touch gives makers an intuitive way to add interactivity to objects

thought, 'The material here is amazing; I need a way to display it!'

"I made this wooden box with a peephole on it, and whenever you look in the door there's a small screen powered by a Raspberry Pi, and it displays one of those webcam feeds at random. Every time you close that door and open it again, I've got a way to detect that action with a simple sensor, and it switches the feed. From your home, you can spy on people from all around the world.

"This is very disturbing, and I wanted it to be disturbing. People should be disturbed by this! You're not supposed to look, but you can – it's a strange feeling. And also, the interaction of opening the peephole is quite voyeuristic. You look at it for one second and you get the concept – that's the kind of stuff I'm looking for.

"I did that installation. I posted the whole breakdown of the project on Instructables and another site so [that] people could have a look and make their own, and recently, I did a virtual version of it: a bot on Instagram and Twitter that posts an image from one of these feeds with the caption along the lines of 'Someone in the United States with a Logitech camera didn't set up their credentials'. I get strong reactions, but I want to sensitise people to this: if I can do it, anybody can do it. It doesn't require any knowledge at all. I want people to be aware of this issue. I find it so interesting: how much personal data you give up on the internet without knowing. I haven't found the object yet, but that's what my next project is going to tackle.

"For my day job, I'm a freelance coder for the Google Arts and Culture Lab, mainly doing web stuff. It's like a catalogue. If museums have a collection that they want to show to the world, they can just use that website as a portal to display their images and get visibility. And so we at The Lab are promoting these artworks. But there are so many artworks, so how do you do that? You just can't make a slide show; that would be terrible, so boring!

"We need to find ways to display that kind of content in a meaningful way. The best example would be the website we did that shows images from NASA. They had somewhere in the region of 200,000 pictures, and so we scanned all of the pictures virtually, and used machine learning to detect labels on top of the images. There's a rocket, there's a dog in this one, this one is space, the moon, all of those labels were linked to those pictures, and we mapped [it] so [that] it looked as if you were looking out →

[into] space. The problem we have is how do you make sense of large amounts of data? How do you make it interesting? How do you use tools like machine learning to facilitate your work and arrange things in a cool way?

"The first build I did that got any attention used a globe as a way to show my holiday pictures to people. I'd had an amazing trip and wanted to show it to people, but what's more boring than somebody showing you their pictures? So I took this globe, put a pin in each of the countries I'd visited, [then] ran a wire from each pin to an Arduino. And so with capacitive touch, you can detect whether the user is touching Japan or Italy or wherever. The Arduino was linked to my computer and was acting as a keyboard, so whenever I would touch Italy, it sent the keystroke 'I' for Italy. I'd linked this to a screen and a web server, so whenever the 'I' key was pressed, it displayed photos of Italy. On the axis of the globe, I had a rotary encoder, and when the position of the encoder

Left
It's one thing to look at a website showing insecure camera fieeds, but something else entirely to see the same thing through a physical peephole: you become complicit in the voyeurism

changed, you display the next photo in the series. It was a bit scrappy, but I think it worked because the interaction was so simple. The relationship between the object and what it does was so obvious.

"I find that people around me know that I like to hack stuff; I got a telephone from a friend who contacted me to tell me that he was getting rid of this old object. I like to salvage objects or take them from second-hand shops.

"Usually when it happens, first, I need to see the object, and then I have an idea for a new interaction; it doesn't go the other way round. I'm not looking for a specific type of phone, I just have the phone in my hand and think. 'What would be nice to do with this specific object?'

"All of the code for my projects is open-source on GitHub, you can find it with instructions. A few weeks ago, a science teacher from the US contacted me; he did an assignment with his students, and they used the globe project as the basis for the assignment. Open-source should be standard, because it's how people share knowledge. I love it. You find ideas for things you would never have thought of doing.

"I use other people's code all the time: Googling stuff, checking on Stack Overflow. If somebody's done it already, why would you do it yourself? I'm all for giving credit if I've taken code for someone, or cite them in the sources.

"When I first started, if I hadn't been able to take someone else's code to do something, I wouldn't have done anything, it would have been too hard to sort. When I look for answers on forums I'm always so amazed, because it takes time to do a tutorial or write something about how to make something, and the people who do it, do it for free. I think it's amazing – how can you be so generous that you put the information out there for others to use, and then answer questions about it on Stack Overflow? Whenever I do a project, I try to do a write-up because somebody might want to build the same thing." ◻

Budding builder

My journey in woodwork

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**

For my seventh birthday, I asked for a carpentry set.

My dad made it out of real tools – a little yellow flat-head screwdriver, a small ball pein hammer, a junior hacksaw and a dovetail-saw, some clamps – which I am still using – and a bright orange, plastic tool-box.

One of the first projects we made together was a bird table. I collected about 30 straight twigs, about as thick as my then thumb, and cut them all to length. This kept me busy while Dad made the rest of the bird table – the stand from a thick branch, the platform and pitched roof from plywood, the struts from sticks as thick as my young wrist. I nailed my twigs to the roof – giving it a sort of thatched appearance. To be fair, I probably nailed two or three of the twigs in place. I didn't have a great aim then. I remember hitting my finger and letting Dad do the rest.

Fast forward 20 or so years, and my woodwork was mainly for DIY. I have fond memories of straddling a pine door (I didn't have a Workmate), and being hit by the fresh smell as the shoosh, shoosh of a sharp plane removed the edge in satisfying curls of shavings. I have less fond memories of trying to hang the door.

I then had the opportunity to attend a wood-turning course at my local college. In the first session, I was given a foot (300 mm) long, one inch (25 mm) square piece of pine, a round-nosed scraper – a type of chisel that has a round nose and scrapes the wood off – and I was shown how to use a lathe. Thirty very happy minutes later, and I had a pile of woodchips and a rough, round piece of wood with lumps in it. I was hooked.

My first project was to make a bowl. I used a lump of wood I'd found in a skip. It was pale yellow and heavy. After extensive research, (this was before the internet was really useful) I decided it was probably ramin. A wood I had not come across before – or even since.

> **Just because straight lines and balls look simple, they are not easy to make!**

The second project was to make four matching spindles or legs – which was a lot harder than I'd thought – even with the help of a template and spring calliper. Looked at together, the legs are obviously not identical. However, at the corners of a stool, they match. I used ash for this project, and it's still my favourite wood – for smell, look, feel, and joy of working.

For the final project, I made a standard lamp from sycamore and elm. The light colouring of the sycamore contrasted well with the dark of elm. But I discovered elm is hateful to turn – it's 'fluffy' – it does not cut nicely, and is hard to make smooth. I also learnt a lot about design in that project – just because straight lines and balls look simple, they are not easy to make!

Propelled by community

Projects pushed by people power

Drew Fustini

@pdp7

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

I spent some time in Copenhagen last year, where I got to see some really awesome spaces and projects. On a visit to illutron, an 800 m² industrial barge that houses a floating artists' collective and hackerspace, I turned the corner and found myself outside a warehouse belonging to the Copenhagen Suborbitals group. Copenhagen Suborbitals is a crowdfunded amateur space programme who have successfully launched five rockets and two capsules, including the most powerful amateur rocket ever flown, and the first amateur rocket flown with a payload of a full-size crash test dummy. Happily, they agreed to show me around their warehouse, so I got to take a look inside this impressive community project. One of the highlights of my visit was getting a demo of their newest bit of kit: an Arduino-based reaction control system using compressed gas to manoeuvre the capsule (**hsmag.cc/6U6445**).

Another DIY space propulsion project that has really excited me recently comes from Michael Bretti, who runs Applied Ion Systems (AIS). AIS is an open-source R&D program for electric propulsion. Their open-source thrusters and testing machines are comparatively low-cost (with costs starting at $500 plus, as opposed to $25,000 in academia and industry) and easy to manufacture, putting this technology within reach of nano-satellite groups, hobbyists, and educators.

As well as designing, making, and sharing his own work, Bretti is providing resources for hobbyist nano-satellite groups and educators, including projects and experiments that explore a number of advanced science and engineering concepts from an accessible, maker-friendly point of view. Check some of them out in the DIY Science section of the AIS website (**hsmag.cc/CbC4EV**) and find Bretti on Twitter: **@Applied_Ion**.

The final project I wanted to talk about comes from regular HackSpace magazine contributor Jo Hinchliffe, also known as **@concreted0g** on Twitter. Hinchliffe runs Open Research Rocketry (ORR), who research, build, publish, and fly open-source rocket designs, many of which carry payloads for research projects. ORR is currently looking for people to get involved with their new open-source flight computer project: a logging altimeter with dual deployment that returns data to an open online database. This data would be used to drive real science and datasets in rocketry, something that would be incredibly useful for comparing performances of different rocket kits and equipment configurations.

> **ORR are currently looking for people to get involved with their new open-source flight computer project**

If you'd like to get involved with the flight computer project or take a look at the files for the ORR's high-power amateur rocket, you can find more details at **hsmag.cc/FleiX7.** Issue 12 of HackSpace magazine includes a number of rocket-themed articles, including one by Hinchliffe about using the software OpenRocket to design your own rockets.

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**

LEGO

Please give yourselves all a hearty pat on the back for titling a boxout on the right size of hole to use for a Lego Technic axle 'Axle Holey'. I had the *Beverly Hills Cop* theme in my head for a solid afternoon as I was reading issue 33.

Alfie Jones
Guildford

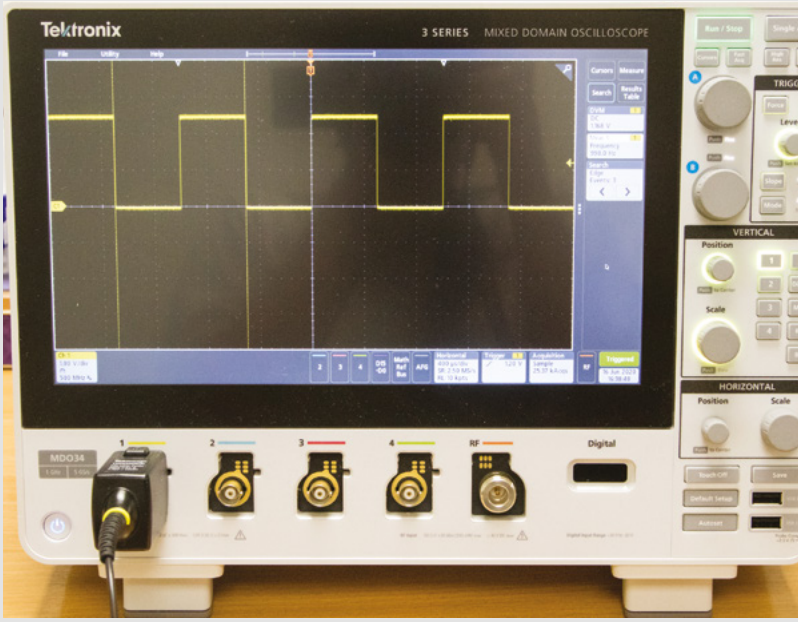
Ben says: The pat on the back goes to Matt Bradshaw, who did a brilliant job modifying Lego to use as a shelf in his shed. If you missed the article, the whole magazine is available as a free download (hsmag.cc/jgLLpo), so if you're new to the magazine, you can always catch up on what you've been missing. And there's more of Matt's excellence in this issue: turn to page 48 for an incredible dive into beats, bleeps, and product design.

DEBUGGING

I'm probably late to the party as a reader Down Under, but I wanted to say thanks for the hardware debugging article. I now know where to turn when I'm fixing my own idiotic mistakes, but more importantly, I'll remember the golden rules of how to design a circuit so that an idiot like me can fix it in the future!

Robert Wilson
Penrith (the one in Australia)

Ben says: 'The golden rules for designing easy debugging are: 1) Make it easy to *observe* behaviour, and 2) Make it easy to *control* behaviour.' Dr Andrew Robinson continues his debugging series on page 72 of this issue.

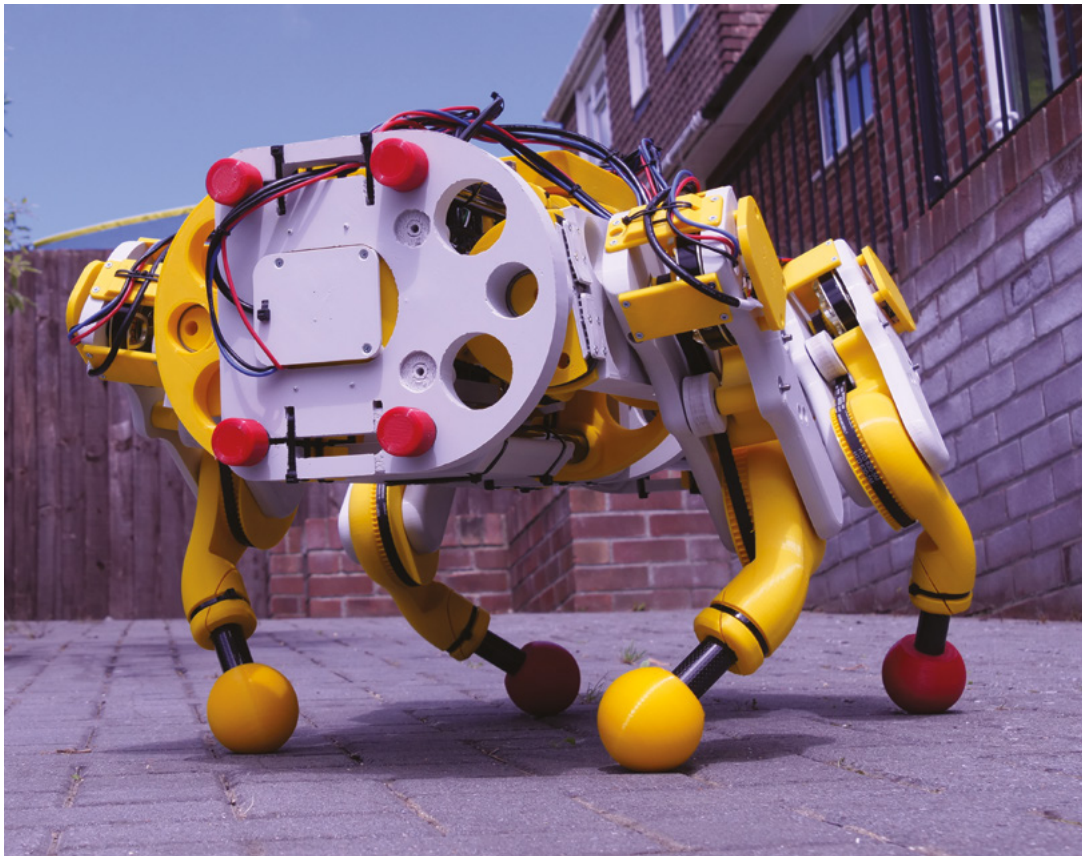


OPENDOG

I remember seeing James Bruton's openDog ages ago, maybe in a thumbnail on YouTube. I didn't even bother clicking on it – what's the point, when there's no way a bloke working on his own can come anywhere close to Boston Dynamics? I'm happy to say I was wrong though, not because of the robot itself (impressive though it is), but because I've learned so much from watching him. Apart from considerations of torque, weight, and feedback from sensors on the joints, I've learned that if your router bit is cutting aluminium too fast, it's hot enough to melt the swarf back onto the workpiece. That's stuck in my head now, never to leave!

Dave Eastwood
East Yorkshire

Ben says: James's openDog project is truly amazing, and the videos are incredible too in the amount of detail they go into. What's almost as impressive is James's decision to junk openDog v1 and start again with v2. It must have been hard, but that's how we get better and learn.



REGULAR

CROWDFUNDING NOW

Bittle

Build your own robotic pet

From $225 | **kickstarter.com** | **Delivery: December 2020**

There's something fantastically futuristic about robot pets that walk around the house. When Petoi released Nybble – a robotic cat – via crowdfunding in 2018, they unleashed the dream of electronic friendship for hundreds of people. Now they're back with a new kit – Bittle. This time they've created a dog, and from what we've seen, it looks like a sleeker, faster robotic companion for you to build and program.

There's a couple of things that really make Bittle stand out, compared to other walking robot kits we've seen: it moves much faster thanks to high-speed servos, and spring-loaded legs give a little for more natural walks.

The walking body has eight servos – two on each leg – plus a ninth to move the head. This is enough for quite a bit of motion, such as walking, and the Kickstarter video shows it at a moderate trot. For many uses, this is a sweet spot where it keeps weight, cost, and power consumption low enough, but still provides enough range of movement for some interesting movement. However, if you're hoping for a Boston Dynamics-style running, jumping, super-canine, you might want to temper your expectations. There's a reason the Boston Dynamics dog costs $75,000, while the full kit for Bittle is $225. That's not cheap, but it's at the lower-cost end of quadruped robot kits.

The real value in Bittle will depend on the programming environment. Controlling servos is easy enough from just about any language, but if you have to start from this level, it's a very involved process to just get to a walking dog. If you can start with a reliable and easy way to get your Bittle to move around and focus on the higher-level things (such as what you want to get them to move to), it's a more rewarding experience. While we've not been able to test out the Bittle, it's based on an improved version of the NyBoard that drives the Nybble robotic cat. This provides a range of instructions that are available via IR or serial port. There are mounting points for a Raspberry Pi model A+ or Zero, so you can use this to do any higher-level processing you want, and then offload the actual movement to the NyBoard. If this works reliably, then it should provide an easy way to program your robot to move. There's also a block-based programming environment based on TinkerGen (**ide.tinkergen.com**) planned.

With robots such as this, it really comes down the minutiae of how they move and what the programming environment allows. Without having one to play with, we can't say for sure, but based on what we've seen, the Bittle looks like it'll be a really fun way to add some motion to a Raspberry Pi. We're only slightly disappointed that they chose to do a dog for this robot rather than another cat.

BUYER BEWARE

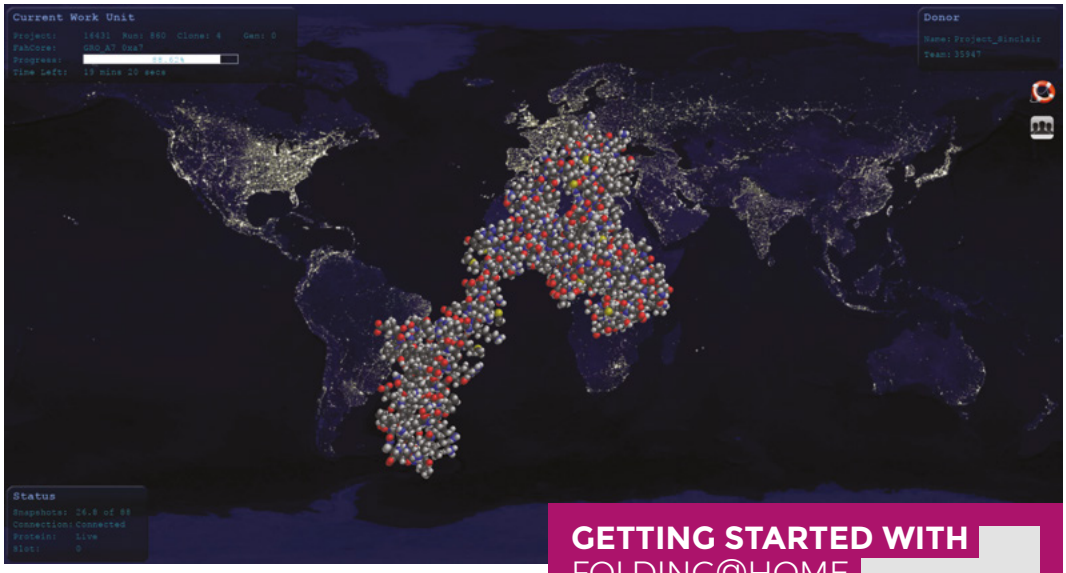
When backing a crowdfunding campaign, you are not purchasing a finished product, but supporting a project working on something new. There is a very real chance that the product will never ship and you'll lose your money. It's a great way to support projects you like and get some cheap hardware in the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

REGULAR

Sponsored by **Arm DevSummit**

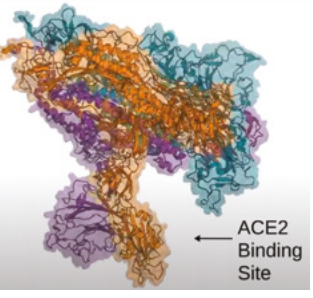
Folding@Home and the fight against COVID-19

Volunteers contribute computing power to help advance medicine



Right
The Folding@Home client viewer lets you see a visualisation of the molecule you're processing

Below
The opening motion of the 'Demogorgon' COVID-19 spike reveals the ACE2 binding site for interaction with human cells



Folding@Home is a project that allows a global volunteer network to contribute their processing power to help understand how proteins fold. This might sound obscure, but through the power of crowdsourced computing, researchers can understand how biological processes happen on a tiny scale, and from this, we can start to understand how different chemicals and agents might affect them. Traditionally, researchers studying protein folding have had to rely on expensive supercomputers to do their work. With Folding@Home, that has changed. Instead of utilising supercomputers, researchers can now submit workloads to the Folding@Home volunteer army of computers around the world and have their processing done for free. The more people that download the Folding@Home software and run it on their machines, the faster researchers can crack the secrets to challenging problems by submitting their processing jobs to the network (see box opposite).

GETTING STARTED WITH FOLDING@HOME

It's incredibly easy to get started with Folding@Home. Just head to **foldingathome.org** and download the appropriate client (available for Windows, macOS, and 64-bit Linux). Once this is installed, run the program to start folding. You can also set up a profile and join a team – various teams compete to see who can do the most work. There's a leader board at **stats.foldingathome.org/teams-monthly** and teams engage in a little friendly rivalry to see who can perform the most work units.

The easiest way of getting started with distributed computing to help fight COVID on an Arm device is to use the Fold for COVID system available from **foldforcovid.io**. It runs on a wide variety of Arm-based boards, including Raspberry Pi models 3 and 4. This is a Balena-based system that lets you download an image with everything set up and ready to go. Once you've burned the SD card, just power up your system and you're ready to go.

You can also download Folding@Home and Rosetta@Home client applications from Neocortix to volunteer your mobile phone in the efforts: **neocortix.com**.

Ben Hardwidge spoke with Dr Greg Bowman, Director of Folding@Home and Associate Professor at Washington University School of Medicine, to find out how this project – which has been backed by an all-star cast of tech industry heavyweights such as Microsoft, NVIDIA, AMD, and Arm – has been using its computing power to help the global fight against COVID-19.

"The simulation of the COVID-19 spike," says Bowman. "We've been calling it 'the Demogorgon' because of this opening motion it has – the opening of the three receptor-binding domains reminded us of the mouth of the Demogorgon from *Stranger Things*."

Analysing this opening motion was a key project for Folding@Home, as there was no way to observe it using standard experimental techniques. The aim was to understand how the COVID-19 spike protein opens up to bind to a protein called ACE2 on human cells. "Specifically, the spike has three receptor-binding domains that directly bind to ACE2," Bowman explained on his blog. "For the spike-ACE2 interaction to form, the spike's three receptor-binding domains must open up to reveal the binding interface."

In the image of the 'Demogorgon' spike on this page, each of the three proteins that form the spike has been given a different colour. These three proteins all need to spread apart to open up access to the ACE2 binding site, so that it can interact with the surface of human cells and initiate infection.

After the opening motion was visualised, the Folding@Home Twitter account explained that Folding@Home's next step was to "help figure out where the protein spends the majority of its time using thousands of simulations run by our donors. This kind of information will help to prioritise drug development efforts."

The Demogorgon isn't the only target for Folding@Home's work on COVID-19. At the moment, Bowman says Folding@Home is collaborating with Diamond (**diamond.ac.uk**) in the UK, an X-ray beamline company that's currently using macromolecular crystallography (MX) to study COVID-19.

Exploring beamline technology in depth is beyond the scope of this feature, but in basic terms, Diamond's techniques can enable researchers to observe the shape of biological molecules at atomic resolution experimentally, rather than virtually. If you head to **diamond.ac.uk/Covid-19.html**, you can see a video of how the COVID-19 crystallography setup works.

This is just the tip of the iceberg of the work Folding@Home has done to advance medicine. As well as working on COVID-19, the team has worked on Alzheimer's disease, Huntingdon's disease, cancer, and many others. You can find out more at **foldingathome.org**.

UNLOCKING THE POWER OF ARM FOR DISTRIBUTED COMPUTING WORKLOADS

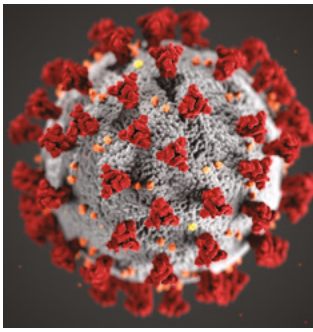
During the first half of 2020, as the COVID-19 virus began to spread, there was a concerted effort by tech industry leaders to support the efforts of Folding@Home and Rosetta@Home (a similar project to Folding@Home), the two leading distributed computing research projects. Initially, support for Arm-based devices for these projects was lacking. Recognising the opportunity to make a difference, a volunteer working group was formed by a team at Arm, working with its broader developer community and partners including Neocortix, Packet, Hivecell, and others, to perform the work necessary to bring these projects to Arm-based devices.

Traditionally, Folding@Home and Rosetta@Home have worked on desktop computers and leveraged GPUs to provide large amounts of processing power. However, the proliferation of mobile and IoT technology has caused an explosion in the availability of total compute power globally, much of it taking the form of smartphone handsets or other small devices. While more powerful desktop machines offer robust capabilities, they are outnumbered by lower-powered Arm-based machines, such as Raspberry Pi computers and mobile devices. This opportunity has led to the foundation of startups like Neocortix, who seek to offer distributed computing capabilities and platforms worldwide.

Work on Arm clients for both systems started at the beginning of 2020 with a team of experts from Neocortix, Balena, Linaro, and Arm working together to get both of these projects running on Arm machines. The Rosetta@Home version arrived first on 31 March. Since then, two Arm-based volunteer teams (crunching on Arm and Fold for COVID) have reached places 11 and 15 on the leader board (**boinc.bakerlab.org/rosetta/top_teams.php**). Not bad for teams that have only been around a few months!

Following Rosetta@Home, thanks to efforts by Neocortix, the more challenging technical implementation work to offer an Arm client for Folding@Home was completed in July 2020. At the time of writing this article, the Folding@Home client is now available for Arm devices in beta form from **foldingathome.org/beta**.

If you've got an Arm machine, why not join the effort, and together, let's get through this. ◻



Above
COVID-19 has caused suffering and chaos throughout the world in 2020

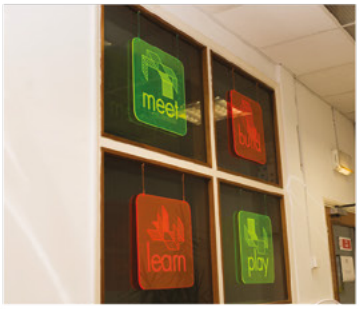
FIND OUT MORE

You can hear more about Folding@Home and how home computers are powering the future of medicine at the Arm DevSummit. Dr Greg Bowman will be speaking at 9am PDT on Tuesday 6 October. Find out more, and join for free at **hsmag.cc/fZDhrW**.

To learn more about how the COVID-19 work has been adapted to run on Arm processors such as that in Raspberry Pi, tune in to hear from Neocortix CEO Lloyd Watts at 10:50am PDT on Wednesday 7 October. Find out more and join for free at **hsmag.cc/2XoVcL**.

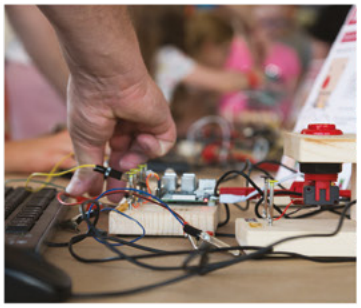










Build a Makerspace for Young People

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

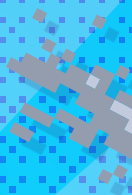
Sign up today: rpf.io/makerspace

HackSpace
TECHNOLOGY IN YOUR HANDS

LENS

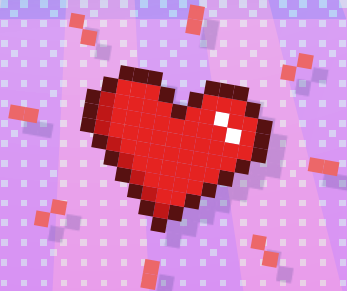
HACK | MAKE | BUILD | CREATE

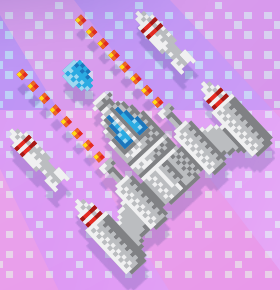
Uncover the technology that's powering the future



DIY ARCADE

BUILD A FULL-SIZED ARCADE CABINET





Games consoles might be fast and have great graphics, but for us, they're no match for the entertainment value of a proper arcade machine.

Standing up and feeding coins into the slot – each life costing real money.

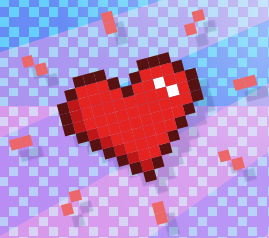
We're not alone – makers and enthusiasts around the world are recreating the gaming experience of the past with their own arcade machines. Some people prefer to find and repair old machines, but here we look at how to build your own. The advantage of building one is that you can make it just the size you want. Need it to fit in the corner of a room? No problem. Need it to fit on a tabletop? Now you can. Need to extract the children's pocket money from them 10p at a time? This is how you do it.

Of course, not everyone games in the same way, but makers around the world are finding different ways of enhancing their gaming experience. Whether that's through complete systems, or unique controllers (take a look at page 78 for our take on a pedometer-based games controller).

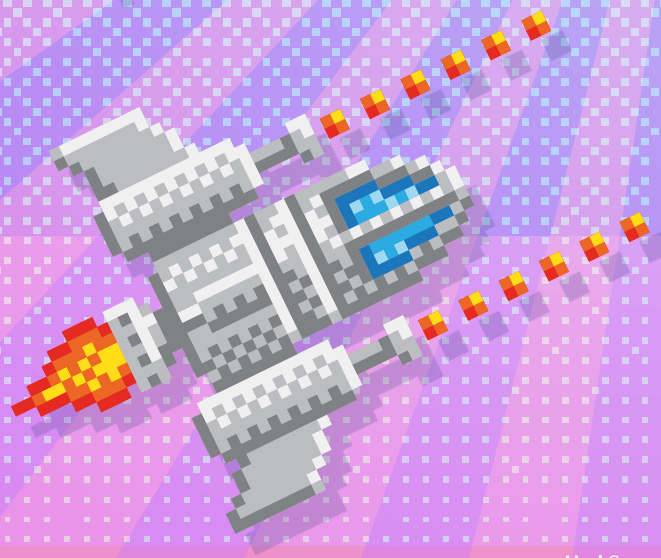
Whatever gaming you do, why not see if you can put your maker skills to use and make it even more fun? Now, join us as we travel back in time 30 years and recreate our youth. →

BUILD YOUR OWN ARCADE MACHINE

Build a full-height arcade cabinet powered by Raspberry Pi

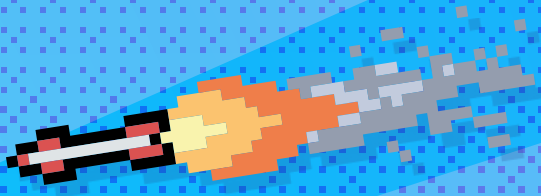


BY DR. ANDREW LEWIS



There's something special about the comforting solidity of a coin-eating video game monolith, and nothing screams retro fun like a full-sized arcade cabinet sitting in the corner of the room. Classic arcade machines can be a serious investment. Costing thousands of pounds and weighing about the same as a giant panda, they're out of reach for all but the serious collector. Thankfully, you can recreate that retro experience using modern components for a fraction of the price and weight.

An arcade cabinet is much easier to make than you might expect. It's essentially a fancy cupboard that holds a monitor, speakers, a computer, a keyboard, and some buttons. You can make your own cabinet using not much more than a couple of sheets of MDF, some clear plastic, and a few cans of spray paint.



Right
Build a full-sized arcade cabinet and play the classics the way they were meant to be experienced

If you want a really authentic-looking cabinet, you can find plenty of plans and patterns online. However, most classic cabinets are a bit bigger than you might remember, occupying almost a square metre of floor space. If you scale that down to approximately 60 cm^2, you can make an authentic-looking home arcade cabinet that won't take over the entire room, and can be cut from just two pieces of 8 × 4 (2440 mm × 1220 mm) MDF. You can download our plans from **hsmag.cc/issue35**, but these are rough plans designed for you to tweak into your own creation. A sheet of 18 mm MDF is ideal for making the body of the cabinet, and 12 mm MDF works well to fill in the front and back panels. You can use thinner sheets of wood to make a lighter cabinet, but you might find it less sturdy and more difficult to screw into.

