

MAKE | BUILD | HACK | CREATE

# HackSpace

TECHNOLOGY IN YOUR HANDS

hsmag.cc

April 2019

Issue #17

# ARDUINO

## Circuit Boards

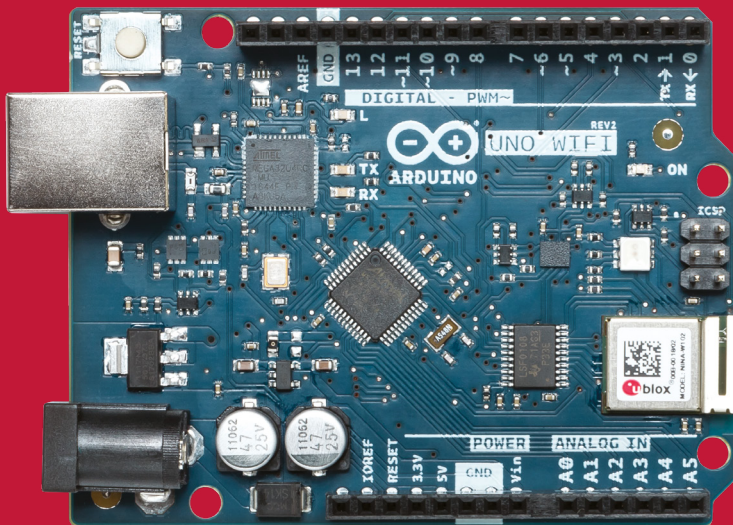
Design your first PCB with KiCad

## Polyphonic Synth

Turn a Teensy into a music machine

## Make Vinegar

Ferment your own condiments



OPEN SOURCE MACHINE TOOLS

Build your own workshop

# THE ULTIMATE GUIDE

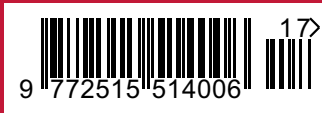
BOARD, SHIELDS, AND PROGRAMMING

Flap Flap Flap Flap Flap Flap



Control games with a wave of your arms

Apr. 2019 Issue #17 £6



RADIO SPRESENSE CHIPS WAVEFORMS

# The Building Blocks of IoT

Digi-Key is Your One Stop Shop for the Internet of Things (IoT) Components.

Antennas

Cellular, M2M

Zigbee/Thread

Sensors

Development Systems

Expansion Boards

Cables/Connectors

Wi-Fi

**Digi-Key**<sup>®</sup>  
ELECTRONICS

LP-WAN

Bluetooth

Transceivers

**DIGIKEY.COM/IOT**

1,300,000+ PRODUCTS IN STOCK | 750+ INDUSTRY-LEADING SUPPLIERS | 100% AUTHORIZED DISTRIBUTOR

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. © 2019 Digi-Key Electronics, 701 Brooks Ave. South, Thief River Falls, MN 56701, USA

ecia  
MEMBER



# Welcome to HackSpace magazine

Arduino is more than a company; it's an ecosystem, a movement, and a state of mind. It's hundreds of microcontroller boards that all run the same code, it's libraries for almost any hardware a hobbyist is likely to come across, and it's a range of development environments for different ranges of experience. For many people, the word Arduino is almost interchangeable with microcontrollers. This issue, we take a look at exactly what this all means in our cover feature.

Perhaps the most interesting recent development in Arduino is the Arduino IoT Cloud, which aims to make it easy to develop secure, robust Internet of Things appliances. We take a look at how all this works, by exploring the features and creating our own IoT lamp.

If you'd rather use a different environment, we're also looking at CircuitPython, with Sophy Wong's Flappy Birds outfit, which is quite possibly the best games controller ever made. Flip to page 100 to get flapping.

## BEN EVERARD

Editor [ben.everard@raspberrypi.org](mailto:ben.everard@raspberrypi.org)

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

get in touch at [hsmag.cc/hello](https://hsmag.cc/hello)

### GET IN TOUCH

[hackspace@raspberrypi.org](mailto:hackspace@raspberrypi.org)

[hackspacemag](#)

[hackspacemag](#)

### ONLINE

[hsmag.cc](https://hsmag.cc)



PAGE 48

SUBSCRIBE TODAY

## EDITORIAL

### Editor

Ben Everard

[ben.everard@raspberrypi.org](mailto:ben.everard@raspberrypi.org)

### Features Editor

Andrew Gregory

[andrew.gregory@raspberrypi.org](mailto:andrew.gregory@raspberrypi.org)

### Sub Editors

David Higgs, Nicola King

## DESIGN

### Critical Media

[criticalmedia.co.uk](https://criticalmedia.co.uk)

### Designers

Lee Allen, Harriet Knight, Sam Ribbits

### Photography

Fiacre Muller, Adam Gasson, Nigel Barker

## CONTRIBUTORS

Lucy Rogers, Andrew Huang, Cameron Norris, Plunkett Doyle, Mayank Sharma, Dave Astels, Jo Hinchliffe, Graham Morrison, Matt Bradshaw, Andrew Clarke, Sophy Wong, Les Pounder, Marc de Vinck, Richard Smedley, Gareth Halfacree

## PUBLISHING

### Publishing Director

Russell Barnes

[russell@raspberrypi.org](mailto:russell@raspberrypi.org)

## DISTRIBUTION

Seymour Distribution Ltd  
2 East Poultry Ave,  
London EC1A 9PT

+44 (0)207 429 4000

## SUBSCRIPTIONS

Mann Enterprises Ltd,  
Unit E, Brocks Business  
Centre, CB9 8QP

[hsmag.cc/subscribe](https://hsmag.cc/subscribe)



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

HackSpace magazine is published by Raspberry Pi (Trading) Ltd., Station Road, Cambridge, CB1 2JH. The publisher, editor, and contributors accept no responsibility in respect of any omissions or errors relating to goods, products or services referred to or advertised. Except where otherwise noted, content in this magazine is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported (CC BY-NC-SA 3.0). ISSN: 2515-5148.

# Contents



122

06

## SPARK

- 06 Top Projects**  
Cool stuff, made by you
- 16 Objet 3d'art**  
All aboard the Marvel hype train!
- 18 Upcycling**  
Recapture the joy of 1990s gaming
- 20 Meet the Maker: Will Ferraby**  
Making knives by hand in the Steel City
- 26 Columns**  
Stop! Collaborate and listen!
- 28 Letters**  
Safety is serious, so take it seriously
- 30 Hackspace NextFab**  
Makerspace and business incubator

33

## LENS

- 34 Arduino**  
The ultimate guide to the ultimate maker board
- 50 How I Made: Heating system**  
Control your central heating with a phone and a Pi
- 56 Open Source Machine Tools**  
Making things that make other things
- 60 Interview: Dr Adrian Bowyer**  
The co-founder of RepRap on 3D printing
- 68 Improviser's Toolbox Sponges**  
Cheap, bouncy, and conductive when wet...

## Tutorial

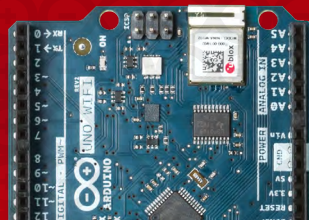
### School of making: Vinegar



90

How to make vinegar,  
step 1: first, make cider...

## Cover Feature



# ARDUINO

Boards, add-ons,  
and programming

34

100



116



20

Interview

Adrian Bowyer



60

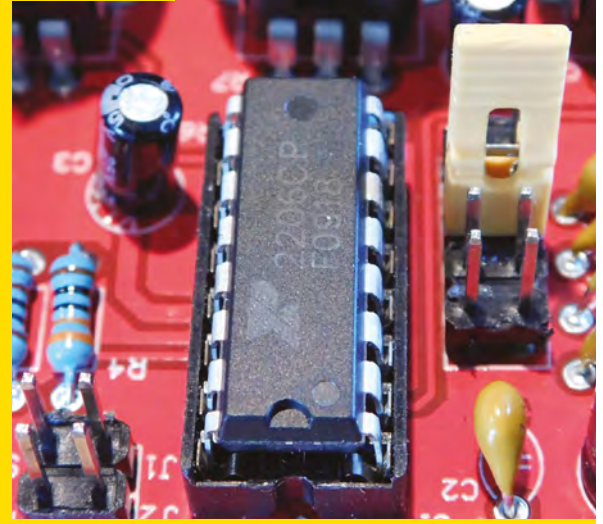
What it's like to lead the UK's biggest research project



16

Direct from Shenzhen

Waveforms



114

Bleeps and beats in an incredibly cheap package

73

FORGE

- 74 **SoM Electronics**  
The everyday miracle of the integrated circuit
- 80 **SoM KiCad**  
How to lay out a PCB ready for fabrication
- 84 **SoM Arduino**  
The art of troubleshooting hardware builds
- 90 **Tutorial Vinegar**  
The home-brew way to make fried potatoes tastier
- 94 **Tutorial Polyphonic synth**  
Get some tunes out of your DIY music box
- 100 **Tutorial Wearables**  
Make a games controller you can wear!
- 106 **Tutorial Networked E-ink**  
Send information to a low-power display

113

FIELD TEST

- 114 **Direct from Shenzhen Frequency generator**  
Trigger square, sine, and triangular waves
- 116 **Best of Breed**  
LoRa – long-range radio communication for Internet of Things
- 122 **Can I Hack It?**  
What's inside this projector that we can use?
- 124 **Review LumiDrive**  
An integrated display/controller for loads of LEDs
- 126 **Review Sony Spresense**  
A microcontroller with an added helping of UNIX
- 128 **Review ESP Game Engine**  
Build and test games with or without hardware
- 129 **Book Review Self-Sufficiency**  
How to live like Tom and Barbara from *The Good Life*

Some of the tools and techniques shown in HackSpace Magazine are dangerous unless used with skill, experience and appropriate personal protection equipment. While we attempt to guide the reader, ultimately you are responsible for your own safety and understanding the limits of yourself and your equipment. HackSpace Magazine is intended for an adult audience and some projects may be dangerous for children. Raspberry Pi (Trading) Ltd does not accept responsibility for any injuries, damage to equipment, or costs incurred from projects, tutorials or suggestions in HackSpace Magazine. Laws and regulations covering many of the topics in HackSpace Magazine are different between countries, and are always subject to change. You are responsible for understanding the requirements in your jurisdiction and ensuring that you comply with them. Some manufacturers place limits on the use of their hardware which some projects or suggestions in HackSpace Magazine may go beyond. It is your responsibility to understand the manufacturer's limits.

# Illuminated waterways map

By AlexT9


[hsmag.cc/WwpRCq](https://hsmag.cc/WwpRCq)

I had just discovered my local makerspace, and was immediately drawn to the laser cutters. I have always enjoyed looking at unique maps, so I knew I wanted to do a map-related project. At some point, I got the idea to fill in laser-etched wood with opaque-coloured epoxy. The idea grew and became way more complex (as most projects always do...) into fully cutting through the wood and backlighting the epoxy. I used Photoshop and Illustrator to convert a high-resolution waterways map into the vector format needed by the laser cutter. After cutting the wood, I used a blue pigment powder with two-part epoxy resin to fill in the waterways. Standard LED strip-lighting provides the backlight for the resin. I think the coolest aspect of the map is being able to easily visualise different river basins. The Mississippi river basin is huge! ▣

**Right** ▣  
It took Alex about eight hours to find the right level of detail for the rivers






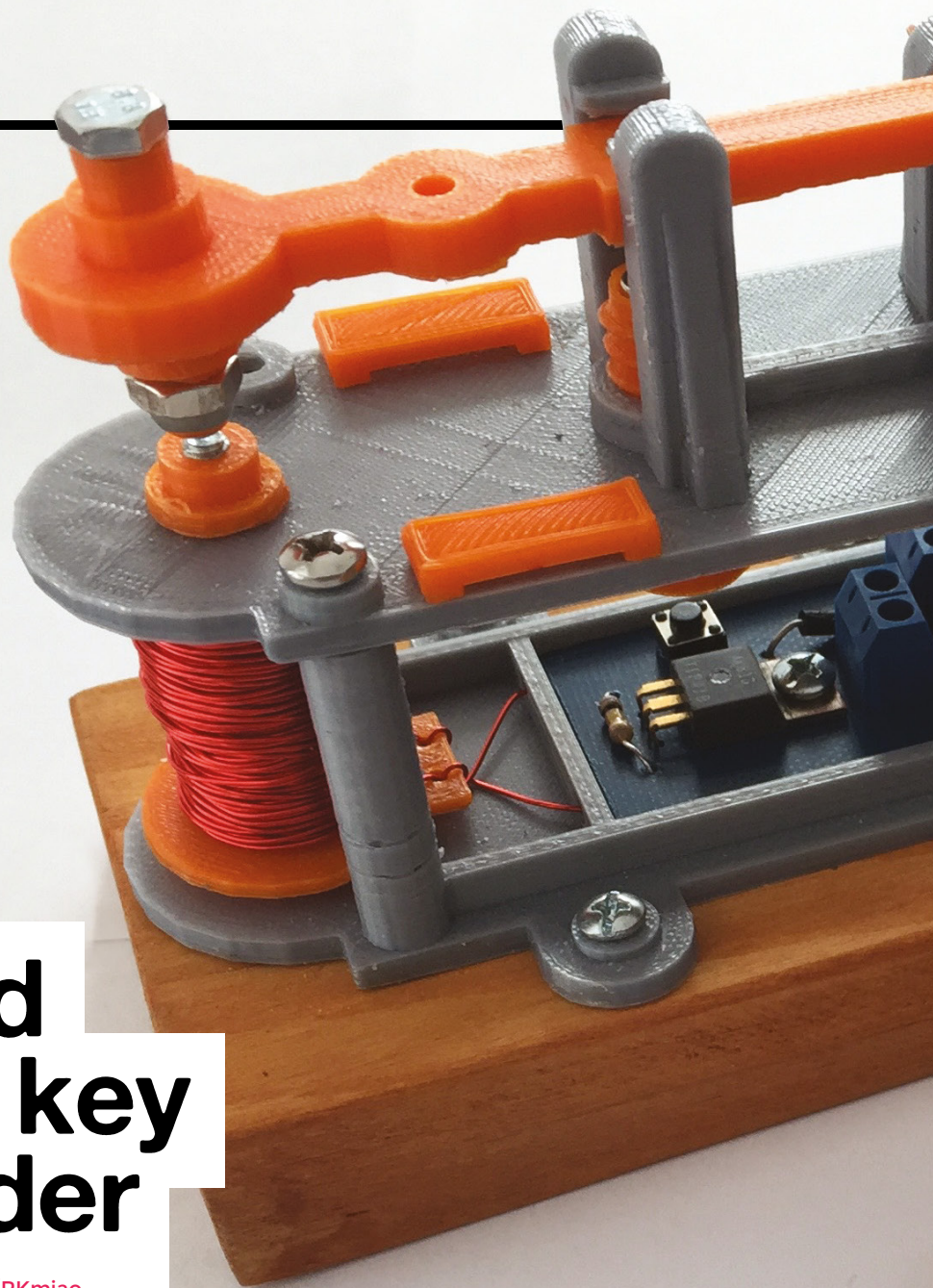
**Right**   
To see the  
sounder in  
action, visit  
[hsmag.cc/NqKEKs](https://hsmag.cc/NqKEKs)