The sides of the machine should be cut from 18 mm MDF, and will be 6 feet high. The sides need to be as close to identical as possible, so mark out the pattern for the side on one piece of 18 mm MDF, and screw the boards together to hold them while you cut. You can avoid marking the sides by placing →

Right
Begin by cutting both MDF sheets in half vertically, using a jig-saw or circular saw. This will make them much easier to handle. Then cut the largest pieces to size. If you make a mistake, you might be able to reuse larger pieces

YOU'LL NEED

- **8 × 4 sheet of 12 mm MDF**
- **8 × 4 sheet of 18 mm MDF**
- **Sheet of clear acrylic –** at least 600 mm × 900 mm × 2 mm (for screen bezel)
- **Sheet aluminium checker plate – approximately 600 mm × 600 mm** (optional for decoration)
- **2 cans of 400 ml matt black acrylic spray paint**
- **Can of 400 ml matt red acrylic spray paint** (optional for decoration)
- **Roll of cosmic space wallpaper** (optional for decoration)
- **6 metres of 18 mm chrome car detail moulding** (optional for decoration)
- **Raspberry Pi 3 or 4, and a suitable case and power supply**
- **microSD card, the larger the better**
- **USB encoder to arcade joystick with connecting wires**
- **2 × digital arcade joysticks**
- **8 × arcade buttons**
- **Electronic coin acceptor** (optional for coin acceptor)
- **12 V 1 A power supply** (optional for coin acceptor)
- **12 V relay** (optional for coin acceptor)
- **BenQ 24.5-inch FHD monitor and HDMI cable**
- **Universal TV wall mount**
- **4-gang power strip with 3 m cable**

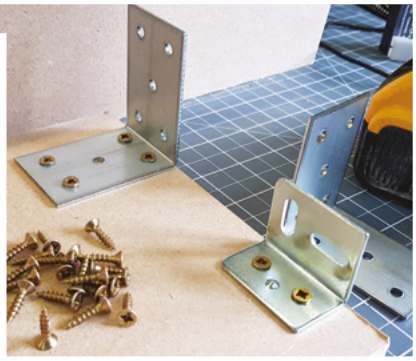
Above
Once you have the larger cabinet pieces fitted together, you can double-check your measurements by test-fitting the other panels

FOLLOW THE OLD ADAGE MEASURE TWICE AND CUT ONCE

the screws through the waste areas of the MDF. Keep these offcuts to make internal supports or brackets. You can cut the rest of the pieces of MDF using the project plans as a guide.

Attach the side pieces to the base, so that the sides hang lower than the base by an inch or two. If you're more accomplished at woodworking and want to make the strongest cabinet possible, you can use a router to joint and glue the pieces of wood together. This will make the cabinet very slightly narrower and will affect some measurements, but if you follow the old adage to measure twice and cut once, you should be fine. If you don't want to do this, you can use large angle brackets and screws to hold everything together. The cabinet will still be strong, and you'll have the added advantage that you can disassemble it in the future if necessary.

Keep attaching the 18mm MDF pieces, starting with the top piece and the rear brace. Once you



Right
Metal brackets are only as good as the screws you use to fix them in place. Try to find screws that are long enough to get most of the way through the MDF without causing dimpling on the outside. For 18mm MDF, 16mm screws work well. For 12mm MDF, you can get away with using 12mm screws, provided that they aren't countersunk and your brackets are made of a thicker metal

Left
Aluminium checker plate is a good way of protecting your cabinet from damage, and it can be cut and shaped easily. Plastic works too, but it can be brittle and is more difficult to work with

have these pieces attached, the cabinet should be sturdy enough to start adding the thinner panels. Insetting the panels by about an inch gives the cabinet that retro look, and also hides any design crimes you might have committed while cutting out the side panels.

The absolute sizing of the cabinet isn't critical unless you're trying to make an exact copy of an old machine, so don't feel too constrained by measuring things down to the millimetre. As long as the cabinet is wide enough to accept your monitor, everything else is moveable and can be adjusted to suit your needs.

MAKE IT SHINY

You can move onto decoration once the cabinet woodwork is fitted together. This is mostly down to personal preference, although it's wise to think about which parts of the case will be touched more often, and whether your colour choices will cause any problems with screen reflection. Matt black is a popular choice for arcade cabinets because it's non-reflective and any surface imperfections are less noticeable with a matt paint finish. Wallpaper or posters make a great choice for decorating the outside of the cabinet, and they are quick to apply. Just be sure to paste all the way up to the edge, and protect any areas that will be handled regularly with aluminium checker plate or plastic sheet. The edges →

PAY TO PLAY

Although there's no real need to include a coin slot on a home arcade machine, there is something that feels very authentic about putting a coin into a slot and hearing the sound of a credit being added to your game. It also makes you think twice before pushing the Continue button to carry on when you've just been blasted out of the galaxy by big guns for the third and final time. If you search online for coin acceptors, there are plenty of options to choose from and some of them aren't much more expensive than an arcade button. For a home arcade machine, the cheapest option is probably going to be OK to use.

Linking the acceptor to your arcade machine doesn't take a lot of effort, but you need to be careful connecting your wires. Some coin acceptors have a 5V logic output, but others output 12V directly. The best way to deal with this is to power your acceptor from a 12V wall supply, and use the output of the coin acceptor to trigger a mechanical or solid-state relay. Use the relay as a regular button (usually mapped to the Select button) on your arcade machine.

Most cheap coin acceptors do a simple comparison between the coin inserted and a reference coin clamped in place on the mechanism. If the coins match, then a small bumper inside the mechanism retracts and the coin falls out of the bottom of the acceptor. If the coin doesn't match, the bumper guides the coin back out of the coin reject slot.

Quick Tip

MDF generates a lot of dust – at the very least you should wear a mask and goggles, work in a well-ventilated area, and clean up dust with a vacuum regularly.

Above
Take care when cutting and peeling off the protective layer from the outside of your bezel. If you accidentally remove the wrong piece, you'll either have to use masking tape or flip over the plastic and start again on the other side

THE BEZEL SHOULD BE A PIECE OF CLEAR ACRYLIC

Below
Be gentle while sanding the bezel, especially around the edge of the protective layer. You just need to lightly scuff the surface of the plastic, so use a fine sandpaper and a light touch

of MDF sheets can be finished with iron-on worktop edging, or with the chrome detailing tape used on cars. You can buy detailing tape in 12 mm and 18 mm widths, which makes it great for finishing edges. The adhesive tape provided with the chrome edging isn't always very good, so it's worth investing in some high-strength, double-sided clear vinyl foam tape.

You've made your cabinet, but it's empty at the moment. You're going to add a Raspberry Pi, monitor, speakers, and a panel for buttons and joysticks. Positioning the monitor securely inside the cabinet is very simple, using an MDF brace with a VESA bracket bolted through it. Set the monitor at a height that's comfortable for you, and screw the brace into position so that the screen is set back about six or seven inches into the case and parallel to the front edges. Making the monitor look like it's part of the cabinet means making a custom bezel to surround the monitor. This isn't as difficult as you might think, and can be done quite easily at home. The bezel should be a piece of clear acrylic the same width as the inside of the cabinet, and high enough so that it will reach from the joystick console to somewhere near the top of the inside of the cabinet. When you finish the bezel, the plastic will be held in place by an MDF frame and some double-sided foam tape. Make the MDF frame now, so that you can measure your monitor position more accurately. The frame should be positioned so that it matches the position of the front edge of the monitor as exactly as possible, so that when the plastic is placed

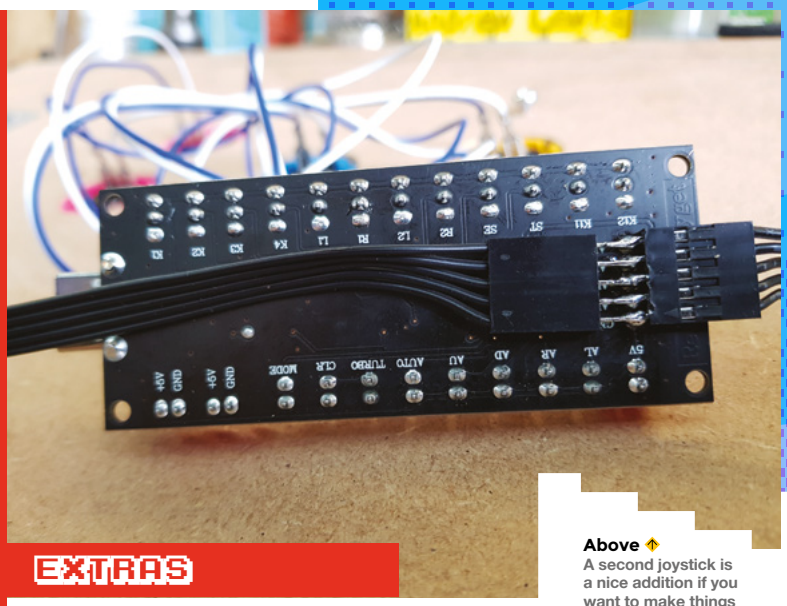
Left
Most bezels have a plain black outer, but they can be as colourful or detailed as you want them to be. It's not too hard to get a gradient or starburst effect with a little practice, or to create a star field by spatter painting white acrylic before you apply black spray paint

Quick Tip

When you're building a project, you're going to make some mistakes with your cuts, and cumulative errors will happen. Don't rely on the measurements in your plan – check the actual measurements of the project as you're going, and make adjustments if you need to.

against the monitor, the edges will rest against the MDF frame.

To make the bezel, you need to blank out the areas of the sheet of plastic that aren't in front of the screen. Set the plastic sheet in place in the cabinet, and then measure and mark the position of the monitor on the plastic sheet. Carefully cut through the protective covering on the side of the plastic sheet that will be closest to the monitor, and remove the outer part of the covering to expose the plastic. Be very careful not to scratch the plastic with your blade as you cut, and make sure that the protective covering doesn't get pulled off from the area where the monitor will be. You'll be using spray paint to colour this area of plastic, and any lifted edges on the protective sheet will make a noticeable defect in the bezel. You also need to create a slightly roughened surface to help the paint stick to the plastic, but don't want to make any deep scratches. →



Above
A second joystick is a nice addition if you want to make things easier for left-handed players. You can add a second digital joystick to your USB controller by soldering the second set of wires to the same connection as the first joystick. You can't use both sticks at the same time, but the player can choose which stick they want to use

EXTRAS

Emulation and gaming are popular hobbies, and that means there is no shortage of joysticks, buttons, and dials for your arcade cabinet. For a basic single-player cabinet, six game buttons and one joystick should be fine for most games, with extra buttons for Select, Start, and special functions. A simple digital joystick is ideal for classic emulator games, although some more modern games might be able to use analogue joysticks, too.

Fitting the buttons and joysticks into the panel is very easy with a drill and some appropriately sized Forstner bits. The size of drill will vary depending on your choice of buttons, but is usually between 25 mm and 30 mm in diameter. Most switches have a clip that will hold them in place when they're fitted, so just drill through the cabinet and insert the button into the hole. If you want to make the switches more secure, you can add a dab of hot glue to the back of the panel once the button is in place.

How you connect up the buttons to Raspberry Pi will depend on what sort of games you are going to play. There are plenty of controller boards available, and these are great for emulators where you want to use something that appears to the system as a standard game controller. You just need to plug in your choice of buttons and joysticks, and the controller board will appear as a standard USB gamepad to the system. For other games and applications that don't support a gamepad or need complex keyboard commands, it might be a better option to use an Arduino variant like the Adafruit Trinket M0 to mimic an HID keyboard or mouse. You can see a good example of how to do this here: **hsmag.cc/91zYMt**.

Quick Tip

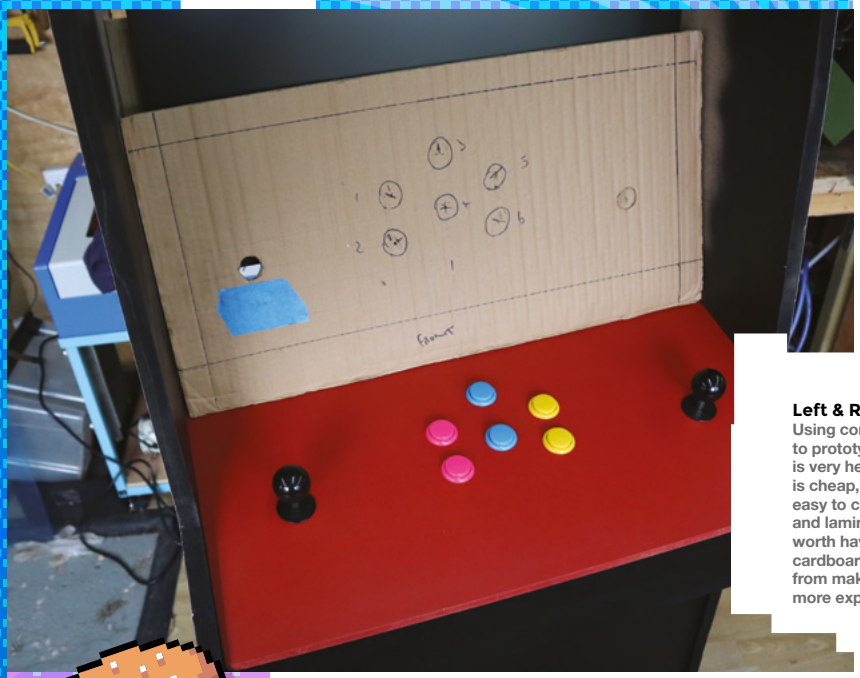
Where panels join at odd angles, it's best to cut the edge of the panels at 45 degrees so that any gaps will be hidden inside the machine.

Right
Your Raspberry Pi can be fitted anywhere in the case, but it's convenient to place it on the rear of the screen brace. Choose a suitable case, preferably with active cooling, and fix it to the screen brace with screws or foam tape. The back of the screen brace is also a good place to put a 4-gang power strip, so that you'll have somewhere to plug in your monitor, Raspberry Pi, speakers, and any other peripherals that need their own power supply. Normal powered PC speakers are fine for this project, and they can be mounted in the top of the cabinet above the monitor, or behind the monitor on the screen brace using a metal bracket

Gently sand the exposed plastic sections with 600 grit paper, and then wipe it clean with a damp sponge to remove any dust. Leave the plastic to dry, and then do a final check to make sure there's no dust on the surface before you apply the spray paint. Apply two coats of black paint, spraying in alternating directions to get an even finish. Once the paint has dried, you can peel off the remaining protective coating from the back of the bezel, clean it with a soft cloth to remove any dust, and then position it in front of the monitor. If everything fits, fix the bezel in place with foam tape.

INSERT POCKET MONEY TO CONTINUE

Joysticks and switches will mostly be located on the arcade machine's control console, although it isn't unusual to add some buttons to the front and side of the machine, or even to use foot controls. It is wise



Left & Right
Using corrugated cardboard to prototype panel designs is very helpful. Cardboard is cheap, readily available, easy to cut, and easy to glue and laminate. It's always worth having a few pieces of cardboard in stock to save you from making mistakes with more expensive materials

RETROPIE

RetroPie is a great one-stop solution for installing on your arcade machine. RetroPie is a Raspbian-based solution for setting up and managing arcade, console, and classic PC games on Raspberry Pi. RetroPie can be installed onto an existing Raspbian image, or installed directly to a microSD card as a complete image. It supports lots of emulators and controller hardware by default, and has good documentation online. You can get RetroPie at **retropie.org.uk**.

Below
If you're adding a coin acceptor to your machine, you need to think about what happens to the coin when it comes out of the bottom of the acceptor. You could simply let it fall into a bucket or a metal cash box at the bottom of the machine, or (if you don't need a giant money box in your house) you could make a simple channel to redirect the coin back out of the front of the machine, as shown in this image

to use a cardboard template to figure out the best positioning for joysticks and switches before you drill any holes. Start with a piece of cardboard the same size as your cabinet's control console, and lay the components out on the cardboard, moving them around until they feel comfortable to use. Transfer the positions from the cardboard to the MDF control console and then cut the necessary holes.

You should have everything you need now to start playing games. Make sure that you've connected the Raspberry Pi to the monitor, speakers, and game controllers. Also check that you've connected power to the peripherals, and you've connected a wireless keyboard and mouse to the Raspberry Pi. Install an operating system (RetroPie is a good choice) onto your microSD card and get ready to start playing! →

Quick Tip

Coin acceptors typically work on 12 volts, and wiring them incorrectly can damage your controller board and Raspberry Pi.

OTHER GAMES BUILDS

From mini-machines to custom controllers, there's a world of other options

Not ready to commit to building a full arcade cabinet? Don't worry: there are plenty of other options for the space-constrained maker. These are a few of our favourites, but there are literally hundreds of other options, so dive in and find something that tingles your maker instincts.

RASPBERRY PI HANDHELD

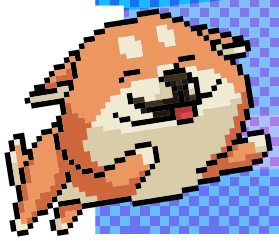
We've looked at massive gaming machines in this feature, but the brains behind them – a Raspberry Pi board – are quite small. You can go the other way and build your own portable Raspberry Pi-based games machine. You can build one from scratch, or make use of a huge range of different cases that are on the market.

PICADE

The Picade is an arcade machine that's been sliced in half and sent through a shrink-ray. It comes in versions with eight- and ten-inch screens that fit neatly onto a tabletop or bar counter. It combines this screen with a Raspberry Pi and a set of arcade controls, meaning that it's a complete gaming system, just a little smaller than its full-sized brethren. It's a perfect project for anyone who wants to build their arcade machine but who can't quite find the space to make it fit.

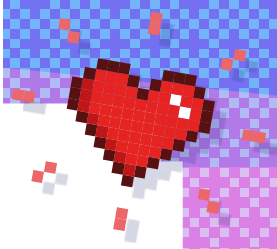
ESP32

The ESP32 microcontroller is in slightly unusual territory. It's not particularly resource-constrained like the ATmega chip in the Arduboy, but at the same time, it's not overflowing with processor power and memory like a Raspberry Pi. It's got a decent chunk of memory and processor power and is even capable of running some emulators (such as NES: **hsmag.cc/N1NOuR**). There are several projects to help you out on the software side of things if you want to build your own ESP32-based controller, such as the FabGL (**fabglib.org**) graphics library.



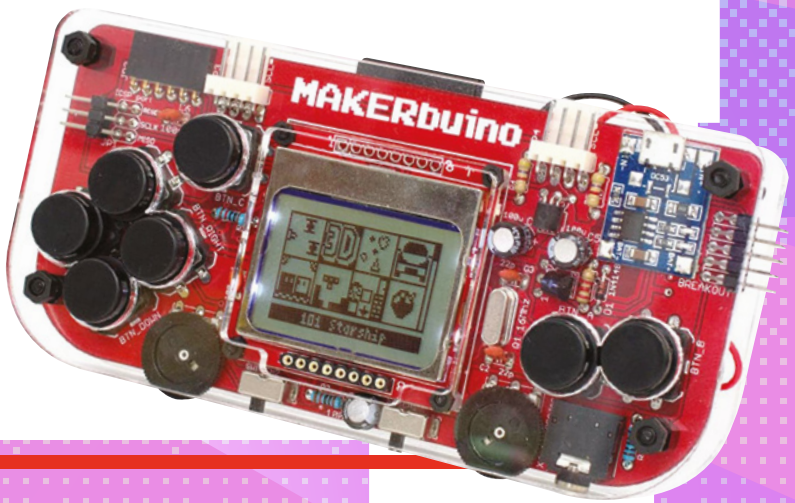
CONTROLLERS

One of our favourite ways of using our maker skills to add a little more entertainment to games is building DIY controllers. There are USB and Bluetooth libraries for Arduino and CircuitPython (among others) that make it easy to build your own games controller that's more entertaining than the classic thumbpad. For example, take a look at our pedometer watch games controller on page 78.



ARDUBOY

There are plenty of games frameworks for Arduino, but none has taken off as well as the Arduboy. This tiny console is great for people who like working with constrained environments. It's based on the ATmega32U4 8-bit microcontroller with 2.5kB of RAM. There are many other handhelds that take the same hardware into slightly different form factors. Perhaps the most popular of these is the MAKERbuino, which is a bit bigger and assembled as a kit (the Arduboy comes premade).



SUBSCRIBE TODAY

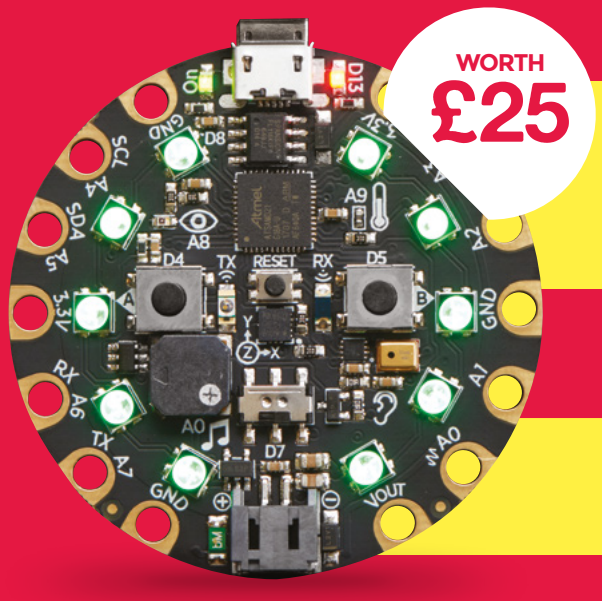
Subscribe for 12 months | **Rolling monthly subscription**

£55 (UK) **£90** (USA)

£80 (EU) **£90** (Rest of World)

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

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



FREE!

Adafruit Circuit Playground Express

With your first 12-month print subscription

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

SUBSCRIBE on app stores

From £2.29

Available on the App Store

GET IT ON Google Play

Buy now: hsmag.cc/subscribe

FEATURE

How I Made

AN ALEATORIC DRUM MACHINE

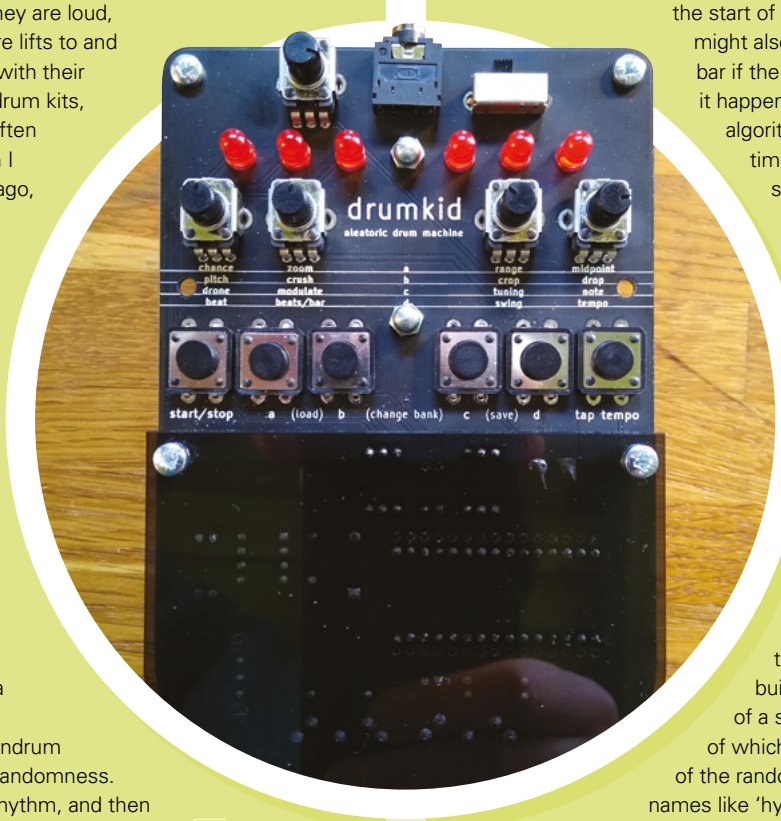
How one maker ditched the drummer

By **Matt Bradshaw**

Drummers are difficult people. They are loud, they require lifts to and from gigs with their unwieldy drum kits, and they often play clichéd beats. So when I started a band a few years ago, I decided to design my own drum machine instead.

While I wanted to avoid recruiting a real drummer, I also wanted to avoid the sound of a dull, robotic drum machine. Simply hitting 'start' at the beginning of a song and 'stop' at the end is liable to make for a flat live experience, and even if you program a drum machine to follow a song's dynamics exactly, there is still a sense that you might as well be playing along to a backing track.

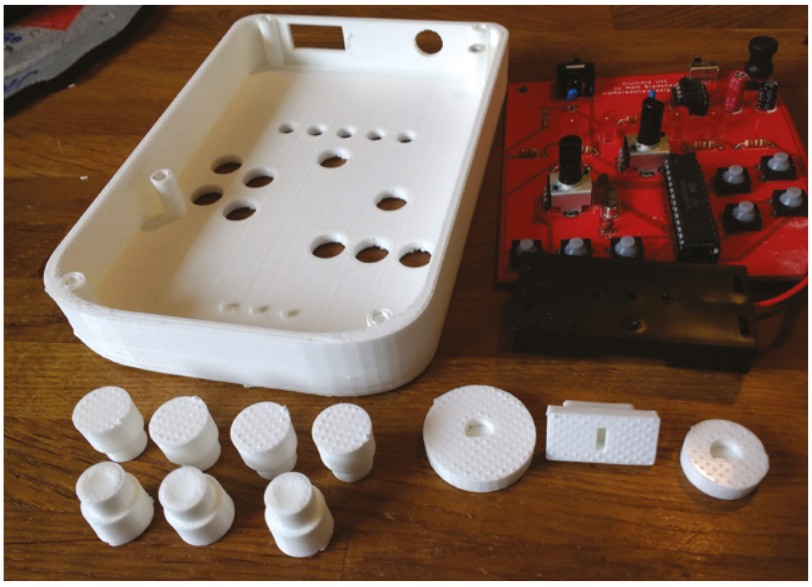
My way around this conundrum was to base my design on randomness. I would start with a vanilla rhythm, and then allow the user (in this case our bassist) to decide how much randomness to feed into the beat. For instance, while a normal rock rhythm might only feature the bass drum at the start of the bar, an extra bass drum hit might also be added near the end of the bar if the random number controlling it happened to be high enough. The algorithm I used changed over time, but suffice it to say that it sounded good even in early versions, and we ended up basing our band's sound around the machine, which we called 'DrumKid'.



Above
A finished DrumKid

ORIGIN STORY

This early version of DrumKid was actually a web app running on an iPad which I borrowed from my office. I was working as a web developer, so JavaScript was my strongest language, and therefore the simplest way to build my idea. The app consisted of a series of large sliders, each of which controlled some element of the randomness. The controls had names like 'hyperactivity' (how likely the virtual drummer was to hit a particular drum) and 'sloppiness' (the chance of getting the timing wrong for a given hit), as well as a



Left
The Nintendo Gameboy provided inspiration for an early form factor

few standard audio effects like reverb, delay, and filter.

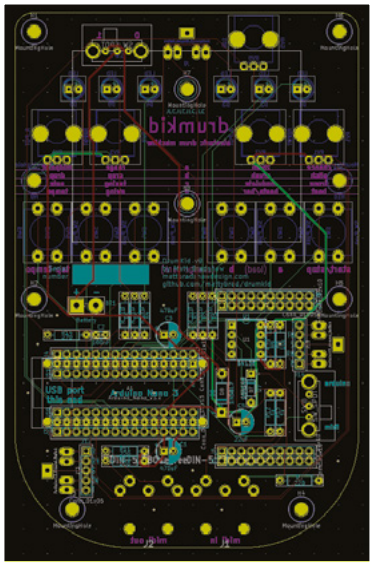
The iPad version sounded great and was intuitive to play, but it had some drawbacks. Firstly, bringing an expensive, breakable, nickable device to a gig is always slightly worrying, but it was also somewhat unreliable. During one particularly stressful sound check, other bands tutted as DrumKid refused to boot up, until we eventually gained access to the venue's WiFi network.

That episode made me resolve to build a more robust version of DrumKid that wouldn't crash every time Safari updated itself, although the real impetus came a couple of years later when the 2019 Hackaday Prize was launched. Its theme was 'design for manufacture', which sparked an idea – I had been looking to transition my career from web development to 'making' of some sort, and I realised that this would be a great excuse to work on a version of DrumKid that I could actually sell. The prize also gave me a deadline of around four months to finalise my entry, which was good motivation.

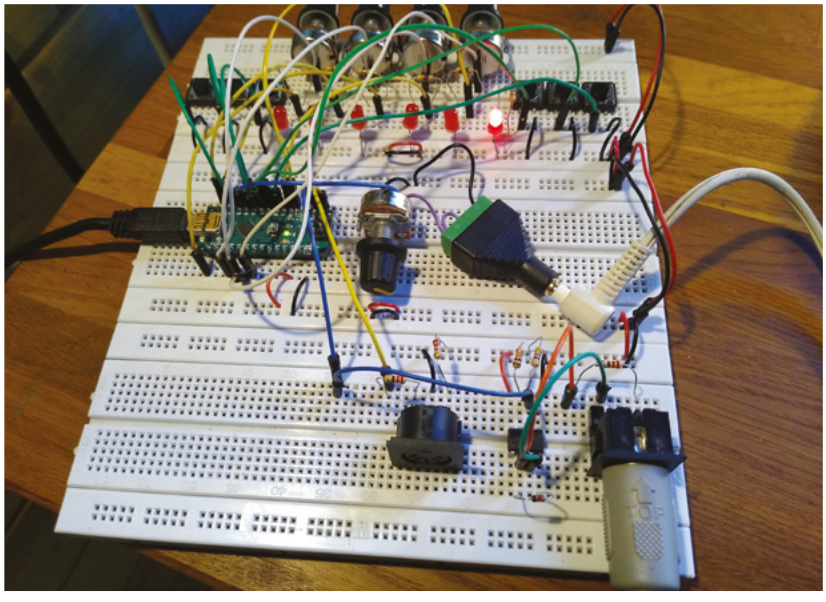
BEYOND HACKING

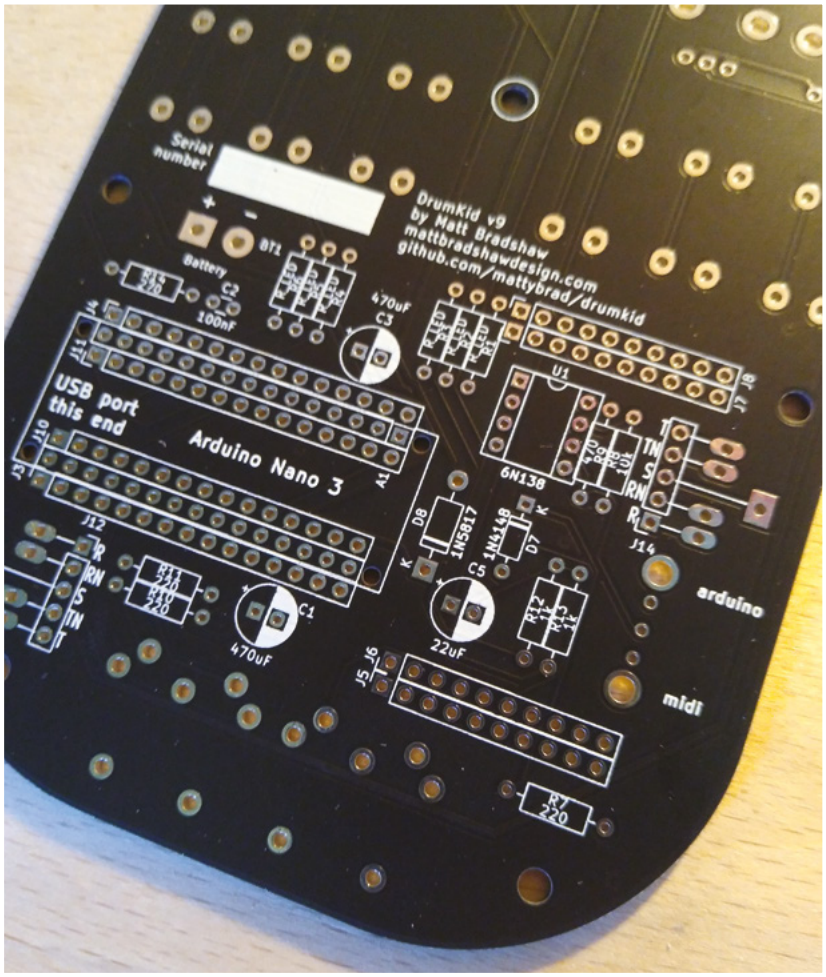
Like many makers, my shelves are littered with half-finished projects, or builds that worked once but have since been scavenged for components. My favourite part of the making process has always been simply proving that the idea could work, often on a breadboard or inside a hastily constructed wooden box. The concept of designing something that could be reliably duplicated was anathema to me, but I resolved to change my mindset and view this as an inherent (and interesting) part of the project.