# 3D-printed telegraph key and sounder

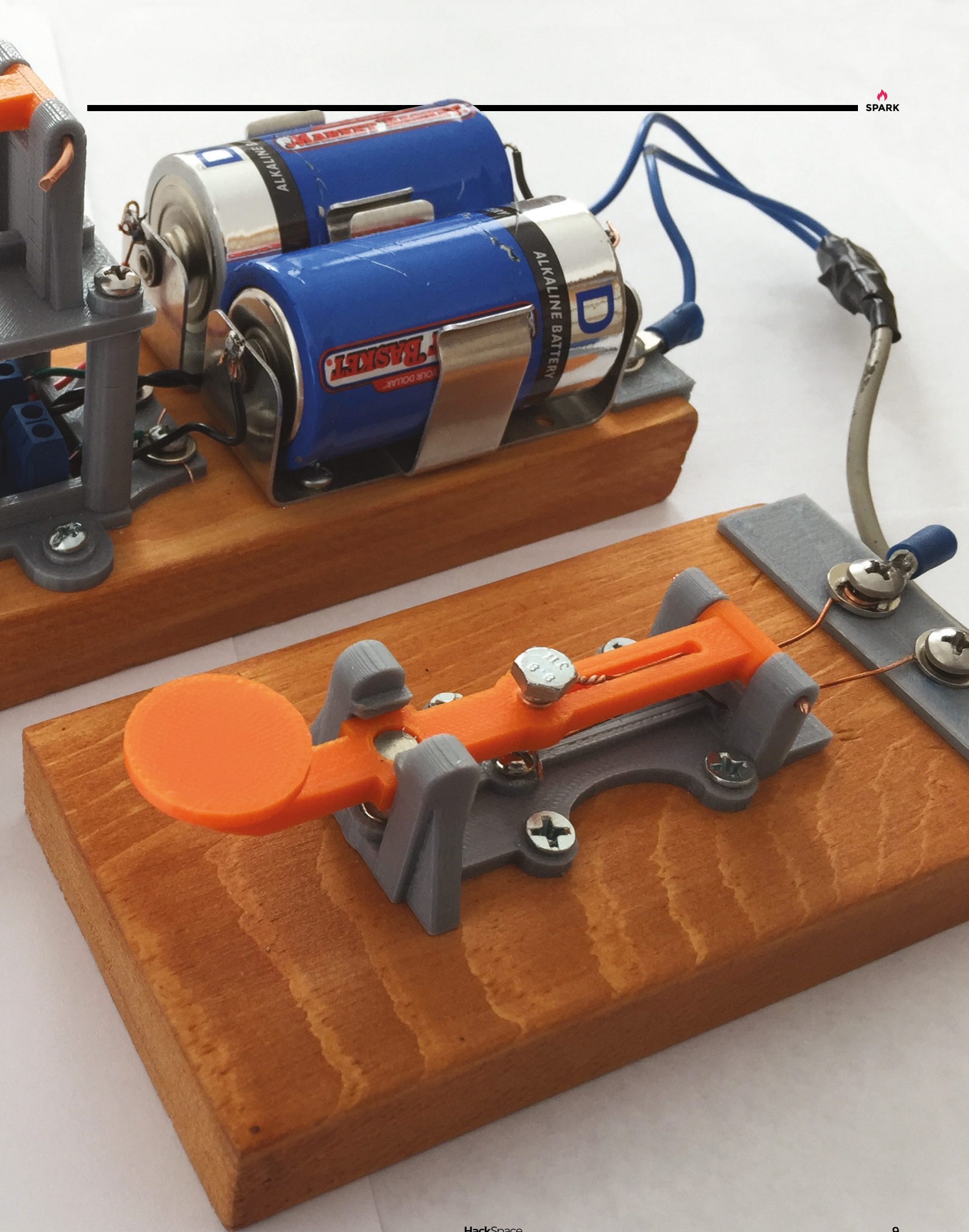
By Mattosx

 [hsmag.cc/RKmia0](https://hsmag.cc/RKmia0)

**T**his is a complete telegraph system – including both the key and sounder device, ready to be printed. It was designed in Tinkercad, and was printed on a Dremel 3D20 printer. There are some modern improvements, such as using neodymium magnets, instead of springs, to hold up the key and sounder arm. The sounder also uses a MOSFET to improve the sensitivity and reliability of the key. It can even record telegraph messages on a strip of paper with a marker. I am showing this retro-tech to my students as a part of an electricity unit where students can test their ‘texting’ capabilities old-school. 











# Networked balloon LEDs

By **Samy Kamkar**

[samy.pl](http://samy.pl)

**S**amy Kamkar is an independent security researcher, intent on showing the world how everyday kit, such as smartphones, is horrifically insecure. He's also a pretty good person to bring to a party. For this project, he created a network of independently addressable LEDs suspended within hydrogen balloons, all connected via ATtiny24 microcontrollers, and controlled by a gesture-based interface running on a laptop, all for a friend's birthday party.

Samy went through a ton of engineering options, considering everything from wires, batteries, to lift gas – to the extent that he even attempted to make his own hydrogen gas. He started off thinking that he had to put batteries in each balloon, but ended up sending 48V up to each balloon through the tether wire. If you've got half an hour, we highly recommend listening to the talk he gave at the recent Hackaday Superconference: [hsmag.cc/sWziYH](http://hsmag.cc/sWziYH). □



**Left** □ Filled with helium, each balloon has a payload of around twelve grams. That's not a lot to work with!

# The Joy Spreading Machine

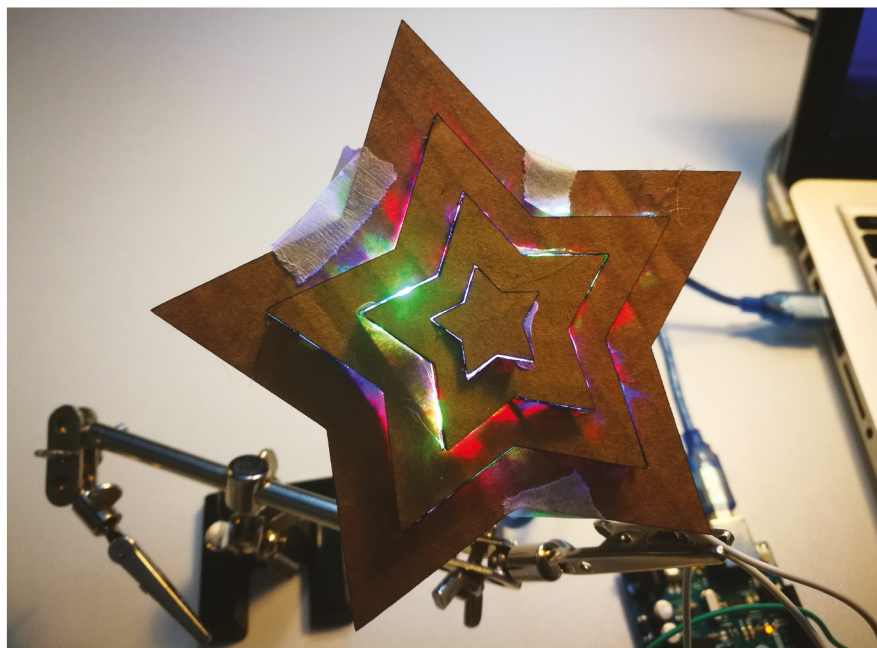
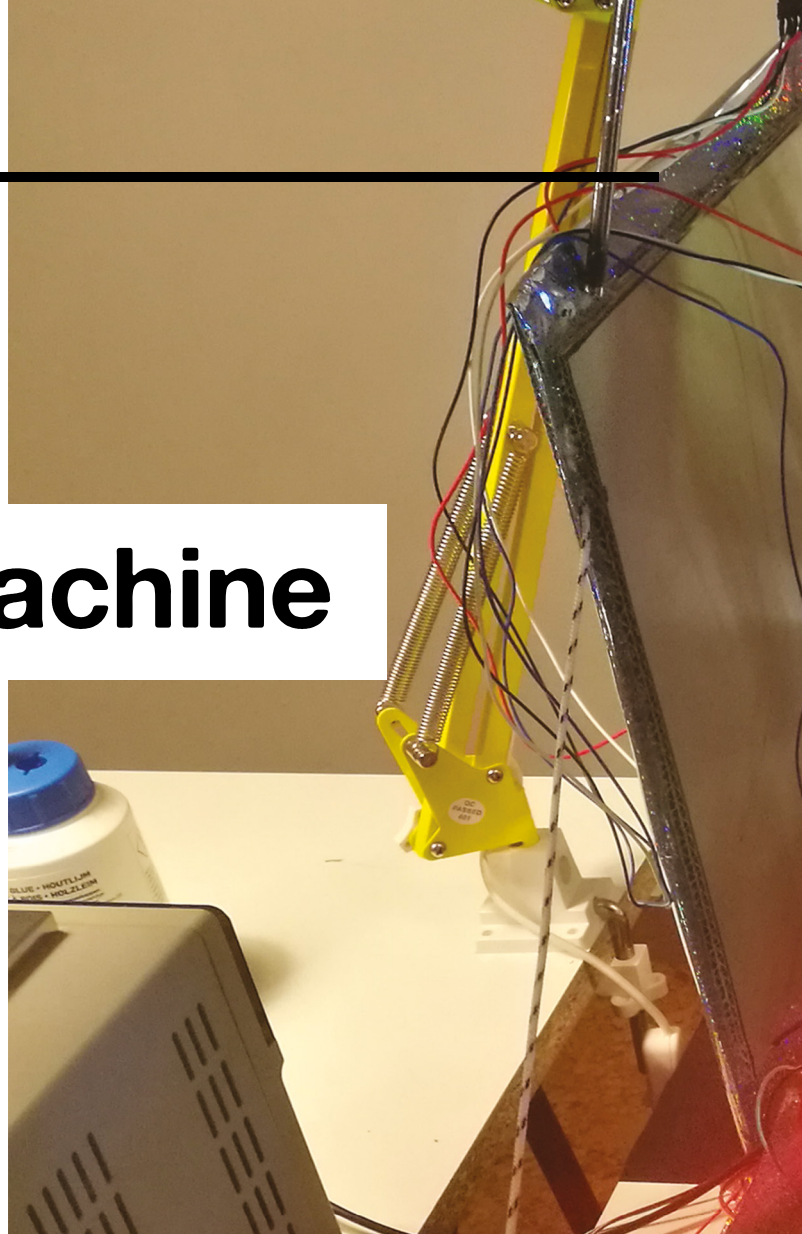
By Enrico Miglino

[hsmag.cc/GrWmAb](https://hsmag.cc/GrWmAb)

**T**his project is part of a series of original Art-a-tronic creations with educational purpose; together with the electronic parts and components, these projects are controlled by a micro:bit board, while the building uses poor and recycled materials: cardboard, adhesive tape, hot glue, metallic clips, popsicle stick, and other materials all easy to find at home.

Thanks to its great versatility and affordable price, the lighting effects of this project are obtained with some Adafruit NeoPixel addressable LEDs.

The project has been named 'The Christmas Joy Spreading Machine', as it has been created to celebrate Christmas through some metaphor of the traditional Christmas celebration: the lights, the music, the frenetic movement in the urban environments. □



**Right** □  
This project is part of a two-part book Enrico is writing, called *Physical Computing: Advanced Projects for Makers*



# Crazy Circuits dress

By Kitty Yeung

 @KittyArtPhysics

**T**his dress uses my painting of flowers as the fabric. I cut the fabric according to the outlines of the flowers and stitched them into a dress. There are LEDs embedded underneath the flowers to light up like fireflies. The LEDs blink according to the wearer's heart rate, which is detected with an DFRobot EKG monitor.

The microcontroller is an Arduino Nano, with Crazy Circuits' sewing breakout board. There are also additional LEDs under the tulle. The circuit is constructed with Crazy Circuits' sewing components and conductive tapes.

I recently published the construction process and story on Hackster.io: [hsmag.cc/EgjbeM](https://hackster.io/hsmag.cc/EgjbeM). 



**Right** 

We can see this working brilliantly at 127 BPM – disco heaven!



# Objet 3d'art

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

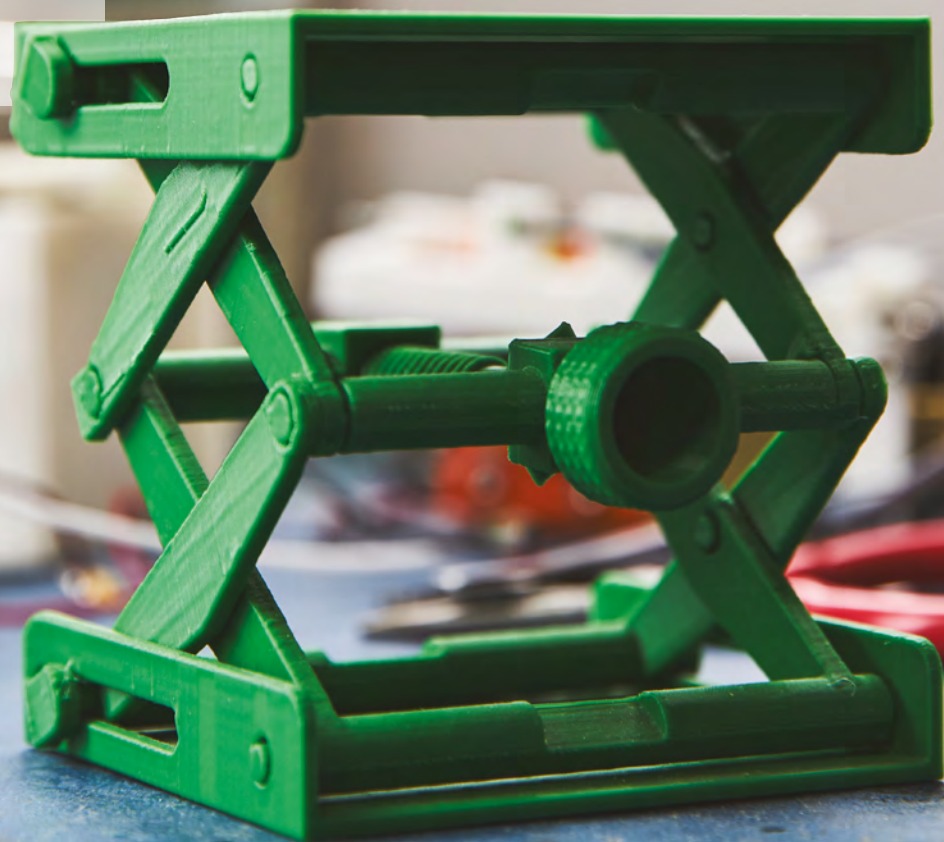
**W**

**e spotted this in the RepRap workshop when we went to meet Dr Adrian Bowyer (page 60).**

It's a little screw jack, printed on a RepRap machine in eSUN PLA. The most impressive thing to us is that it came off the print bed like this – already assembled, with no need to sand or smooth components to make it work.

You're not going to get a car on it, but if you want to raise or lower the height of a lens, it's perfect. □

[hsmag.cc/HbOhNe](http://hsmag.cc/HbOhNe)

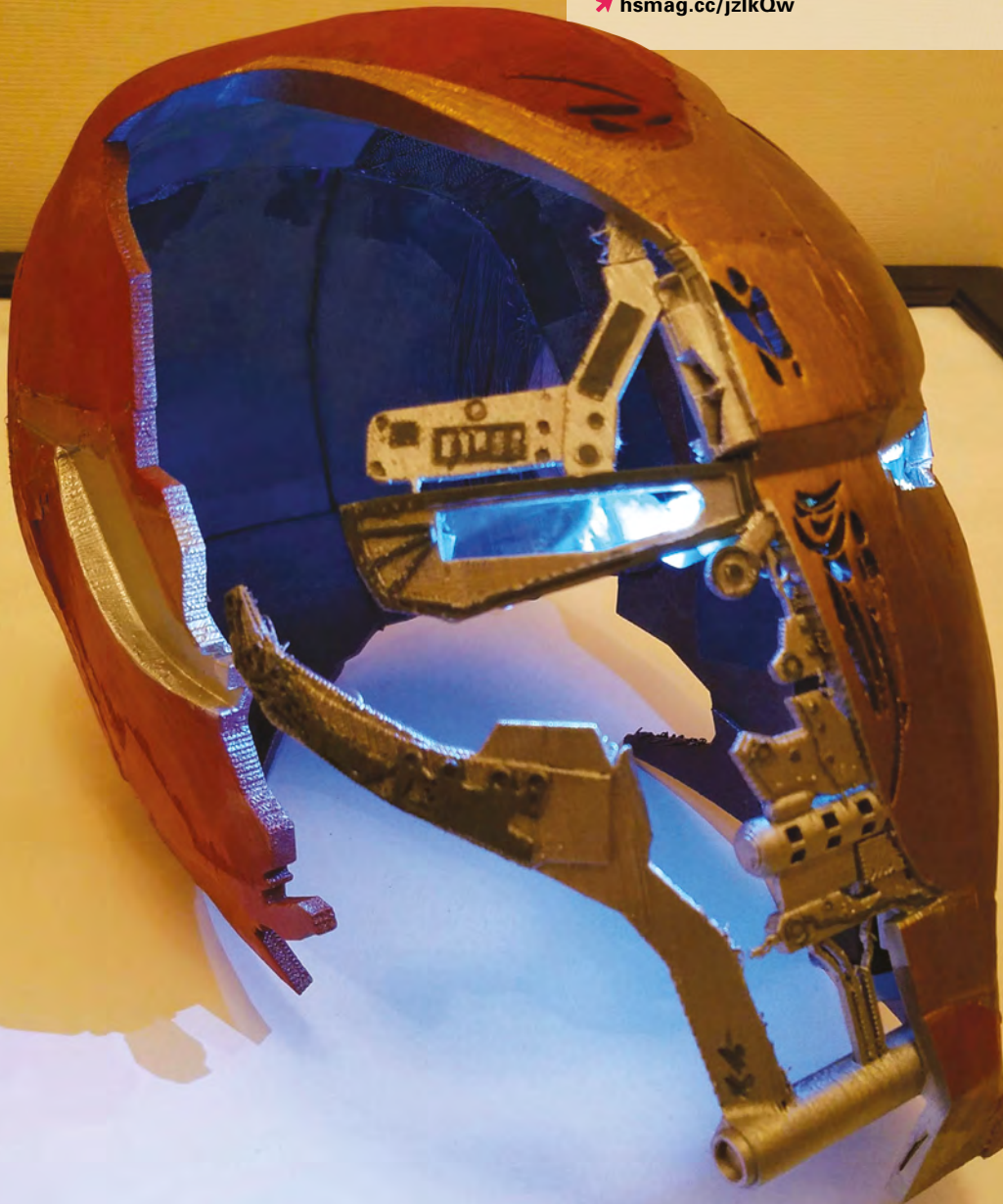




**T**he hype for Marvel's *Avengers: Endgame* has got us all excited at HackSpace Towers, so here's a film prop that you can make in time for the premiere. It's

designed by Tommy Cantwell, a 15-year-old high school student from Fredericksburg, VA. He designed it in Fusion 360, printed it out over the course of two days on his Prusa i3 MK3, with another two days to sand and paint. Head to [hsmag.cc/jzlkQw](https://hsmag.cc/jzlkQw), grab the design files, and make your own. □

➔ [hsmag.cc/jzlkQw](https://hsmag.cc/jzlkQw)



# SNES-style RetroPie build

Gaming like it's 1999

By Mark Thorpe

[hsmag.cc/UOIQuJ](https://hsmag.cc/UOIQuJ)

**A**t one end of the upcycling scale people are taking old furniture, repainting it, and adding new knobs/handles: it's simple, it makes an old thing more useful, and it works.

At the other end, there's this Raspberry Pi Zero W-based build from Mark Thorpe. It takes an old games cartridge and re-uses it as the case for a Pi Zero

W, running the RetroPie games emulation platform. Mark's taken something old, added something new, and retained the spirit of the cartridge's original use, so you get the Proustian rush of wasted 1990s afternoons when you pick it up. He's even added a Pimoroni LED SHIM to the display, for a bonus rainbow effect.

The materials for the build cost about £30 from your local internet retailer, and Mark has documented the build here: [hsmag.cc/UOIQuJ](https://hsmag.cc/UOIQuJ). □



**Above** ♦ RetroPie gives you an interface that's easy to use with a controller

**Right** ♦ For the true retro experience, don't forget to blow into the cartridge before powering on





# Meet The Maker: Will Ferraby

Making knives by hand in the original Steel City



**W**ill Ferraby makes knives by hand, in a small workshop in the old cutlery-making area of Sheffield, very near the river Sheaf that once powered the factories there.

He makes just four knives a month, using the last stock of a steel first made in the area over 100 years ago.

"The steel I use, Silver Fox, is special not just because it's an old steel, and not just that it's the last of the Sheffield knife steel, though that adds to it. It is genuinely a very nice steel to work with, and it performs really well. I've not found anything better.

"I like that I can source it locally, because that's a key part of the ethos of why I do things. And I like that it's one of the very, very rare steels that's actually got


a name rather than a very long number. It was named after the guy who invented it; his surname was Fox and his nickname was Silver Fox because he had silver hair, he was getting on a bit.

"The very first batch was made about 100 years ago. It was a razor blade steel, and they dabbled with it, slowly increasing the level of carbon. There are different sorts of Silver Fox; this is Silver Fox 100, which means it's got 1% carbon in it.

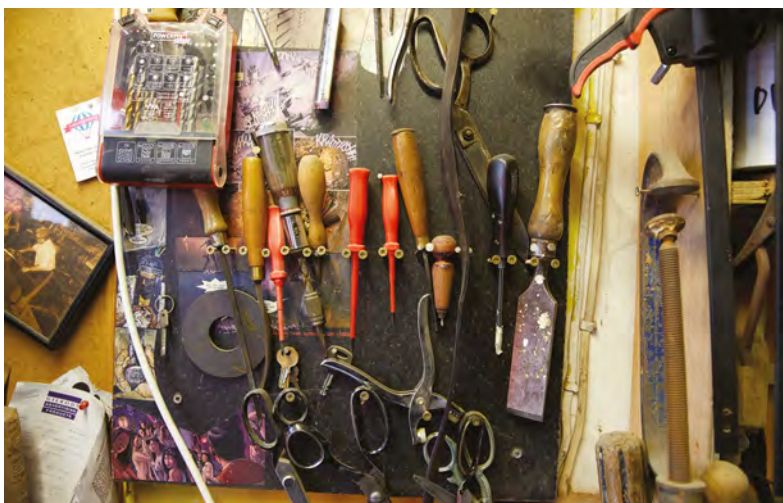
"For everyone who goes on about how good carbon steel is, carbon steel is rubbish, because it doesn't have any of the other alloys in it, and they're in there for a very good reason: to make the knife better.

"They say the first stainless steel knife was made nearby, at Portland Works. 1913 I think it was. They'd been playing about with it for ages. We're so used to the name 'stainless steel', but in the early days they came up with all sorts of rubbish names. 'Evershine', things like that.

"You see people nowadays making things out of Damascus, [where two kinds of steel are layered together and forged into patterns]. Is Damascus better than using just one steel? No, it's worse. And especially if you do it by hand, it's much worse. The hardening process is about crystals, so the metal crystals, they start out big and the hardening process makes them go very small. It has to be done at the right temperature. It's not an art, it's a science; it's either right or it's wrong. Damascus is using two steels all wrapped together, folded up nicely. Those two steels will require a different optimum temperature for hardening. Whichever you choose for the knife, the other one is not going to be hard. So, along the edge, you're going to have layers of hard steel and soft steel. You don't want that soft steel anywhere near the edge. →

**Below**  With old tools, even new makes are imbued with a bit of patina

**Images**  
Nigel Barker





Above  
"I don't do many commissions, but the last one I did, the client said: 'you're the knife maker, you make the decisions.' That's a great client!"





“The most important part of a knife is 1 mm back from the edge. And all the edges should be rounded. It shouldn’t have a bevel on it. If you were to sharpen it on a diamond steel, you’re putting a flat bevel straight into that, and the shoulder of that bevel stops you slicing cleanly with it. People always ask me what angle is the best for sharpening, but there’s no such thing. The steeper it is at the beginning, the stronger the edge is – there’s an obsession with really, really hard knives and a really acute angle – but that means you’ve got a really weak edge, and if it’s too hard, it’ll chip. Some Japanese knives are single-bevel knives, but they’re very, very specialist for slicing off. There’s a big difference between slicing something in half and just shaving little bits off the side.

### GOING PRO

“I started making knives when I was nine. I made knives all through my teenage years until I left home, and I was still really into knives, but I trained in environmental conservation. From there, I got into teaching woodland skills and survival skills, so I did that until I was about 26. It was a really nice job, but I

was behind a computer for a lot of the time, organising, doing risk assessments, reviews, funding applications, and I found myself one day with my knife in my hand, just about to stab it through the keyboard, and I thought, ‘it’s time for a move’.

“I’ve had five assistants over the years, but I’m back to it just being me now, which is nice because I can be late. It’s lovely being your own boss!

“It’s funny. I didn’t go in slowly. I decided that it was going to work. If it was going to cost me 110% of my energy, it would work. I was sure.

“As far as planning and stuff like that, I say follow your gut instinct and just go for it, and don’t think about it too much. There are a lot of barriers that are bigger when you think about them too much. Take a run-up and jump over them. Most of the barriers are in your head.


“I decided that I was going to make the best knife in the world. But there’s no such thing: it’s all about the process. If you feel that you’ve run the race, then you stop running. If you feel like you’ve found what you’re looking for, you walk around with your eyes shut.

“I’ve only been working totally professionally for eleven years now, but I’ve still got eight or nine →

“

**I started making knives when I was nine. I made knives all through my teenage years until I left home**

”

**Left**  “The most important part of making a knife is getting the right profile, to give it the ability to get sharp and stay sharp”



**Above** ■  
Will's workshop would be recognisable to a cutler from 100 years ago

years more on all the new knife-makers that are coming along. There's a massive boom in knife-making courses done by people who've only been doing it for a year. It's internet knowledge, and it's mostly nonsense.

"Funnily enough, I agreed very recently to do my first, kind of, lesson. It was with my friend's girlfriend who was really, really active in the Syrian refugee crisis. She went out to Greece and was working day and night. She was there for two years helping these people. At the time I thought that if there was anything I could do to even that debt towards her, then I would do. Her boyfriend asked me to teach her how to make a knife, so that's the first lesson I'm giving.

### NEW OLD THINGS

"It's funny in a way. I would almost describe myself as an anti-technologist, as in, the less technology the more I would be happier. If I had to drill everything out by hand, yeah, not that happy, but if it was powered by water, yeah, pretty happy.

"I would love my whole workshop to be powered by water power, that's my big thing at the moment.

Sheffield's got an amazing history of waterwheels. All the knife-making used to happen along the three big rivers. One of my pals works on the rivers and does a lot of projects there. One year he worked out that they'd got as much energy out of the river as was physically possible, by weirs and putting waterwheel after waterwheel after waterwheel. And all that energy, downstream, it's just the same as it would be. There's no waste.

"I am a passionate knife-maker, but overall I'm more passionate as an environmentalist. I think there's always a price for anything. If you've taken a shortcut and things are easier in some way, there's a price somewhere. That's why I don't want things to be done the easy way, which I suppose is at the heart of the handmade movement.

"It's an interesting one; I think it's to do with debt, and visual debt, and the way that nowadays the price you pay for things you're not seemingly paying for it, but it's happening either in the air, in the environment, on the other side of the world, all the stuff that's made in China; we don't even know who's making it.

"People like the idea of an in-house product, because the debt doesn't go out somewhere else. Energy in, energy out. It's simple as well; I think people like a simple life.





**Above** ♦  
 “This was my girlfriend’s granddad’s hammer (which he bought second-hand). It’s had a bit of life.”

**Below** ♦  
 The forge marks on each knife are different every time, and reduce suction when cutting through food

“I guess with handmade things, people get a deeper level of happiness. It’s an antidote to consumerism: people buy stuff to make themselves feel happier, and the less happy they make themselves, the more they then have to buy stuff. One of my things is that you buy a knife, you don’t have to buy another one. Even though it seems pro-consumer, the idea is that you buy less.

**GET A HANDLE ON IT**

“I think we’re happier when we’ve got anything old. If you’ve had a pair of shoes, for example, for 15 years, you get to like them more and more. Or, if you have a pair of trousers for so long that when they wear out, you’re sad, because they’re your favourite pair of trousers – but they weren’t when you bought them!

“I guess the value in old things is part of the vintage movement – it’s old, so it’s good. In order for stuff to get old, it’s got to be good in the first place.

“A pal of mine is really into saws. He bought a really old one, and he sharpened it. It took him a long time, but he never has to buy another saw. I found this fascinating. A new saw is better than an old saw, it will cut better because of the way they make them. But how much do you enjoy sawing? And how much do you enjoy sharpening?

“If I really enjoyed chopping out blades, I would carry on doing it. And if I thought that peeps really enjoyed the thought of me chopping out blades – or could even tell the difference — then that would be something as well. You’ve got to make your life as pleasant as possible.

“There’s an interesting thing with handmade versus factory. In order to have a business, you’ve got to

have a balance. It’s got to be efficient, and you’ve got to do all of the enjoyable bits. If you did everything by hand, it would be just pointless. My blades are cut out on a laser machine. If I did that by hand, like I used to do it not very long ago, I wouldn’t have any time to spend on designing a handle. That’s why everyone that makes a blade entirely from scratch, their handles are rubbish. If they factored in every single minute of time that they put in, they’d have to charge over £1000 for it. I mean, I do charge over £1000 for some of mine, but that’s because the handles are so interesting.

“This handle that I’ve just finished: I couldn’t be happier with it. I spent an entire day looking for a bit of wood with this exact curve. I took my time mixing up the resin to get the right colours, and I’m entirely happy with it. But is it the best kife in the world? No. There’s always a new idea coming along.

“The inspiration for this handle comes from an amazing tattoo – it was going up a girl’s leg from the very bottom all the way up. It looked amazing, especially with the contours. I looked at it, and my girlfriend looked over my shoulder and said: ‘I bet I know what you’re thinking’. You get inspiration from all over the place. The wood has been chopped up and then set just the right distance apart in a mould that I built. The resin is dark blue. No-one else is doing them like this. The resin runs all the way through like a stick of rock.

“I really like olive, beyond all other woods, because it’s got a truly wild grain. You’re always surprised. However long you spend looking at the side of it trying to work out what it’s going to look like, it will always surprise you. ‘What do you want for your birthday?’ ‘A nice surprise!’ It’s the best thing in the world!” □



**Left** ♦  
 Will uses a lot of olive wood, as it’s hard, looks good, and is available as a by-product of pruning rather than felling trees

# Go further together

Get by with a little help from your friends



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](http://guildofmakers.org)

**H**ave you heard the quote, 'If you want to go fast, go alone. If you want to go far, go with others'? I didn't think this applied to me – until I was invited to spend time with Rob Ives in his workshop. Rob ([robives.com](http://robives.com)) makes automata and paper crafts, such as flying pigs and skiing dinosaurs, and I have admired his work for years.

Knowing that Rob has a laser cutter, I asked if we could work together on some laser-cut mini mannequins. The 3D-printed mannequins I had made for a dressmaker ([hsmag.cc/WobOAB](http://hsmag.cc/WobOAB)) – took too long to print and I wanted more of them.

As I was trying to figure out how to slice my 3D CAD model for laser cutting, Rob was getting on with some other work. I was getting frustrated at my lack of CAD skills, and was about

to give up, when Rob suddenly said, "Google says there's some software called Slicer for Fusion 360" ([hsmag.cc/ixbhBA](http://hsmag.cc/ixbhBA)). Slicer "slices and converts 3D models into 2D patterns that you can cut out of any flat material" – which is exactly what I wanted – better yet, it is free! Why did I not know about this before?

Before long, a waft of burning cardboard permeated the workshop, and laser-cut jigsaw pieces were assembled into 3D models. We both got very excited

by the possibilities that this software offers. But I wonder how long it would have taken us to discover it if we had been working alone.

Rob had acquired a CNC router, but hadn't quite got around to getting it to work properly. It needed to be connected to a PC via a parallel port. Rob had an old PC and we turned it on. It beeped. It beeped some more. Nothing else happened. I took the back off the PC, and poked and blew at stuff. I plugged it all back in. It beeped. We were both about to give up when I videoed the beeping PC, and posted it on Twitter, with a call for help. Two people told me to lift out and reseat the memory

modules – which to me sounded a bit like 'turn it off and on again'. But I respect both of those people, so I followed their instructions. When I switched it on, it made the familiar Windows 'tune', and the monitor

flickered into life. We had been about to give up on the whole project but, with a little help from our friends on Twitter, we connected the PC to the router. The software took quite a lot more discussions and headbanging – but the CNC router is now up and running.

Although I probably wouldn't want to do it all of the time, occasionally working with others does provide the push to get out of my current comfort zone, and into amazing new things. Or, as the quote says – to go further. □

**Before long, a waft of burning cardboard permeated the workshop, and laser-cut jigsaw pieces were assembled**

# Towards security

Staying safe in an open-source world



**Bunnie Huang**

🐦 @bunniestudios

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

**K**erckhoffs's principle – the notion that a well-designed cryptosystem remains secure even if everything about the system, except the key, is public knowledge

– is a guiding tenet in the design of secure systems. Systems that unravel when inspected are said to rely on 'security through obscurity'. Renowned cryptographer, Bruce Schneier, goes even further to state that "obscurity means insecurity" ([hsmag.cc/Paokbi](http://hsmag.cc/Paokbi)).

I agree with Kerckhoffs and Schneier.