Below
I made use of the 'flip board view' function in KiCad to add components on both sides of the PCB



Below
A breadboard layout allowed for easy development





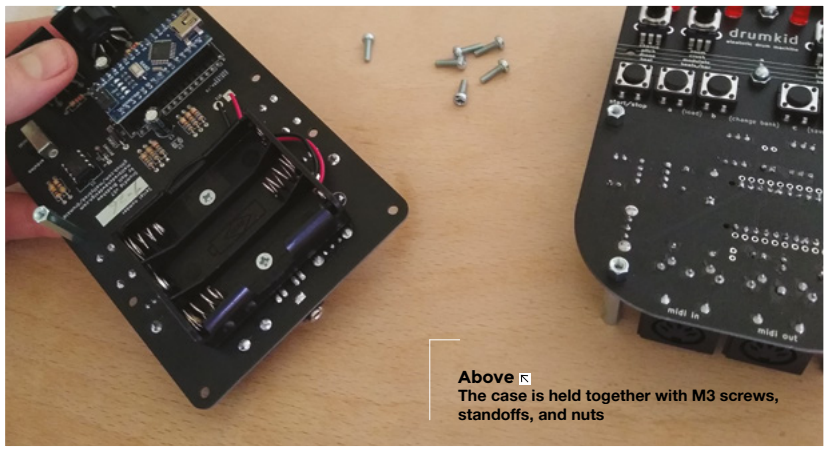
Right
The latest PCB has extra space for custom user hacks

I began by writing a specification, where I listed the features that I wanted DrumKid to have. I knew I wanted it to be handheld, battery-powered, cheap, robust, and easy to play. I knew I wanted the majority of the iPad version's features to be implemented, although I was aware that I might have to compromise on audio quality. I also knew that I wanted the design to be open-source, repairable, and hackable. Armed with these requirements, I started sketching ideas and experimenting with breadboard circuits.

I quickly settled on using the Arduino platform as the main 'brain' of the design. Arduino microcontrollers are usually seen as not being powerful enough for audio, but there is a great little code library called 'Mozzi' which allows you to squeeze a surprising amount of audio from even low-end Arduino boards, so I used that.

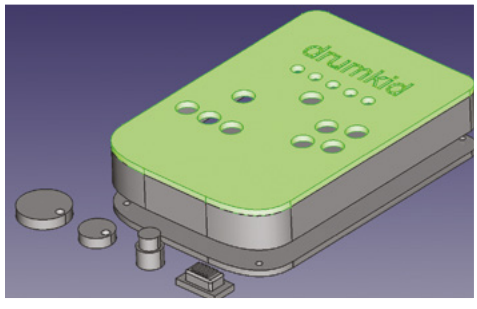
HONEY, I SHRUNK THE CODE

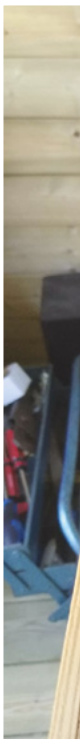
I gradually ported each feature from DrumKid's JavaScript code to its new Arduino code, testing it at each stage with an Arduino Uno. While an official Uno costs around £20, you can buy the basic components (an ATmega328P chip, a crystal, some capacitors, and a resistor) for much less, so I initially based my design around this 'DIY Arduino' circuit.



Above
The case is held together with M3 screws, standoffs, and nuts

Below
FreeCAD is great for designing 3D-printed prototypes





Above
There were many prototypes en route to the final version

Left
After a while I developed an efficient home factory

Although the basic circuit and code were working well at this point, I knew that I would need to learn some new skills before I could build something that could be reliably manufactured. I began to teach myself printed circuit board (PCB) design using Shawn Hymel's excellent YouTube series, *An Intro to KiCad*. KiCad is a free program in which you first create a schematic of your circuit, then use it to lay out a PCB. Once you have chosen where each component should be placed on the board, you go through a process known as 'routing', where you draw copper traces (either on the top or bottom of the board) to connect your components to each other.

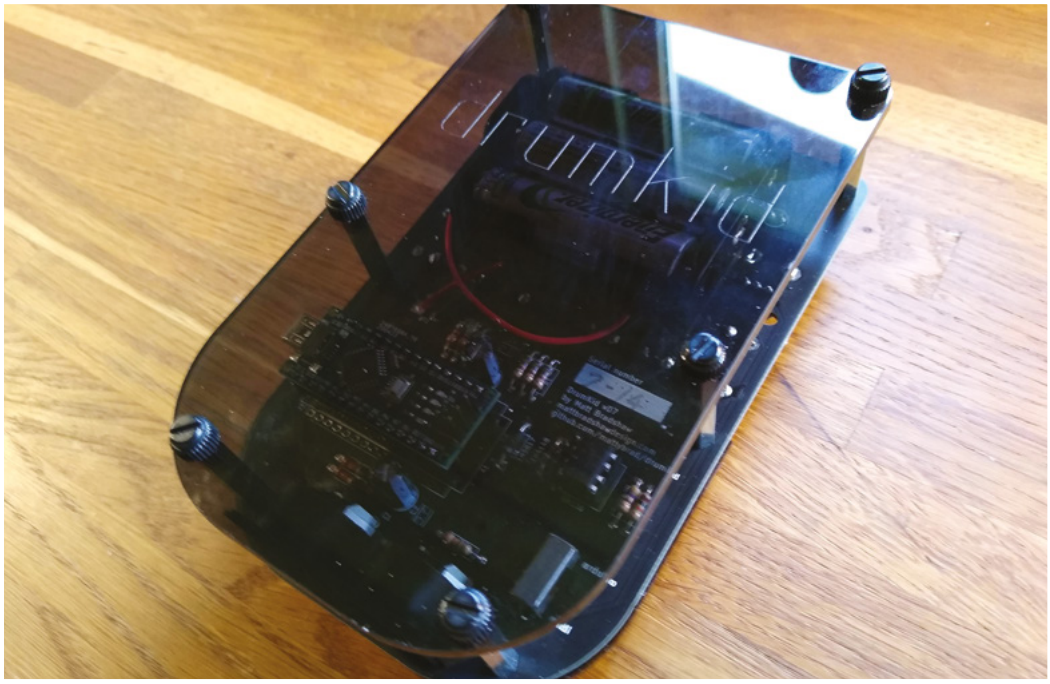
Once my PCB design was done, I exported it as a set of 'Gerber' files and ordered a test-run of five boards from JLCPCB in China. Ordering small batches of simple boards from China can be very cheap (especially if you choose a slower shipping option). I haven't yet found a viable alternative in the UK, although OSH Park seems to be a good option in the US.

LITTLE BY LITTLE

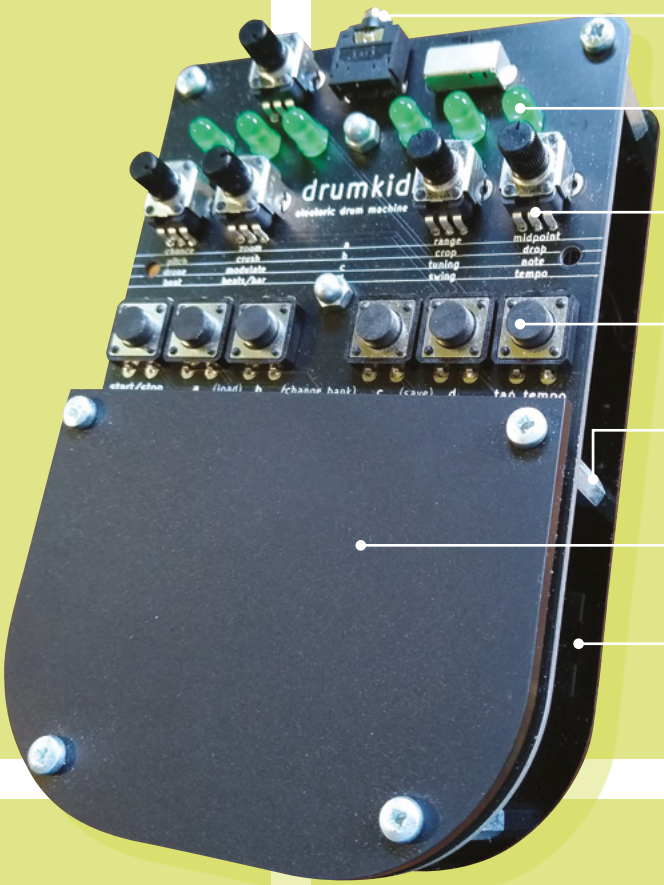
It took me a few iterations of the PCB to settle on a good design. I moved from two batteries to three so that I wouldn't have to boost the voltage, and spent quite a bit of effort on filtering noise from the signal. I hadn't previously encountered the concept of a decoupling capacitor, but had to become very familiar with them in order to produce a nice audio output. I also made various stupid mistakes, such as forgetting to check for short circuits in KiCad before exporting (resulting in an unusable batch of PCBs) and somehow adding the word 'DrumKid' in huge copper letters as part of the circuit (a mistake so obvious that the factory noticed it during their checks, allowing me to fix it before production).

While working on the circuit board, I was also trying to figure out the enclosure.

FEATURE



Right
The reverse of the finished product



Audio output

LEDs for visual feedback

Potentiometers to adjust values

Buttons to switch between modes

Metal standoffs (open-sided case)

Laser-cut palm rest

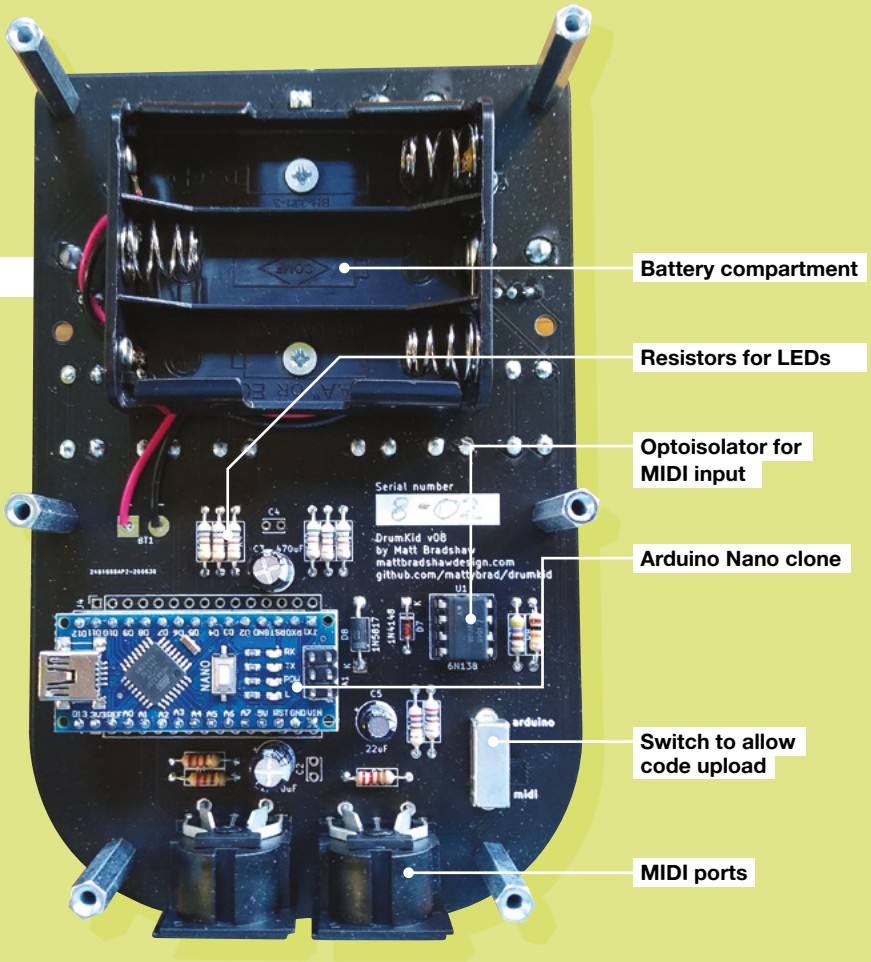
Laser-cut rear cover

I realised quite early on that 3D-printing would be impractical for the final design, but I decided to print a case anyway, just so that I had a starting point. The aim was to evoke a Game Boy, with LEDs in place of the display and a set of controls for each thumb. The 3D-printed prototype felt comfortable to hold, but was awkward to play because I had decided to only include one potentiometer, so you had to press a lot of buttons before you could adjust the desired parameter.

After some experimentation, I settled on a design with four potentiometers, which made DrumKid a lot easier to play. I also chose to employ an open-sided design, with the circuit board itself as the instrument's front panel, laser-cut pieces to protect the delicate components, and metal standoffs to keep the front and back sections apart. I didn't have ready access to a laser cutter, but was contacted by an extremely helpful maker on Twitter who cut the pieces I needed and sent them to me in the post.

THE FINAL PUSH

By this stage I had a single, fully working DrumKid, and reasonable confidence that I could build more. I made another three identical DrumKids, this time using plywood parts from a local maker who had a laser cutter in his shed. I gave each DrumKid to



What is an 'aleatoric' drum machine?

The word 'aleatoric' means 'characterised by chance'. There are many examples of music which uses chance as part of the composition process. John Cage notably used the *I Ching*, a Chinese divination text, to help him compose the piano piece *Music of Changes* in 1951, allowing the book to make decisions about tempo, dynamics, and other parameters.

DrumKid can be called aleatoric because drum hits are either added to or removed from a beat according to chance, and the volume of each hit is also determined by chance. Turning DrumKid's knobs alters the flow of random numbers into a beat, creating rhythms that change constantly but are still under the control of the player.

Computers actually find it difficult to produce random numbers (i.e. chance) on their own, so one of the first lines of DrumKid's code initialises the random number generator using a truly random value read from an unconnected (floating) pin on the microcontroller.

a local musician, so that I could get some feedback before going into production. Around this time, the results of the Hackaday Prize were announced – DrumKid reached the finals and received an 'honourable mention' award for best production, which was really exciting. I was able to use the money from the award to stock up on components for the first proper batch of DrumKids.

Armed with both new-found enthusiasm and helpful user feedback, I spent the next few months refining the design. I added MIDI input and output ports, which allowed DrumKid to control or synchronise with other instruments. I also switched to using an Arduino Nano clone, which meant that users would be able to update their own firmware via USB. This, along with the breakout pins on the circuit board and the open-source schematics and code, was intended to make DrumKid as 'hackable' as possible. While I was aiming to make an instrument that could be easily played straight out of the box, I was also hoping that hackers would modify their units, upload their own samples, and hopefully even augment the hardware.

I was also hoping that hackers would modify their units, upload their own samples, and hopefully even augment the hardware

RELEASED INTO THE WILD

Eventually, a year after I started designing the first non-iPad DrumKid, I had a stack of 20 identical DrumKids in my shed, and concluded that I was ready to begin selling them. A couple of months later, there are now about 70 DrumKids dotted around the world, in about 15 different countries, which is quite a change from building one-off hacks.

Although DrumKid's outward appearance hasn't changed much since launch, I have gradually improved the design. The shiny plastic parts were impossible to keep clear of fingerprints, so I switched to a matt finish, and realised I still had space on the circuit board, so I added more breakout pins and spaces for extra audio jacks, to encourage hacking.

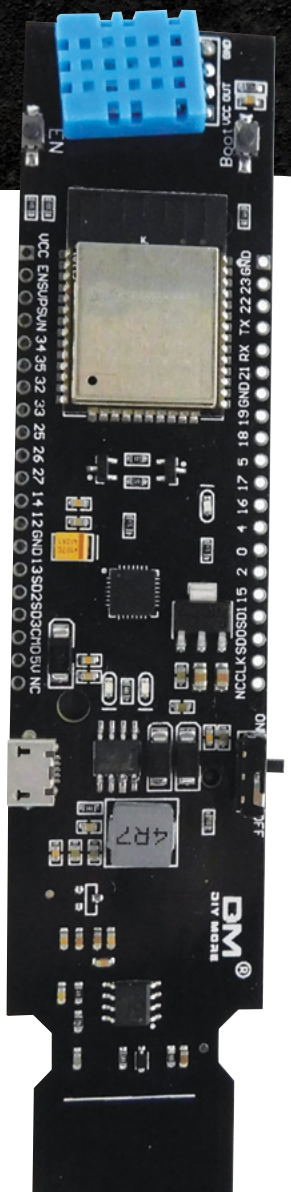
I've learnt a lot from this project. I'm hoping it is the first of many instruments that I design for other people to play, but I think that the experience of designing for others will also affect the way I build my own personal projects in the future. If you would like to build a DrumKid, the breadboard layout is available at **hsmag.cc/x1QhpR** (or you can buy a ready-made one from **mattbradshawdesign.com**). □

FEATURE

IN THE WORKSHOP: Soil moisture

By **Ben Everard**

Preparing the garden for a perfect growing season next year



Left
The ESP32 module and DHT11 are mounted together on the top half of the PCB

This summer has certainly been funny weatherwise. Blazing hot in spring, then a bit damp throughout June and July with occasional blasts of heat. That means it's been a bit difficult to get into a routine of when to water. On busy days, it's easy to forget until you suddenly see a wilted green mess where your vegetables used to be. Yes, it's a bit late in the year to be worrying about this, but now is the time to get things up and running ready for next year.

There's a range of soil moisture probes around with different features. The cheapest and most common have two prongs that test the resistance of the soil, but these have the disadvantage that they gradually break down due to electrolysis. The other option uses capacitive sensing to read the soil moisture – these are slightly more expensive, but should last much longer. This is the option that I went with.

As well as a sensor, we need a controller, and for almost anything wireless, the ESP32 is my controller of choice at the moment. As well as having WiFi, it's got support for deep sleep which should mean that I can get a long run time (I'll be running it by battery in the garden).

It wouldn't be too much work to build a circuit from scratch, but to make matters even easier, a few companies make boards that combine everything into a single PCB. I opted for an ESP32 soil moisture meter from DIY More. These are available on direct-from-China websites for around £10, not including the 18650 Li-ion battery, which come in at about £5 for two 3600 mAh capacity batteries. This module also includes a DHT11 temperature and humidity sensor. These aren't particularly accurate, but as it's included, I decided I may as well monitor the output.

A key part of the puzzle for this project is energy efficiency. After all, there's no point in this if I just replace watering every day with recharging the batteries every day. There's a whole bunch of techniques that you can use to reduce power consumption on the ESP32 such as reducing the clock speed and minimising the WiFi usage, but I decided to focus my efforts on just one – deep sleep.

You can see the complete code for this project at: **hsmag.cc/VmEzta**.

On the ESP32, deep sleep isn't really like putting your computer to sleep; it's more akin to turning it off and back on again. Everything stored in RAM is lost (there are ways around this for small amounts of data), and the sketch starts from the beginning again. For our purposes, this isn't a problem as we just want to read the sensor values and send them off to the internet. When it's in deep sleep, it uses very little

power (around 10 microamps), so we don't need to worry too much about this level of current consumption when we have 3600 mAh to play with. When the unit is taking readings, it's a short amount of time so power usage isn't a problem.

Putting an ESP32 into deep sleep is fairly straight forward in Arduino. First, you need to set a wakeup source with:

```
esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_
TO_S_FACTOR);
```

This is done with two defined values: the amount to sleep in seconds, and the multiplication factor to take this into microseconds, which is what the ESP32 uses.

When we want to go into deep sleep, we then use:

```
esp_deep_sleep_start();
```

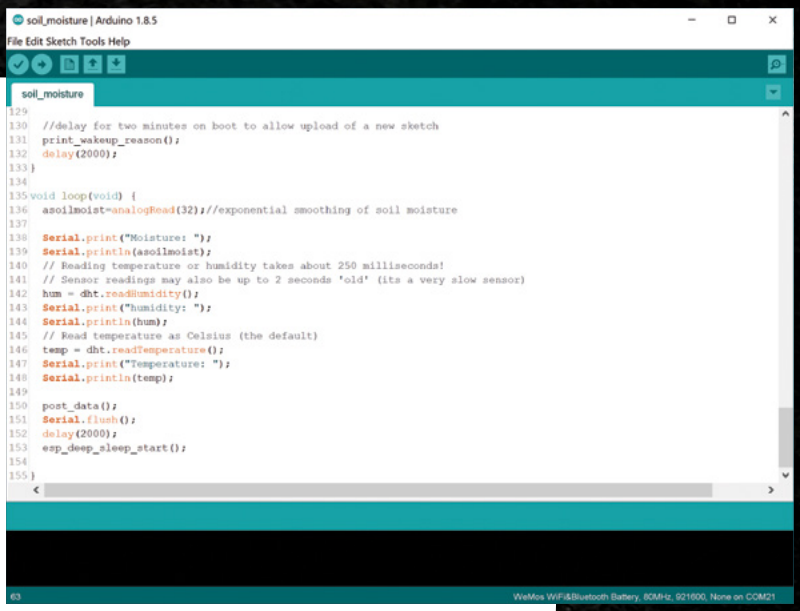
In principle, that should be enough, but for some reason, I couldn't persuade my device to sleep for more than 25 minutes. No matter what value I put in the `TIME_TO_SLEEP` definition, the board would wake up after a maximum of 25 minutes. After searching for a solution for a while, I decided to go with the simplest hacky solution, and that is to count how many wake-ups there have been and only read the sensors / do some processing after a certain number of wake-ups. In other cases, I'll just shut the device down again.

This does cause a slight problem because variables aren't held between boots, so how can we know how many times it's booted? The solution is in the real-time clock (RTC). This is the part of the ESP32 that wakes up the main core after a certain amount of time. There's a small amount of memory on the RTC that we can use to store bits of information like this.

In the Arduino language, we can declare a variable that will be held in the RTC memory with the following:

```
RTC_DATA_ATTR int bootCount = 0;
```

Then, we can reboot the machine every time we don't need it with the following at the very start of the setup:



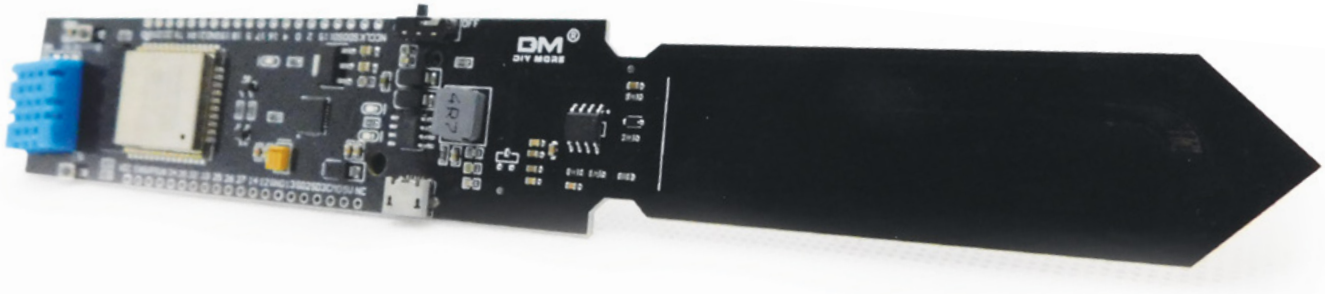

```
++bootCount;
  sleepus = TIME_TO_SLEEP * uS_TO_S_FACTOR;
  esp_sleep_enable_timer_wakeup(sleepus);
if(bootCount !=3) { esp_deep_sleep_start(); }

bootCount=0; // this will only run on the 3rd
boot.
```

This is a pretty hacky solution, but it did give me a chance to play with the RTC variables. If you can spot what the bug is in my code to make it necessary, please get in touch.

How long will it last on a battery? We've no idea! The current consumption of the unit is very low (under 1 mAh per hour), but the batteries aren't necessarily their stated capacity, and even if they are, they won't necessarily respond particularly well to this style of use. The one feature that this board doesn't have, that we'd like, is a way of checking the battery level. It'll just suddenly stop working at some point. Hopefully, it'll take several weeks (or ideally a few months) before I find out how long it lasts. The final part of the puzzle will be calibrating the moisture level. That will just be a case of seeing what readings it gets at different levels of moisture, and seeing what corresponds to unacceptably dry. ◘

Above
The deep sleep option on the ESP32 is great for getting large uptimes from battery-powered devices

Below
The pointy end has the capacitive sensor and gets stuck in the ground



HackSpace magazine meets...

Robin Baumgarten

Let there be light – lots of light!

Robin Baumgarten is a great example of a person who makes unique things. His job wasn't advertised anywhere; instead, he's making a living out of creating beautiful things and seeing what happens. And unlike, say, Picasso or Michelangelo, he's doing it now, with LEDs, Arduinos, breadboards, jumper cables, and custom-designed PCBs.

The idea of getting paid for doing exactly what you like has always fascinated us, so we spoke to Robin to find out how the magic happens, what he's working on now, and what it is about springs that humans find so appealing. →

Right
Robin exploring quantum computing at the University of Turku, Finland

HackSpace Let's start at the beginning. How did you embark on a career making things that people play with?

Robin Baumgarten Games were always a passion of mine, and when you're a programmer, making your own games is something that you can do easily. When I was at university for my masters in London, I made an AI for a game that existed already – do you know Introversion Software? They made Prison Architect, and before that, they made a game called DEFCON. I made an AI for DEFCON. As my masters project, they gave me the source code for the game, and I hacked a new AI into it. That was really good fun. I didn't dabble too much with AI afterwards, but I really enjoyed making things for games. That was the starting point of me getting into games. I started making mobile games, really tiny experiments, and of course, some were more successful than others.

I experimented with Unity as everyone does; the common games engine. And it's a very small step from there to experimental hardware. You can connect an Arduino to Unity fairly easily; it's just a serial connection. Then let's use some sensors as the input for the game – that's very easy to do. It's a great project for game jams, which are similar to hackathons but focus on games.

Game jams are also zero risk. Even if you don't come up with anything, you still have fun with your friends at the weekend; there's nothing lost.

I did a lot of those for a while, and Line Wobbler was one of the results from the game jams. It was the first one where people said: 'This is really good, you should develop this further'. So I submitted it to a lot of games festivals. Even the biggest games conference in the world, GDC in San Francisco, they have a section for alternative controllers, for people making things with Arduinos and Raspberry Pis. And almost none of them are commercial; it's students, hobbyists, tinkers experimenting. It's like a tiny Maker Faire within the games conference. There are these massive titles with big expensive stands, and then you go one hall further, and there are these tiny little stands with people showing off controllers made out of cardboard and aluminium foil. When you move the controller one way, the aluminium slides and makes a contact, and a little spaceship floating around a screen turns left or right. It's so much fun; for me, it's the best part of the conference.

Because the conference is so big, there are all these representatives from Sony, Bandai Namco walking around, so it's possible to make contact with big arcade companies as well.

When I showed Line Wobbler at GDC, I spoke to someone from Bandai Namco, and we had this back-and-forth about making an arcade machine out of this one-dimensional LED game.

In the end, I think they were a bit risk-averse… they didn't say 'no' outright, but they did say 'maybe not'. I visited their headquarters in Tokyo as part of that, but nothing came out of it.

> "Game jams are also zero risk. Even if you don't come up with anything, you still have fun with your friends"

HS These creations look like they'd be more at home in an art gallery than a games arcade.

RB Well, museums are also getting more into showing experimental games. I showed Line Wobbler at the V&A at an exhibition one or two years ago, in London and Dundee, and I think it should have moved to Australia by now. I think museums are very keen to show experimental stuff, because video games you can play at home; you wouldn't go to a museum just to play Pac-Man. But this experimental stuff is much more interesting to museums, because you need to see it in person to really appreciate how weird it is and how it works.

I think that's a niche within a niche that works, because it's not something that a lot of people are doing. It's hard to mass-produce, almost impossible. But for me, as an independent person, it works well enough to sustain me.

By contrast, mobile games are almost impossible to succeed at, because everyone's doing it. There are hundreds of games released every day and big companies [are] throwing advertising money around, so [for] a standalone person, it's very hard to make a living out of it. With alternative hardware games, there's almost no competition, but the market is much weirder and smaller.

It's kind of a trade-off, but for me, it's one that has been working so far. I've been building on the success of Line Wobbler and thinking about what comes next. I enjoy working with springs and LEDs, so I thought, 'What else can I do? I've made a one-dimensional game already, so what happens if I make a two-dimensional [game]?' In one way it's not quite as exciting, because screens are two-dimensional. So I made Wobble Garden; it's basically a field of 228 springs, each surrounded by a ring of 16 LEDs. I liked the way the light reflected off the prints and the patterns it made, and that was basically the birth of Wobble Garden – without any immediate goals, I'd made a minigame, a variant of Twister where you had to touch the right springs, but none of them were as captivating as Line Wobbler was. At first, I was a bit down about it – I can't make a compelling game out of this. But then I thought that despite it not being a compelling game, I realised that it looked cool it was. So, that's why I started to go in that direction. →

Above
Robin's current work in progress, Wobble Sphere

Above
Just touching the springs is enough to trigger the animations, but as humans we love anything that goes 'boing'

HS It's funny what you said about the interactive hardware controllers being like a Maker Faire within the games conference. That's what the magazine is all about. What sort of things do you use when you're prototyping?

RB I mostly use Teensy at the moment; I love those. Line Wobbler was standalone, so it was a Teensy controlling everything, connecting to the LEDs and the sound and accelerometer. But Wobble Garden uses a PC in the background. I have Python on there for the visualisations, and I push them out to an Arduino that does the rendering and gets the touch data back to the PC.

For that one, I also used a Raspberry Pi for a while. Right now I'm using a PC because I was too lazy to optimise the code properly, and I needed the raw power to get the frame rate back up. But there's nothing to stop a Raspberry Pi from working, especially if it's a smaller version. I made a version with 36 springs, and that worked fine on a Raspberry Pi – nice and smooth. But now the bigger version with [over] 200 springs, I set myself a goal of having at least 100 frames per second, because I really like the smoothness of the animations. I want to keep it super-smooth, surprisingly nice animations. A lot of LED projects I've seen are very slow and jittery, and I don't enjoy that at all. I wanted smoothness and low-latency feedback. Maybe that's a legacy from my game developer days, where 60 fps is what I want, and low lag is what you need when you play a classic shooter. I want to get that back.

It's going away a bit, especially with mobile phone games; when you touch your mobile phone. There is, I think, a 100 ms delay, which our brain gets rid of efficiently, but it does feel different if the delay comes down to 5 ms or something. So there's this technical component that I've tried to optimise as much as I can.

There's not too much else in terms of hardware. I decided to make my own circuit boards, but that's fairly easy once you get started. I enjoy the puzzle aspect of it as well, routing all the traces.

HS Is there an accelerometer on each one of the springs?

RB No. Only Line Wobbler had an accelerometer – it's like a joystick. Wobble Garden and Wobble Sphere are touch-sensitive, so the direction doesn't matter anymore; they're basically just buttons. Otherwise there would be far too many wires and far too much data. The advantage is also that, because it's capacitive touch, when you touch the metal it makes a contact. So I don't need to have any electronics inside the springs; they're basically like wires transmitting the touch signal.

> "I made a version with 36 springs, and that worked fine on a Raspberry Pi – nice and smooth"

On the circuit board I have a contact, so the spring sits on a contact, and from there I measure the capacitance. How that works is that a capacitive touch chip charges the spring with a little current. If you touch it, it gets conducted over your skin, and if you don't touch it, it stays there.

I don't need to have any electronics inside the spring, which makes it very robust. Even if kids pull on it – even if the spring is pulled entirely off, it would work. I mean, it would look ugly, and you still need to maintain it, but it shouldn't break anything, which is important when you're designing for exhibitions.

In a way, it's not as elegant from a conceptual perspective because the springs don't do anything – it doesn't matter if you wobble the spring or not, but I noticed that when people are interacting with it that it doesn't make a difference to them. They like to wobble them anyway, and when they wobble, they make a touch contact. It invites a greater variety of interaction; some people just tap it, but other people really wobble it quite hard. From an engineering perspective, I don't need to worry too much about wires or cabling breaking – that was a nightmare with Line Wobbler before that.

HS What are you working on now?

RB My latest project, Wobble Sphere – it's a polyhedron with 72 faces, and each of the faces is a custom-built circuit board that's exactly the right size.

The idea came from Wobble Garden. A friend had a 360-degree camera, and it has an effect called miniature planet, which distorts the image. We used it to take a photo of Wobble Garden, and it looks like a ball – like a small planet. It looked cool. I wanted to build one in real life. The next problem was how do I build a round PCB, which is almost impossible. So I thought I'd make it tiled, like a football with pentagons and hexagons. I found a website that describes a lot of the shapes really well and has all the dimensions already worked out for you.

I decided to make this one; it only had two different versions of the PCB – a pentagon and a hexagon. I exported the design from Inkscape into my PCB software as my board outline, and that's all I needed to do to get the shape exactly right. Then there was a bit more work to get the LED signals going across the boards without having too many wires, but that wasn't too difficult. And then the next step was to figure out exactly how it was going to stay together.

The springs helped me out here: I'm using these door stopper springs that are usually screwed to the wall. Each PCB has a hole in the middle for the spring to go through. For Wobble Garden, I mounted these springs onto wood, but for Wobble Sphere, I 3D-printed holders so →

that I could get the angles exactly right. It's basically angled screw holders; they hold the spring in exactly the right way.

Once it's screwed together, it's surprisingly solid – I think it's because it's a sphere, so it has a natural strength, like the way an egg-shell is really hard to compress. So it's really solid, with a 3D-printed skeleton on the inside and an Arduino with a power line going out.

A lot of the technical details are very similar to Wobble Garden – the touch controls, for example, were already made. I was just figuring out the topology of the whole thing: how do I address LEDs in a 3D space? I had the software already for the flat version, so I just needed to figure out how this would change on a sphere. The main difference is that, on a sphere, the distance between pixels isn't just a straight line. But there are forums for that, right? You need to do that when you navigate Earth, so this knowledge has been around for a long time, and it's fairly easy to calculate the distance on a sphere. And the distance is all you need to know to create these circles, these rings, on the surface of the sphere. There isn't much more to the animations than those circles and these particle effects that look like little lights travelling alongside the rings. You don't need to know the shape of the object either, just the distance and the neighbours.