However, we must be careful not to assume the inverse is true: openness does not mean security. In hardware, every time you peel back a layer

for scrutiny, underneath is another problematic layer. Underneath proprietary boot ROMs lie microcoded state machines. Under the microcoded state machines are the analogue circuitry, trusted to establish

the power-on reset state. And under the analogue circuitry is a set of IC fabrication masks made by machines trusted to faithfully create, edit, and repair the masks. And under these mask-making machines lies a huge closed-source code base, trusted to render the mask patterns, which again runs on machines with proprietary boot ROMs... so clearly, the solution must be to build from the ground up a completely trustable, open-source, silicon foundry, right? Except between the foundry and you lies a courier who

can swap out trusted chips for malicious ones. An open-source silicon foundry built for the purpose of security is a billion-dollar investment that is defeatable by a thousand-dollar bribe paid to the right person, at the right time.

Transparency or openness of design, by itself, does little to secure a supply chain, because the entire situation is one huge TOCTOU (time of check to time of use) problem. It is like opening your source code for scrutiny, only to have a third party compile the code and send the binary unencrypted over the internet, and trusting it because 'open means secure'.

Of course, the solution to this problem in the software world is to hash the binary and sign it, thus defeating attempts

to tamper with the executable. Unfortunately, there is no equivalent in hardware because 'hashing' hardware involves the end-user taking it apart and inspecting every transistor and wire, which is both impractical

and also likely to render the hardware non-functional.

Thus, while open-source hardware engenders solid benefits for security by empowering us to weed out design failures, it addresses a separate problem domain from supply chain attacks. While an open-source hardware design is arguably more trustable than a closed-source hardware design, it is important to remember that open source is necessary, but not sufficient, for any given machine to be trusted. □

**Systems that unravel when inspected are said to rely on 'security through obscurity'**

# 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](http://hsmag.cc/hello)



### ROBOTS

A simple request: can we have some more robot builds please? I've mastered things with wheels, and I'm wondering how feasible it would be to make something like a Boston-dynamics-type walking robot (a simple version, of course).

**Rob Jones**

Middlesex

**Ben says:** Pi Wars is just around the corner at the end of March, which always gets us thinking about automatic creatures doing our bidding. I totally agree that, for a robot to be worthy of the hallowed pages of HackSpace magazine, it has to be more than just an off-the-shelf model though, so that complicates things a little bit. We're working on it!

### COLIN FURZE

In the States we have loads of attention-seekers – on the internet there are even more shouting about themselves. But too many of them don't have anything to say. Colin Furze has plenty to say! I spent about half an hour reading about Colin Furze [in issue 15] and since then a whole bunch of afternoons wondering how he's not killed himself yet. But I'm learning at the same time! It's always great to see someone who loves doing whatever they do – and even better when I love doing it too.

**David Peterson**

Minnesota

**Ben says:** A lot of YouTubers are an acquired taste, but it took us all of 30 seconds to acquire the taste for Mr Furze. Everyone should be watching his stuff – it's what the internet was made for.

### DANGER! HIGH VOLTAGE!

Your publication regularly has how-tos involving IoT devices (vulnerable to spying), drills, blades, hammers, and the things that could be dangerous in the wrong hands. So why do you insist upon warning me every time you do something with electricity? I'm not an idiot – I don't need to be told that high voltages can hurt me. Stop treating your readers like children!

**Thomas McKenzie**

Glasgow

**Ben says:** No. If we print a warning, it's because we want readers to be safe and it's because we don't want anything bad on our conscience. If that made you feel like an idiot, I'm afraid there's nothing we can do to help.



**BUYER BEWARE** 

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

# CROWDFUNDING NOW

## Maker LED Display


WiFi-connected blinking lights

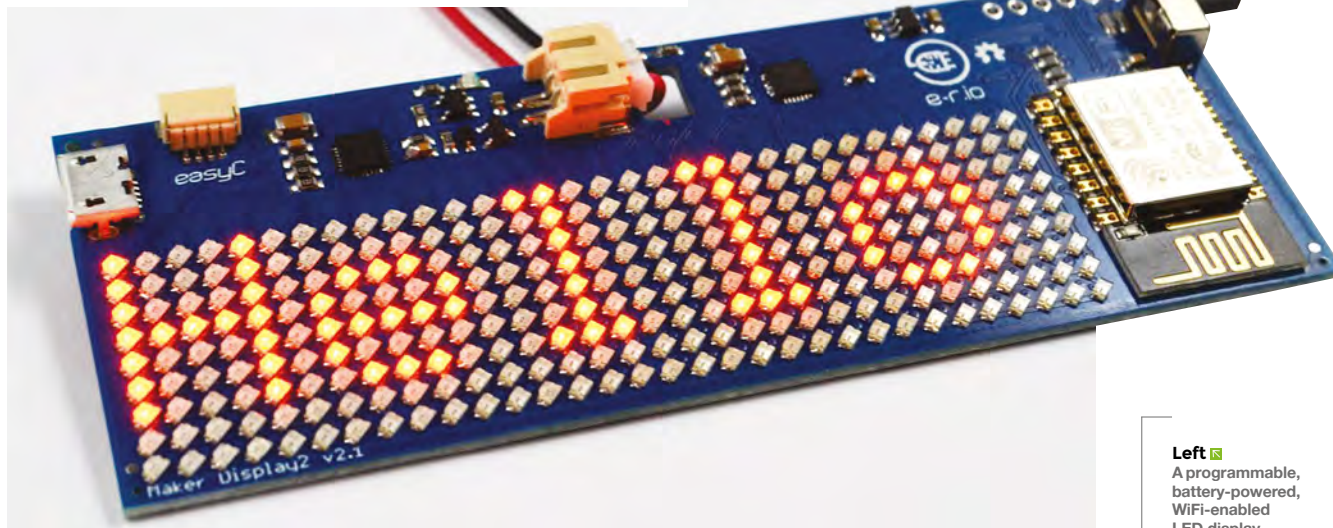
From \$29 | [hsmag.cc/ERVEsF](https://hsmag.cc/ERVEsF) | Delivery: May 2019

**L**ED matrices are a staple maker project. You can get modules and connectors for most common microcontrollers to wire everything together. However, wouldn't it be easier if everything came on one big board? That's what the Maker LED Display is. An ESP8266 controller (which has in-built WiFi), packaged together with a battery charger, and either a 32x8 LED display or a 64x8 display. Either size comes in a range of colours (though each display is a single colour LED, rather than RGB).

The LEDs are driven by IS31FL3731 drivers, which means that you still have four GPIO pins left for general use. In addition, there's an I<sup>2</sup>C connection in the EasyC form factor – this is similar to, but not as common as, the Grove or Qwiic connectors. An

EasyC connector comes with the board, so you don't need additional hardware to connect I<sup>2</sup>C devices.

There's nothing unique about this board – you could do everything that it can do by connecting together a few commonly available modules. However, by bringing everything together into one device, it keeps the cost down and makes it easier to use. There's an Arduino library to help you get your code started, and 3D printer files for enclosures. This looks like it'll be the quickest and easiest way to custom programmed LED displays. 



**Left**  A programmable, battery-powered, WiFi-enabled LED display

# Space of the month: NextFab



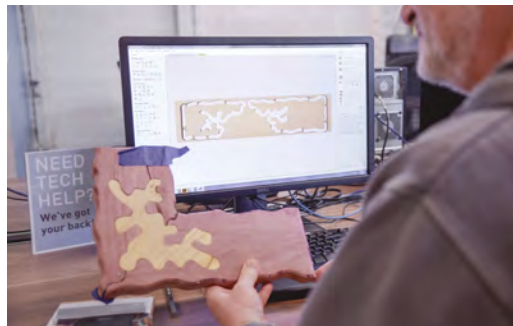
Marcelle Rice

[nextfab.com](http://nextfab.com)

In 2009, NextFab opened its first location in West Philadelphia's University City Science Center. Since then, NextFab has expanded to become a regional network of makerspaces, including two locations in Philadelphia and one in Wilmington, Delaware. With over 1000

members, NextFab provides access to tools and training but, most of all, it fosters a community of creative and collaborative people.

NextFab also offers a range of professional services including corporate packages, product development, and startup support. The RAPID Hardware Accelerator programme, which provides business coaching and seed funding to eight teams each year, has recently been accepting applicants for the Spring 2019 cohort. Accepted applicants will go through an extensive twelve-week program to develop their business and prototype their product, and will receive up to \$25,000 in funding. □



## CONTACT US

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





Below  
In West Philadelphia born and made, in the makerspace is where I spend most of my days. Chilling out, maxing out, relaxing all cool. Doing some angle grinding out side of the school



# 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](https://rpf.io/makerspace)**



# LENS

HACK | MAKE | BUILD | CREATE

Uncover the technology that's powering the future

PG  
50

## HOW I MADE HEATING CONTROLLER

Control the central heating from  
a smartphone with Raspberry Pi

PG  
56

## OPEN-SOURCE MACHINE TOOLS

Using concrete and scrap to make  
things that make other things

PG  
60



## INTERVIEW DR ADRIAN BOWYER

One of the originators of open-  
source 3D printing on where we  
are and where we're going

PG  
34



## ARDUINO THE ULTIMATE GUIDE

The boards, the add-ons, and the  
programming you need to take your  
project to the next level

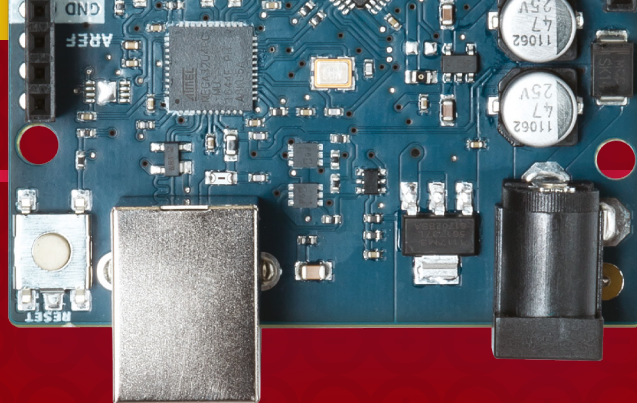
PG  
68

## IMPROVISER'S TOOLBOX SPONGES

Flexible, absorbent, and  
cheap starting points for  
bizarre household robots

# THE ULTIMATE GUIDE TO ARDUINO

Discover the boards,  
add-ons, and programming  
environment that make it  
easy to create interactive  
electronics projects



# A

## arduinOs are stripped-down computers.

They've got a little storage, some RAM, and a processor all packed into a chip. You write code for them on a normal desktop or laptop, then upload this over USB to run on the board. They're cheap (ranging from a few pounds to a few tens of pounds), widely available, and easy to use.

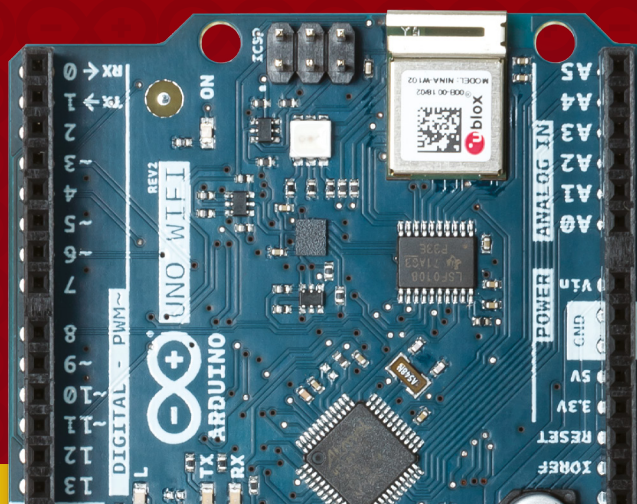
While the word Arduino can mean specifically boards that are designed and manufactured by the Arduino organisation, there's a whole range of Arduino-compatible boards available. These range from boards that are almost identical to official boards, to boards that are wildly different but that can still be programmed using the official software. We'll be looking at all boards –

both official and compatible – here, as few makers restrict themselves to just one type.

We use these boards to add tiny programmable brains to our project. They can be combined with LEDs to add visual effects, speakers to add sounds, motors to add movement, and a vast range of sensors to bring in information to be processed. It's a cliché to say that the possibilities are endless, but they really are.

What's more, they've been designed to be as easy to use as possible. The programming environment has many of the complexities of many programming tools, and they're designed so you can get started without having to know anything about electronics – add-ons known as shields can just be pushed on top.

Now, let's dive in and take a closer look at the hardware. →



# OUR FAVOURITE ARDUINO- COMPATIBLE BOARDS

Pick the best board for  
your next project



**Arduinos are boards made by the Arduino LLC company. However, as the designs and software are open-source, there's a large number of 'Arduino-**

**compatible' boards.** Exactly how compatible depends on the boards. All of them can be programmed from the Arduino IDE. Some of them are also pin-compatible with official Arduino boards, and can use the same hardware.

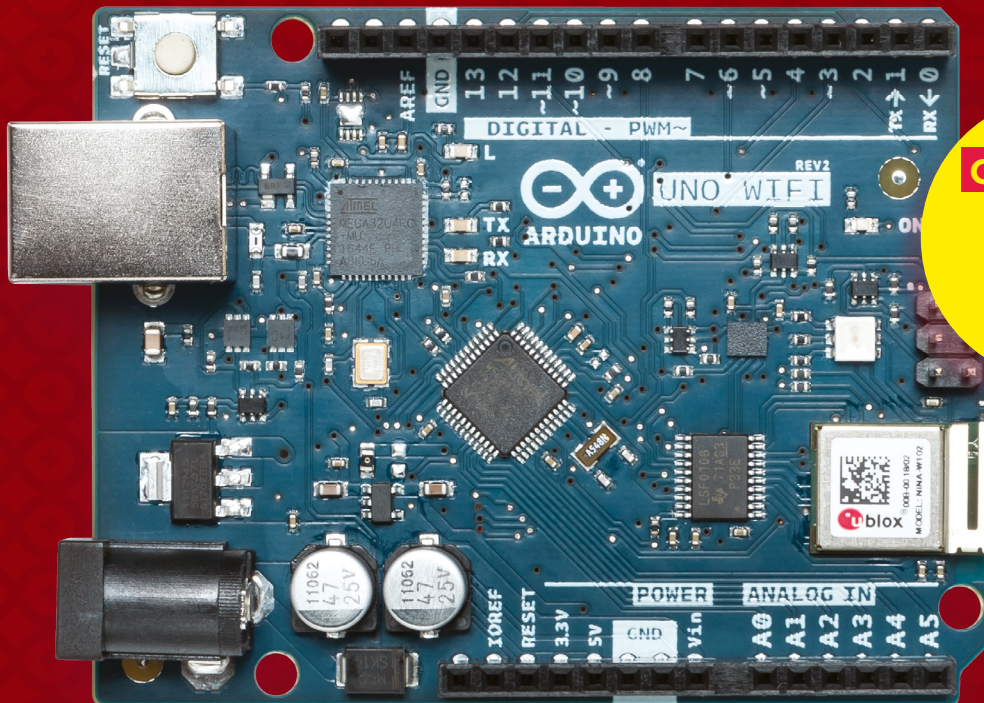
There are literally hundreds of Arduino-compatible boards, and we can't cover them all here, so let's instead take a look at a few of our favourites. Each one is specialised for a particular use.

# ARDUINO UNO WIFI REV2

ARDUINO | €38.90 | [store.arduino.cc](https://store.arduino.cc)

**T**he Uno has been the board of choice for beginners for nine years. While it's been through a few revisions in that time, the first major change came out in 2018 with the Uno WiFi Rev2. Including WiFi makes this an ideal starter board for anyone looking to make internet-connected projects.

Perhaps the most attractive thing about this board is the number of add-on 'shields' that have been created for it over the years. There are hundreds – possibly thousands – of these that make it easy to add more features to your projects. Unlike many other boards in this form factor, the Arduino Uno WiFi is 5V, so is guaranteed to work with all shields built for the original Uno. □



**ALSO  
CONSIDER**

**ARDUINO  
UNO REV 3**

FEATURE

# ARDUINO MKR WIFI 1010

ARDUINO | €27.90 | [store.arduino.cc](http://store.arduino.cc)

**T**he MKR range makes it easy to build high-quality, reliable Internet of Things projects. Each board includes a battery charger and secure crypto element on a robust PCB. There's a range of official add-on boards designed for the needs of small and medium businesses, which includes the capability to connect to CAN buses and communicate via RS-485.

As well as this, there's extra software support in the form of Arduino IoT Cloud (see page 44) that makes it really quick to hook your projects up. There are a few MKR boards that connect to the wider world in different ways, including LoRa and Sigfox, but for most projects, good old-fashioned WiFi is the best option. □

**ALSO CONSIDER**  
FEATHER HUZZAH  
SPARKFUN ESP32  
THING



**BEST FOR**  
INTERNET  
OF THINGS

# FEATHER M0 ADALOGGER

ADAFRUIT | \$19.95 | [adafruit.com](http://adafruit.com)



**ALSO CONSIDER**  
TEENSY 3.6

**BEST FOR**  
DATA  
STORAGE

**A**dafruit's Feather range is made up of lots of boards the same shape, with the same pinouts, but with different microcontrollers at their heart. This means that one set of accessories works with all of them.

One common set of builds that we do involves sensing and storing information. This could be environmental data – such as the amount of pollution in the air, or how much moisture there is in soil, or information about how a device is used. For this, SD cards are a great solution, as they're small, cheap, and easy to use with both microcontrollers and computers. The Feather M0 Adalogger has a microSD card reader on the board itself, so you don't need any extra hardware. □

# METRO M4 FT. ATSAM51

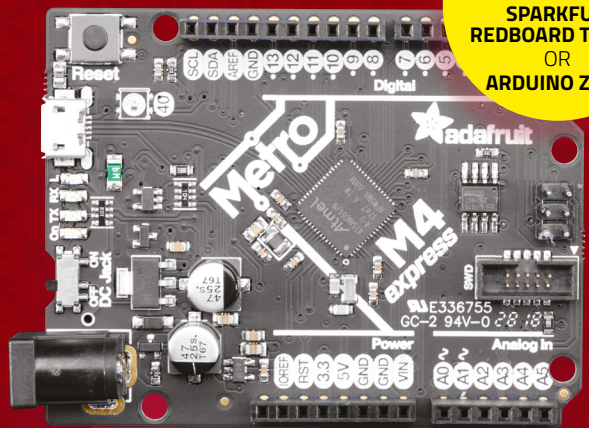
ADAFRUIT | \$27.50 | adafruit.com

**T**he original Arduinos all used AVR microcontrollers. These are easy to work with, but lack a bit of oomph compared with the more modern ARM

Cortex-M series microcontrollers used in many newer boards.

Metro M4 Express keeps the traditional Arduino Uno form factor, but pops in a more powerful processor. When it comes to raw processing power, the Metro M4 Express flies past the Arduino Uno and even the faster official board, the Arduino Zero. As well as having a faster processor, it's got a floating point unit, so if you're doing maths, the advantages are twofold.

If you need to crunch through a lot of data, this is a great choice. □

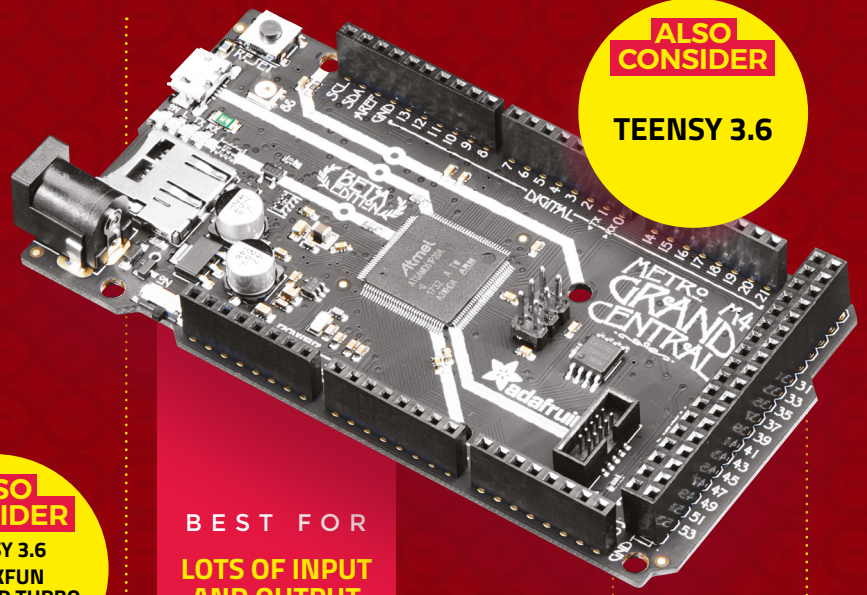


**BEST FOR**  
**PROCESSING POWER**

**ALSO CONSIDER**  
TEENSY 3.6  
SPARKFUN REDBOARD TURBO  
OR  
ARDUINO ZERO

# GRAND CENTRAL M4 EXPRESS

ADAFRUIT | \$37.50 | adafruit.com



**ALSO CONSIDER**  
TEENSY 3.6

**BEST FOR**  
**LOTS OF INPUT AND OUTPUT**

**W**e looked at this board in issue 16, but it's got a whopping 54 IO pins, and the same blistering-fast processor as the Metro

M4 Express. It's in the same form factor as the Arduino Mega and Arduino Due, so there's a range of add-ons already available, even though this board is brand new.

While this is quite a big board, there's a good chance that if you need this many IOs, then you're working on a big project. Whether it's to control lots of lights, lots of motors, or get lots of inputs from sensors, when we need lots of IOs, the Grand Central is now our board of choice. □

FEATURE

# RASPBERRY PI 3B+

RASPBERRY PI | \$35 | [raspberrypi.org](http://raspberrypi.org)

**M**

any people have been using Raspberry Pis to run the Arduino IDE but, since March 2018, you've also been able to program

Raspberry Pis directly from the Arduino software. This is particularly useful because it links into the cloud IDE so that you can control your Linux machines from anywhere on the web.

Unlike traditional microcontrollers, you can run multiple sketches simultaneously, and even install and remove software. This ability to manipulate machines with a few clicks makes it easy to update embedded systems without having to physically access them or learn to use remote administration tools aimed at sysadmins.

Head to [create.arduino.cc/devices](http://create.arduino.cc/devices) to get started. □



**BEST FOR  
LINUX  
COMPATIBILITY**

# ESP8266

VARIOUS | \$2-3 | various sites



**ALSO  
CONSIDER**

**BLUE PILL  
ESP32**

**BEST FOR  
LOW PRICE**

**T**

here's a wide range of boards that are built around the ESP8266 modules that provide USB connectivity and break out the IO pins. They're

available on many direct-from-China websites for around £2 each. The stand-out feature at this price range is the WiFi connectivity. Usually, they come with a Lua firmware, but also support Arduino. There is essentially no manufacturer's support for these boards; however, there's an active community that provides libraries and tutorials for many common uses.

The pictured board has nine 3.3V GPIOs and one analogue in; however, this can only take between 0 and 1V (a simple voltage divider can be used if you've got a larger range than this). They're small, cheap, and well-connected – perfect when you just want to add the ability to remotely control an object. □

**ALSO  
CONSIDER**

**BEAGLEBOARD**



# TOP 10 ADD-ONS

Traditionally, Arduino add-ons have been called 'shields', and were designed to slot directly into the pins on boards in the Arduino form factor. However, as Arduino-compatible boards now come in many shapes and sizes, we can't be so restrictive. Here, we've included any hardware that can easily work with Arduino-compatible boards, regardless of form factor.

There are so many Arduino add-ons that it's impossible to pick one as 'best'. It depends on the form factor of your underlying board, your power requirements, and the features you need for your project. Here are ten of our favourites, but there are hundreds of others that can make great projects.

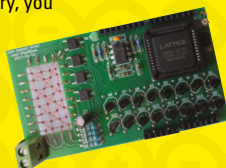
## 1. NeoPixels

We're stretching the definition of Arduino add-on almost to its limit to include these, but they're easy to work with on Arduino and have libraries that work with most boards, and that's good enough for us. You need to be a bit careful with the supply voltages if you're using a 3V board, but with a bit of care you can have your whole word blinking different colours.



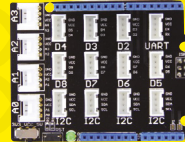
## 2. Core Memory Shield

Back in the early days of computing, RAM was made of loops of magnetic material. Wires crossed through these loops, and electrical pulses through these wires could induce and read magnetism. This way, small amounts of data could be stored. While we now have much faster, smaller, and cheaper forms of memory, you can still geek-out with old-fashioned ferrite core memory using this Arduino shield from [hsmag.cc/pzKiKq](http://hsmag.cc/pzKiKq).



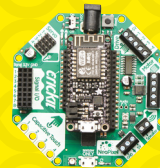
## 3. Grove Base Shield

This is a metashield. It doesn't have any hardware in itself, but it gives you the ability to add hardware easily to an Arduino Uno (or compatible) board. There are 16 ports that break out analogue inputs, UART connections, digital IO, and I<sup>2</sup>C into 'Grove connectors'. There's a range of Grove-compatible hardware that can then be plugged into this. It's easier to build up designs than using an evergrowing stack of shields, and more robust than using a breadboard.



## 4. CRICKIT

This add-on is available for several Arduino-compatible boards, including the Feather range from Adafruit, the Circuit Playground Express, the micro:bit, and the Raspberry Pi. It adds a bunch of input and output options, including a NeoPixel driver, touch sensors, motor drivers, servo controllers, and high-current IO. It's a great board to have on hand as it can be useful in so many projects.



## 5. OpenLog

If you've used Arduino, one of the first things you learned (after blinking an LED) was probably sending data to the serial monitor. This is a great way to get data from your Arduino to the computer it's plugged into. However, what if your Arduino isn't plugged into a computer? The OpenLog stores serial messages on an SD card, so you can log your data as usual and come back to get it later.



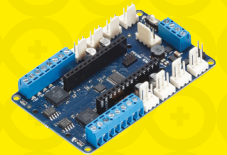
## 6. Adafruit Ultimate GPS FeatherWing

The GPS is truly one of the wonders of the modern world. We take it for granted now, but the ability to know your position to within a few metres anywhere in the world, or get an accurate idea of your speed, is miraculous. This FeatherWing includes a GPS receiver, a real-time clock, and a battery backup which all makes it easy to use with any of the Feather range. You can get similar add-ons for other form factors.



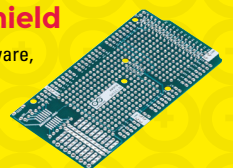
## 7. MKR Motor Carrier

Controlling motion with electronics is an art form. At the simplest level, it's just a case of a small DC motor, but as you look for more power or more precision, you need to consider more factors. The MKR Motor Shield from Arduino adds the ability to control four motors (two with encoders) and four servos to any of the MKR boards. There's also a charger for 2S and 3S LiPo batteries, which should keep your motors running when not near a power source.



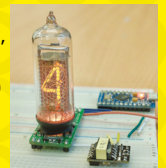
## 8. Protoboard shield

If you design your own hardware, you may well have started out with a breadboard. This makes it really easy to prototype things, but it's not very permanent. One option is to design a PCB, but if it's a one-off project with simple hardware, then it can be much quicker to solder it together on some protoboard. You can find this in the right shape to slot straight into most popular form factors for Arduino-compatible boards. There are a few different formats. Shown is the official Arduino protoboard for the Mega form factor.



## 9. Nixie tube drivers

Nixie tubes are gorgeous numerical displays from the days before LEDs. Put enough volts into the right pins and you're rewarded with a digit glowing orange-red. They can be a little tricky to drive from Arduinos, as they require a few hundred volts to run. The exixe board packages up everything you need, and lets you control them via SPI. There's even an Arduino library to take the hassle out of using these retro displays. Get yours from [hsmag.cc/xtdbjd](http://hsmag.cc/xtdbjd).



## 10. 3D printer

Many 3D printers are built on Arduino-compatible systems. The popular Marlin firmware is designed to work on the Arduino Mega 2560 and the RAMPS 1.4 add-on board. Many 3D printers use other hardware, but it's all built on this Arduino-compatible platform, including the Anet A8 (pictured). In a sense, then, all these 3D printers are really just Arduino add-ons.



# ARDUINO IDE

Let's get ready to program



Figure 1 Main interface

1. The main code area
2. Button to compile the code, but don't upload it to the board
3. Button to upload the code to the board (and compile if changes have been made)
4. Output from the compiler and uploader
5. Details of the currently connected board

T

**he Arduino IDE is, for many people, the place where hardware programming happens.** It is, by development environment standards, a fairly straightforward piece of

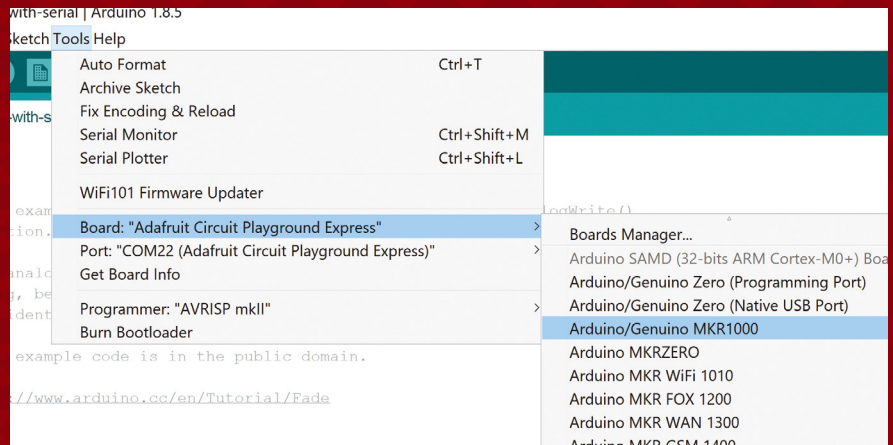
software, but it does the basics well: There's code highlighting, tabs for multiple files, and the ability to manage the compilation and hardware options. Let's take a look at the main features.

**Figure 1** shows the main interface with a program (known as a 'sketch' in Arduino-speak) that turns the built-in LED on and off every second. There are two functions: one called `setup` that runs when the board is turned on (or reset), and one called `loop` that runs repeatedly. This is the basic organisation of every Arduino sketch.

Perhaps the best feature of the Arduino IDE is its ability to support a vast range of different hardware. All this hardware requires different options to the software responsible for compiling and packaging the code, so you need to make sure that the software knows which hardware you're using. You can select the correct option in Tools > Boards (see **Figure 2**). If your hardware isn't listed there, you can add more boards via URLs (find the correct URL

## PROGRAMMING LANGUAGE

The Arduino IDE uses a language based on C++. If you've programmed in any similar language before, take a look at the example sketches included with the Arduino IDE. They're well commented, and take you from the basics upwards. Alternatively, there are loads of great resources to help you get started from any level of programming experience (including none). Graham Morrison's excellent tutorial series started in issue 5 (which you can download from [hsmag.cc/issue5](http://hsmag.cc/issue5) if you don't have it) and continues to this issue.



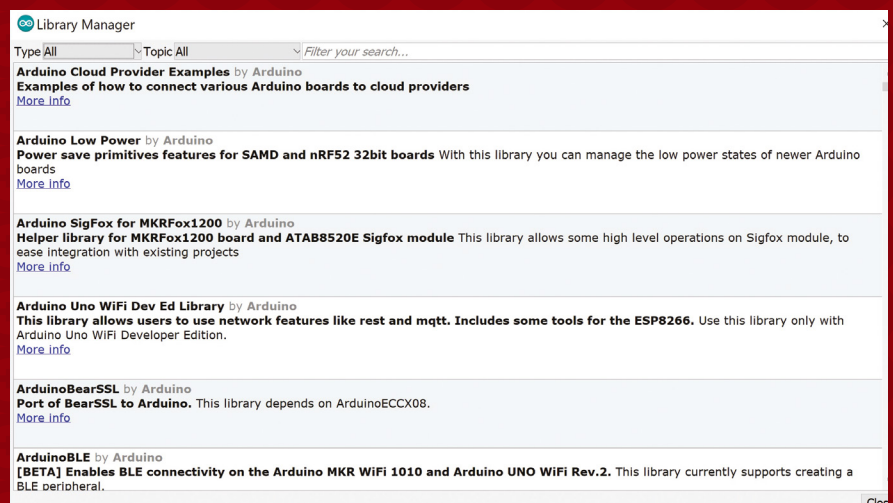
**Figure 2** As well as selecting the correct board, you'll also have to select the right port. You'll see the different options in Tools > Port

from your hardware supplier) in File > Preferences > Additional Board Manager URLs.