Right
The advantage of only having one degree of freedom is that Line Wobber is incredibly intuitive to play

At the moment, I have this fascination with making everything bigger. The next thing I want to do is a massive version of Wobble Garden, six metres long or something with thousands of springs. And if the code only looks at distance and neighbours, it's easy to expand, right?

You just need to plug in the maximum extent and it should work. I'm quite happy with that design at the moment. It works fairly well.

HS Does it look like a virus particle on purpose?

RB It's purely an accident that it looks like a virus, but everyone says that's what it looks like, so I need to lean into it and figure out something to do with it. In a way, it's very contemporary art. I don't know if people want to be reminded of the virus, but I'm just going to roll with it and see what happens.

I also haven't figured out how to represent the virus. It's interesting because we've been given the guidance that you shouldn't touch your face or surfaces, but all my stuff works by touch only, so there's this contrast there... I'm not sure how I'm going to deal with it, but there's an interesting conflict there.

Also, at the moment, no new events are happening, no museums are commissioning new work, so there hasn't been a very big response yet. We'll see where it goes. But I didn't know where Wobble Garden was going in the beginning either, and I found good uses for it.

I do know that I want to scale it up – that website I mentioned, where I found the 72 polyhedron sphere... they go up to any size, so it's easy to make, say, a 300-tile polyhedron. It's just quite a bit of work to get that done; it took at least a month to make the Wobble Sphere, but that was a few days of work, a few days of thinking, a few days of 3D printing, and so on. Now that I have the models in Fusion 360, it's going to be a bit easier to change the angles.

Right
Wobble garden has found new purpose as a science visualiser

Above
For all that the controller is experimental, Line Wobbler is quite a traditional game

Line Wobbler was a fairly traditional video game in a very unusual package. You had this strange LED strobe with a spring controller. But the game itself was very traditional; you fight enemies, you get to the end to make it to the next level. It's typical in a way.

With the newer things, Wobble Garden and with Quantum Garden, which is one version of Wobble Garden, and now Wobble Sphere, it's a bit more free-form. There isn't an immediate goal or purpose, and part of it comes from the way I built these things. I started very bottom-up, from an interesting interaction. Springs are cool and LEDs are cool, and I built up from there.

I didn't necessarily have an end goal in mind. Sometimes it's good to meander and go in unexpected ways; sometimes that also means that there isn't a finished state. You never get to the point where you feel like you've got a product. With Line Wobbler, that worked pretty well; here's a thing, it's complete, it's finished, people like it. But with Wobble Garden, for example, it's an interesting interaction, people enjoy doing it, but they're not entirely sure what they are doing. Is it a game? Is it an art piece? Is it a thing you hang on your wall?

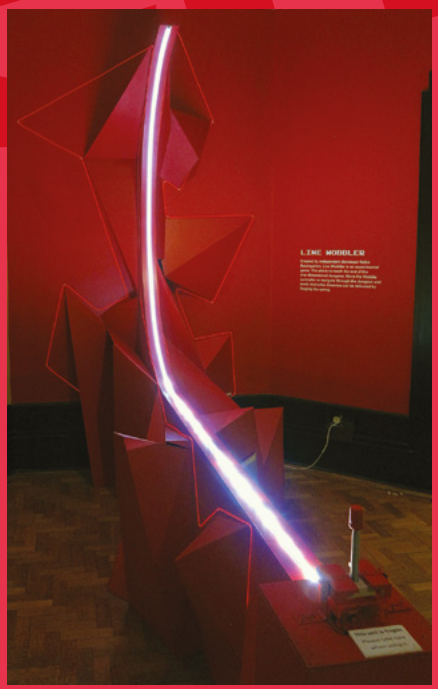
In the beginning, I was struggling with that. I thought that my work needed to have a purpose; this needs to be a thing, and how do I describe it to people? But now I'm much more relaxed. Maybe this is just almost like a platform for things to exist, and I don't know what those things are yet.

Quantum Garden came from this randomly. There was a collaboration with a quantum computing research group, and they wanted to visualise quantum computing somehow, because it's very abstract.

From there we said 'OK, I have a cool platform that looks very appealing and interesting, and they have perhaps algorithms that they want to visualise, or candidates for things that might look cool'.

There were several iterations of Quantum Garden as well; the video that you've probably seen, using these abstract circles that pulse, that was the first version. And [it's] still almost impossible to explain what's going on, because the quantum algorithms that are going on are very advanced.

Now we've changed it to another one which is actually possible to explain to a layperson. A lot of people have heard of Schrödinger's cat; this is Schrödinger's equation. With this one, it's possible to explain it, and it's possible to interact with Quantum Garden and see how the equation works. When you tap it again, the quantum state collapses… there's a quantum state of between zero and one, and it collapses to one or zero, which is how quantum computers work. This sort of stuff is easy to explain when you've got a visualisation that you can interact with. But Wobble Garden's creation only found meaning after the fact. I made a version of it that suddenly has a scientific purpose which was cool and very interesting, and I'm still working on it and exhibiting it. But it was definitely not planned in the beginning; it was an accidental meander towards science art. But right now I'm open to seeing where it takes me – it's a turbulent process, but I enjoy the fact that it's not set in stone. □



Left
All things being well, Wobble Garden will make its way to Brighton for the 2021 Festival of Light

FEATURE

IMPROVISER'S TOOLBOX

PLYWOOD

Put on your safety goggles and head to the workshop

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.

Plywood is a type of engineered wood material that's suitable for all kinds of construction projects. Put simply, it is made up of several glued sheets of peeled veneer, though the actual construction process is a little more involved.

It begins with logs that usually have a greater diameter than the average log from which wood is cut. The log is soaked in water for several hours, before it is mounted on a peeling lathe and turned on its long axis. A long knife peels off a thin layer of wood in a continuous veneer sheet, more commonly referred to as ply.

When dry, this thin ply is very strong along the grain, but very weak in the other direction. To overcome this, several layers of this veneer are glued together, alternating the grain direction between the layers. This stack is then heated and pressed to form a rigid panel.

"THE PLYWOOD IS FURTHER TREATED WITH SOME RESIN AND VARNISH TO MAKE IT STRONGER, DURABLE, AND WATER-RESISTANT"

Because the grain direction of the layers of veneer alternate, the panel is extremely strong in all directions. In most cases, the plywood is further treated with some resin and varnish to make it stronger, durable, and water-resistant. In some cases, the plywood is also laminated, which makes it highly resistant to chemicals and natural phenomena.

The standard size for plywood in the UK, and many other countries, is 4×8 foot (1220 mm × 2440 mm). Compared to some other types of wood materials, plywood has a low cost and a high strength to weight ratio. Thanks to its construction, plywood also has high impact resistance, as the laminated surface distributes loads from impact over a larger area on the opposite face, which effectively reduces the tensile stress.

Wooden objects from ancient Egypt around 3500 BC made from veneer, that was sawed and stitched together with cross-wire, are the earliest known instance of the use of a plywood-like material. It wasn't until 1797 that British naval engineer Samuel Bentham patented the process to make thick panels by using multiple layers of sliced veneer combined with glue. Plywood made its way across the pond not long after, where it was instantly commercialised.

Today, plywood is a broad term, and there are several varieties of plywood for different applications. For instance, there's softwood plywood that is useful for industrial applications; hardwood plywood is popularly used for flooring; while flexible plywood is useful for furniture. Of course, with the right kind of tools, you can use it in many unique and exciting ways, as we'll show over the next few pages.

ROCKET BOOKSHELF

To fulfil her son's love for space, Vineta's created a rocket-styled bookshelf. While she modestly calls herself an amateur woodworker, she's carved the entire bookshelf from a single sheet of plywood, which makes it fairly cheap to replicate. Working with wood is an instrument-intensive affair, and this build is no different. You'll need a jig and a jig-saw, along with some screws, sandpaper, and some measuring instruments like a beam-compass and a square. She first cuts the plywood to a more manageable 6"× 4" piece, and marks a line halfway down the middle and 24" from

> "She's supplemented the textual instructions with visual illustrations"

the edge. This marks the widest point of the rocket. She then lays out her design using an extension cord on a 6" sheet of paper, which is then traced onto the plywood before it is jig-sawed. Creating and connecting the shelves is a little more involved, but Vineta has explained the process in great detail. She's supplemented the textual instructions with visual illustrations at every major step, which makes it very easy to replicate. If you sign up to her email list (registrations are free), she'll send you a PDF construction plan of the bookshelf with all the measurements. →

Project Maker
Vineta Jackson

Project Link
hsmag.cc/YP1n4n

Left ◆
It might appear that the bookshelf is balanced on the rocket fins, but it's actually attached to the wall

FEATURE

CHILDREN'S CHAIR

Project Maker
Steven De Lannoy

Project Link
hsmag.cc/NGYYiT

Now that you've constructed a bookshelf for your child, give them a chair to sit on and visualise their space adventures. Steven's chair is simple, yet sturdy, and easily modifiable depending on the size of your child and the amount of plywood at your disposal. The good thing about the chair is that it doesn't need any screws, which makes it safe for children of all ages. The chair is made with five panels of plywood of different sizes. You can cut them to size using either a circular saw with a straight-edge guide or a table saw. Then begin the construction by following Steven's templates to first create the two side panels. It'll take some doing, and when they're done, attach wooden beams to both the panels to help support the back panel and the seat. Now assemble the side panels with the help of three temporary construction beams, before sliding and gluing the seat panel in place. The bookshelf below the seat panel wasn't part of the original plan – it was created with some leftover wood and slid in place pretty much like the seat itself. Finally, sand the surfaces, apply varnish, and perhaps even some pipe insulation to the side panels to make the seat more child-friendly.

> "The bookshelf below the seat panel wasn't part of the original plan"

Left
The pictured chair is designed for an average-sized six-year-old, but Steven's plans can easily be modified to suit smaller and larger children

DESK ORGANISER

Working with wood gives you two things: a beautiful piece of wooden furniture and leftover wood. Jacob's desk organiser, or 'bloks' as he likes to call them, can be made from any amount of leftover plywood. Begin by grabbing the items you want to place in your organiser, and lay them out on a piece of paper in the same manner as you'd want them in the finished product. Trace the items, leaving a 1-inch border around the whole block and in-between the items. You'll then have to figure out how deep you want the slots to be. Then grab the appropriate number of plywood sheets to create that height. Jacob used four sheets of the ¾" plywood for the pictured organiser, with one acting as the base and the other three with slots for the items. Cut out and trace the template on the plywood and then jig-saw them. Finally, glue all the plywood and clamp them down as you sand the edges.

Project Maker
JACOB

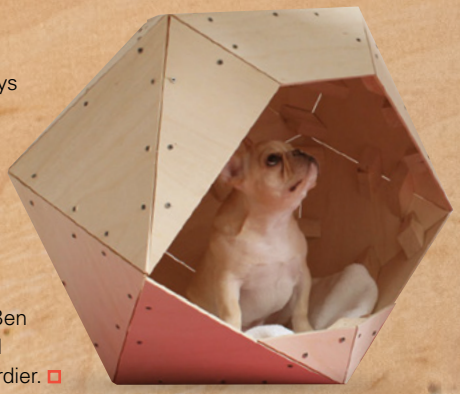
Project Link
hsmag.cc/cJrtyX

Left
Jacob's applied a couple of coats of polyurethane to give his organisers a nice shine

Below
Make sure you sand the triangles thoroughly before assembling them. Still, it'll be a good idea to place some cushions to make the pet house more comfortable for your pet

GEOMETRIC PETHOUSE

If you're a mathematically inclined pet parent, Ben's geometric pethouse is just the thing you need. It's made up of a bunch of equilateral triangles that should be about the same height as your pet when it's sitting. Ben's dog Fletcher is about 14" when perched, and so Ben used a standard single sheet of plywood cut into two 14" wide strips. He clamped the boards together, and then measured and cut 20 triangles using a circular saw. He's also used a couple of 2×3s to create the angled support blocks to connect the triangles. Ben's shared two ways to cut these, depending on the type of saw you have. Then sand the triangles and screw the blocks onto the triangles, before assembling the pieces together. You'll have to cut some of the triangles to create room for your pet to enter their new digs – Ben suggests screwing some additional blocks to make the contraption sturdier.



Project Maker
BEN UYEDA

Project Link
hsmag.cc/38cglF

HackSpace
TECHNOLOGY IN YOUR HANDS

FORGE

HACK | MAKE | BUILD | CREATE

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

Weather-resistant prints with ASA

Add some 3D-printed bling to your outside spaces

Ben Everard

@ben_everard

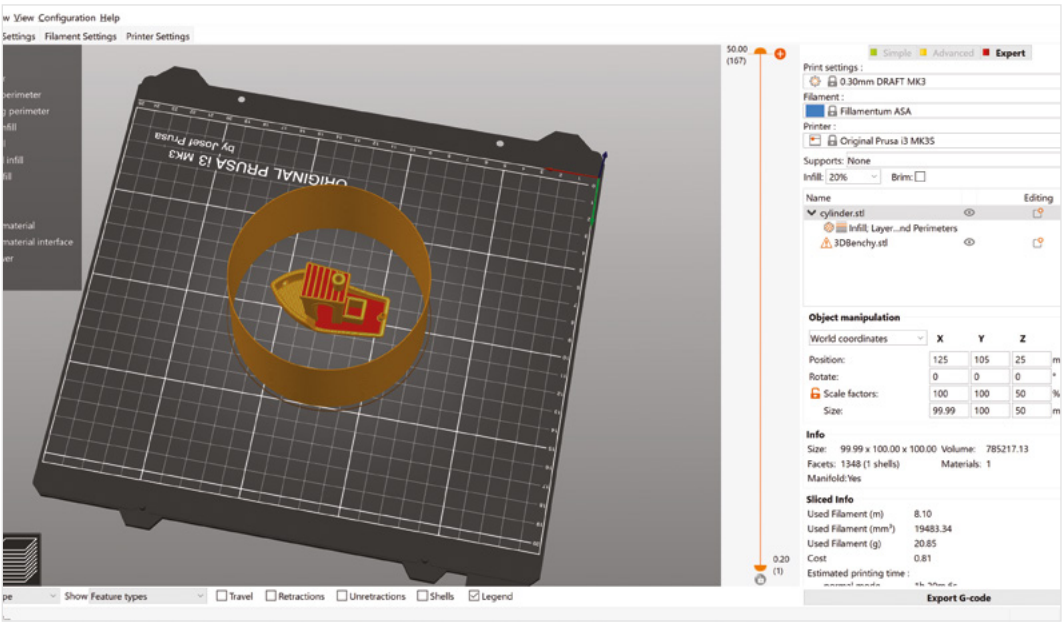
Ben loves cutting stuff, any stuff. There's no longer a shelf to store these tools on (it's now two shelves), and the door's in danger.

Figure 1
With larger, thinner objects like this planter, it can be hard to get layers to stick together properly if you don't have an enclosure

The big selling point of ASA is its weather-resistance. You can print bits with this plastic, leave them outside, and expect them to last. Other 3D-printable materials are prone to breaking down with UV exposure. It's also more temperature-resistant than most common 3D-printable filaments, which is an extra bonus for anything that might be exposed to both UV and heat like something on a car dashboard.

However, with this, there are a few downsides. The most important thing is safety. You may not realise it, but all 3D-printable filaments give off fumes in the form of volatile gases and fine particulate matter. PLA and PETG give off very little, but some, such as ASA, give off non-trivial amounts. There's no easy way to say how much of a problem this is – different brands, different printers, and different setups will give off different amounts. There are a couple of solutions to this – you could print in a well-ventilated room, or you could have fume extraction on your printer. To complicate matters further, you also need to avoid drafts (we'll get to this), so simply placing your printer by an open window isn't a good option.

So, if you've got a place where it's safe to print, let's take a look at how to print it. First up, you need a printer that can get hot. You'll need a bed



Left
When printing without an enclosure, you might have better results if you put a cylinder around your object to stop it cooling too quickly

Below
We solved the layer adhesion issue in Figure 1 by enclosing the planter in a cylinder

Below
ASA prints really well (provided you avoid the problems with layer adhesion)

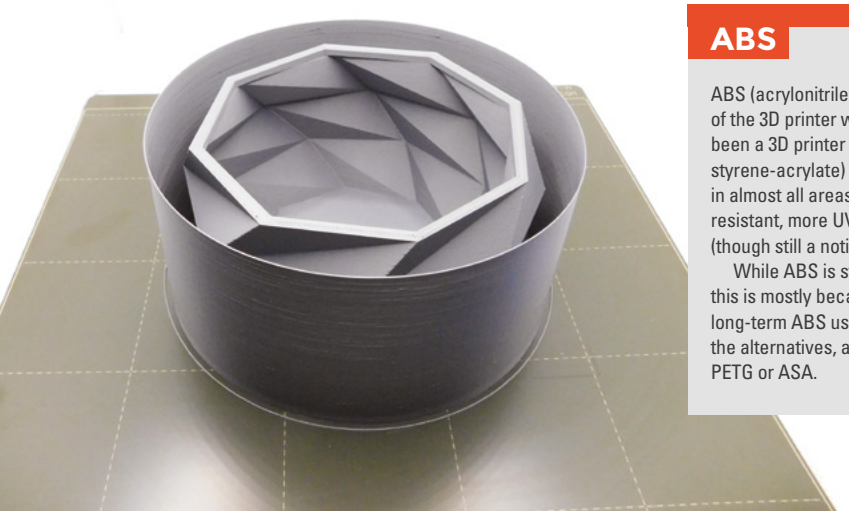
temperature of around 80–100°C, and a hot end at 230–260°C.

It's quite a sticky plastic, so bed preparation is important. On PEI sheets, you'll probably want to use a smooth finish rather than textured. On glass, you'll need a release agent, such as hair-spray. Some people find that adding Kapton tape to the bed helps provide a more even bed temperature, and thereby minimises warping.

Depending on how large and how accurately you need to print, you might find that shrinkage isn't a significant problem, but it does happen. So, if you need dimensionally accurate parts, you'll need to make allowances for it. The big problem with ASA is layer adhesion. This is partly due to how stable it is at high temperatures – when each layer goes down, the previous layer is already solid. This means that ASA prints can be weak in the Z direction, and you may find larger prints have gaps where the two layers haven't stuck at all.

> "Prusa recommends printing a hollow cylinder around **the object as you print it**"

The key to good ASA printing is temperature management. If it cools down too quickly, it's prone to layers not sticking together and other problems, so turn the part-cooling fan right down, or off completely. Ideally, print in an enclosure, but if that's not possible, at least minimise drafts (while, of course, ensuring that the room is properly ventilated). Prusa recommends printing a hollow cylinder around the object as you print it. In our experiments, we've found that this is a definite improvement on no enclosure, but isn't a complete solution, particularly with larger and taller prints.



ABS

ABS (acrylonitrile butadiene styrene) has been a staple of the 3D printer world almost as long as there has been a 3D printer world. However, ASA (acrylonitrile-styrene-acrylate) is a plastic that supersedes ABS in almost all areas. It's as strong, more temperature-resistant, more UV-resistant, and emits fewer fumes (though still a noticeable amount) when printing.

While ABS is still a popular 3D printing filament, this is mostly because of the momentum. If you're a long-term ABS user, it might be time to take a look at the alternatives, and see if you're better served by PETG or ASA.

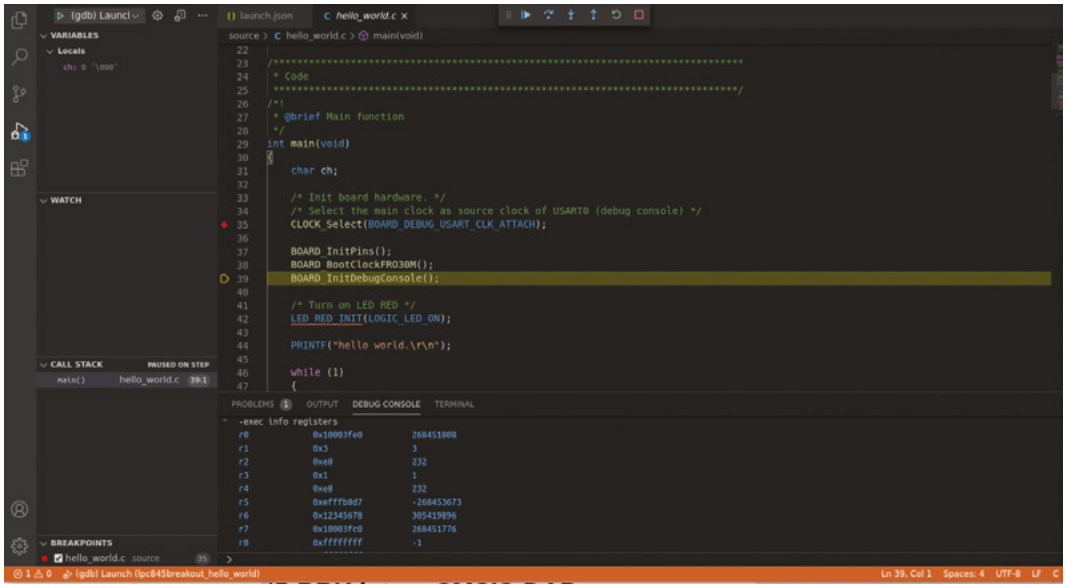
Debugging microcontroller software

Get your project working faster by taming the debugger

Dr Andrew Robinson

@fortyfourmu

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



Above Harness the debugging power of GDB with the ease of use of VS Code

Reduce the time it takes you to debug software and discover the powerful hidden features of GDB, the GNU Project Debugger – it isn't as scary as it first seems!

DEBUGGING SOFTWARE

In this final of three articles on debugging, we look at debugging software on microcontroller projects.

Debugging software can be tricky at the best of times, but when the code's running on a tiny microcontroller, it's many times harder. As highlighted in the last two articles, debugging is easier if you can observe what is going on in a system and exercise particular functions. With no screen and keyboard attached, these aspects are particularly challenging for debugging software in microcontrollers.

HIDDEN IN THE BOX

Without care, debugging on a microcontroller is a bit like Schrödinger's cat: the act of observing it makes it behave differently.

Debugging can be similar (though slightly less strange than the quantum world); if you want to collect data on how your program runs, you have to add code, and this will make your program run differently.

Serial connection

Adding a pair of serial lines provides a basic way of enabling input and output of debugging information. It allows messages to be printed to a terminal on a PC, and can take user input from the keyboard. The drawback of this method is that software is needed to actually transport the data and respond to it.

Printing status

To show the values of variables, or the flow of execution through your program, you'll typically add print statements at key moments. While with experience you'll get a better feel of where to place print statements to get a good overview of what's happening in your program, typically, to debug an issue, you'll need more. This means recompiling, reprogramming, resetting, and then recreating the bug each time you need to investigate a new variable or program location. This long-winded process gets tiresome and protracted very quickly. Inserting print statements also actually changes your program. If you're particularly unlucky, inserting print statements can change how your program runs. The author had experience of this with a bug that accessed uninitialised memory. By chance, inserting a print statement caused the memory to be correctly initialised, and therefore hid the bug. This led to much time being wasted.

> **If you're particularly unlucky, inserting print** statements can change how your program runs

Another major drawback of printing debugging information is the amount of time spent on printing and the amount of data generated. For resource-constrained microcontrollers, this can be quite significant. Commenting the print statements out, or removing them after debugging, takes further time.

A logging framework can help manage this; for compiled languages, a flag can be set that prevents the print statements being included in the final code, meaning no loss of efficiency. Some frameworks allow statements to be categorised by severity, ranging from information through warnings to critical errors. They can also group and control printing at file or module level, and offer formatting options such as colour coding. If a framework is available, then it's definitely worth spending a bit of time investigating, rather than immediately reaching for the print statement.

Accepting configurable parameters

To be able to adjust variables or call functions, you'll have to write code to parse the input that is sent from the console. Typically, you won't have options for all the variables and functions you'll need when debugging a particular issue, so you'll have to recompile and reprogram to add options to set everything required. Parsing input can quickly become complicated and a source of further bugs and, as with introducing print statements, will change the code the microcontroller is executing. As such, other than for changing a few configuration settings, using serial input is not recommended.

```
andrew@mcuxpresso:~$ openocd -f interface/cmsis-dap.cfg -f target/lpc84x.cfg
Open On-Chip Debugger 0.10.0
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.org/doc/doxygen/bugs.html
Info : auto-selecting first available session transport "swd". To override use '
transport select <transport>'.
adapter speed: 10 kHz
adapter_nsrst_delay: 200
cortex_m reset_config sysresetreq
Info : CMSIS-DAP: SWD  Supported
Info : CMSIS-DAP: Interface Initialised (SWD)
Info : CMSIS-DAP: FW Version = 1.10
Info : SWCLK/TCK = 1 SWDIO/TMS = 0 TDI = 0 TDO = 0 nTRST = 0 nRESET = 1
Info : CMSIS-DAP: Interface ready
Info : clock speed 10 kHz
Info : SWD DPIDR 0x0bc11477
Info : lpc84x.cpu: hardware has 4 breakpoints, 2 watchpoints
```

Above OpenOCD successfully connected to SWD debug interface

Using a debugger

Debuggers offer a much richer debugging experience, but can have a fairly steep initial learning curve. A good debugger should allow a running program to be paused and then stepped through line by line. It should also allow the contents of the memory and processor registers to be examined, and changed if required. Breakpoints can be set at a specific line in the program which, when reached, will cause execution to pause. Watchpoints are similar, but pause execution when a variable or register is assigned a particular value. More advanced features of debuggers include handling multiple threads, and integrating with power analysis for low-power applications.

GDB

GDB is a free debugger which was created as part of the GNU project to build a free UNIX-like operating system in the 1980s. Development continues and new features are still being added. Most microcontrollers use the GCC compiler, which was also developed as part of the GNU project. As such, →

PERMISSIONS **TROUBLE**

If OpenOCD can't talk to your USB debug interface, check the permissions on the device. CMSIS-DAP often uses **/dev/hidraw0** and **/dev/hidraw1**. Use **dmesg** to look at the devices that get created when you plug your board in, and then check and change the permissions if necessary. You could create or download udev rules to automatically set the correct permissions.

QUICK TIP

JTAG is such a simple protocol, you can create your own debug interface by bit-banging GPIO pins on a Raspberry Pi with appropriate software.

SOURCES

OpenOCD comes bundled with configuration files for many devices. However, most operating system packages of OpenOCD are not sufficiently up-to-date, so it is better to build from source (**hsmag.cc/Q9T0HT**) or another up-to-date repository, such as xPack OpenOCD (**hsmag.cc/1Gpdfc**).

if the GCC compiler and assembler has been ported to a particular processor architecture, GDB will often be available for it too.

On first impression, GDB looks intimidating and its command-line interface perhaps a little uninspiring. But when you discover it incorporates a Python interpreter, and you uncover the magic keystrokes to visualise your code, you'll wonder how you managed without it.

GDB can run on a PC and connect to a remote target, and we'll see this later. However, for simplicity, in the first example, we'll debug a program compiled for and running on a Linux system, e.g. a Raspberry Pi.

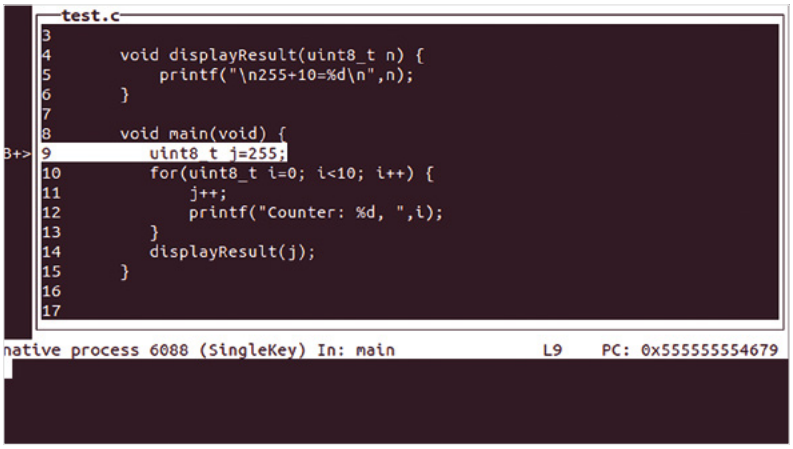
If GCC and GDB are not installed for your Linux computer, install them using your package manager. To install with APT, type the following in a terminal:

```
sudo apt update
sudo apt install gdb gcc
```

If you need GCC/GDB for another architecture e.g. to compile for an ARM microcontroller, then download them from **hsmag.cc/OkRz9H**.

To see what's going on, we'll create a simple program with a deliberate mistake. We'll then use the debugger to go through it to find out why it behaves the way it does. Create the following example program and save it as **test.c**.

Below
Use the lesser-known TUI mode of GDB to step through your code




```
#include <stdio.h>
#include <stdint.h>

void displayResult(uint8_t n) {
        printf("\n255+10=%d\n",n);
}

void main(void) {
   uint8_t j=255;
   for(uint8_t i=0; i<10; i++) {
        j++;
        printf("Counter: %d, ",i);
   }
   displayResult(j);
}
```

Use GCC to compile **test.c** into an executable

```
gcc -g -o  test test.c
```

Include the `-g` flag to include debugging information for the debugger.

Run the code:

```
./test
```

You should see the last line of the program print:

```
255+10=9
```

This may or may not be the result you're expecting. Let's use GDB to show what's going on. We'll use it to show the result of running the program, one line at a time, and inspect the contents of the variables.

First, start GDB with the program to debug:

```
gdb ./test
```

Once GDB starts, add a breakpoint to the `main` function in your program.

```
b main
```

Tell GDB to start debugging:

```
start
```

GDB will pause on the first statement under `main`. Activate SingleKey TUI mode by pressing **CTRL+X** followed by **S**. You should see the first line of your program highlighted. Press **N** to step to the next line, and the highlight should move down a line. Press **N** a few more times until the print statement is highlighted. What value is contained in variable `j`?

Press **V** to show the contents of the variables.

DEBUGGERS SHOW WHAT A LINE OF CODE ACTUALLY DOES

Variable `j` has been incremented from 255 and, because it is only an 8-bit positive integer, it wraps

Left
See the contents of your variables at any point in your program

around to become 0. This may not have been what the programmer intended, but GDB has allowed us to see what is happening.

Add another breakpoint at the **displayResult** function by typing:

```
b displayResult
```

Press **C** to continue running until that breakpoint. Press **V** and notice that GDB doesn't show any variables as there are none defined in the **displayResult** function. Press **W** to show how the function was called: in this case, it shows **displayResult** was called from line 14, which is in **main**, and it was passed the value 9 for argument **n**.

GDB Single Key TUI commands summary table

KEY	MEANING
R	run
C	continue
S	stop
N	next
O	step over (doesn't step into functions, steps to immediately after their return)
F	run to the end of the current function (step out)
V	info locals (show current local variables)
W	where (show where function was called from)
Q	quit single key mode

GDB CONTAINS A PYTHON INTERPRETER!

The later versions of GDB have a Python interpreter built-in. This can be useful for formatting debugging information for display or exporting to file. You can also use it to programmatically control other features of GDB, such as breakpoints, and create dynamic behaviour, e.g. setting another breakpoint depending on the state of a variable in the current execution run. Note that some builds of GDB disable the Python interpreter, in which case you may need to rebuild GDB from source yourself.

DEBUGGING ON BARE METAL WITHOUT AN OS

When you run a program under an OS, the OS manages the screen and other hardware outside of your program. The OS can start, stop, and monitor your program, and watch for break conditions. If your program crashes, the OS continues to run and allows you to examine the result. This is not the case when running applications on microcontrollers, where your program runs on 'bare metal'. If your program stops, there is nothing else running to manage interaction with the hardware I/O. For these situations, chip manufacturers include debugging hardware on the chip. This may allow breakpoints and watchpoints to be set, execution started, stopped and traced, and access to read and write flash and memory. However, the facilities provided will vary between models and manufacturers. Also, because there is a dependency on physical hardware resources, breakpoints might be limited to just two or three, and trace buffers will only be able to trace a limited number of instructions.

Remote debugging

GDB can debug a program on a remote system as well as the PC it is running on. Again, it can sometimes be fiddly setting up this remote link, so it can be useful to understand how the remote connection is made.