Libraries are packaged bits of code that are easy to reuse. There are hundreds of them for Arduino, most of which give you the ability to control hardware without getting bogged down in the nitty-gritty of how they work. The Library manager (in Sketch > Include a Library > Manage Libraries) gives you the ability to easily install and use these libraries (**Figure 3**). Many will also install example sketches (in File > Examples) that give you a chance to see how to use them.

One of the problems with embedded coding is that it can be hard to see what's going on. The serial monitor gives us a solution to this. It's a way of shovelling data through the USB connection between the computer and the board. Using the command `Serial.println()` (you'll also need `Serial.begin(9600)` or similar in your `setup()` function), you can send data from your board to the computer and display it using the serial monitor for text (in Tools > Serial Monitor) or the Serial plotter for numbers (in Tools > Serial Plotter). You can also send data in the other direction if you need to control your hardware from a computer. >

**Figure 3** Before diving into some complex coding, it's always worth asking yourself if there's a library that will make your life easier



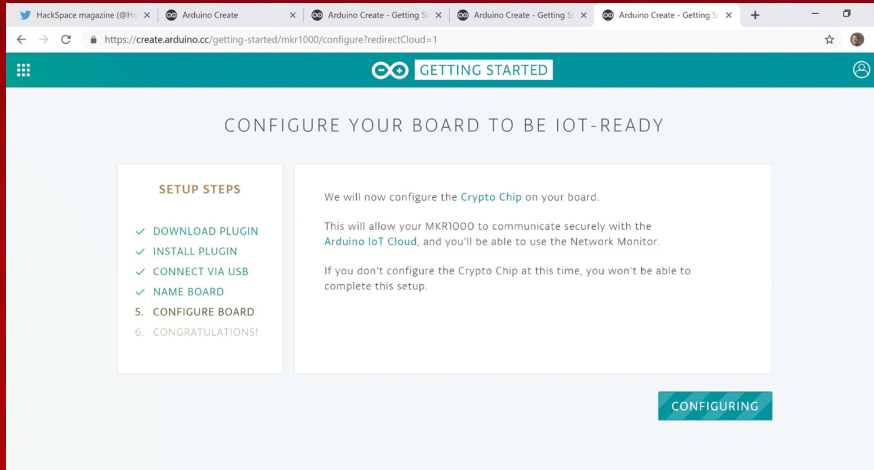
# EXPLORING THE ARDUINO CLOUD WITH A CONNECTED LAMP

Securely send data back  
and forth on the internet



**Arduino Cloud is the latest innovation from the team at Arduino.** It lets you control official WiFi-enabled boards via the internet, and send data back and forth

between a website and an Arduino. It's built to make it really easy to build secure IoT appliances using the familiar Arduino IDE. To get started, you'll need an account at [create.arduino.cc](https://create.arduino.cc).



We're going to create an internet-connected light which uses an Arduino MKR WiFi 1000, a motor controller, and a high-power (3W) LED. You'll be able to turn the LED on and off from both the internet and the light itself. This project gives a good overview of the different features we can use with the Arduino Cloud, and you should be able to take these skills and build a wide variety of different IoT projects. To build your first project, head to [create.arduino.cc](https://create.arduino.cc) and click on Arduino IoT Cloud. Click on New Thing to create your project. You'll need to follow the instructions to link your hardware with the project. Once it's created, you'll see the Cloud interface – there are two modes: the design mode where you can specify what cloud properties you want, and the view mode where you can interact with your thing. You can flick between these by selecting the pencil icon (for design) or the eye (for view).

In the design mode, you'll need to create three properties: Brightness, PWMLevel, and OnOff. The cloud IDE will automatically create variables with these names. This puts us in a bit of a conundrum: on the interface we'd like capital letters, while in the code our variables usually start with lower case. We decided the interface was more important, so have variables starting with capitals in the code. If your programmer sensibilities can't cope with this, you can switch to lower case.

To create a property, click on the + icon on the design tab and enter the following:

#### Brightness, which is:

- Type: Percentage
- Permission: Read and Write
- Update: When the value changes

#### PWMLevel, which is:

- Type: Int
- Min value: 0
- Max value: 255
- Permission: Read only
- Update: When the value changes

#### OnOff, which is:

- Type: ON/OFF
- Permission: Read and Write
- Update: When the value changes

**Left** All the MKR boards come with a crypto chip that can be used to ensure your communication with the Arduino Cloud stays secure. This is handled automatically

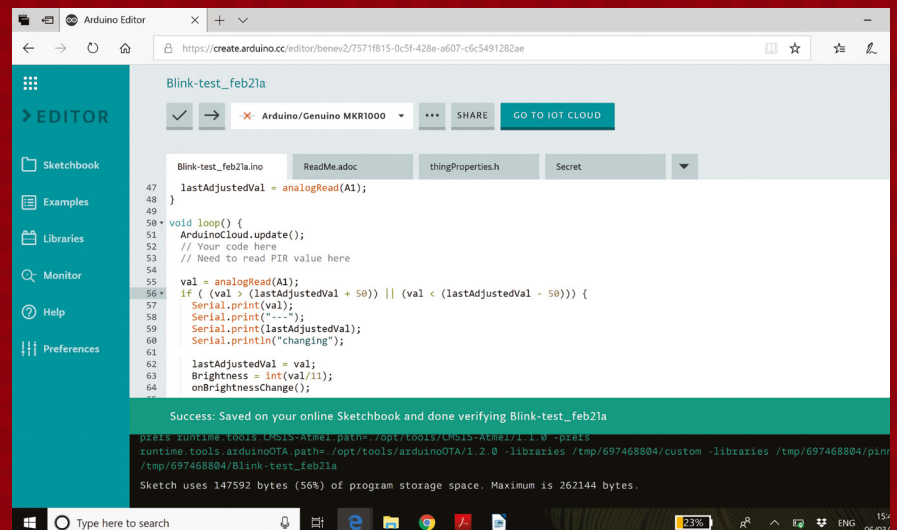
If you switch to the view tab, you should now see that you have widgets for Brightness, OnOff, and PWMLevel. Brightness and OnOff are interactive – you can vary them from this webpage. This won't actually do anything at the moment as we don't have any code listening for changes, so let's move on to that next. In the design tab, click on Edit Code. This will open the web-based IDE.

## START CODING

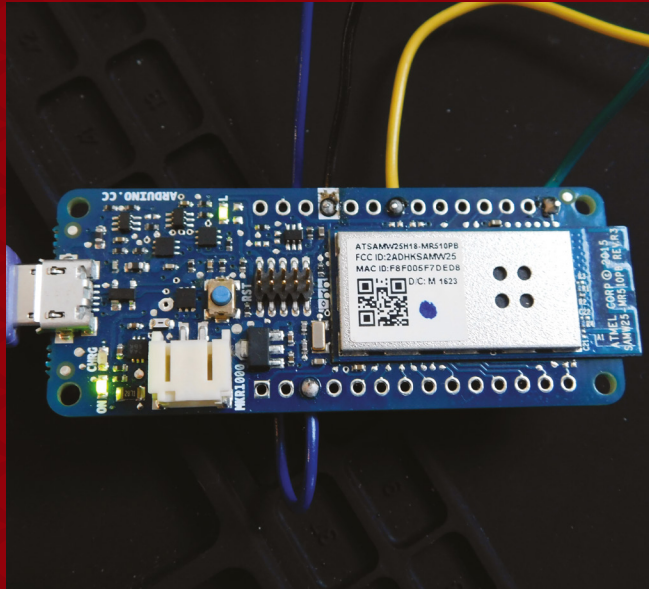
The Web IDE is laid out in a fairly similar way to the offline IDE, and the main part of the interface is taken up with a tabbed text editor. There are four files created by default: the main sketch, a readme, thingProperties.h (which will be managed automatically), and Secret. The Secret tab lets you add things that you don't want included when you share the project. By default, it contains entries for the WiFi SSID and password. You can enter these now.

Let's build our project iteratively. To start with, we'll just use the built-in LED on the board as the light, and only take input >

**Below** If you're familiar with the Arduino IDE, it shouldn't take long to find what you're looking for on the cloud IDE



## FEATURE



**Right**  You'll need to solder some wires onto the MKR WiFi 1000 to connect it to your lamp controls

from the internet. We'll be able to use the Brightness adjuster to turn the PWM level up and down, and when the OnOff button is switched off, we'll turn the light off. If it's off and the Brightness is turned up, it'll come back on, and if it's switched back on, it'll restore to the previous brightness level.

To do this, we need a variable to store the previous brightness if it's turned off, so add the following to the main INO file directly under `#include thingProperties.h`:

```
int prevBrightness = Brightness;
```

We also need to set the `pinMode` on our built-in LED to `Output`, so include the following in the `setup` function:

```
pinMode(LED_BUILTIN, OUTPUT);
```

Under the main `loop` function, you'll find two functions that are called when the user makes a change on the web interface: `onOnOffChange` and `onBrightnessChange`. In these, we can include the code:

```
void onOnOffChange() {  
  if (OnOff) {  
    Brightness = prevBrightness;  
  }  
  else {  
    prevBrightness = Brightness;  
  }  
}
```

```
Brightness = 0;  
}  
onBrightnessChange();  
}  
void onBrightnessChange() {  
  PWMLevel = Brightness*2;  
  analogWrite(LED_BUILTIN, PWMLevel);  
  if (Brightness > 0) { OnOff= true; }  
}
```

The first function here just turns the light on and off, storing and restoring values in the `prevBrightness` variable. It also calls the

## DEBUGGING

The most useful tool in debugging Arduino software is the serial monitor, which lets you receive information from your board. In the Web IDE, you can open it via the Monitor option on the left-hand panel.

The default code will include some output which is particularly useful in ensuring that you're connecting to WiFi and the Arduino Cloud, but you can also add your own data to this using code like:

```
Serial.println("some debug info");
```

`onBrightnessChange` function as this handles the actual PWM values set on the LED.

We created `PWMLevel` as a property rather than a regular variable, to show how these are handled in the web interface, and it's also a handy way of debugging any problems.

With this code added, you can upload it to your board (you'll need it plugged in via USB), and you should be able to use the web interface to change the brightness on the LED. It will take a few moments for the board to start up and connect to the WiFi. See the 'Debugging' boxout if you're having problems.

## ADD A LIGHT

That's the basic part of our software sorted, now let's move on to adding some hardware to make it a bit brighter. We used a 3V, 3W LED that draws up to 1 amp. You don't need additional resistors when powering it from 3V, so we can just wire it into our circuit. You need to make sure that you have adequate cooling for your LED. Ours was OK as long as it had airflow around it, but some need heatsinks or other ways of keeping the temperature down.

We can't just pop this LED on a GPIO pin, because that's far too much current. Instead, we need something to convert the low-power GPIO signal into a high-power signal for the LED, and we used a motor driver. Specifically, a SparkFun Motor Driver – Dual TB6612FNG, although any driver that can take PWM signals and handle 3V at 1A should work.

To wire this up, we need to connect the positive leg of the LED to AO1, the negative to AO2, pin six from the Arduino (the LED pin) to PWMA, and then set the configuration with AI1 and AI2 – AI1 goes to positive, and AI2 goes to ground (see the hookup guide for more details on this: [hsmag.cc/aVbFmx](https://www.hackspace.com.au/magazine/2017/07/10/arduino-wifi-1000/)). You also need to take the STBY (standby) line to positive.

We also need to hook this up to a power source. The Arduino's 3V line can't handle this much current, so you'll need an external 3V power supply. Since both the board and the LED can run off 3V, we can use one supply to power everything

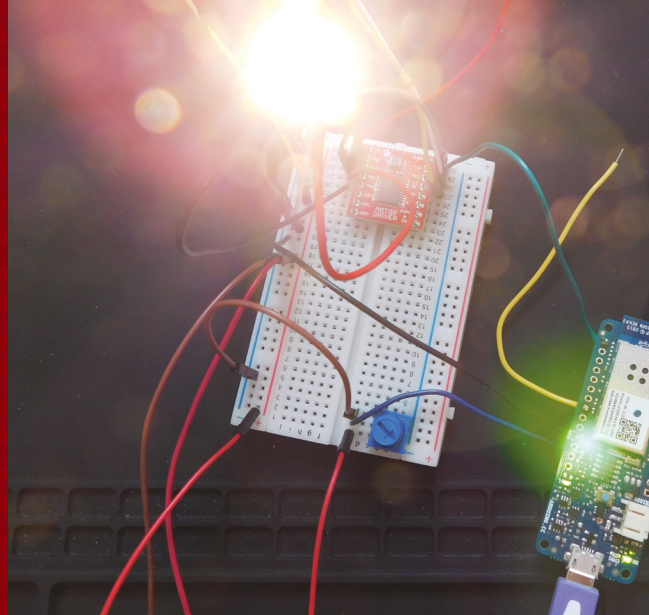
(connect VM and VCC to 3V, and ground to ground). We do need to connect ground from the Arduino to the power supply ground.

With that all connected, you can power up your Arduino and you should now find that you can turn the bright LED on and off from your Arduino IoT Cloud webpage.

## TURN IT UP

Now, let's switch it around so we can also adjust the brightness physically. First, we need a way of inputting the brightness we want. We've opted for a potentiometer which gives us a nice twiddly knob to adjust. The exact value of the potentiometer doesn't matter as it'll function as a potential divider, but about 10kΩ should work well. There are three pins arranged in a line. Connect one end to 3V, one end to ground, and the middle one to the A1 pin on your Arduino.

We need a little debouncing to know if a change is a real change or just a little bit of wobble in the reading. To do this we'll need two variables; declare them at the same place as `prevBrightness`:



**Left** ♦  
As long as your motor driver can cope with 1 amp at 3V, and can take PWM input, you should be able to use it for this project

```
int val;
int lastAdjustedVal;
```

In the `setup` function, we need to set the `lastAdjustedVal` (this will be the value where we last made an adjustment to the brightness:

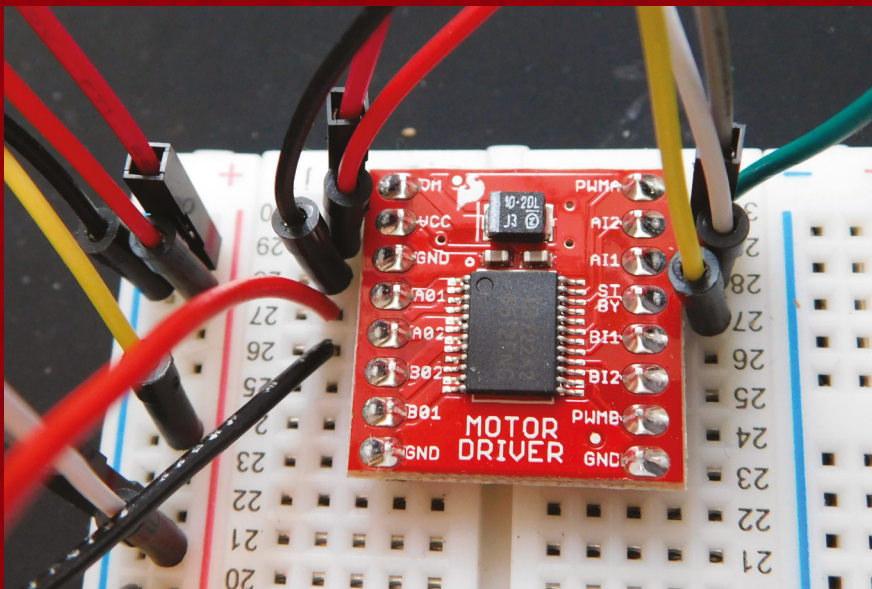
```
lastAdjustedVal = analogRead(A1);
```

The only other bit of code for this sits in the `loop` function and processes the value of A1:

```
val = analogRead(A1);
if ( (val > (lastAdjustedVal + 40)) ||
    (val < (lastAdjustedVal - 40)) ) {

    lastAdjustedVal = val;
    Brightness = int(val/11);
    onBrightnessChange();
}
```

**Below** ♦  
Remember, don't look at the bright LED when turning it on



This checks if the current value of the potentiometer is within +/- 40 of the value when we last processed an adjustment. We picked 40 because we found the value jumping by about 30 without the potentiometer being changed.

This just maps the potentiometer reading to a brightness level by dividing by 11 (this isn't quite accurate, as the analogue reading only goes up to 1024, but it's close enough for our purposes).

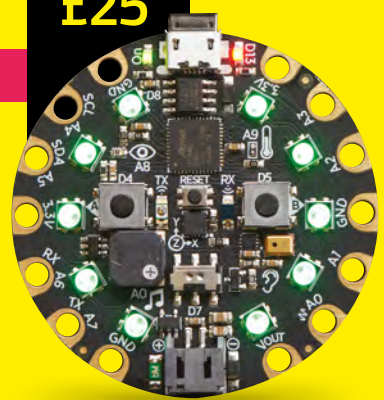
With that wired up and the new code uploaded, you now have a 3W internet-connected LED lamp. More importantly, you've also become familiar with the Arduino IoT Cloud, and can use this to easily, and securely, create a huge range of IoT appliances. □



# CIRCUIT PLAYGROUND EXPRESS

WITH 12-MONTH PRINT SUBSCRIPTION

WORTH  
£25



FROM JUST  
£55

12-month  
subscription  
from £55:

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

Offers and prices are subject  
to change at any time

Visit: [hsmag.cc/subscribe](https://hsmag.cc/subscribe)



# SUBSCRIBER BENEFITS ↓

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

## OTHER WAYS TO SUBSCRIBE

### Quarterly subscription

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

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

### Rolling subscription from just £5 a month:

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

## DIGITAL SUBSCRIPTIONS ALSO AVAILABLE



**Visit: [hsmag.cc/subscribe](https://hsmag.cc/subscribe)**

How I Made

# CENTRAL HEATING CONTROLS

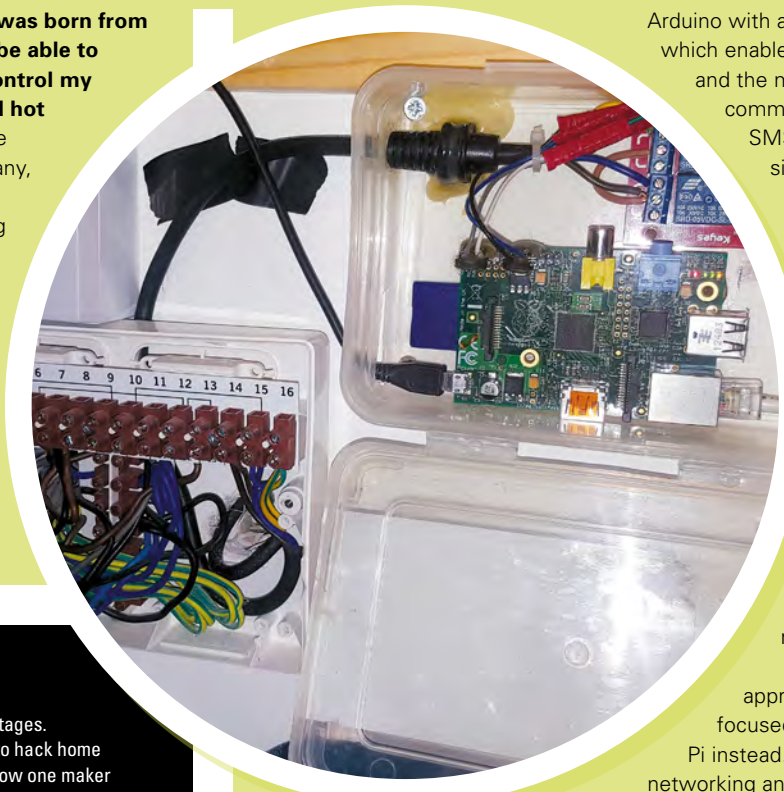
Automating my hot water system with a Raspberry Pi

By Plunkett Doyle

**T**his project was born from a desire to be able to remotely control my heating and hot water. There were not many,

if any, solutions available when I first started thinking about it, so I set out to develop my own and, as I had played around a little bit with both the Raspberry Pi and Arduino boards, this seemed like an obvious starting point.

My first iteration was quite simple and none too elegant; it used an



Arduino with a SIM card module on top, which enabled me to use a prepaid SIM and the mobile phone network to communicate with the Arduino by SMS. The code allowed for very simple on/off instructions, but I had lots of problems with SMS because the signal was not very strong in my airing cupboard – this often meant delayed or undelivered messages. Also, there was no feedback mechanism, so I didn't know if a request had been successful or not. The actual control of the system worked fine (via relays – we'll look at this in a bit), so I focused on the method of communication. I decided to scrap the SMS approach with the Arduino and focused on using the Raspberry Pi instead. My background is in networking and telecoms, but my coding skills were limited, so I had to learn as I went along. I hacked around with various options, trying to build the web app and server stack from scratch, before

Above ♦  
Original Raspberry Pi version  
with two-relay board

## Safety

Heating systems run at mains voltages. This article isn't a guide for how to hack home heating, but a demonstration of how one maker hacked his. If you choose to make any hardware changes to yours, you're risking injury or death to yourself or an expensive repair bill. Make sure you fully understand the risks and how to mitigate them before considering a project like this.

**Figure 1** ♦  
Current standard controller removed to show backplate

stumbling upon WebIOPi (see boxout) which is an excellent out-of-the-box web app framework that contained everything I needed, with some great online tutorials and code examples. It combines Python, HTML, Java, and CSS, and enables the development of a fully working web app and server infrastructure with built-in troubleshooting capabilities. The examples also include a macro function which I was able to expand to have one running for each heating zone – this can be set with one On and one Off time for any 24-hour period. I have found from using it that this is more than sufficient because it's so easy to set and change.

WebIOPi was a major breakthrough moment, and soon thereafter I had my first working web app, and was feverishly switching LEDs on and off on a breadboard via my smartphone. The responsiveness was really impressive – the LED would switch instantly upon pressing the button on my app, which changes colour when on (so I also had the necessary feedback mechanism missing in the SMS approach). This was very satisfying and provided the motivation needed to build out the relays and get a working model into operation ASAP.

**GETTING INPUT**

With a basic interface that worked, the next stage involved simply replacing the LEDs with the relays. I opted for cheap mechanical relays at a couple of quid each

WebIOPi was a major breakthrough moment

– which have worked fine and haven't let me down so far. The output that switches the LED on/off is simply connected to the relay control pin. Note: I did have to experiment with the relay boards and

adjust my code accordingly, as some relays activate with a high signal and some with a low signal.

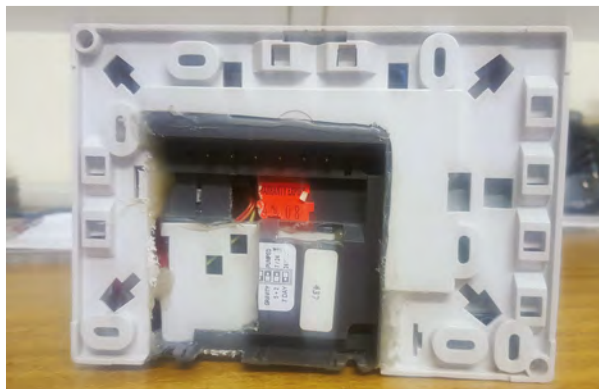
At this stage, it is important to

have a basic understanding of how my heating system works to understand how the device interfaces with it. I am not a heating engineer; this has all been self-taught. What follows works for me,

but your system may be different. My goal from the start was to replace my current controller/timer, which is the device used to switch heat/hot water on or off, and to set programs based on timers. (Figure

1 shows the controller removed from the backplate.)

My current controller is extremely user-unfriendly and difficult to program; so much so, it is rarely used for this purpose. Instead, we simply use it as an on/off switch. It also has no remote capabilities, so we can't change the settings when out of the house. >



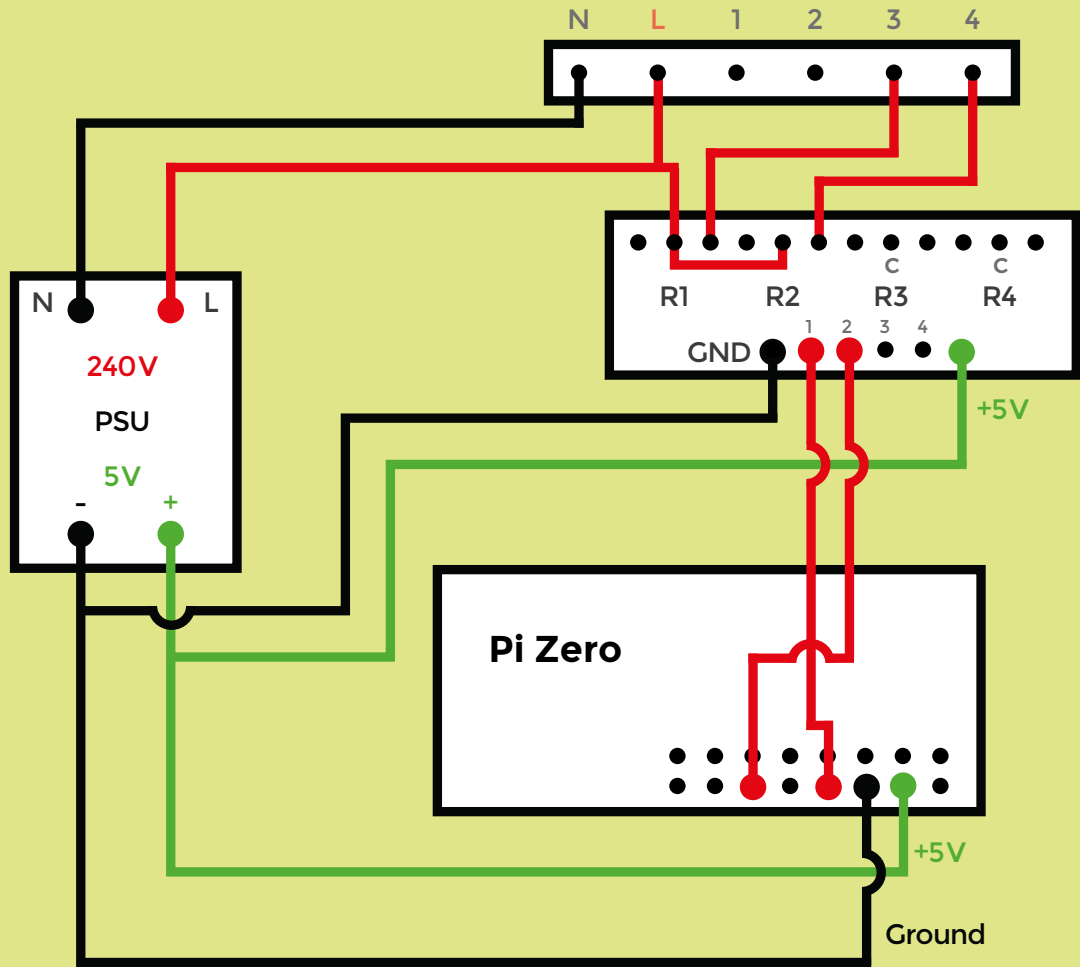
**Left** ♦  
The old controller connections merged into the back of the junction box

**WebIOPi framework**

Further information on WebIOPi framework can be found on the website, including tutorials, support, etc. The tutorial section is well worth a read, and much of my code is based on the examples there, especially on the 'framework basis' and 'Macros' submenus.

**Web:** [webiopi.trough.com](http://webiopi.trough.com)

## FEATURE



**Right**   
The circuit for Plunkett's heater. Some power supplies also require an earth connection

## Electronics

The schematic drawing shows only two relays in use, as per my own system. However, the code has been manipulated to allow up to four relay control.

The Pi Zero and the relay board receive the +5V supply from the PSU as shown, while the 240V side supplies the relays. The easiest way to think of a relay is as a switch similar to a light switch, for example, it has two states, on and off, which are controlled by a signal sent to the input pins (shown on the bottom of the relay board as four pins between GND and Vcc). Each relay has one control pin, and when a high signal is sent from the Raspberry Pi (e.g. when a button is pressed on the app) to that pin, the relay switches state and allows the flow of electricity (240V) from the L input pin to the relevant output pin. This then sends power through that pin and to the thermostat, and the process described earlier is initiated, resulting in heating or hot water switching on.

I already have thermostats in my system; I just wanted the ability to access it remotely with on/off and basic timer functionality. Also, I have serious concerns about allowing a so-called 'smart' device or corporation to have control over my heating, and ultimately my fuel bill, not to mention the potential security risks.

### HEATING SETUP

In a nutshell, my system has two heating zones: one upstairs and one downstairs, each controlled by a thermostat. When I switch the controller on for heat (using the one-hour boost function), the thermostats will make a call for heat for both zones (assuming the zone is colder than the temperature set on the thermostat). This

call for heat is simply a signal to open the motorised valve for that zone (or both in this case). When the motorised valve opens, it activates a contact switch which sends power to the boiler and pump, which subsequently heats the radiators. Once the

thermostat detects that the required temperature has been reached, the process is reversed and the motorised valve for that zone is closed off, and power to the boiler

I just wanted the ability to access it remotely

is turned off – or, the more likely scenario in my case, the one-hour timer expires and switches power off to the boiler. At this point, I need to highlight that the controller is connected to 240V mains power, so make sure you fully understand the risks, or consult a qualified professional. Mains electric can be fatal,

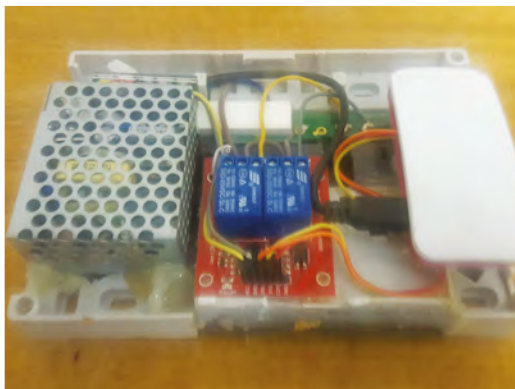
## Control more than your heating

Of course, the device is not limited to only control heating (although this particular device is tailored as such) but could, in theory, be interfaced with any mains application, giving you the ability to control up to four devices from your smartphone. I'm considering building another iteration which is integrated into a four-plug extension lead; for example, this way I could remotely control any four items. There is also no reason why more devices couldn't be added with the addition of more relays and some code manipulation. It could be useful for remote properties, allowing the ability to remotely control lights, heating, cameras, etc.

and you may only get one chance, so don't take any chances.

The enclosure needs to be robust enough to protect not just the electronics, but also people from the mains electricity inside of it. It needs to be sufficiently insulated and secure so that someone's not going to accidentally come into contact with mains electricity – and, obviously, you shouldn't open it while the power is switched on.

**Figure 2** ♦  
The first attempt to integrate the PSU with two-relay control



[Ed note: In some jurisdictions, there are legal requirements around wiring appliances directly into a junction box like this. Make sure you follow any local regulations.]

The first iteration of my new app-based model is shown on page 50 (it is an original Raspberry Pi, hence the Ethernet connection, but my newer models now use the Pi Zero W, which works great with WiFi). This one is actually hidden in the airing cupboard and connects into the junction box at the back of the controller; this is still my current working model and has worked seamlessly for two years now. The only downsides to this approach are that it requires an external power supply (for the Raspberry Pi and relay control pins) and that if I switch heat on via the app, the existing controller does not indicate that it's on. This is not a major issue, and the upside is that you can retain your existing controller and hide your new gadget out of sight, which tends to appease the house-proud.