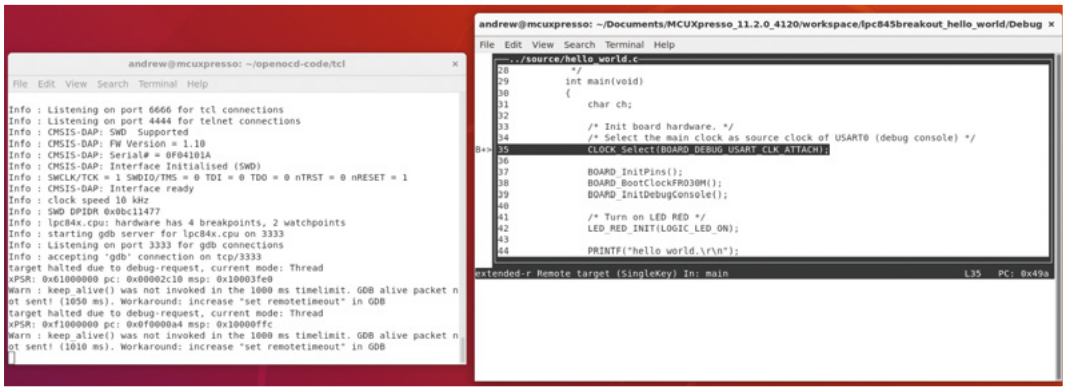
> **Note that some builds of GDB** disable the Python interpreter

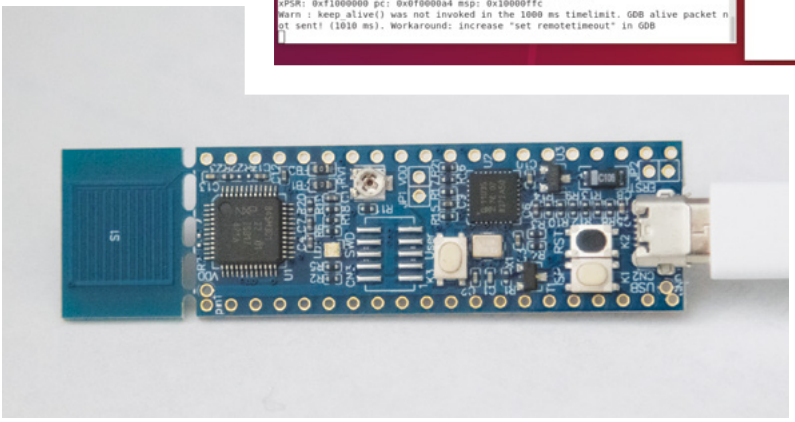
To communicate with embedded hardware, there is reasonable standardisation, and most chips offer JTAG or SWD, or both. JTAG consists of four communication lines, whereas SWD requires two which are bidirectional. To connect with a PC, you need a JTAG or SWD interface, and these typically connect over USB. You can either use a standalone interface, or some development boards have one built-in. ARM provides a reference implementation of a USB-SWD interface called CMSIS-DAP, which is commonly included on development boards.

OpenOCD acts as the connector between GDB and the debug interface. It can control the debug interface over USB and interpret the data sent over JTAG or SWD. Both the USB control messages and the JTAG/SWD messages differ between devices, and different PCBs may have multiple devices sharing the debug connection. As such, OpenOCD needs to be configured with this information. This is done by specifying one or more configuration files for →

QUICK TIP

GDB typically connects to OpenOCD over a network connection on port 3333. This offers further flexibility as you can debug a remote machine located somewhere else running OpenOCD.





Above, Below
Step through code running remotely on microcontrollers

the Interface, the Board, and the Target (i.e. the chip). The configuration files are actually Tcl/Tk scripts. However, in many cases, the configuration files for most popular development boards are included or are available to download online.

In this example, we'll be connecting to an LPC845 development board; you'll have to adapt it depending on the hardware you're using.

Before starting GDB, start OpenOCD with the necessary config file options for your device.

For instance, for this example:

```
openocd -f interface/cmsis-dap.cfg -f target/
lpc84x.cfg
```

Here we specify we are using the CMSIS-DAP interface and are debugging an LPC84x chip (which supports the LPC845 microcontroller we are using).

You should see a series of messages that contain "Info : Listening on port 3333 for gdb connections". Often the number of hardware breakpoints and watchpoints supported by the hardware are shown, in this case, 4 and 2. Now start GDB to connect to OpenOCD. Note: you need a version of GDB built for the target architecture (in this case ARM).

Start GDB followed by the program that has been flashed onto the microcontroller:

```
arm-none-eabi-gdb hello_world.axf
```

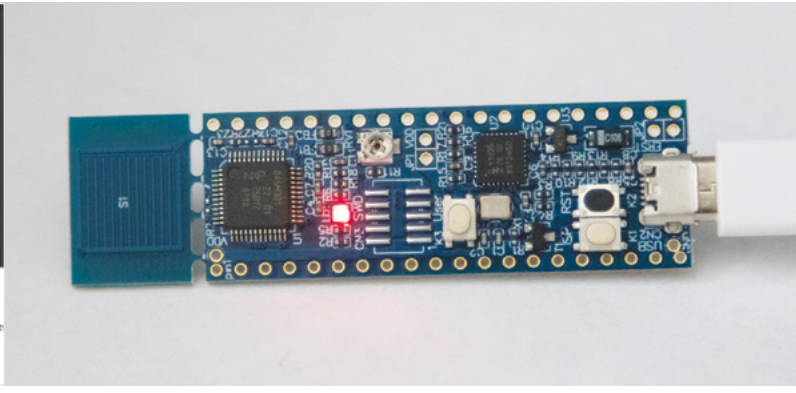
Once GDB starts, tell it to connect to OpenOCD. In this case, since it's on the same machine as GDB, we use localhost:

```
target extended-remote localhost:3333
```

Type `start`, press **CTRL+X** and then **S** to enter Single Step TUI mode, then press **N** to step through. We notice that after we step over `LED_RED_INIT(LOGIC_LED_ON);` the LED turns on!

SHOWING MORE DEBUG INFORMATION

There are times when you need more information from the process to debug exactly what is happening, or to determine what mode it may be in. We can see the values in the processor registers and the actual instructions the processor will execute. To show the processor registers, in GDB type `layout regs`.



Press **N** to step through and notice how pc (the program counter) increments.

To show the actual machine instruction the processor has just executed, type **`layout split`**.

OpenOCD debugging commands in GDB

bp addr	breakpoint
rbp addr	remove breakpoint
wp addr	watchpoint
rwp addr	remove watchpoint
mdd addr mdw addr mdh addr mdb addr	display doubleword/word/halfword/byte from memory
mwd addr data mww addr data mwh addr data mwb addr data	data,write doubleword/word/halfword/byte to memory

Graphical debugging

GDB is very powerful, but there's no doubt it can be a steep learning curve. Luckily, GDB allows itself to be attached to a GUI front-end. There's a range of GUIs to choose from including DDD and Eclipse, but we're using Visual Studio Code.

Open your project in VS Code. To debug your project, you need to set up an appropriate run configuration. On the menu, go to Run > Open Configurations.

Set the 'program' entry to the file name of your compiled program. This will usually be an AXF or HEX file.

Set 'miDebuggerpath' to the full path of the arm-none-eabi-gdb you installed.

Set 'targetArchitecture' appropriately, e.g. arm.

Change 'stopAtEntry' to true.

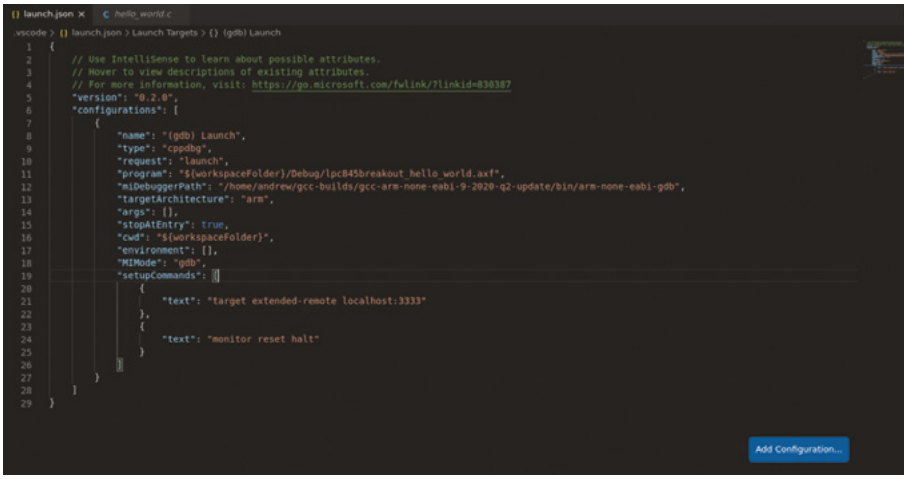
Clear out the existing 'setupCommands' and replace with text entries for 'target extended-remote localhost:3333' and 'monitor reset halt'.

Save the file.

See the image above for a full example.

From a separate terminal, start OpenOCD as before if it is not already running. In VS Code, click the debugging view (the bug next to the play icon) or press **CTRL+SHIFT+D**. Click to the left of the line of code where you want to create a breakpoint. Click the play icon at the top.

You should see that OpenOCD recognises a new connection from GDB, and VS Code will then highlight the point where the program is running. Use the Debug toolbar at the top of the editor to run, step, reset, etc. your program.



Above Example configuration for debugging a microcontroller with GDB and OpenOCD

You can still use some of the more advanced functionality of GDB from VS Code through the Debug Console. VS Code reminds you to prefix commands with **`-exec`**. To show contents of the registers, type **`-exec info registers`**.

GDB offers significant debugging power and flexibility, e.g. using Python to programmatically debug or help with custom visualising or formatting of data structures. Combined with the ease-of-use features and graphical richness of tools like VS Code, this makes GDB more accessible.

> **GDB is very powerful, but there's no doubt** it can be a steep learning curve

Understanding the modular approach and how the debugging chain is made up helps you to know where to look if a debugger fails to connect to the target hardware. The open, well-defined interfaces of tools like GDB allow deep stacks to be built with established tools. As such, if a new chip works with GDB, chances are that any of the graphical debuggers that connect to GDB can also be used, and users can follow a universal debugging workflow they're already familiar with. ◻

PYOCD **PACKAGE**

You may wish to consider pyOCD, an alternative to OpenOCD written in Python: **hsmag.cc/9FYWoE**. Since it's maintained by ARM, it's focused on ARM processors and limited JTAG interfaces! Also, keep an eye out for VS Code extensions under development that provide debugging support for microcontrollers.

Watch games controller

Turn an ESP32-based watch into a Bluetooth games controller

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.

We've been playing around with the LilyGo T-Watch 2020 for over a month now, and it's great fun. In this tutorial, we'll explore the hardware and take a look at how you can use its basic functions. We'll ignore the watch part of this tutorial because that introduces a little more complexity, and instead use the watch as an innovative games controller. We'll look at three key input methods on the watch and use them to create a mini Bluetooth keyboard that uses the pedometer to keep you fit. The controls will be:

- Run on the spot to move forwards
- Press the left-hand side of the screen to turn left
- Press the right-hand side of the screen to turn right
- Tap the power button to perform an action

This is obviously quite a limited set of controls, so it won't work for every game. We chose them for simple racing games (which work well with the running input), but you could map them to other actions for other games.

Before we get too tied into the code, let's take a look at the hardware on the T-Watch that we'll be using.

The processor at the heart of the watch is an ESP32, which has integrated WiFi and Bluetooth (we'll be using the latter to emulate a keyboard). This is quite a beefy processor, but we won't be pushing it to its limits. It's well-supported in the Arduino ecosystem, and there's loads of example code and libraries around to help you make use of it. The downside of this processor is that it can take a little work to stop it draining power. We're not going to worry too much about that for this project, as we'll assume that you're not going to use the running-on-the-spot controller for more than an hour or so at a time.

At the front of the watch sits a 240×240 TFT touchscreen. This is capable of quite nice graphics and is set up by the drivers using the Adafruit GFX library, so it's easy to draw things on the screen. You could do something quite graphically interesting with this, but for our controller, we just output some debugging information. The screen burns the battery quite quickly, so if you find it's a problem, you might want to disable it in the code. The touch interface is hidden behind some internal gubbins – it's only exposed to us via a single function to get the current touchpoint.

Right
The final controller isn't much to look at, but it lets us play our games

The heart of our controller is the pedometer, which comes courtesy of a BMA423 accelerometer. This includes an internal step-detection system, so we don't need to actually detect the steps ourselves – the accelerometer does it for us.

The final piece of the puzzle is an AXP202 power management chip. For the most part, this works in the background, but it also handles our button presses as it's the power button. A long press (over six seconds) turns the watch off and on; a short press is detected by our code and converts to an action.

This bundle of hardware is all controlled via one main class, the TTGOClass. This provides us with pointers to functions to get what we need.

Left
We really enjoyed using the watch with racing games such as SuperTuxKart

> We don't need to actually detect the steps ourselves **– the accelerometer does it for us**

Before we get to the hardware, though, let's take a look at the interface with the computer. Bluetooth Low Energy (BLE) is an adaptable protocol that can send packets of almost any data. It gets a bit complex if you get down to the nitty-gritty, but fortunately, there are some wrappers that tidy everything up for us. The ESP32 can simply emulate a wireless keyboard. The only slight issue with this is that keyboards have separate press and release events for each key. We've used a slightly hacky solution that checks for a change in the state, and then releases all the keys before immediately re-pressing the relevant ones. This makes it a bit more robust.

The crucial bit of the code that handles the keyboard emulation is then:

```
  if(bleKeyboard.isConnected() and (tapping_right
!= was_tapping_right or tapping_left != was_
tapping_left or stepping !=was_stepping or
button_press)) {
    bleKeyboard.releaseAll();
    if(stepping) {bleKeyboard.press(go_key);}
    if(tapping_left) {bleKeyboard.press(left_
key);}
    if(tapping_right) {bleKeyboard.press(right_
key);}
    if(button_press) {bleKeyboard.press(button_
key);}
  }
```

The key variables can either hold **char** or **uint8_t** data types. In our case, they're defined with:

```
uint8_t go_key = KEY_UP_ARROW;
uint8_t left_key = KEY_LEFT_ARROW;
uint8_t right_key = KEY_RIGHT_ARROW;
char button_key = 'z';
```

This obviously requires us to set the variables **stepping**, **tapping_left**, **tapping_right**, and **button_press** to say whether or not these events have taken place. Let's have a look at this now. →

GOING **FURTHER**

Watches are, by their very nature, small devices, so there's not much space to get extra input on them. You could put more buttons on the screen, but you'll probably struggle to press them accurately, especially when running. If you need more input, you could use the watch only for step-based input, and then use an additional controller for other buttons – there's no need to try and do everything with just this.

SETUP

You will need to set up your Arduino IDE to work with the LilyGo T-Watch 2020 by following the setup info at **hsmag.cc/GILca1**. This also brings in the libraries needed to control the watch hardware, so you don't need to install these separately.

You will also need to include the ESP BLE Keyboard library from **hsmag.cc/LnMzwX**.

STEP CONTROL

The most original part of this controller is the step control. This simply uses the step-detection feature of the BMA423 accelerometer to see if you're currently stepping and, if you are, presses the up arrow button on the keyboard.

The BMA423 keeps an internal step count and initially, we just tried to see if this was incrementing, but unfortunately, there's quite a delay in this count, so this approach didn't work. Instead, we had to use the step interrupt that fires every time a step is detected.

There are two parts to this – the setup, and the loop. The setup is:

```
pinMode(BMA423_INT1, INPUT);

attachInterrupt(BMA423_INT1, [] {
    irq = 1;
}, RISING);
```

```
ttgo->bma->begin();
ttgo->bma->attachInterrupt();

ttgo->bma->enableStepCountInterrupt();

last_step_time = millis();
```

The watch isn't brilliantly documented, but there are a good few examples at **hsmag.cc/TzAVnp**. From here, you can find the setup code like this to get all the bits working.

As you can see, we also record the `last_step_time`. We'll use this to make sure that the wearer is currently walking fast enough for us to trigger the button press.

The following code in the loop section of the Arduino sketch then sets the `stepping` variable appropriately:

```
if (irq) {
    irq = 0;

    bool  rlst;
    do {
        rlst =  ttgo->bma->readInterrupt();
    } while (!rlst);

    if (ttgo->bma->isStepCounter()) {
    stepping = true;
    ttgo->tft->setTextColor(GREEN, BLACK);
    ttgo->tft->drawString("stepping         ",
22, 60, 4);
    last_step_time = millis();
    }
}

  if(millis()-last_step_time > stop_step_speed) {
    stepping = false;
    ttgo->tft->setTextColor(GREEN, BLACK);
```

Below
You can access the electronics easily if you want to poke around with what's inside the watch

KEYBOARD VS GAMEPAD

In this code, we've emulated a keyboard, but there is also a BLE standard for creating gamepads. There is a library for working with them on the ESP32 that is very similar to the keyboard library we've used (**hsmag.cc/1lu0aA**). One big difference is that keyboard keys are either on or off, but gamepad joysticks can be analogue. This could give you the ability to vary the speed input based on the speed you run at. How well this will work will depend on the game. We decided to use simple keyboard input because it's compatible with a wider variety of games.

ADDITIONAL HARDWARE

This controller uses most of the hardware in the watch. The only additional bits are a real-time clock (a PCF8563) and a haptic feedback motor on pin 4. Altogether, this is quite a capable, wearable computer that you can turn into a watch. This author does also have his own watch firmware that we'll take a closer look at in a future issue. You can see the code at: **hsmag.cc/FE9XqP**.

```
    snprintf(buf, sizeof(buf), "not stepping");
    ttgo->tft->drawString(buf, 22, 60, 4);
}
```

When an interrupt is fired, the `irq` variable is set to true, and we then read all the interrupts that come from the BMA. The `isStepCounter()` function tells us if this particular interrupt is a step interrupt, and in there, we can do the necessary processing. In this case, we set the variable, put some text on the screen so that we've got some feedback, and update `last_step_time`.

The final piece of the puzzle is checking if it has been long enough since the last step to reset the `stepping` variable and consider the user stopped.

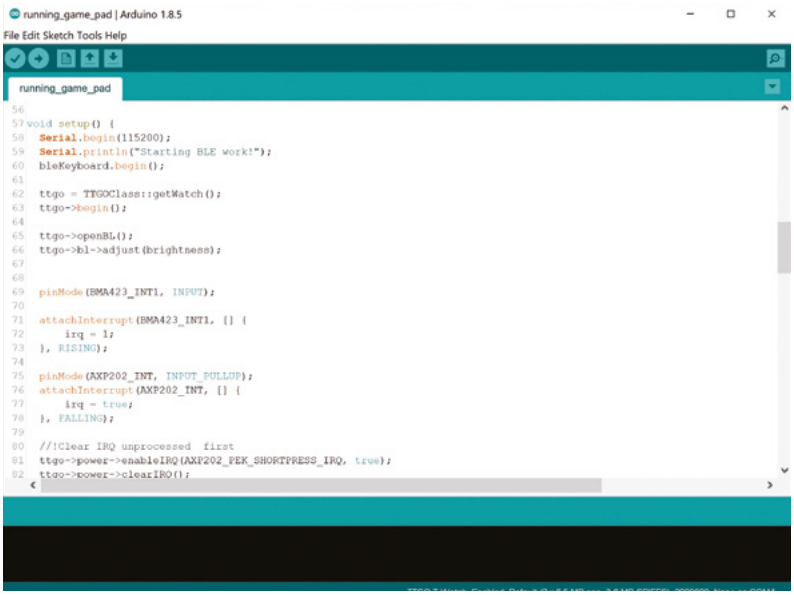
UP AND RUNNING

The only button on the watch is attached to the power management chip, so we can't read it like a regular GPIO. Instead, it's accessible via an interrupt in a similar way to the step counter. Take a look at the code for full details.

The final piece of the puzzle is the touch interface which we do with the following code (the debug print commands have been removed for brevity):

```
int16_t x;
int16_t y;
tapping_left = false;
tapping_right = false;
if (ttgo->getTouch(x, y)) {
  if(x<100){
    tapping_left=true;
  }
  else if(x>140){
    tapping_right=true;
  }
}
```

This simply detects if there's a tap, and if there is, checks whether or not it's on the left- or right-hand side of the screen (taps in the middle of the screen will be ignored), and sets the appropriate variables.



That's the main guts of the code – all that remains is to load it to your watch, then pair it with your computer. Once everything's connected, it's time to get running. Bear in mind that once paired, it'll act as a keyboard whenever you take a step, so you may want to turn Bluetooth off on your computer if you're walking about.

Above
You'll need to use the Arduino IDE to compile and upload the code to the watch

Right
Press the side of the screen to turn on/off

TUTORIAL

Raspberry Pi, meet Arduino

Need the versatility of an Arduino and the horsepower of Raspberry Pi? Here's how to use both in your project

PJ Evans

@mrpjevans

PJ Evans is a developer and wrangler of the Milton Keynes Raspberry Jam. He runs a LoRa gateway, which is probably the nearest he'll get to his own radio breakfast show.

Often Raspberry Pi computers get unfairly compared with microcontrollers such as Arduino. In fact, they are very different beasts, despite their diminutive similarities. Raspberry Pi has the power of its operating system and better connectivity; Arduino has better support for analogue devices and faster reaction times. So what happens when you need both in a project? You use both, as it's not too complicated to get a Raspberry Pi computer and Arduino talking! To demonstrate, we're going to read the analogue input from a simple £1 photocell using an Arduino and then relay that information to a Raspberry Pi 4.

JUST USB

We'll start with the easiest, simplest way of getting our two friends talking. In the case of an Arduino Uno, simply take a Type 1/2 USB cable and connect the two devices together. The Arduino's serial console is now available to Raspberry Pi OS as **/dev/ttyACM0**.

You can even use the Arduino IDE in Raspberry Pi OS Desktop to reprogram the Arduino. Try building the circuit in **Figure 1**, then download the code from **github.com/mrpjevans/ard2pi/** and send it to the Arduino. Now on the command line from your Raspberry Pi 4, enter this:

```
screen /dev/ttyACM0 115200
```

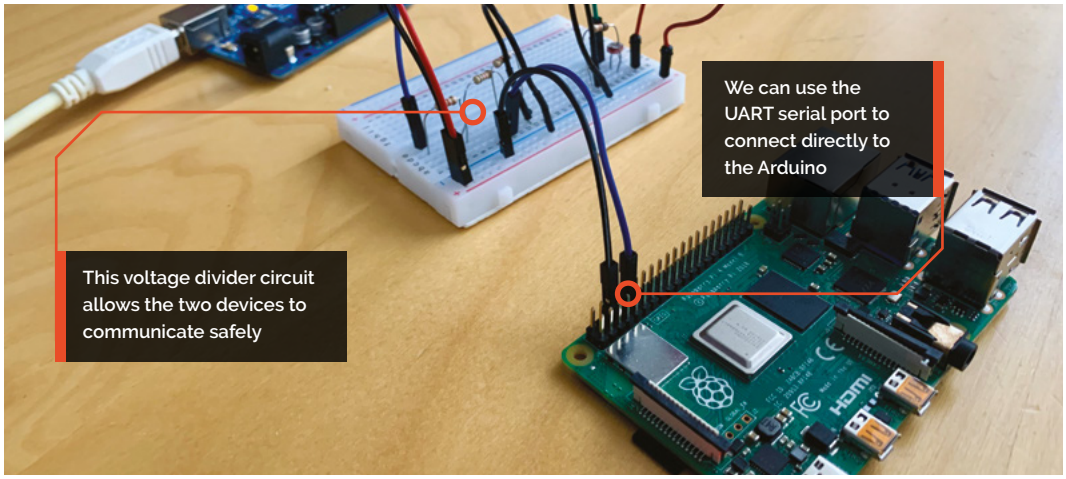
You should see a list of numbers coming in; that's the readings from the photocell. Cover it with your finger and see it change! To quit, press **CTRL+A** followed by **K** and **Y**.

DIRECT UART CONNECTION

A USB cable is simple but bulky. If space is an issue, we can do the same job by connecting Raspberry Pi 4 and the Arduino directly using their serial ports. However, there's a catch. Raspberry Pi runs at 3.3V, and the Arduino at 5V. As soon as the Arduino sends data to Raspberry Pi, it will fry the circuits. The solution is to add a voltage divider, which reduces the signal to 3.3V. First, run `sudo raspi-config` and select Interfacing

YOU'LL NEED

- **Arduino Uno**
- **Raspberry Pi** (except the original)
- **USB cable**
- **Photocell** (e.g. PDV-P8001)
- **10 kΩ resistor**
- **1 kΩ resistor**
- **2 kΩ resistor**
- **Breadboard and wires**



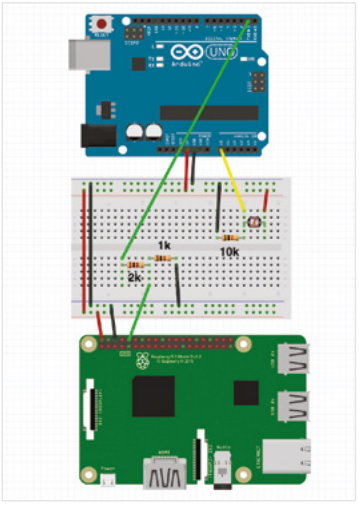


Figure 2 A voltage divider lets us connect serial ports

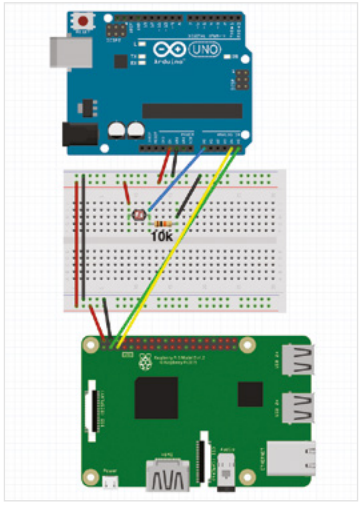


Figure 3 I2C lets us connect multiple Arduinos

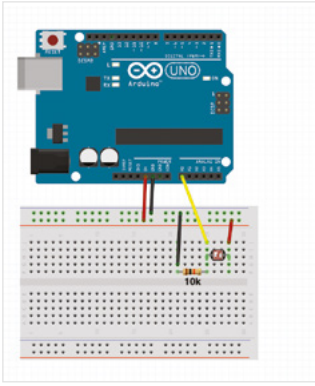


Figure 1 A basic circuit that we can connect via USB cable

Options, then enable the serial port but not the serial console. Build the circuit in **Figure 2** and then on Raspberry Pi, run the **screen** command as before but now pointing to the primary UART:

```
screen /dev/serial0 115200
```

MULTIPLE ARDUINOS WITH I2C

Direct wiring is an elegant solution, but we can get even simpler. I²C (inter-integrated circuit) is a standard interface available on both Raspberry Pi 4 and Arduino Uno that allows a direct-wire connection to multiple devices. Luckily, a quirk of the way they communicate means that we can safely disregard the voltage divider circuit this time. Follow the wiring in **Figure 3** and then reprogram the Arduino with the code from **github.com/mrpjevans/ard2pi/blob/master/ard2pi_i2c/ard2pi_i2c.ino** and use your favourite text editor to create **readi2c.py** on Raspberry Pi with the code from **github.com/mrpjevans/ard2pi/blob/master/readi2c.py**, and run these commands:

```
ip3 install smbus
python3 readi2c.py
```

The first command only needs to be run once. If you get an error, run **sudo apt install python3-pip** and try again.

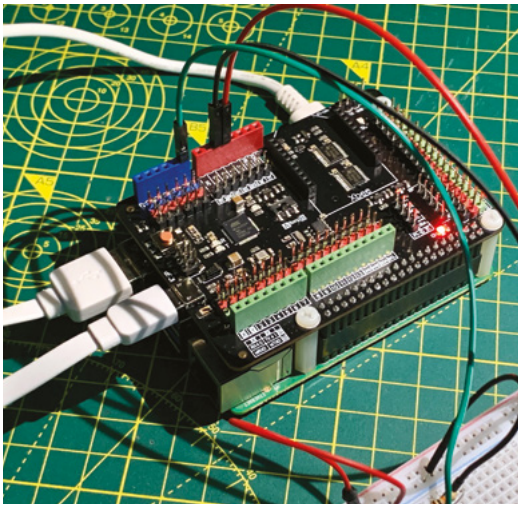
In this scenario, the Python script 'requests' the data and the Arduino responds instantly with the reading.

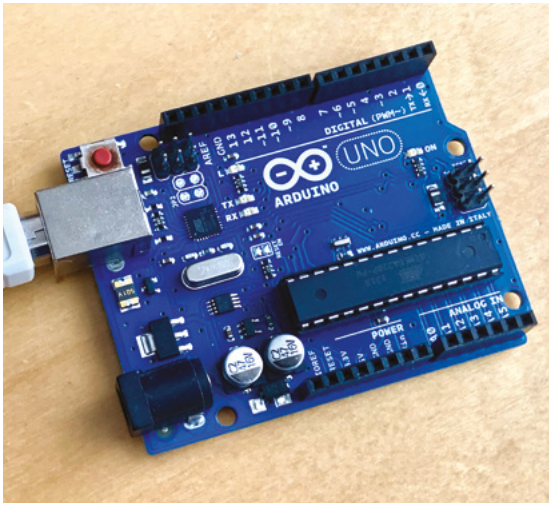
HAT-TRICK

Our final solution is the perfect choice for compact yet demanding projects. DFRobot's DFR0327 is a Raspberry Pi HAT with an on-board Arduino Leonardo. It has a dazzling array of inputs and outputs, with a built-in voltage divider to ensure safe communication throughout. A mount for a standard XBee/ZigBee radio module also features. It's designed for projects where Raspberry Pi does the heavy lifting, such as object recognition, and you take advantage of Arduino's superior reaction times. Not only are all the standard Arduino ports available, but there's a full breakout of the GPIO as well. ◻

QUICK TIP

Warning! Arduinos run at 5V and Raspberry Pi 4 at 3.3V (3V3). The Arduino can destroy your Raspberry Pi 4 if wired together incorrectly.

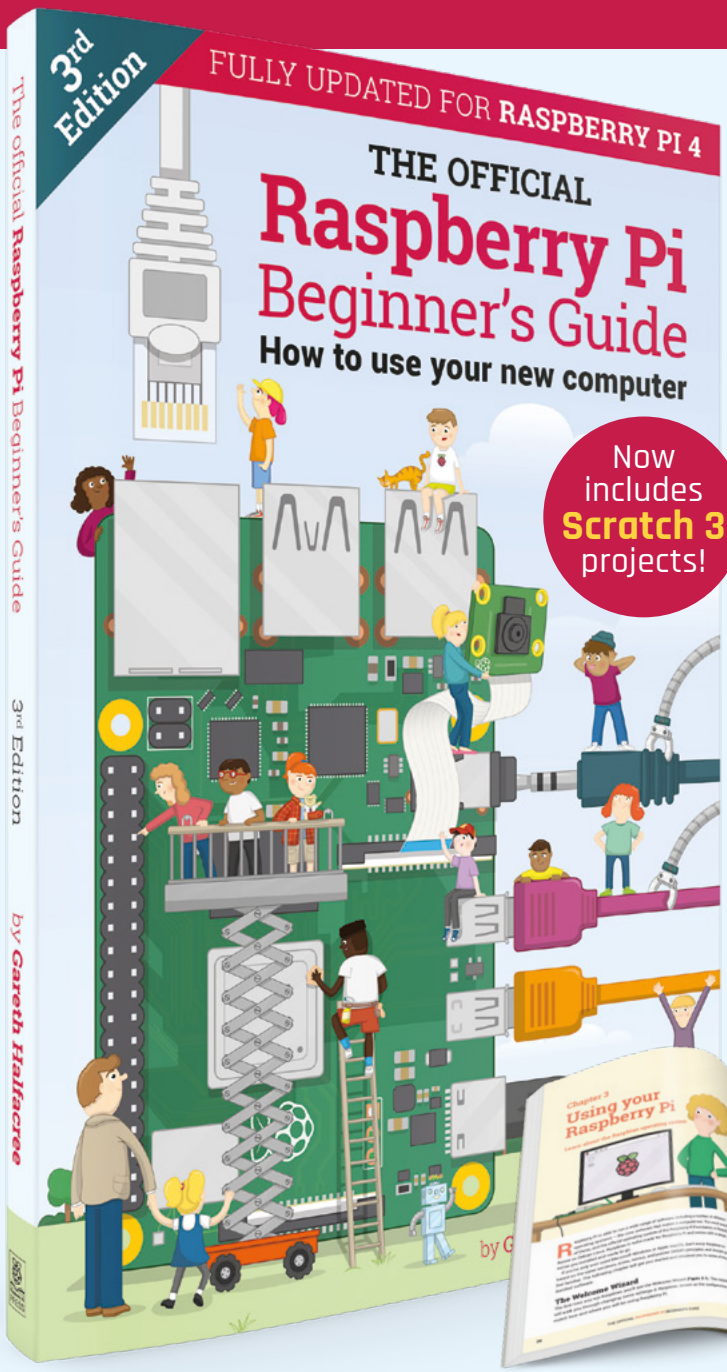




Left A straightforward USB cable is by far the easiest way to connect an Arduino to Raspberry Pi

Far Left This Arduino Leonardo-compatible HAT combines the best of both worlds in an elegant package

THE OFFICIAL Raspberry Pi Beginner's Guide



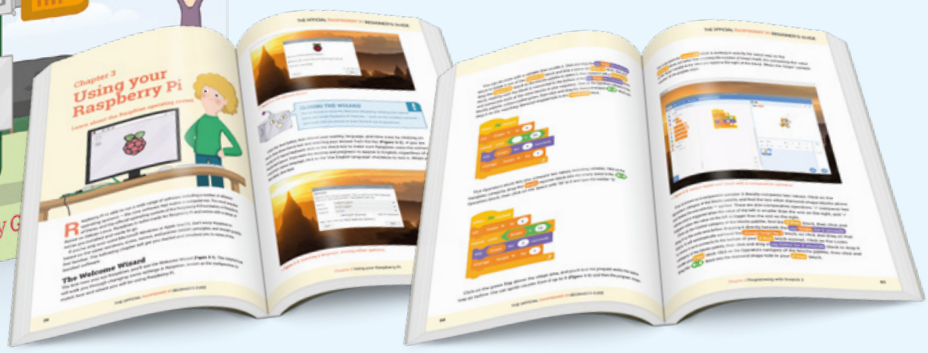
The only guide you need to get started with Raspberry Pi

Inside:

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

Plus much, much more!

£10 with FREE worldwide delivery



Buy online: magpi.cc/BGbook

BOOK OF MAKING

VOLUME 2

ONLY £10

WHSmith

BARNES&NOBLE

Available on the App Store

GET IT ON Google Play

THE BEST PROJECTS FROM HACKSPACE MAGAZINE

THE ULTIMATE SKILLS, TRICKS, AND MAKES

AVAILABLE NOW

hsmag.cc/store

FROM THE MAKERS OF **HackSpace** MAGAZINE

TUTORIAL

Moulding and casting tips for makers

Knowing the skills needed to easily duplicate parts isn't as difficult as you might think

Gareth Branwyn

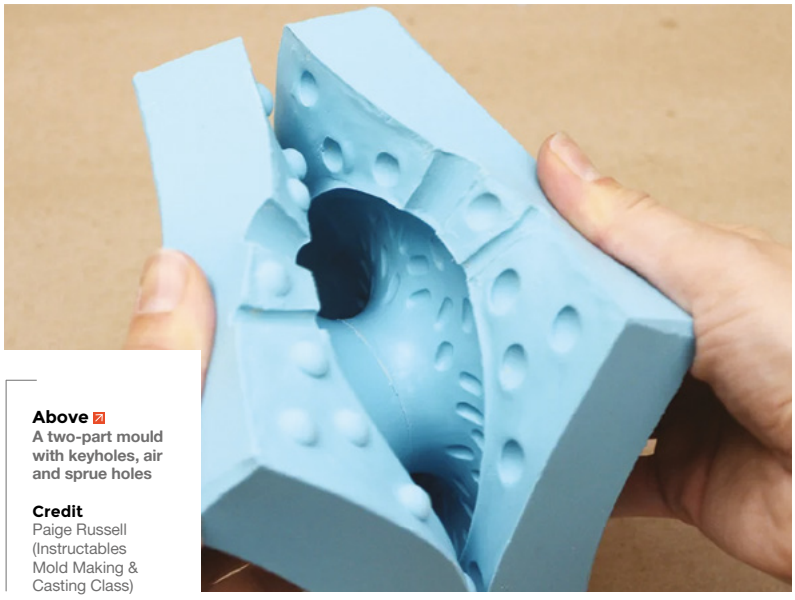
@garethb2

Gareth has been a lifelong practitioner (and chronicler) of DIY tech, media, and culture. He is the author of ten books, including *Tips and Tales from the Workshop*, and is a former editor for Boing Boing and Wired.

Moulding and casting can often seem intimidating to those not versed in the art. But it doesn't have to be. These days, there are all sorts of sophisticated materials for making everything from simple and complex moulds, and an even greater variety of materials available for casting what you wish to duplicate. In a pinch, you can even do respectable moulding and casting with common household and shop material, such as hot glue, decorator's caulk, and even jelly/jello.

How thick should your mould be? You don't want the sidewalls of your mould (the area around the object to be cast) to be too thin, but if you make them too thick, you're just wasting moulding material (and money). Your moulds should be about 1⁄2" to 1" away from the piece you're moulding. It's best to make it a little thicker, closer to 1", if you plan to use a zipper cut (also called a jeweller's cut) when removing the original object from the mould. This will provide more stability, but otherwise, 1⁄2" to 3⁄4" or so is fine. A zipper cut is used in a one-part block mould where you remove the object inside by making zigzag cuts through the silicon moulding rubber. This allows the mould to close up tighter when you are ready to pour your casting materials.

How many are you going to make? Take into consideration the number of copies you wish to make to determine the cost of your mould-making materials. One-offs can be made with much cheaper moulding materials.



Above
A two-part mould with keyholes, air and sprue holes

Credit
Paige Russell (Instructables Mold Making & Casting Class)

Using Lego for mould boxes If you've looked at any moulding and casting how-tos online, or done any yourself, you likely already know this trick, but it's still worth mentioning. Lego bricks make for a perfect, reusable, and resizable mould box, and nearly every hobbyist (and pro) who does casting uses them. They can be used for one-part, two-part, and block moulds.

Reinforcing foam core mould boxes Frank Ippolito, of **Tested.com**, recommends reinforcing pinned-together foam core mould boxes by wrapping them in plaster-infused bandage material. This adds strength and stability and holds the box together tightly so that the weight of the mould material doesn't warp and 'blow out' your mould box.

Don't forget the mould release agent! You can buy commercial release agents to spray or dust on, and you can use things like cooking oil, petroleum jelly,

Above
Glycerine and gelatine work as a mould (youtu.be/AS7dlRPryP8)

cooking spray, and talcum powder. There are tons of recipes online for homemade release agents.

Quick, easy, reusable moulding material You can make decent-quality moulds using a material called Oyumaru, sometimes sold as 'Blue Stuff'. You heat it in water until pliable and then mould it around your object. When you're through with the mould, you can reheat it and reuse.

Use marbles for casting 'keys' Two-part moulds use something called casting keys. These are protrusions on one side of the mould that go into indents on the other side. This ensures that the two parts line up properly and lock into each other for the pour. Marbles make easy positive keys that you can press into clay that you're using to create support around the object you're moulding.

Allow space around an armature When making moulds for latex rubber casts with armature wires inside, make sure your wireframe 'floats' in the mould so that the armature does not protrude out of the final casting.

Recycle old, failed moulds Failed/obsolete moulds can be reused with the same material type if you cut it into small chunks and use it to fill big spaces in your moulds. This will save you money on mould material.

Casting your own knobs If you're working on, say an electronics project, and you want to finish it off with your own custom-designed control knobs, doing so is easy. You can find silicone moulds in all sorts of shapes and sizes (used for making sweets). All you have to do is position your threaded hardware in the centre of the mould and fill with casting resin or other casting material. The YouTube channel Stuff I Made has a video on the process (**hsmag.cc/dkrvML**)

Mould levelling When pouring a cast into a mould with a lot of real estate, getting the pour level can be an issue. To help with this, place the mould on a board and sink four screws into the board like table legs. By adjusting the screws, you can make sure the casting resin is perfectly level in the mould.

Elongate your pour When pouring silicone rubber into a mould, hold up the bucket, or cup of your rubber mix, as high above the mould as possible. This will help break bubbles that might be in the mixture.

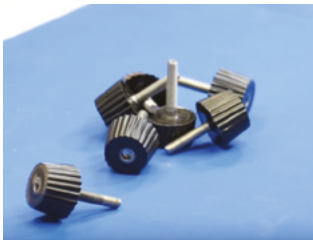
Getting bubbles out of epoxy To get the bubbles out of epoxy (if you don't have a vacuum chamber), use a heat gun or hair-dryer.

Use a vacuum chamber/pressure pot To remove bubbles from resin casts as they cure, use a vacuum chamber or pressure pot. There are dozens of videos on YouTube on how to make pressure pots from cheap tanks used in paint spraying (such as the Harbor Freight Air Pressure Paint Tank) and ones on making smaller pots using little more than some PVC piping, a pressure gauge, and a safety valve. Here's a good video on making a PVC pot (**hsmag.cc/hcjBFn**).

Calculate how much casting material you will need To calculate volume of solid casts, if possible, immerse the object in a bucket sitting in a pan that's filled to the rim with water, and push the object down until it is fully immersed. Measure the volume of the overspill, and that should be quite close to the volume of your object. Add 10% volume to your projected amount of casting material to compensate for leaks or spills.

DETAILS LAYER

Scott of the YouTube channel, Essential Craftsman, has a wonderful statement that speaks to something very important in successful moulding and casting: details layer. The idea here is that, in every step of a project or activity, the precision with which you do one step will carry over into the next, and the next, and the next. Over time, mistakes and imperfections can compound, so it's important to do each step as well and thoughtfully as possible. This concept is critical in moulding and casting. The better, more hi-res the mould, the better the copies. The more time you spend properly setting up the mould box, the better the results. The more precise you are with calculating proper casting mixtures and volumes, following proper demoulding times, etc., the better your casts. It all compounds. Because details layer.



Above
Cast your own knobs to add a personal look to your projects

Credit
Stuff I Made

THANKS

Andy Birkey, Miguel Valenzuela, Becky Stern, Chris Gilroy

Below
Put threaded screws into the corners of a board to make an adjustable, level table for pouring

Credit
Andy Birkey

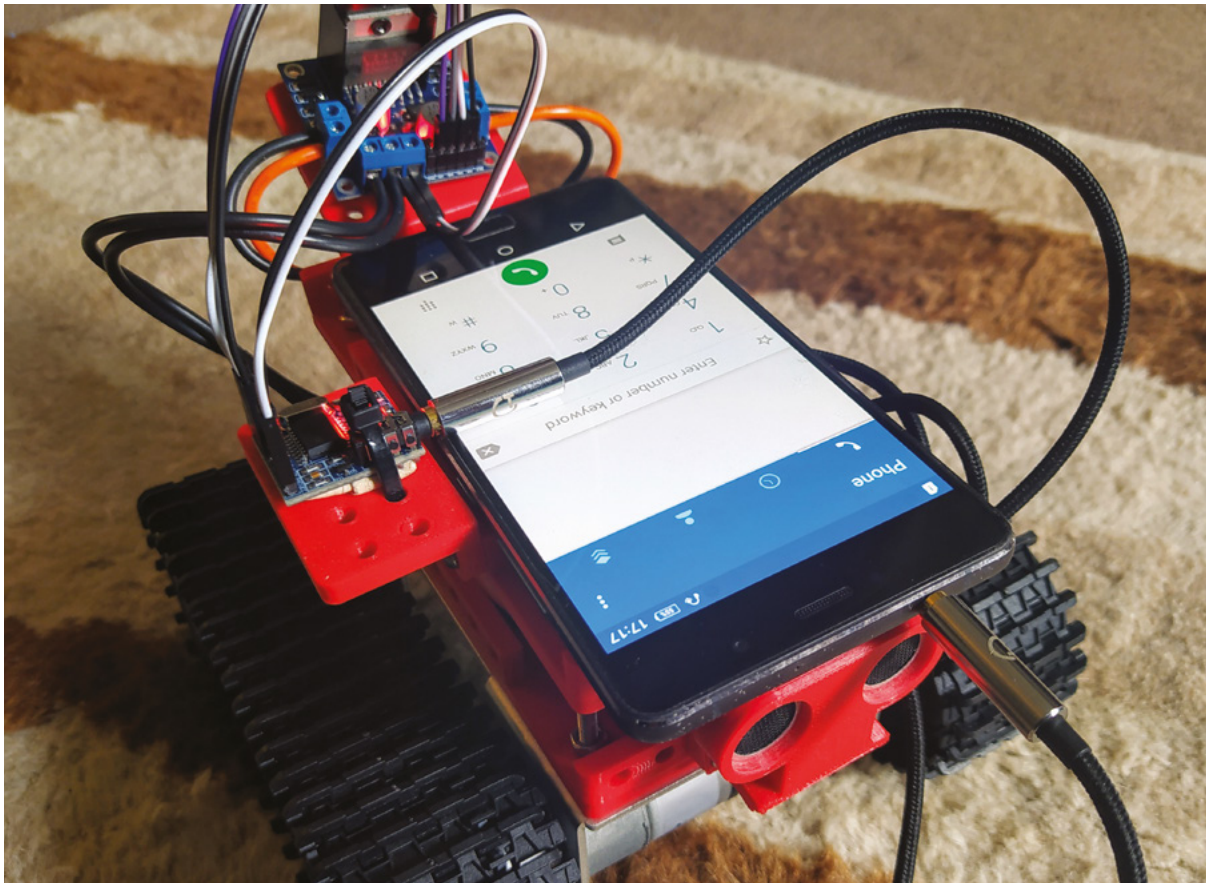
Playing with DTMF

Dual-tone multi-frequency is a telecommunication signalling system. Let's explore how it can be used as a control system in projects

Jo Hinchliffe

@concreted0g

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



Above
We used the MTV robot as a test platform for numerous experiments with DTMF control

Dual-tone multi-frequency tones are the type of sound you hear when you are asked by an automated telephone menu system to press a number on your keypad to interact; for example, 'press 1 for option A'.

As most keypads on phones have twelve keys, this means that there are separate tones for 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, *, and #, but the full DTMF system also has tones A, B, C, and D available, giving it 16 different tones to utilise. These days DTMF tones are more often generated in software, but the way they are generated in older hardware is quite ingenious.

A DTMF generator chip is capable of generating eight tones of different frequencies. Four of these tones are lower frequency, and four are higher frequency. The generating IC connects these frequency outputs to either a 4×3 switch keyboard or, for full capability DTMF, a 4×4 keyboard. The low-frequency group of four tones are connected to the four rows of the keyboard, and the high-frequency group are connected to the columns of the keyboard (**Figure 1**). Pressing a key on the keyboard essentially creates an algebraic sum of the two tones so that each key in the 4×4 keyboard matrix has a unique tone. As these tones sit within the range of human voice and

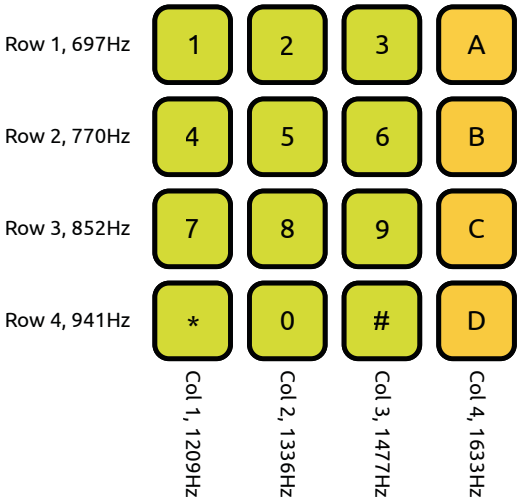


Figure 1
The DTMF keyboard layout and the corresponding tones for each row and column

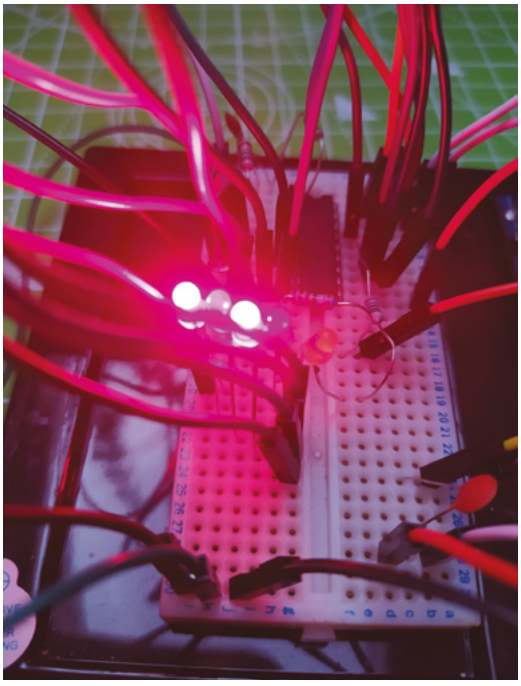
hearing, they are perfect to be transmitted on voice-capable systems, such as telephones.

In a hardware DTMF system, the receiving end decodes the tone pair and usually creates a 4-bit binary output which represents the original key press. A common IC, which we have played with for this article, is the MT8870 IC which has four output pins that when connected, to perhaps four LEDs, will display a binary representation of the original key press. The 4-bit outputs are listed in **Figure 2**.

The MT8870 IC is available in many formats, and also in some pre-built modules. We first explored the MT8870 by building this circuit on a breadboard (**Figure 3** overleaf).

We attached a yellow LED to the STd output, and four red LEDs to the Q1, Q2, Q3, and Q4 outputs. The yellow LED momentarily flashes when the IC detects a new DTMF tone has been received, and the red LEDs show the decoded 4-bit binary output

DTMF tone	Binary output	Left track	Right track
0	1010	FW	FW
1	0001	REV	STOP
2	0010	FW	STOP
3	0011	STOP	STOP
4	0100	STOP	REV
5	0101	REV	REV
6	0110	FW	REV
7	0111	STOP	REV
8	1000	STOP	FW
9	1001	REV	FW
*	1011	STOP	FW
#	1100	STOP	STOP
A	1101	REV	STOP
B	1110	FW	STOP
C	1111	STOP	STOP
D	0000	STOP	STOP



(**Figure 4**). We made an input for this circuit by connecting a 3.5mm stereo jack socket with the right channel connected as input – we then used an audio cable to attach the breadboarded circuit to our mobile phone headphone socket. We turned everything on and then tapped some keys on the keypad of our mobile phone to generate some DTMF.

CRANK IT UP!

Initially we didn't get anything happening until we realised that when we plugged the 3.5mm jack into our Android smartphone, it detected it as headphones and, for safety, reduced the output volume. Make sure you put your sound up to full volume!

Having built the breadboard version of the DTMF decoder, we also picked up an MT8870 module online for around £4. This is essentially the same as the circuit we built on the breadboard but uses an SMD version of the MT8870 on a PCB with everything added and ready to go. As this is functionally identical to the circuit we breadboarded, we continued using this module as it was more robust than the breadboard version we had put together. We discovered with the MT8870 module that there are some with the VCC and GND pins reversed when compared to others; they are labelled correctly but routed in different positions. Make sure you double-check your pins if you have more than one instance of this module. →

YOU'LL NEED

- MT8870 DTMF decoder module
- 3.5 mm stereo jack lead
- Jumper wires
- Android phone
- A cheap Bluetooth headset
- L298N motor driver board and motors

We also used, and you may optionally want

- MT8870 18-pin DIP IC
- 3.57MHz crystal
- 2 × 100 kΩ resistors
- 1 × 390 kΩ resistor
- 5 × 220 ohm resistors
- 2 × 100 nF capacitors
- Breadboard and wires

Figure 4
Breadboarding a DTMF decoder using an MT8870 IC capable of decoding tones from a mobile phone and displaying the output on LEDs

Figure 2
The first two columns show the key presses and the corresponding 4-bit binary output generated for each key. The final columns show how this controlled the MTV robot motors when we connected the MT8870 to the L298N

TUTORIAL

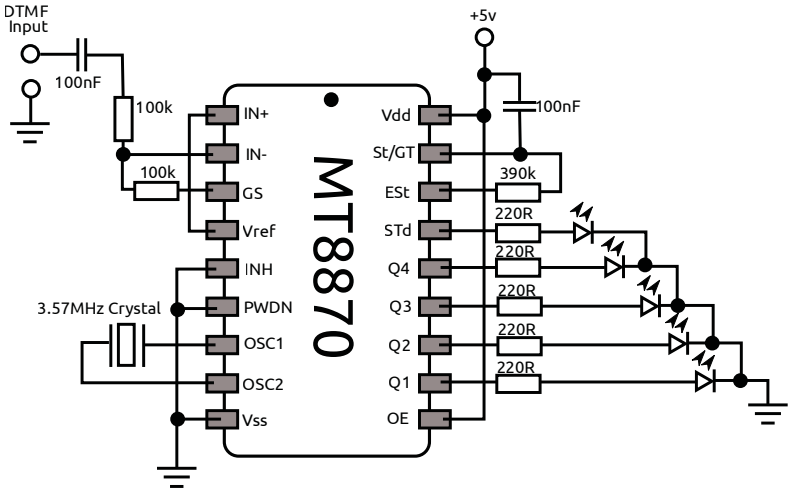


Figure 3 Schematic for a DTMF decoder circuit using the MT8870 IC

The 4-bit outputs from a receiving end of a hardware DTMF system are at logic level, so it means that we can use these 4-bit words to interact with other hardware. At this point, we'd seen a lot of people jump into projects where they connect the four 'Q' output pins of the MT8870 to a microcontroller to control what happens when certain states are detected. There are lots of examples online of people doing this, but we wanted to start with something simpler. In fact, we decided that we didn't need to involve a programmable microcontroller at all.

> We decided that we didn't need to involve **a programmable microcontroller at all**

In issue 32, we used the L298N motor controller on the open-source modular tracked vehicle (MTV) robot that we designed and built. The L298N motor controller can be set up to turn two attached motors off and on – it can also toggle the polarity of the motor supply, reversing the direction in which the DC motors turn. As we have a 4-bit binary output from the MT8870 module, we can use those bits to control the four input pins on the L298N to make a crude but working DTMF control system for the robot.

We wired up the motors of the MTV to an L298N motor driver, and we connected Q1, Q2, Q3, and Q4 on our MT8870 module to the input 1, 2, 3, and 4 respectively (**Figure 5**). The MTV was set up to use a three-cell LiPo battery, but conveniently the L298N motor driver has a regulated 5V output when it's powered by 12 volts – we used this to power our MT8870 module. We kept the system wired in the first instance by connecting our mobile phone to the module with a 3.5mm stereo jack. Pressing different numbers on the keypad toggles different pins on the L298N, resulting in different combinations and directions of movement of the motors driving the two tracks of the MTV robot. These are listed in the final two columns in **Figure 2**.

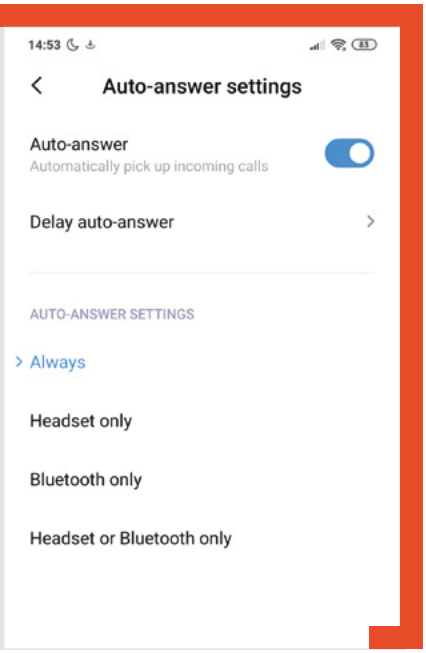
SEND & RECEIVE

An obvious next step is to consider how to send and receive DTMF tones to a system wirelessly. The first option was to grab another phone and plug it into the MT8870 decoder module. You can then, of course, call that phone from another handset and use the other handset to press keypad keys and send tones. This works really well and, of course, that means that your DTMF-enabled robot, or another system, is controllable from any phone, anywhere on the planet.

There are some caveats and challenges, however. For instance, we found that there is a considerable time lag in phone calls. In our case of testing it with the sprightly MTV robot, there were numerous crashes because the delay made it quite difficult to control. Of course, if you were using DTMF to control a slower or static non-time-sensitive system, then this wouldn't be a problem.

Another challenge with the phone-to-phone system is how to get the receiving phone on the robot/system to automatically answer calls. With a little research online, we discovered that modern Android phones can be set up to do this. Open your dialling application and then open the menu and select 'Settings', then select 'Calling accounts' – you should find an 'Auto-answer settings' option in which you can set the phone to automatically pick up all calls. Obviously, you may not want this to be enabled for all calls; however, we also discovered there are numerous free Android applications which will allow auto-answering and can be configured to only automatically answer calls from a certain number.

The other issue with phone-to-phone connectivity was that we could not get our DTMF applications to send their audio into the call, so while we managed to use the 0–9, *, and # keys from our in-call keypad, we could not find a method to send the A–D tones. A classic, almost historical, hack which we are keen to try for this is to build a hardware DTMF dialler using a TP5089N IC with switch keyboard, amp, and speaker that can audibly play DTMF tones. This, in turn, could be held up to a handset to play DTMF sequences 'down the line'.



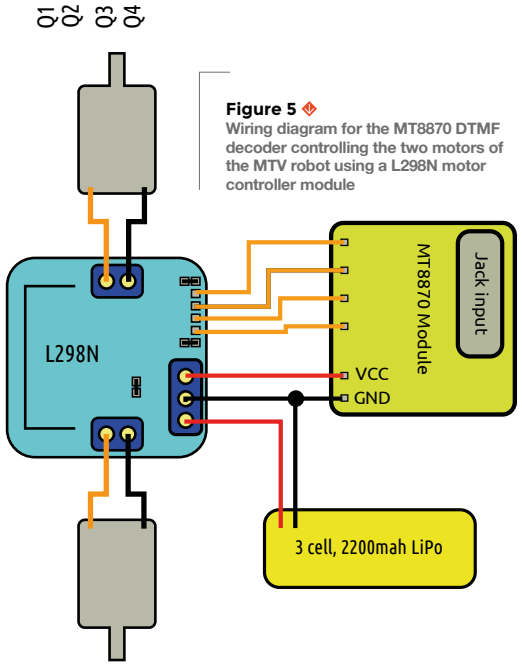
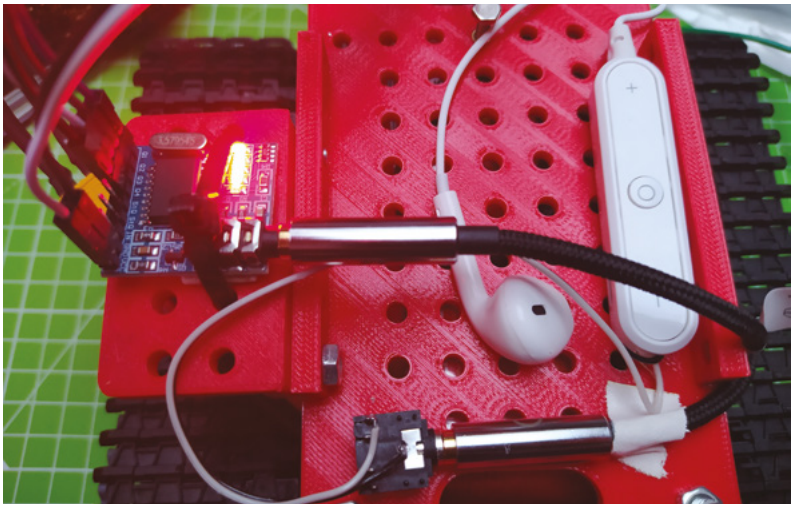


Figure 5
Wiring diagram for the MT8870 DTMF decoder controlling the two motors of the MTV robot using a L298N motor controller module

Using the phone keypad allows us to send 0–9 and the * and # tones, but not the A–D tones. To check that those tones were working, we downloaded and tested some free DTMF-generating Android applications which had the full keyboard, complete with the A–D tones (**Figure 6**). Interestingly, we had a mixture of results. There were a lot of free DTMF generator apps – some worked perfectly, but others didn't work as well. Perhaps differences in volume output, signal quality, or indeed how they generate the DTMF may come into play. Some applications have more advanced settings that allow you to create a queue of DTMF tones to be played back in sequence, and also have settings to vary the duration of tones and the duration of gaps between tones. All of this is fun to explore and play with.

Below
Hacking a cheap Bluetooth headset – replacing one of the earbuds with a 3.5mm jack socket output allows a cheap wireless Bluetooth to DTMF link to be created



LET'S GET WIRELESS!

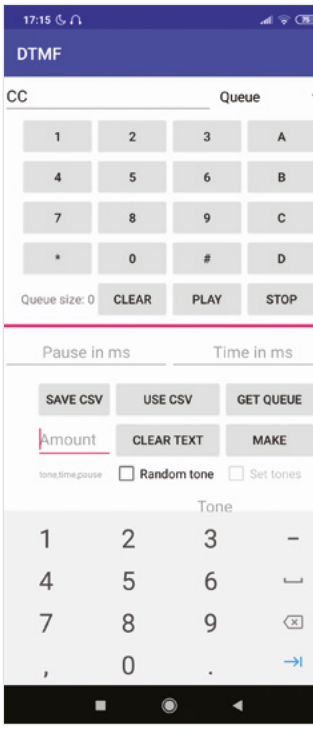
Another option we explored successfully was using a hacked Bluetooth headset to create a wireless audio link to the MT8870 decoder. We bought an unbranded set of Bluetooth headphones online for £3.75. When they arrived, we verified that they worked as intended by pairing them with our Android phone and playing audio through them. They worked surprisingly well, although we discovered that they summed both sides of the stereo audio image to mono for each earbud; this didn't matter at all for our hack, but might be annoying if you wanted them to listen to music with.

We double-checked that the Bluetooth headset played the DTMF tones from our free DTMF tone generator applications and confirmed that they worked. We then cracked the case on the Bluetooth headset and desoldered one of the earbuds. We replaced the earbud wire with a pair of wires connected to a 3.5mm jack socket so that we could easily connect it to our MT8870 module with an audio cable. Our earbud connections consisted of two signal lines and no ground connection to the earbud, which explained why they summed the audio in each channel, so we connected our hacked-in wire to one of the signal lines and soldered a ground wire to the negative battery connection.

The MT8870 module is set up with the incoming signal received from the right-hand audio channel connection of the 3.5mm stereo jack socket. That equates to the middle isolated ring on a jack plug, and so we spent some time with a multimeter working out which pin on the jack socket we were connecting to the newly hacked headset output to make sure our signal went through the correct audio channel. Connecting it all together, we now have a working Bluetooth connection to our MT8870 module. Pressing any of the 16 keys on the DTMF generator applications, or sequencing and sending queues of DTMF tones, works flawlessly.

Above
The combination of the tiny Bluetooth headset and the MT8870 module made for a tiny control system for the MTV robot. If we rewired it without the audio cable, it would be even smaller

Figure 6
A screenshot of just one of the many free DTMF tone generator applications we found for Android phones

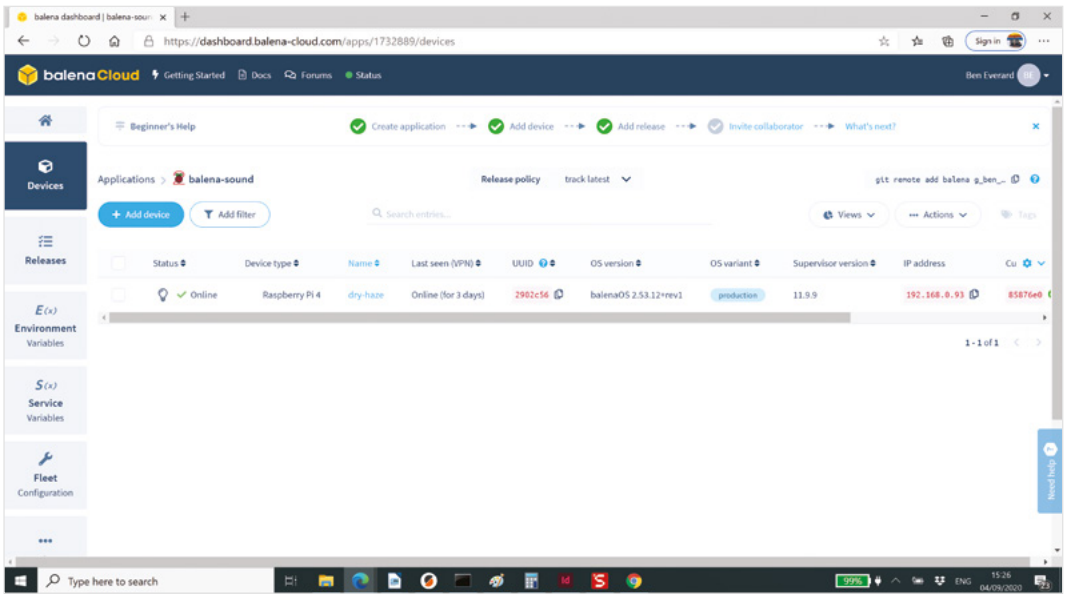


TUTORIAL

Sponsored by **Arm DevSummit**

Stream audio on Raspberry Pi the easy way

Use containers to deploy audio software in just a few clicks



Right
You can manage all your machines from one place

Ben Everard

@ben_everard

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

The popular Linux-based operating systems running on most Raspberry Pi computers have a number of features we can make use of which make deploying our required software easier. When we talk about software, we usually mean a few bits of related software that work together. Your embedded device may need your main application code, plus some libraries installed to support that. It may need a database. It may need some external software tools. It may need a web server. Installing and integrating all of these on one device can be a bit of a chore, but installing them all on a whole fleet of embedded devices? Then keeping them all updated? That's a recipe for a raft of frustrating issues and problems.

One option would be to create an SD card image that you can flash onto all the devices with everything you need. This would be quick to do the first time, but if you wanted to update anything, you'd need to go around and reflash every card. Also, what if you didn't want exactly the same bits of software running on each device? Instead, what if you wanted a selection of software (plus configurations) that you could install to a selection of devices?

Containers provide a handy solution to this. They allow you to bundle up a selection of software,

HOW CONTAINERS ARE CREATED

We've looked at how to deploy a container, but how exactly are these containers we deploy created? Let's take a look at the configuration files used to set everything up. They're called Dockerfiles, named after the underlying container technology and contain a series of steps to run. In our audio example, each audio option (Bluetooth, Spotify, AirPlay, and UPnP) is handled by a different Dockerfile. The Spotify one, for example, can be found at **hsmag.cc/DsLLz9** and contains the following:

```
FROM balenalib/%%BALENA_MACHINE_NAME%%-debian:buster

RUN curl -sSL https://dtcooper.github.io/raspotify/key.asc
| apt-key add -v - 2> /dev/null \
  && echo 'deb https://dtcooper.github.io/raspotify jessie
main' | tee /etc/apt/sources.list.d/raspotify.list \
  && install_packages raspotify

# Audio block setup
ENV PULSE_SERVER=tcp:localhost:4317
RUN curl -sL https://raw.githubusercontent.com/balena-io-
playground/audio-primitive/master/scripts/alsa-bridge/
debian-setup.sh | sh

COPY start.sh /usr/src/

CMD [ "/bin/bash", "/usr/src/start.sh" ]
```

The first line tells Balena what the base image for this container is. Because Balena supports a wide variety of boards, it has to match this to the particular machine you're installing the software on, hence the %%BALENA_MACHINE_NAME%% part of the name. This will be replaced by your actual hardware and hence link to the correct image. In this case, we're using the Buster version of Debian, but there's a wide range of options. Take a look at **hsmag.cc/nkRwtL** for details.

The second part of this file (titled 'Audio block setup') gets the sound configured correctly using PulseAudio. First it sets an environmental variable, then it runs a command that downloads a shell script and pipes it into .sh (SH) – in other words, it runs the script.

The final part copies the **start.sh** script (which is in the GitHub repository) and runs it in Bash. In many ways, this is similar to the previous section in that it runs a script from the web. The previous method gets a script from a different GitHub repository, and this method gets it from the same GitHub repository.

That's all there is to building containers – if you're comfortable working at the command line, it's really just a case of converting the commands you'd type into Docker commands that are run automatically.

This is how we build a single container, but this particular project is made up of multiple containers running together. Take a look at the box on Docker Compose for more details.

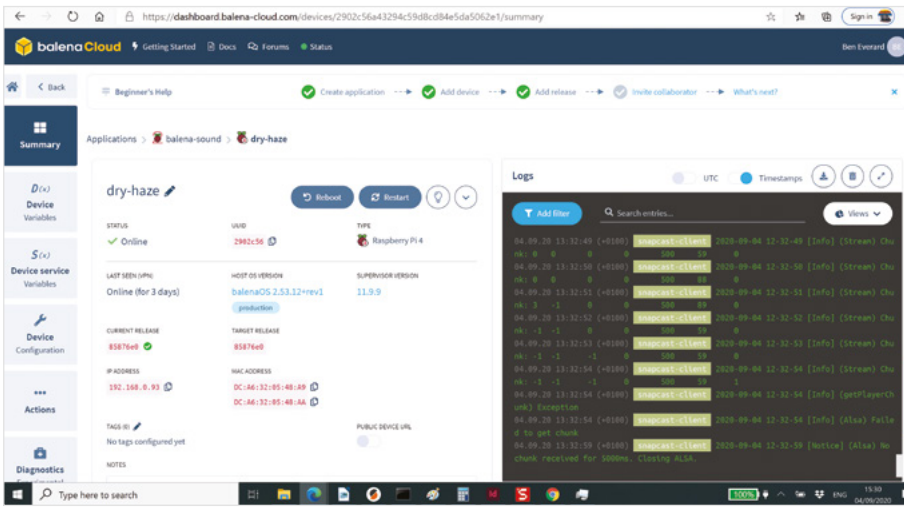
configurations, and other bits and bobs into a 'container' that runs on an embedded Linux device. The underlying technology of containers is a little complex, but once the framework is in place, they're fairly straightforward to deploy and maintain. Each 'container' that you run on your computer runs completely independently. It doesn't know about the other containers running on your computer. This way, if you update one container, it doesn't disturb the other containers. What's more, upgrading is easy – you simply download an upgraded container and run that in place of the previous container. If there are any problems, you stop the upgraded container and restart the previous version.

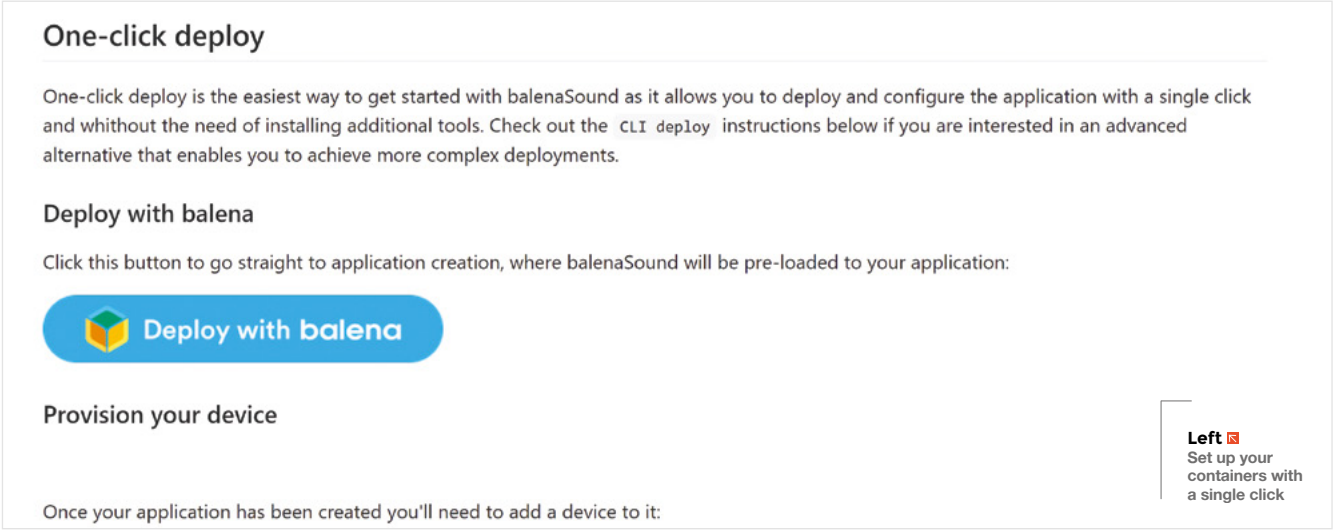
> Once the framework is in place, **they're fairly straightforward to deploy and maintain**

There are a few technologies available for running containers, but Docker is one of the most popular. This is mostly used for running services in data centres and on servers, so the tools are geared towards this use case. However, we can make use of the same technologies on embedded devices.

Balena provides a set of tools for managing containers on embedded Linux devices such as Raspberry Pi computers. We caught up with Developer Advocate at Balena and Arm Innovator, David Tischler about what this means for setting →

Below
View your logs from the web interface





Left
Set up your containers with a single click

DOCKER COMPOSE

A container lets you group together bits of software into a unit that runs independently. However, sometimes you need multiple bits that each run independently. You could put them all into one container, but you might want to mix-and-match them between machines, and update them separately. In these cases, it's better to separate each bit into its own container.

We have a set of Dockerfiles for building containers in both the **core** and **plugins** folders of the main GitHub repository (see here: **hsmag.cc/ISFsNu**). The **docker-compose.yml** file brings these together and tells Balena what we want to do with all the containers. YML stands for Yet Another Markup Language, and is a bit different from the format of the Dockerfiles. It's a hierarchical structure with tabs denoting what level a line refers to.

Continuing on with the Spotify example, let's take a look at the section in **docker-compose.yml**:

```
    build: ./plugins/spotify
    privileged: true
    network_mode: host
    volumes:
      - spotifycache:/var/cache/raspotify
```

Here we tell Balena where to find the Dockerfile (in **plugins/spotify**). Privileged mode gives the container more access to the underlying system than containers typically have (see here for details: **hsmag.cc/y6e1uB**). `Network_mode` sets the way that the system shares its network access with the different containers running on it.

Containers wrap up the file system of the machine into a bundle that can be easily moved between machines. When you build a container using Docker, you're really recreating a file system plus setting some software running. This makes it really easy to get a consistent environment across many machines, but in some cases, you want things outside of this consistent file system. This is particularly for things that you want to be preserved on the system after you upgrade the container. In this case, the Spotify cache. Docker volumes give you storage space outside the main container that will persist even when you upgrade the container.

The full **compose** file contains similar sections for each of the different containers that run on the build.

up your containers by building an audio system. In this system, you can use one or more Raspberry Pi boards (any model) to play music from Spotify, Bluetooth, AirPlay, or UPnP. They'll work together, so you can stream music to multiple rooms around your house and it'll all be in sync.

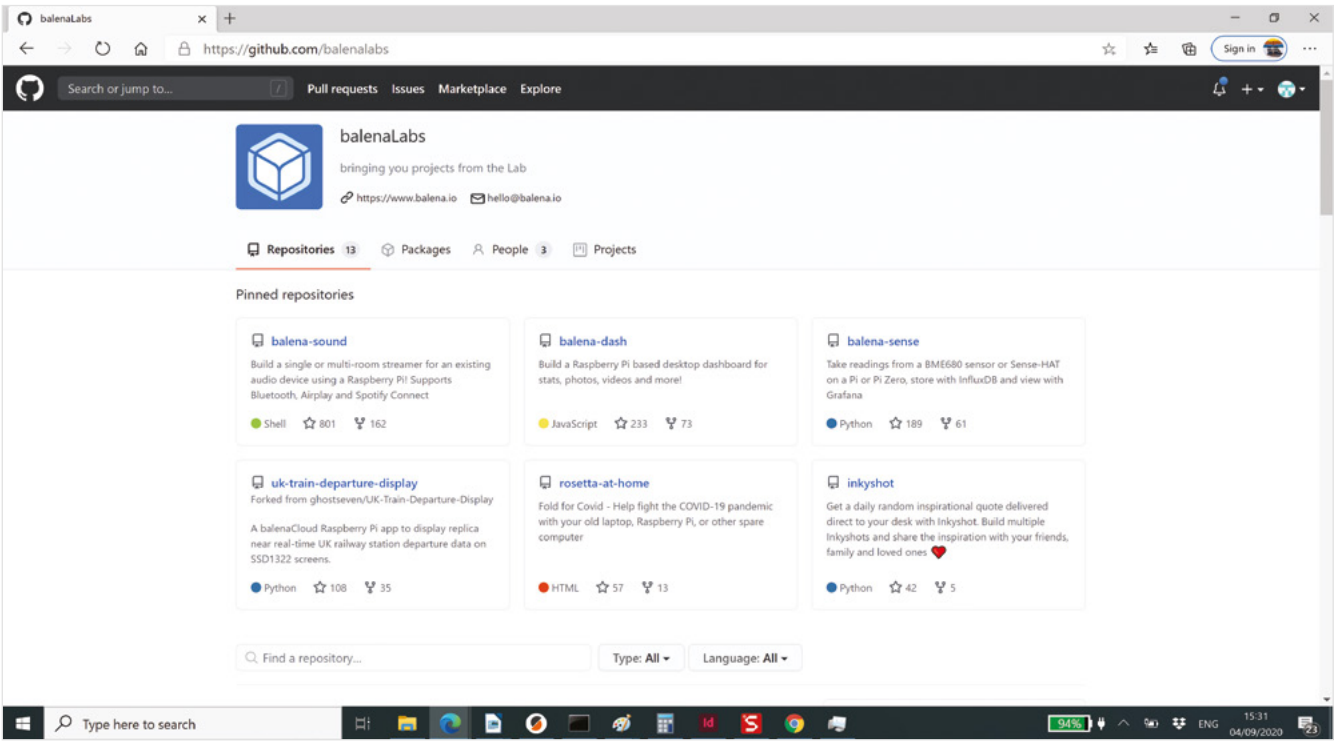
We'll be using the latest version of Balena Sound (3.0) which, at the time of writing, is still in testing, so we'll grab it straight from the v.3.0 source tree here: **hsmag.cc/IY9JPA**. You can get the main version from the source tree here: **hsmag.cc/H8GckY** which may be version 3 by the time you read this, or it might still be version 2.

> They'll work together so you can **stream music to multiple rooms around your house**

From a user's perspective, both work in similar ways, but version 3 should be a bit more robust.

Balena is a cloud-based service, so the first step is to create an account. You can get started with your first ten machines for free – sign up at **balena.io**. Once you've got an account, you need to create a new app. You can do this manually, but the easiest way is via a one-click link. For Balena Sound v3, the link is: **hsmag.cc/uO18qn**.

Enter this, and it'll take you to the Balena dashboard and create the app for you. However, it won't yet have any devices attached to it, so



the next step is to add one (or more). Click on 'Add device', and it'll open a pane where you can select the type of device you want. There's a wide variety available, but Raspberry Pi 4 Model B is a good choice as it has WiFi and audio-out (via the headphone jack). You can use a Raspberry Pi Zero W, but you'll need to add additional hardware to get audio.

If you're using a device with WiFi, be sure to check the 'WiFi+Ethernet' box as this will let you enter connection details so the machine can connect to the network. Finally, hit the download button to get the image to flash to your SD card.

You can flash this in the normal way, such as with Balena Etcher (**balena.io/etcher**), pop it in your machine, and boot it up. You should then see your device listed on your applications dashboard. You can find this at **dashboard.balena-cloud.com/apps** if you've shut the window.

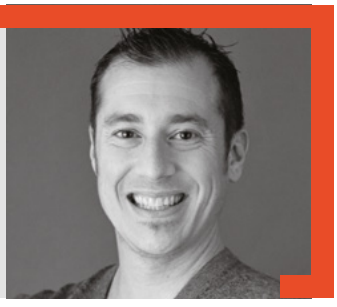
At first, you'll see that it takes a little while to update itself, but once this is finished, you should have a running audio system. You'll need some way of listening to the audio – for testing, this could just be headphones plugged into the audio jack, but for ambient listening, you will probably want to take the output from Raspberry Pi into an amplifier, then drive some speakers off that. One option is to take the output from Raspberry Pi into the input of an old stereo. This is a great way to keep your old audio hardware running in the internet-streaming era of music.

With all this setup, you can now play music through your device. If you set up more than one device on your local network, they will all play the same music perfectly synchronised so you can enjoy your tunes in every room in the house.

We've taken a look at how Balena makes it straightforward to get software in containers out to our devices with minimal fuss. In this case, streaming audio, but it can work with almost any software. There's a range of other examples at Balena Labs (**github.com/balenalabs**), or you can create your own.

Above BalenaLabs have a range of different containers ready to go

ONLINE WORKSHOP



If you'd like to use Balena to help automate IoT deployments, you can join David Tischler's workshop at Arm DevSummit. At 9.30 PDT on Thursday 8 October, David will take you through how to create your first containerised IoT device with Balena.

Find out more and add the event to your personalised schedule at **hsmag.cc/LLY5b0**.

TUTORIAL

Add uncanny eyes to your Halloween project

A microcontroller and a TFT display combine to create a creepy feeling

Ben Everard

@ben_everard

Ben loves combining Halloween with microcontrollaaaagghhs and bad puns.

2020 has been a strange year – **in many ways it feels like we've missed the summer, but there's no denying that autumn now approaches.** For many makers, this means a chance to show off our creepy creations designed for Halloween.

Animated eyes – aka uncanny eyes – have been around in the maker world for a while. Phil Burgess first created the code for the Teensy microcontrollers in 2015. However, even after five years, they're still something of a rarity outside the maker scene and have a real wow factor. Since the original Teensy code, it's been ported to support a range of ARM Cortex-M0 and Cortex-M4 microcontrollers. Adafruit supports it on a number of its boards (see this link for details: **hsmag.cc/RRrqOF**). It's quite hardware-specific, so make sure that it will work on your intended microcontroller before you purchase your hardware.

We've used the Adafruit MONSTER M4SK, which features a SAMD51 microcontroller and two 240 × 240 screens. In this tutorial, we won't be looking at getting the main software up and running. You can sidestep this anyway by downloading precompiled firmware. There's some here (**hsmag.cc/eGBvsR**) for the Circuit Playground Express that HackSpace magazine subscribers receive, and you'll also find it on the Adafruit product pages for the Hallowing and MONSTER M4SK.

PARAMETER PUZZLES

There's a range of parameters you can customise for the eyes. We'll be using the MONSTER M4SK for this article as it makes tweaking the parameters easy. If you're using a different controller, you can make most of the same tweaks, but you'll need to update the settings slightly differently. See **hsmag.cc/IFnHez** for further details.

C:\Users\ben\AppData\Local\Temp\Temp1_eyes (1).zip\M4_Eyes\eyes\hazel\config.eye - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

config.eye

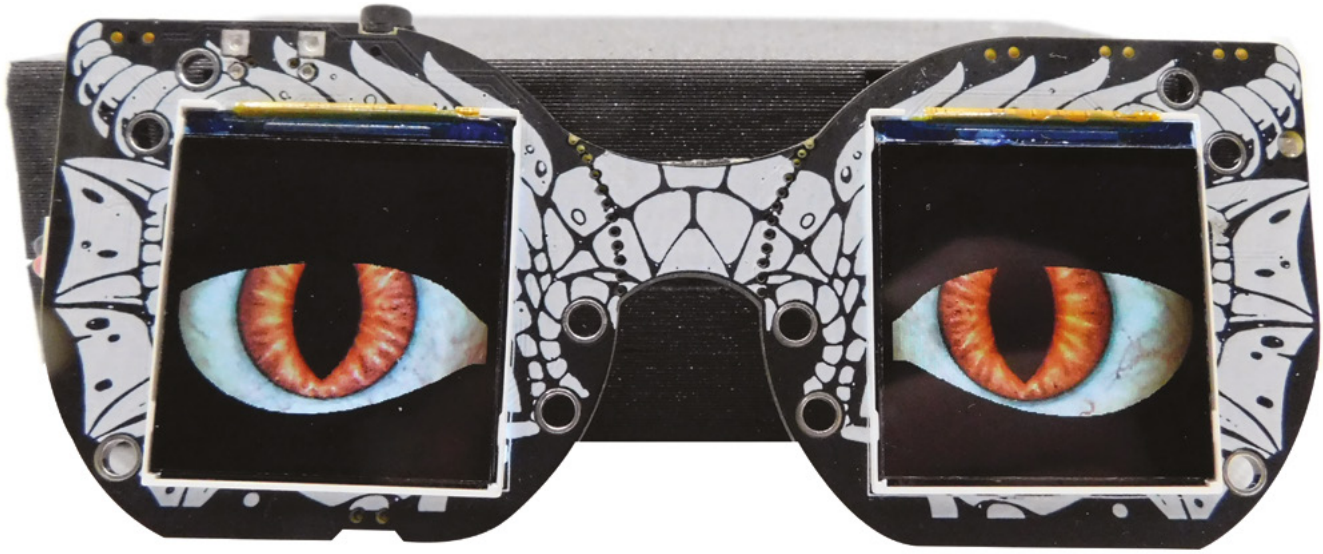
```
{
  "boopThreshold" : 17500, // lower = more sensitive
  "eyeRadius"     : 125,
  "eyelidIndex"   : "0x00", // From table: learn.adafruit.com/assets/61921
  "pupilColor"    : [ 0, 0, 0 ],
  "backColor"     : [ 140, 40, 20 ],
  "irisTexture"   : "hazel/iris.bmp",
  "scleraTexture" : "hazel/sclera.bmp",
  "upperEyelid"   : "hazel/upper.bmp",
  "lowerEyelid"   : "hazel/lower.bmp",
  "left" : {
  },
  "right" : {
  }
}
```

The eyes are controlled by a configuration file called **config.eye**. This is loaded when the microcontroller boots, so if you make any changes, you need to restart the board in order to pick up the new config. This can be done either by turning it off and on again or pressing the reset button.

You can download a range of preconfigured graphics from **hsmag.cc/hpoXt8**. In the zip file, there are a number of folders. Each one includes the **config.eye** file and the graphics. You just need to copy everything to the device, then copy the relevant **config.eye** file to the root of the microcontroller USB drive.

You can edit **config.eye** in any text editor, but it is very pernickety, and the formatting has to be correct. If at some point, you find that the eyes load with just basic colours and no images, then it's probably because there's an error in the formatting of the file. Watch particularly for a comma after every item except the last one (which must not have one). The whole section is enclosed in curly braces, and each item is *name : value*, with the name in double quote marks. For example, the basic hazel eyes are:

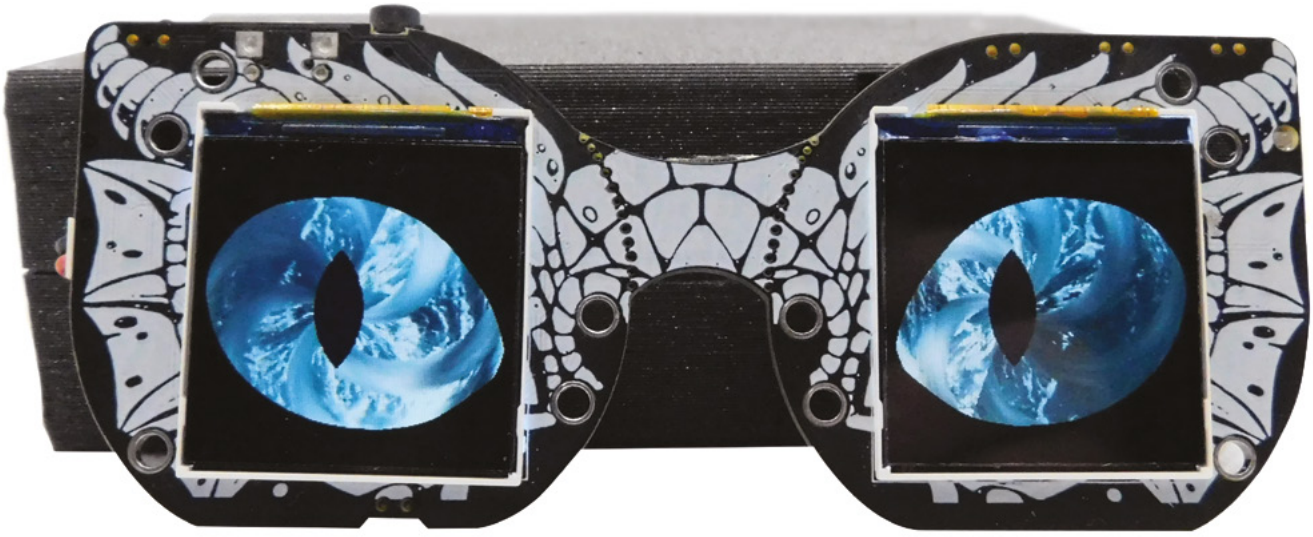
```
{
   "boopThreshold" : 17500,
   "eyeRadius"     : 125,
   "eyelidIndex"   : "0x00", // From table:
learn.adafruit.com/assets/61921
   "pupilColor"    : [ 0, 0, 0 ],
   "backColor"     : [ 140, 40, 20 ],
   "irisTexture"   : "hazel/iris.bmp",
   "scleraTexture" : "hazel/sclera.bmp",
   "upperEyelid"   : "hazel/upper.bmp",
   "lowerEyelid"   : "hazel/lower.bmp",
   "left" : {
   },
   "right" : {
   }
}
```



Above
All these eyes are just a modified text-file away

Left
You can create a wide range of different eyes by combining the examples in different ways

TUTORIAL



Above
A look into my eeeeeyyyyyeeeesss

Below
Want to super-size your eyes? Keep an eye on adafruit.com for details of how to use LED matrixes

Perhaps the easiest way of seeing what things do is changing them and seeing the result. Let's take a look at how one we've created compares to this. We wanted to create a slightly unnatural eye which gave a hint of monster-y-ness without being too over the top.

Eye radius is, fairly obviously, the radius of the eye in pixels. Since the screens we're using have 240 × 240 pixels, a radius of 130 puts the eyeballs slightly larger than the screens. This isn't a problem, as our eyelids cover the edges of our eyeballs anyway.

Speaking of eyelids, their colour is defined with the value **"0x00"** (note the double quote marks), which references a lookup table available at **hsmag.cc/PgwFW8**. Pupil colour, however, is given as a simple RGB set. We've given ours a dark red tint, to make them look a little menacing.

The pupils change size a little bit to give the impression of real, working eyes. There are three variables that affect this: **pupilMin**, **pupilMax**, and **slitPupilRadius**. The min and max variables are proportions of the iris, while the **slitPupilRadius** (for non-round pupils such as cats' eyes) is in pixels. Higher radius values will give a taller, more slitty pupil. The default of 0 will give a round pupil.

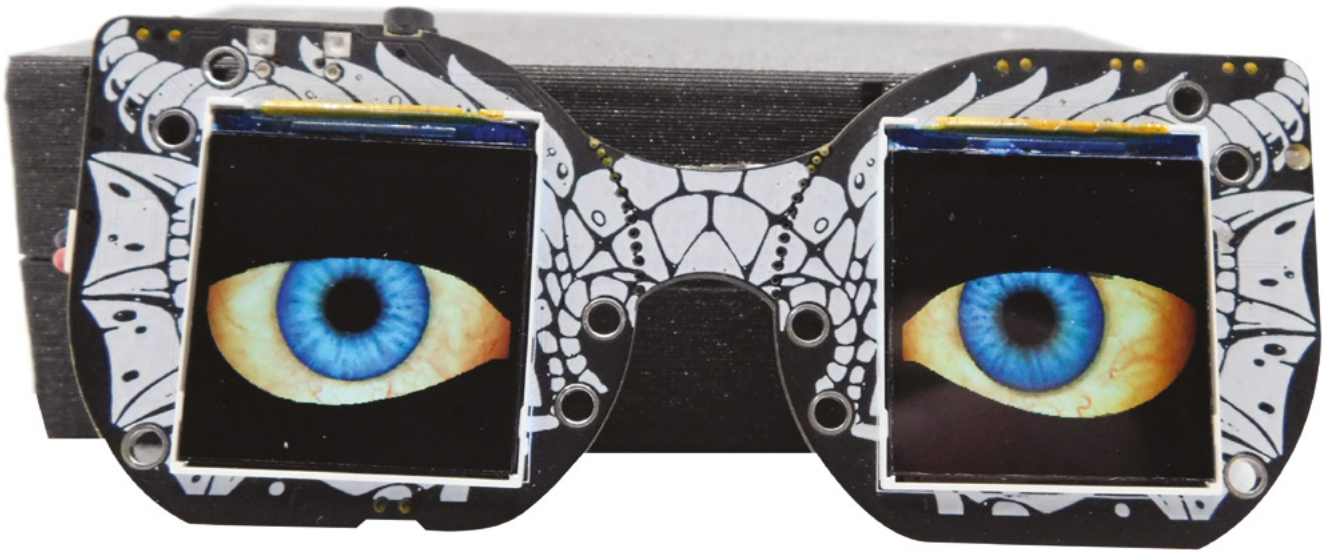
The four images we'll look at in the next section link to different prepared graphics.

```
{
  "eyeRadius"       : 130,
  "irisRadius"      : 70,
  "eyelidIndex"     : "0x00",
  "pupilColor"      : [ 30, 0, 0 ],
  "pupilMin"        : 0.3,
  "pupilMax"        : 0.4,
  "slitPupilRadius" : 60,
  "backColor"       : [ 0, 0, 0 ],
  "irisTexture"     : "skull/iris_red.bmp",
  "scleraTexture"   : "hazel/sclera.bmp",
  "upperEyelid"   : "hazel/upper.bmp",
  "lowerEyelid"   : "hazel/lower.bmp"
}
```

We'd suggest playing around with these different values to see what effect you get.

LOOK INTO MY EYES

Let's now take a look at the graphics. There are four, and each one links to a bitmap file. We've found that the upper and lower eyelid graphics are generally best left as they are or omitted completely. You can play around with these if you like, but we've not found alternatives that look good.



Above
Image filters can easily give your eyes a distinct look and feel

The iris is the part of the eye that surrounds the pupil, and the sclera surrounds the iris (commonly known as the 'white of the eye', although you can make it whatever colour you want).

These files want to be 'landscape' rectangles that will be stretched around the circular shapes. The exact size isn't too important, but there's limited RAM available, so don't make them too large (we used 512 × 128 pixels). If you're feeling artistic, you can create these from scratch, but you can get great effects by playing with a photo editor. We got good results from two ways of creating new eyes: filters, and cropping larger images.

We wanted eyes with a more piercing blue than in the standard hazel. Rather than start from scratch and create a new image, we took the hazel image and put it through a filter (the Arctic filter in Microsoft Photos, though there are many, many different pieces of image editing software that can also add similar filters).

A second effect we wanted was whirlpool eyes: swirling, watery scleras for a sea creature build. You certainly could start from scratch and design a water image, but we decided to use a real photo of water. We searched for white water on a free stock image site and found a suitable photo. We cut out a 512 × 128 pixel section and placed it in a new image. This looked good, but we wanted a whirlpool effect, so we used the smudge tool in the GNU Image Manipulation Program (GIMP) to add diagonal streaks. The final part of this was adding the **irisSpin** option to make the irises rotate. You might have noticed that we wanted the scleras to spin, but we used the **irisSpin** value. This is because there's no **scleraSpin** option. Instead, we simply removed the sclera (the iris radius is the same as the pupil radius) and had just a whirlpool and a pupil.

```
  {
    "voice"           : true,
    "pitch"           : 0.72,
    "gain"            : 1.2,
    "eyeRadius"       : 125,
    "irisRadius"      : 125,
    "slitPupilRadius" : 40,
    "eyelidIndex"     : "0x00", // From table:
learn.adafruit.com/assets/61921
    "pupilColor"      : [ 0, 0, 30 ],
    "pupilMin"        : 0.05,
    "pupilMax"        : 0.15,
    "backColor"       : [ 80, 0, 0 ],
    // From www.deviantart.com/suicidecrew/art/
Fire-Seamless-tile-116721709
    "irisTexture"     : "demon/sclera-water2.
bmp",
    "upperEyelid"     : "demon/upper.bmp",
    "lowerEyelid"     : "demon/lower.bmp",
    "tracking"        : false,
    "left" : {
      "irisSpin"      : 18
    },
    "right" : {
      "irisSpin"      : 18
    }
  }
```

Those are the eyes that we've created, but ultimately it's up to you to decide what effect you want to create for your build. It's possible to get some really creepy, realistic, or just plain fun eyes. We strongly recommend having a play with all of the examples and mashing them up. Eyes have a strange psychological impact on us, and it can be a little hard to understand how changing some values will affect how the eyes make you feel until you've seen them.

HackSpace
TECHNOLOGY IN YOUR HANDS

FIELD TEST

HACK | MAKE | BUILD | CREATE

Hacker gear poked, prodded, taken apart, and investigated

ONLY THE BEST

Electronic snoops and sniffers!

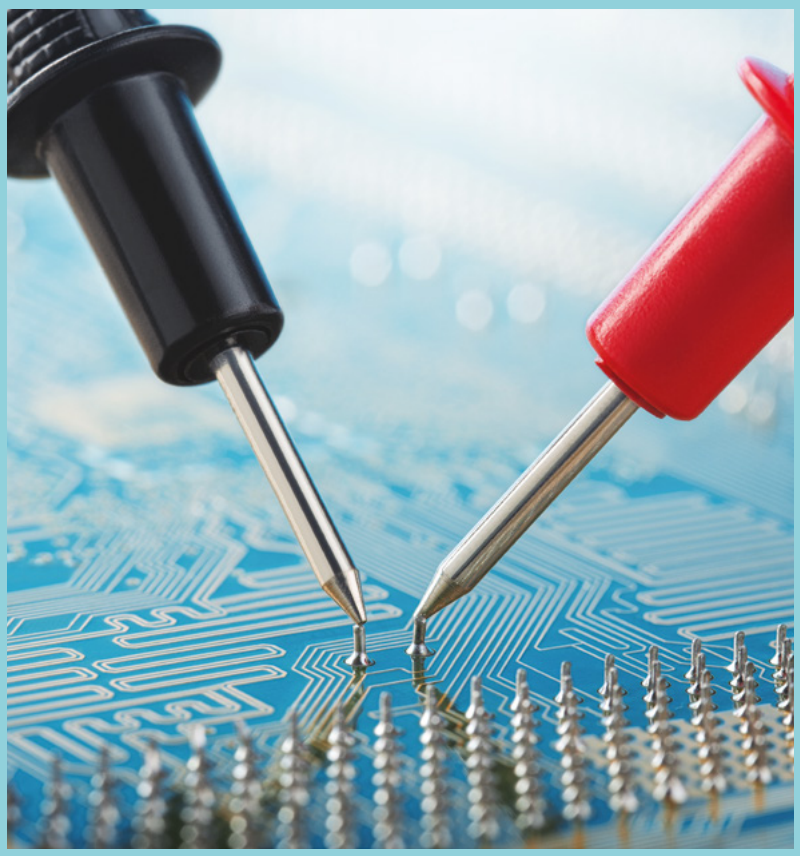
Specialised listening devices for hardware hacking and research

By **Marc de Vinck** @devinck

At some point in your adventures of building electronic devices, you are going to have to reverse-engineer something to see how it works. Maybe tweak it to suit your needs, or just sniff a few bits and bytes out of curiosity. Yes, you have an oscilloscope, we already reviewed them in a previous Best of Breed, but sometimes you need a few accessories or specialised tools to access that data. That's where this Best of Breed steps in. The products range from making it easier to access signals, to very specialised boards for hacking and tapping into your car's computer.

Just like most things in life, there are several different ways to accomplish what most of these boards and peripherals are designed to do. You may find that a standard oscilloscope, along with soldering a few wires here and there, will get the job done. But then there is the time factor. How long does it take to solder wires to a USB cable? Not too long! And did you do it correctly? Yes, of course! But is your solution as robust or reliable as a dedicated PCB? Maybe not.

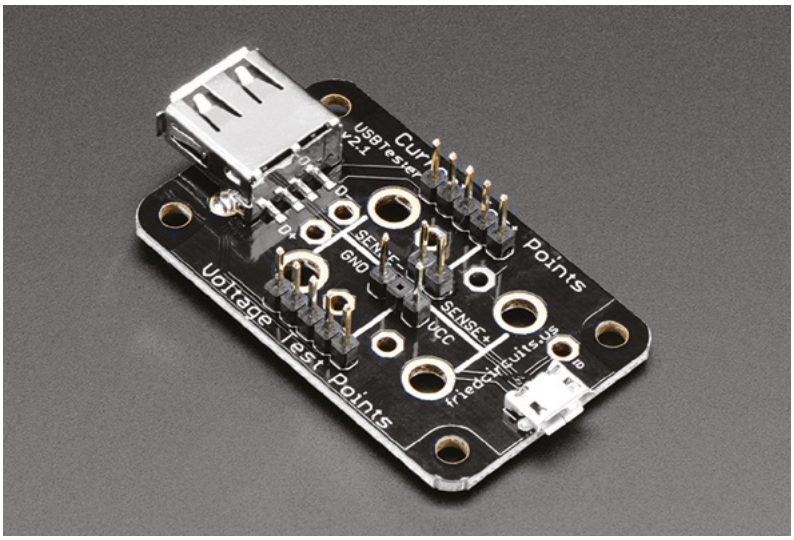
Fortunately, nothing in this roundup is too expensive, and the value of all the products is really high. Check them all out and see if there's anything that may help you more easily sniff out those bits and bytes in your next project.



FriedCircuits USB Tester v2.0 vs I²CDriver Core

FRIEDCIRCUITS $11.95 | adafruit.com

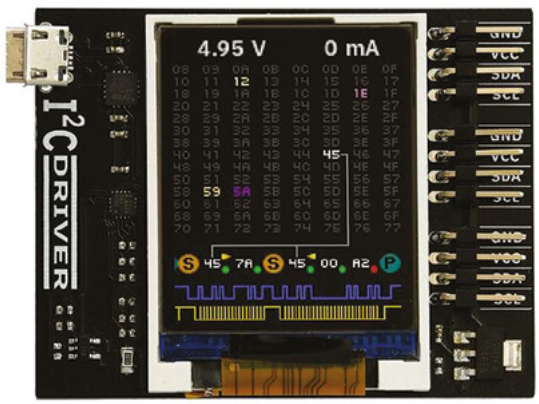
EXCAMERA LABS $33.25 | pimoroni.com



Below
Keep an eye on your USB connections

USB is only becoming more popular every day, but it can be a little frustrating to work with when it comes to accessing those thin little power and signal wires. This handy little board allows you to easily test the voltage coming out of your USB port. Simply connect this board to your project, hook up your oscilloscope or multimeter's test leads, and get to testing.

In addition to testing the voltage, the USB D+/D- pins are also broken out on the PCB, so you can monitor those signals with your oscilloscope. This is a really handy little tool for anyone developing a board that uses USB, or for anyone wanting an easy way to sniff out those USB signals.



Above
Run, monitor, and power I²C devices

I²CDriver Core is a simple, open-source, board for controlling I²C devices. Its biggest feature is the on-board colour display, which is so much more capable than a typical blinking LED found on many other I²C analysers. The display can show the signal lines with a graphical representation of the traffic, a continuous stream of the address maps for all the attached devices, and even has current and voltage monitoring.

The I²CDriver Core has three I²C ports and also includes free software that runs on a Mac, PC, or Linux-based computer. The software features a GUI interface, command line control, and C/C++ and Python integration. And, since it uses a standard FTDI USB serial integrated circuit, no special drivers are needed. I²C is everywhere, and this handy little board will save you time and a lot of frustration when it comes to troubleshooting.

VERDICT

FriedCircuits USB Tester v2.0

A reliable and robust solution to measuring USB voltage.

9/10

I²CDriver Core

A great way to display I²C data.

10/10

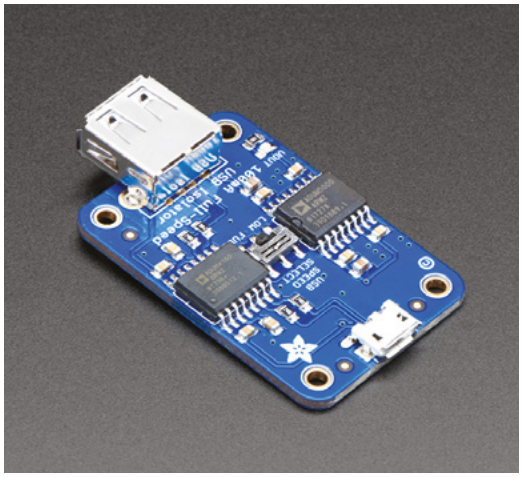
BEST OF BREED

Adafruit USB Isolator

ADAFRUIT ◈ $34.95 | adafruit.com

The Adafruit USB Isolator is becoming more popular now that so many people are using USB-based oscilloscopes and logic analysers. The problem this board solves occurs when the ground of the device you are testing and the testing device itself are connected. This handy little board isolates the ground between the two. It allows for low-speed and full-speed connections. Sorry, no high-speed here, but that's OK for so many other applications.

Eliminate the dangers of back-fed power, background noise, floating grounds, or other hazards of tinkering around with USB-based testing instruments. Just remember the Adafruit USB Isolator is only for 5V power and 3.3V logic USB devices!



Left ◈
Keep your computer safe

VERDICT

Adafruit USB Isolator

When you need it, you need it!

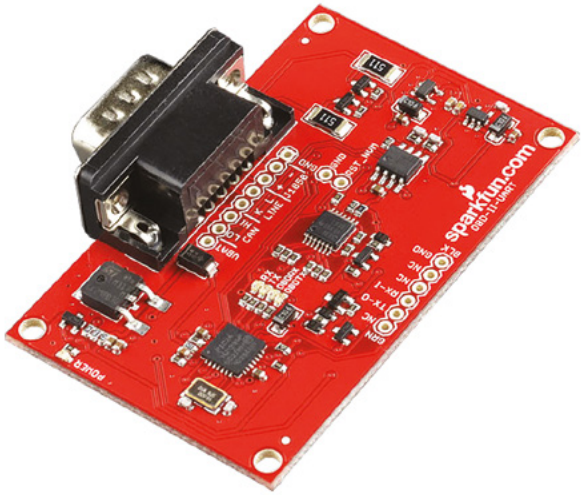
9/10

SparkFun OBD-II UART

SPARKFUN ◈ $52.95 | sparkfun.com

The SparkFun OBD-II UART board, from SparkFun, allows you to easily connect to your car or truck's on-board OBD-II bus port. Once connected, you'll have access to all the standard information that most countries require your car to output since the mid-1990s.

The OBD-II standard requires a compatible vehicle to allow access to the on-board electronic control unit (ECU). This is where you can scan for information like malfunction codes, your VIN number, maintenance messages, and a whole host of real-time data. Just remember to pick up the required special cable that interfaces between the on-board DB9 connector of this board and an OBD-II type port.



Left ◈
Debug your car with this board

VERDICT

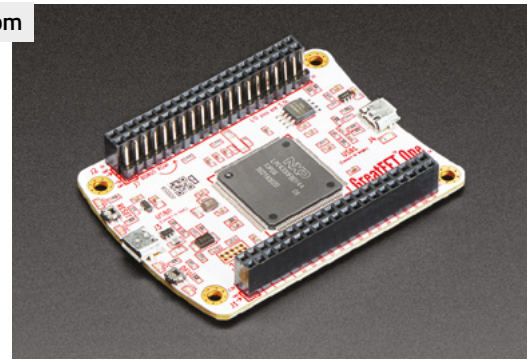
SparkFun OBD-II UART

Grabbing data from your vehicle made easy.

8/10

GreatFET One

GREAT SCOTT GADGETS ◈ **$99.95** | **adafruit.com**



Left ◈
Take control of a USB device

The GreatFET One from Great Scott Gadgets is described as "a hardware hacker's best friend." We have to agree, especially if you want to hack an unknown USB signal. It's a really interesting piece open-source hardware board powered by an NXP LPC4330 with two integrated USB ports. This allows the GreatFET One to be an interface between a USB connection. It's great for reverse-engineering and USB hacking.

If you need direct access to an IC, need a ton of pins to bit-bang some code, or like to get detailed and high-speed access to data, the GreatFET is a solid choice. Just keep in mind that this is for advanced hardware hackers. Head over to the product page for a lot more information about this very powerful board.

VERDICT

GreatFET One

GreatFET is a great board!

8/10

SparkFun microSD Sniffer

SPARKFUN ◈ **$10.95** | **sparkfun.com**

The SparkFun microSD Sniffer not only has a great name, but it's also really handy when you need to check on things when using an SD card. It's a simple concept: the longer end of the PCB plugs directly into the microSD port of the device you are using to send data out. Next, add your microSD card to the on-board card holder, followed by attaching a microcontroller or logic analyser to the provided breakout connection points.

It's a very simple and handy way of reading the data being passed between any device and its SD card. And it sure beats soldering a bunch of wires to the SD card port! SparkFun makes this board in both a microSD and standard SD card layout. You might want to pick up both and add them to your toolbox.

Above ◈
Find out what data is going where

VERDICT

SparkFun microSD Sniffer

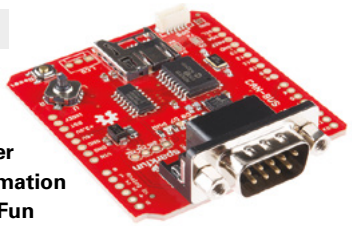
Really handy for SD card sniffing.

8/10

BEST OF BREED

CAN-BUS Shield

SPARKFUN $26.95 | sparkfun.com



The CAN-BUS Shield is another interesting automotive information gathering device from SparkFun which also allows you to listen in on your vehicle's computer. This shield, coupled with an Arduino, allows you to poll the engine control unit (ECU) and receive data about your temperatures, speed, throttle info, and engine statistics. It's handy for creating custom screens in your car or truck with its on-board LCD connector, or you could download the data for offline review thanks to its built-in microSD card holder. Head over to SparkFun's website for more information and a complete user guide and video.

VERDICT

CAN-BUS Shield

A handy shield for accessing your car's data.

7/10

I²CMini Core

EXCAMERA LABS $17.62 | pimoroni.com

So, what can this little board do? A lot! The I²CMini is a USB to I²C bridge which allows you to monitor and control I²C traffic from your computer or another device. The board, although incredibly small, features a USB connector and a Qwiic connector, along with pins for a breadboard – or simply add some header pins.

The I²CMini Core shows up as a standard serial device and is compatible with Mac, Windows, and Linux operating systems. Although there are a lot of appropriate applications for this little board, we can see it being most useful when calibrating sensors prior to installing them in a project.

VERDICT

I²CMini Core

A solid tool with good software.

8/10

Bus Pirate

DANGEROUS PROTOTYPES
$30 | adafruit.com

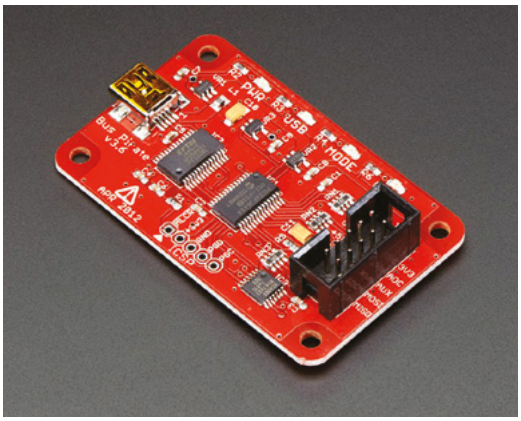
Designed by Ian Lesnet of Dangerous Prototypes and available at Adafruit, this is a universal bus interface for programming integrated circuits. Interfacing a new microchip, especially an unknown one, can be a major hassle. The Bus Pirate eliminates the need for a lot of early prototyping of boards just to get access to the pins of an IC. This board includes JTAG, 1-Wire, UART, MIDI, PC keyboard, I²C, SPI, and generic 2- and 3-wire libraries for custom protocols. Yeah, all of that in one board. If you are looking to hack into an IC, this board is for you! □

VERDICT

Bus Pirate

So many features in such a small board.

10/10



> **"The I²CMini Core shows up as a standard serial device** and is compatible with Mac, Windows, and Linux **"**

GET STARTED WITH ARDUINO®

Robots, musical instruments, smart displays and more

£10 with FREE worldwide shipping

Inside:

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

AVAILABLE NOW hsmag.cc/store

plus all good newsagents and:

WHSmith BARNES&NOBLE

Available on the App Store

GET IT ON Google Play

FROM THE MAKERS OF HackSpace MAGAZINE

REVIEW

Sugru Rebel Tech Kit

Repair and adapt, using this mouldable putty

SUGRU £11.99 | sugru.com

By **Ben Everard** @ben_everard

Sugru is a mouldable putty. It comes in small packets – like the sort of thing you might get sauce in at a café. Open this up and there's a substance with a similar consistency to Blu Tack, that you can mould into almost any shape you want. After 24 hours, it hardens into a stiff, rubber-like state. It's bendable, but no longer re-shapeable.

This reviewer's headphones have been held together by some Sugru for the past couple of years after they broke in his back-pack. It's been strong enough to cope with the day in, day out usage, and flexible enough to have just a little give, so it's not broken again on the joint. He's also used it fairly extensively when he needs glue to fill a gap – it's more controllable than hot glue, so you don't end up with strings everywhere and bits of glue where you didn't want them.

The Rebel Tech Kit comes with five sachets of Sugru, and what the company calls a 'Sugru Remover tool', but other people might call it a guitar pick or plectrum. There's also a 14-page booklet of ideas, and a metal tin to keep it all in.

Sugru also does a Create and Craft Kit that includes widgets for adding texture to your Sugru, a Magnets Kit that includes four neodymium magnets, an Organise Small Spaces Kit that includes D-hooks for hanging things, and a Hacks For Your Home Kit with a slightly longer guide. Each of these come in at the same price of £11.99.

Left
A pack ready to help you stick, mould, and bodge your next project

The range of things they suggest includes using the putty to re-enforce cables, particularly where they join solid bits like USB sockets. This might not be worth it for the vast pile of micro USB cables that most of us have lying around, but if you've got an expensive proprietary cable that you use a lot, an extra bit of strain relief may end up saving you quite a bit of money and hassle. Similarly, it can also be used as strain relief on custom-made cables. Another common use is adding 'little bits' to things. Do you need a particular shaped hook? No problem. Add a knob or custom handle grip? Sugru works really well for that. Need a protector for something fragile that you just seem to keep dropping? Sugru adds bouncy bits to corners and other vulnerable points. For all these things, we've found this solidifying putty works really well.

The price of these kits is more or less in line with other sized packs of Sugru, but you also get the extra 'kit' bits thrown in as well. This makes it a great gift for maker friends, but also handy packs if you're after five bits of Sugru.

Sugru isn't especially cheap – however you buy it, it almost always works out at over £1.50 per sachet. As a general-purpose material, that can quickly add up if you use it a lot. However, for repairs and simple upgrades, it's quick and easy to use, and that can be well worth it. For example, one of the examples from the guide is a headphone hanger. Sure, you could design and 3D-print a similar thing for just a few pence of PLA, but how much time would it take? How much is that time worth to you? Of course, coupled with that is the question of do you prefer to sculpt things with your hands, or do you prefer to design digitally, and perhaps iterate to a look you like?

> **Each packet has a use-by date on it, but it's not** always clear how long you'll have left when you buy it

One thing to think about when buying Sugru is that it isn't shelf-stable. Over time, it hardens in the packet. Each packet has a use-by date on it, but it's not always clear how long you'll have left when you buy it – the pack you get might have been sitting in a storage facility. The packs we got had ten months left on them. If you do want to keep it longer, according to the manufacturers, it'll last about three times as long in the fridge or freezer as it will at room temperature.

This reviewer always keeps a few sachets of Sugru in his workshop, and the metal tin will make it easier to find them when needed. The 'Sugru Remover' may end up being used more for the guitar than for moulding the putty though. ◻

Above
The metal tin should help protect the sachets from damage in the workshop

Below
You get a selection of different colours, so your Sugru can match with what you're making

VERDICT

A great workshop tool and a free guitar pick.

9/10

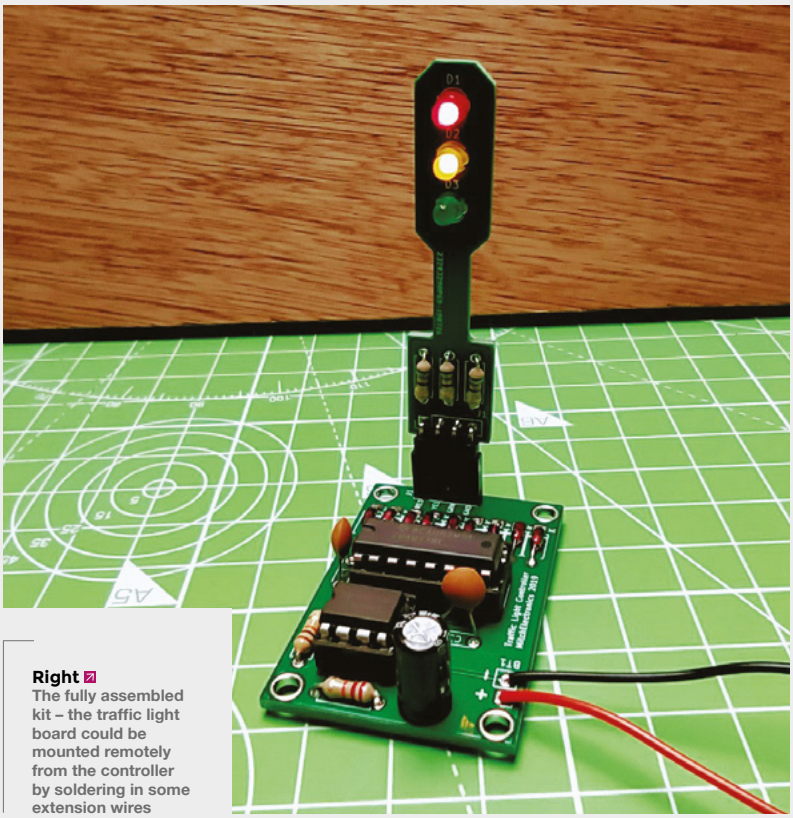
REVIEW

MitchElectronics Traffic Lights kit

Learn to solder and help teeny-tiny cars through junctions

MITCHELECTRONICS £4.99 | mitchelectronics.co.uk

By **Jo Hinchliffe** @concreted0g

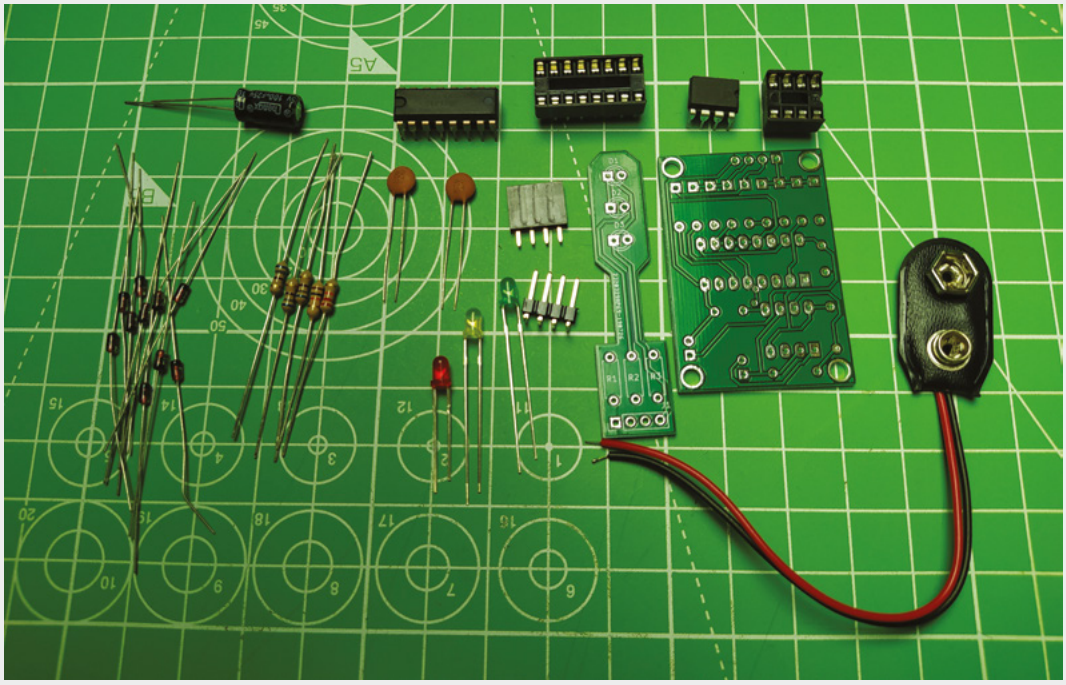


Right
The fully assembled kit – the traffic light board could be mounted remotely from the controller by soldering in some extension wires

MitchElectronics is a UK company making a range of electronics kits. It has a fascinating range of projects available – from programmable logic controller boards, to tiny kits that make a single NAND gate out of a couple of transistors. Some of its kits are tools or utility-type projects, like an audio Datacorder, Logic Probe, or even a Sound Card Scope (oscilloscope) project, and they are targeted at makers or people involved in electronics education.

We have a few of the firm's kits, and we assembled the Traffic Lights one as the basis of this review. The kit comes in a couple of options: firstly, there's the full version, which consists of a controller PCB and a traffic light-shaped PCB that hosts the LEDs. The traffic light board is plugged into the controller board via a simple, but effective, header pin and socket. The other option allows you to buy sets of three of just the traffic light PCB and components.

The kits are nicely packed in an anti-static bag, and the bag has a QR code on it which links you directly to the instructions. MitchElectronics has created nice individual instructions for each kit that include a schematic, a component guide, a bill of materials, and a section describing how the circuit works. The circuit descriptions are well-written and, in the case of the Traffic Lights kit, describe the function



Left
The kit arrived with everything present and correct, and an extra diode

Below
The kit part-assembled. We appreciated the inclusion of sockets for the ICs and good, clear labelling on the PCB silkscreen

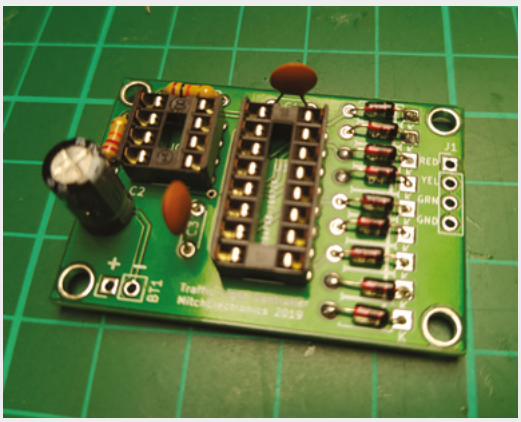
of the 555 IC that is acting as a clock source for the CD4017 decade (0–10) counter. It goes on to describe how the 4017 IC outputs interact with the diode matrix to create the timings for the different LED states that match traffic light function in the real world. There's a separate guide about how to physically install components and construct the kits, and that document applies to the whole kit range. Both sets of documents are well laid out and clear for individuals building the kits, but we feel they could also be useful for educators using the kits to deliver electronics learning.

Soldering the kit was straightforward. The PCBs are nicely laid out, with plenty of space between components and decent pad sizes, so everything fits well and it's hard to create accidental solder bridges. Everything is through-hole, and we just needed the basics of a soldering iron and some snips to assemble the kit. We liked that MitchElectronics included sockets for the ICs, as sometimes these components may be cut from kits to save money. Including the sockets decreases the risk that an inexperienced solderer will overheat the IC, and allows a wrongly oriented IC to be corrected easily.

Building the kit, starting with the smallest components (resistors and diodes) and moving through to the larger components (sockets and capacitors), we found it went together well. We used quite a broad chisel tip on our soldering iron and 0.8mm solder, and this was fine. It could certainly be soldered with a budget-friendly beginner's soldering iron with a larger tip. Satisfyingly, attaching a 9V battery after completing the build, everything worked as it should and the traffic lights operated as intended. As an aside, we wonder if anyone has 3D-printed a case for the traffic light PCB to make it look like a more scale model of a real set of lights!

> **"It could certainly be soldered with a** budget-friendly beginner's soldering iron **"**

We also have the Electronic Dice kit and the Transistor Flasher kit, and we are interested to try some of the even smaller kits, like the individual Logic Gate kits. These kits range in price from £2.49, up to £4.99 for the full Traffic Lights kit as reviewed. We feel that makes them a very viable option for those wanting to host 'learn to solder' workshops, those in more formal education settings, and also those who want a cheap, rainy afternoon activity at home. ◻



VERDICT

With great instructions and education materials, this quality RoHS-compliant kit was a pleasure to build at a decent price.

9/10

REVIEW

Electromaker kits

A pair of excuses to add more NeoPixels to your life

ELECTROMAKER $55 | electromaker.io

By **Ben Everard** @ben_everard

Electromaker has four kits available to buy: an LED mood lamp, an LED disco helmet, a GPS tracker, and a wireless charging stand. We tested out the first two of these. They combine electronics, 3D printing, and Arduino programming to create projects that are both fun and useful.

We tried the lamp project first as that was listed as being easy and completable in two hours. The kit comes with all the electronics you'll need, but you need to supply the 3D printer filament yourself. The base and stand can be any colour, but the diffuser really should be white (or close to white) to allow the LEDs to shine through properly. The first job, then, is to print it all out. This takes quite a while – combined, there's a total print time of about 24 hours, so this is a job to do ahead of time.

Fortunately, once that's done, it all gets a bit quicker. The lighting is provided by Adafruit NeoPixels, and a strip of these is glued around the lamp upright. There are a few connectors and buttons to solder onto wires, but the soldering part of this is all very straightforward. Finally, all this is connected to an Arduino Nano via a breadboard.

The code is provided, but you'll need to compile and upload it yourself. Finally, there's a little assembly to do, and everything should be good to go.

We really like the look of the lamp. The low-poly stand looks great and catches the lights. At first, we were a little unsure about using a breadboard for the Arduino. It's not the most secure attachment, and there is the potential for a wire to fall out. However, it is very flexible and gives you the opportunity to upgrade things easily as your skills improve. We can definitely imagine upgrading our lamp with an ESP32 or other WiFi-enabled controller and use it to display some data.

Once we had our lamp up and running, we moved on to the disco helmet. In many ways, this is a similar kit – there's an Arduino Nano controlling some NeoPixels that are mounted on a 3D-printed body. Perhaps the biggest difference is that the 3D printing is a much bigger job: it took us around 60 hours. Note that if you follow the instructions, some prints take a long time (20+ hours), so you need to run your printer overnight. If yours is too noisy to do this, you'll need to either print at a larger layer height (the above timings are for the suggested 0.15 layers) or print it in smaller sections and glue them together. Since the helmet is already printed in sections and glued together, this isn't too much of a problem.

Above
The final helmet is a great way of illuminating a room or impressing your friends while DJing

The soldering on this project is slightly more fiddly than on the lamp because you need to connect sections of LED strip together at 180-degree angles. This isn't too hard, though complete beginners might struggle a little.

Following this, there's a little modification to components as legs have to be trimmed to make everything fit in the small space, and a bit of hot glue to keep everything in place and you've got one of the coolest party pieces around.

What really makes these projects work for us is the clear, step-by-step instructions. Take a look for yourself at **hsmag.cc/e3eGsB** and **hsmag.cc/y5ine3**. These guide you through the complete build.

There's no denying that both of these projects are involved builds. The skill levels, listed as 'Easy' and 'Moderate', are relative terms. The mood lamp could, perhaps, be completed by someone with no experience, but they'd probably have a difficult time purely because there are a lot of new skills for them to work with – slicing the model for their 3D printer, soldering, and getting set up with the Arduino IDE. That's a lot of new things in one project – none of them are particularly challenging on their own, but together there's a lot of potential for frustration. However, if you've already got some experience in one or more of these areas (as you most likely have if you're reading this), then it shouldn't present too many problems.

> **What really makes these projects work** for us is the clear, step-by-step instructions

As well as the parts you buy, you'll need access to quite a bit of kit – not least a 3D printer and soldering setup. However, the flip side of this is that the end results are complete products. Many electronics kits leave you with bare circuits that often get left in a drawer after they've been built. The final results of both of these projects are genuinely good-looking pieces that you can be proud to display and use in your day-to-day life. What's more, there's scope to go beyond the basics and learn a bit more about the underlying technologies. Both of these provide an interesting backdrop for experimenting with NeoPixels and Arduino boards. Whether that's adding a control board with WiFi connectivity to control the patterns from your phone, or adding some capacitive touchpads to switch patterns with your fingers.

We've had great fun with these kits. They're fun to build, and the final results are great for both impressing your friends and facilitating further learning. ◻

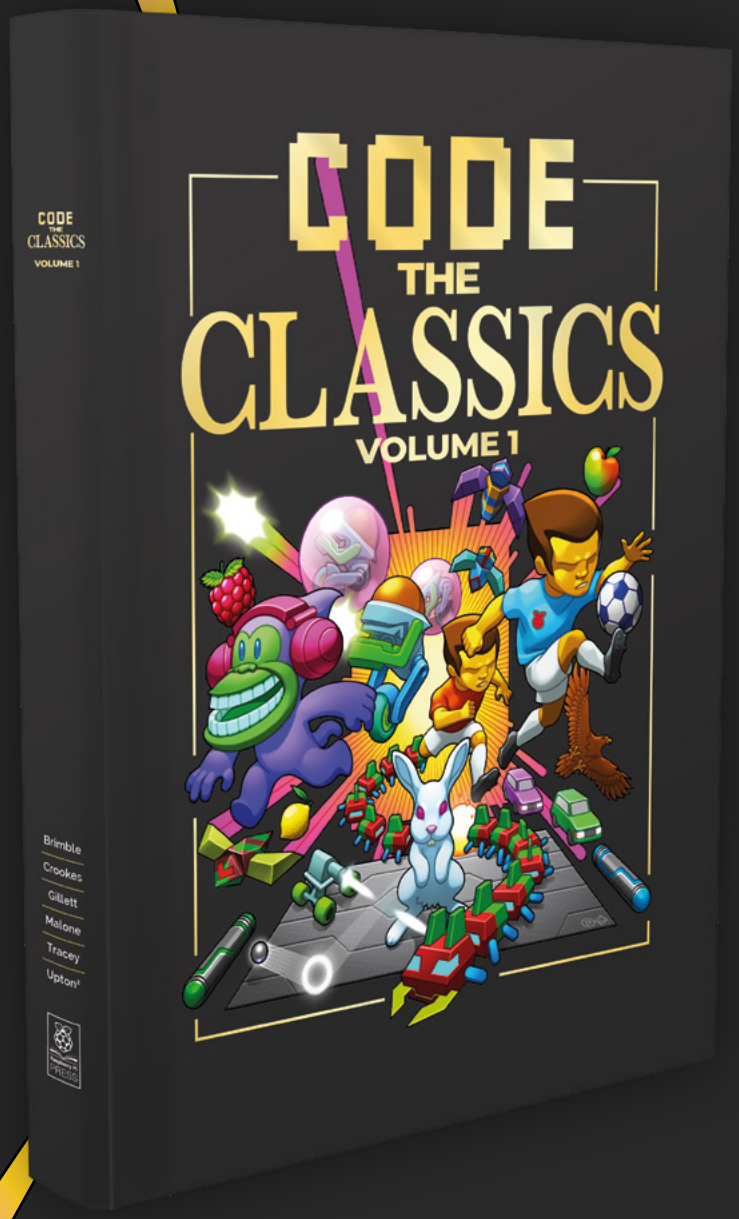
Above
You can create different effects with the LEDs

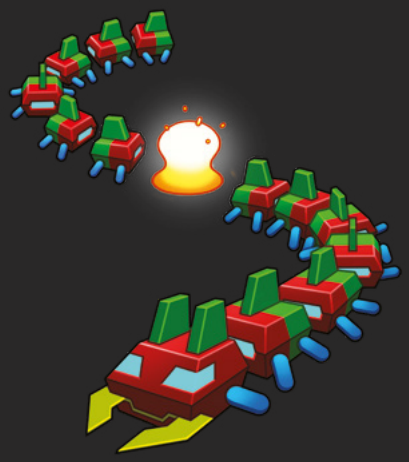
Left
We really like the interaction of the light on the low-poly lamp base

VERDICT

If you've got a 3D printer and soldering iron, these kits are fun to make and they're visually impressive.

9/10





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

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

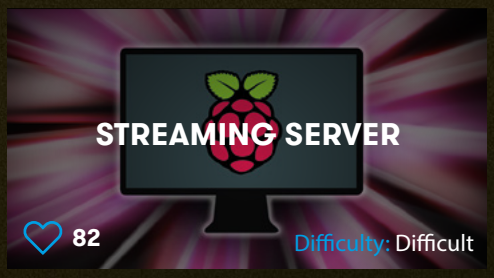
Available now: **hsmag.cc/store**

okdo™

DESIGN THE WORLD

CHECK OUT THE NEW OKDO PROJECT HUB!

Find out more at www.okdo.com/projecthub



Find inspiring projects created by our customers and engineers that show you how to build and programme cool projects and how to get started with some of your favourite boards!

Find out more about our full product range at:

www.okdo.com

@letsokdo