### DROP IN

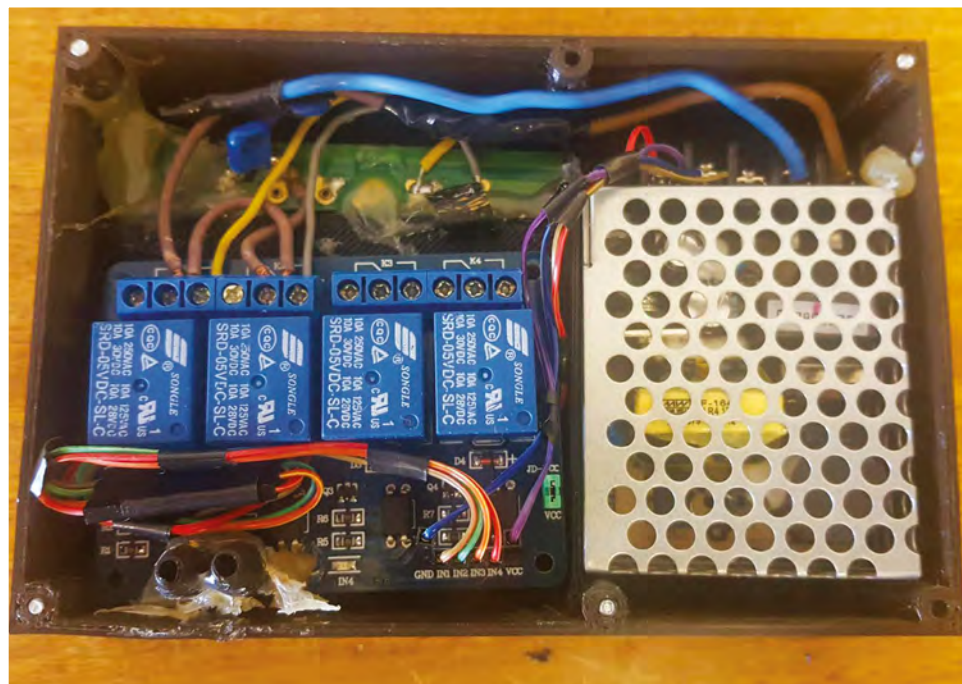
The goal, however, was always to make the device a direct replacement for the controller,



**Figure 4** ♦  
Box fitting to backplate

and be relatively easy to install, so I had to take it further. The controller connects to a backplate, which seems to be fairly standard and is compatible with various controller models; this is where my device will interface with the system. The backplate has six connections: the first two are for live and neutral mains supply, and the other four each have the potential to control one zone. In >

**Figure 3** ♦  
Complete four-relay solution, with the Pi Zero hidden under the relay board





**Figure 5** ♦  
Fusion 360 3D drawing

my example, you will notice only two are in use (3 and 4): one for heat, and one for hot water. In some larger systems, there may be up to four zones connected (e.g. underfloor heating may be a separate zone, or there may be two zones per floor, etc.).

The latest iteration of the device has the capability to control up to four zones; it also has four relays and relevant code updates. (In **Figure 3**, the Pi Zero is located under the relay board.)

The challenge at this stage was how to avoid the requirement for an external power supply, and the obvious solution was to

use the mains power already present and transform it to supply the Pi with 5V DC. After some more research, I opted for a switching power supply which takes a 240V input and outputs 5V DC. This works very well and can be seen in the photo of the

honestly) sort of merged into the back in a bath of hot glue. While the aesthetics leave a lot to be desired, it worked perfectly, and now the device was independent of an external power supply. [Ed note: powering the Pi via the headers like this bypasses the

protection circuits on the Pi, so be careful that you're getting a steady 5V, and not spikes of higher voltage.]

And so it remained, all I needed to do now was to get my hands on a nice, shiny consumer-style box that housed all the

components and included the pin connectors to attach to the backplate. Easy, right? I soon realised this was going to be a challenge. I could find plenty of generic boxes, but how would I attach them to the backplate? I also

While the aesthetics leave  
a lot to be desired, it  
worked perfectly

finished device (**Figure 3**). The first iteration of the backplate-compatible model is a telecoms junction box which has had the connecting pins of an existing controller (a previous one that died from natural causes,

thought about getting something tailor-made, but this seemed like way too much bother and expense at this stage, plus I would need dimensioned drawings first anyway. So I decided the best option was to purchase a 3D printer and maintain control of the process myself. The problem was, I had no clue how to use a 3D printer or software, so back to the bottom of that learning curve!

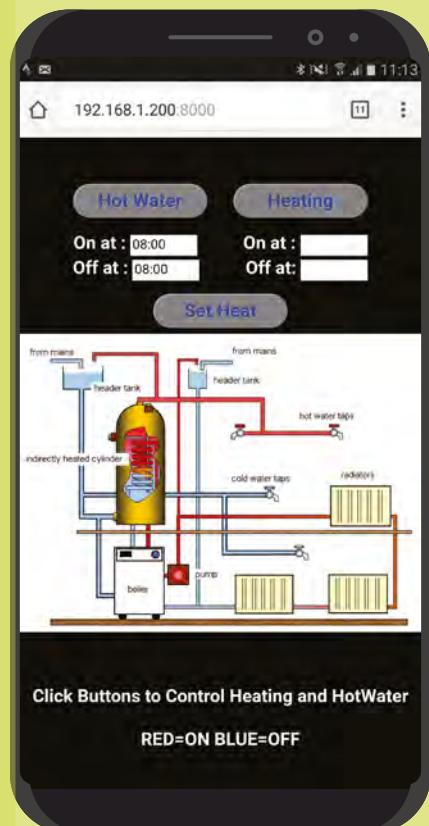
### THE THIRD DIMENSION

To begin with, I needed a 3D drawing, so I began to experiment with various 3D software packages. I'm not the sort of person who spends a long time analysing or

learning the theory of a product, but instead dived right in and started doing. I tried about three different free software products before settling on Fusion 360, which I found to be the most intuitive and functional. I immersed myself in it for about a week; about ten trial prints and a few handfuls of hair later, I had something which resembled a finished product – it still requires a lot of refinement but it works, and it's even more satisfying to know that I produced the box from scratch as well.

**Figure 4** shows how the box connects onto the backplate, and **Figure 5** shows the latest 3D drawing with raised holders for the Pi Zero and the relay board, and a separator to provide some physical isolation between the mains and the 5V side. It took quite a few attempts to get everything fitting properly, and the decision to purchase the 3D printer has definitely been justified, as I had the ability to implement changes immediately. **Figure 3** shows the finished

**Below / Below Right** ♦  
First iteration of app; and latest four-relay version



### External access

WebIOPi lets you control your Raspberry Pi from a web browser on your local network. However, if you want to control it remotely, you'll need some way of pushing that connection through to a public IP address.

There are many free DDNS services out there which offer reasonable performance, but personally, I find them a little bit glitchy, and they require updating when your IP changes. I prefer to use a VPN; if you're lucky you may have a public IP you can assign, although it's pretty rare on consumer-level broadband connections, unless you pay extra.

There are some risks opening your network up to the world like this, and you should be familiar with good security practices.

device without the lid. If you're wondering what the little binocular-like black tubes are for in the bottom left, these simply send the light from the relay state LEDs to holes on the lid (**Figure 5**), which I have covered with clear plastic sheeting. This way, there is visible indication when the heat or hot water

### I had no clue how to use a 3D printer or software

is switched on without adding extra LEDs for this purpose.

So, in conclusion, I now have a fully operational remote control system for my heating and hot water, which is accessible from anywhere that I have a data connection. The necessary components will cost around £25–£30, so considering that a standard controller like my old one will set you back approximately £70, there is definitely room for manoeuvre. □

FEATURE

# Pat's

## Open Source Machine Tools

Build a multipurpose metal working station from automotive scrap



Cameron Norris

@cameronsnorris

Cameron is a technology and communications specialist, passionate about the use of open-source hardware for social innovation.

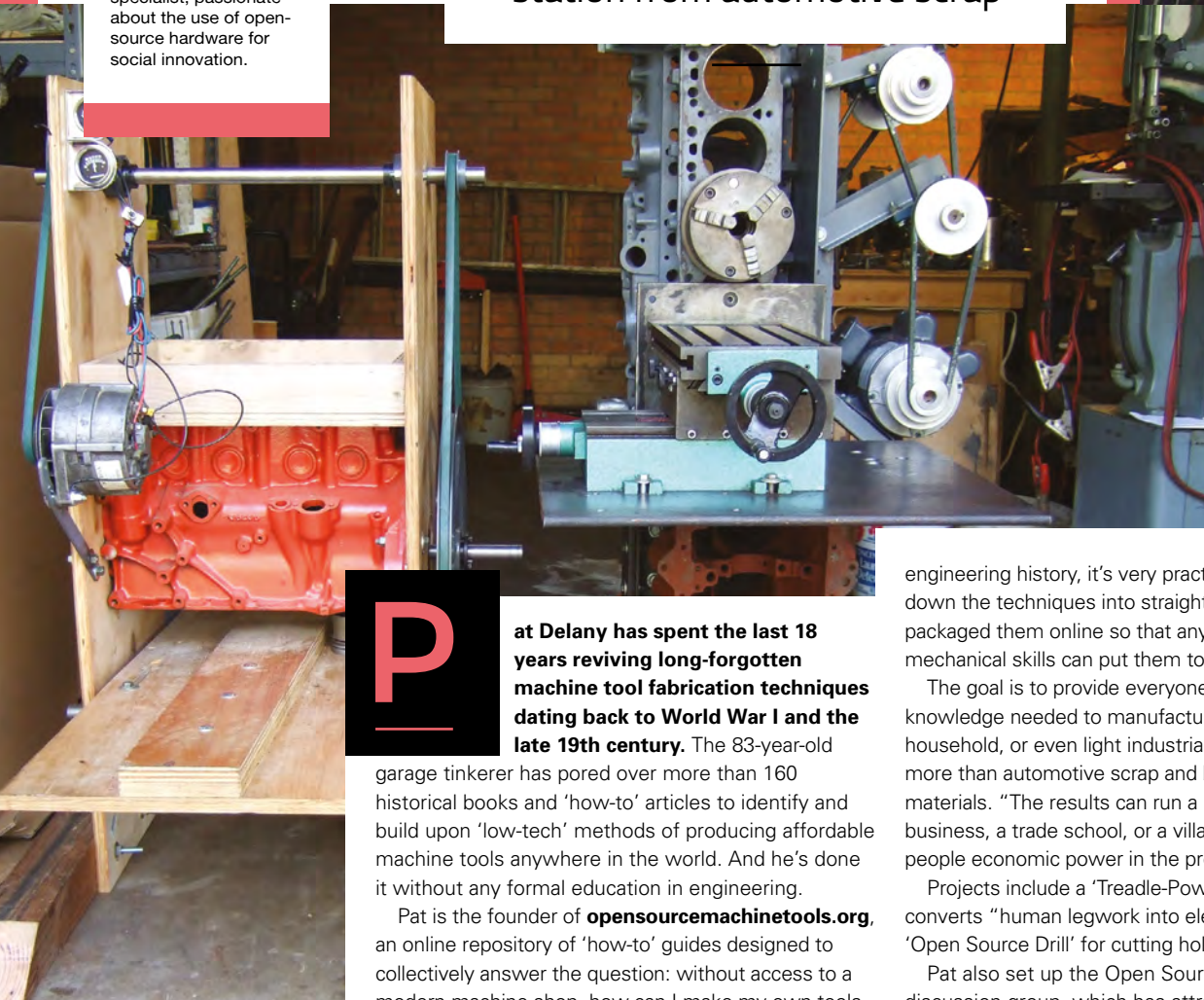


Above ♦ Pat Delany is the founder of the Open Source Machine Tools website

Credit Pat Delany CC BY-SA 3.0

Left ♦ The Multimachine (right) next to the Treadle-Powered Generator (left) at Pat's home in Texas

Credit Pat Delany CC BY-SA 3.0



**P**at Delany has spent the last 18 years reviving long-forgotten machine tool fabrication techniques dating back to World War I and the late 19th century.

The 83-year-old garage tinkerer has pored over more than 160 historical books and 'how-to' articles to identify and build upon 'low-tech' methods of producing affordable machine tools anywhere in the world. And he's done it without any formal education in engineering.

Pat is the founder of [opensourcemachinetools.org](http://opensourcemachinetools.org), an online repository of 'how-to' guides designed to collectively answer the question: without access to a modern machine shop, how can I make my own tools from scratch? As well as being a vast treasure trove of

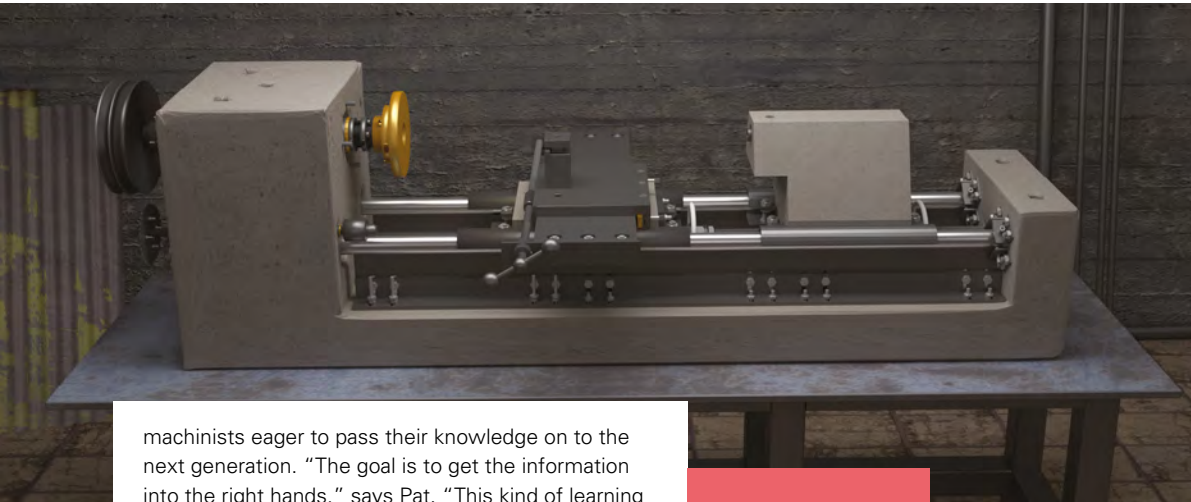
engineering history, it's very practical. "We've broken down the techniques into straightforward diagrams and packaged them online so that anyone with basic mechanical skills can put them to work," says Pat.

The goal is to provide everyone with the tools and knowledge needed to manufacture agricultural, household, or even light industrial goods using nothing more than automotive scrap and basic building materials. "The results can run a metalworking business, a trade school, or a village shop ... and give people economic power in the process," he reveals.

Projects include a 'Treadle-Powered Generator' that converts "human legwork into electricity" and an 'Open Source Drill' for cutting holes in hardened steel.

Pat also set up the Open Source Machine Tools discussion group, which has attracted over 8000 members, many of whom are retired engineers and





**Left** ◆  
A 3D render of the updated concrete lathe design created by Open Source Machine Tools community member, Tyler Disney

**Credit**  
Tyler Disney  
CC BY-SA 3.0

machinists eager to pass their knowledge on to the next generation. “The goal is to get the information into the right hands,” says Pat. “This kind of learning trains someone not just to machine metal, but also to measure and work accurately – the focus required to machine a surface to a thousandth-of-an-inch tolerance will carry over to almost everything in life.”

### PAT'S MULTIMACHINE

Pat's obsession began after he decided to build a horizontal milling machine based on the article ‘*Build the Engine Mill*’ by George Ewen, in issue 15 of *Machinist's Workshop*. After four years of tinkering, Pat successfully expanded the capabilities of George's original design to include the functions of a lathe and drill-press. The result was a low-cost multipurpose metal working station that Pat named the ‘Multimachine’.

In Pat's eyes, what made the Multimachine so significant was its ability to turn basic zinc and aluminium castings into potentially thousands of corrosion-resistant parts and products. An earlier interest in home metal casting led him to discover that common automotive scrap metal could be used to provide a low-cost and seemingly endless supply of these two versatile materials. It's also fairly easy to reach the required temperature with little more than a pile of sticks at your disposal, as the melting points of zinc and aluminium are only around 419 and 660 °C respectively. “Without this machining, the castings are of limited use, but simple machining can turn them into hundreds of different kinds of products,” he explains.

The Multimachine itself is constructed from two engine blocks that are used to support the entire

## HISTORICAL SOURCE

The oldest ‘How-To’ in Pat's collection is by Egbert Pomeroy Watson, who published *The Modern Practice of American Machinists & Engineers* in 1867. Egbert was a machine-shop engineer born in 1835, who became a regular contributor to early *Scientific American*. Today, his work is widely considered by historians as “part of the knowledge base of civilisation as we know it.”

machine assembly. According to Pat, this also provides the surface accuracy necessary to meet existing commercial machine tool capabilities.

The standard Multimachine build consists of five key modules: a vertical milling slide, drive unit, machine mount, overarm, and spindle. The vertical milling slide is used to securely hold and manipulate workpieces. A T-slot plate enables the use of jigs and fixtures for securing larger workpieces. For the drive unit, Pat recommends sticking to a single 450rpm speed until you're confident enough to set up your own variable speed pulley system.

Next, a modified overhead hoist is bolted to the main bearing pads of the engine block to form the machine mount. For the overarm, a steel pipe is filled with concrete and bolted to the inside of the first engine block cylinder, before a tail-stock is added to prevent longer workpieces from bending excessively while being cut. Finally, the spindle is a steel shaft placed through a bearing in the third cylinder of the engine block. The spindle is used to hold and drive the Multimachine cutting tools and must be perfectly aligned with the overarm, which provides additional support to longer workpieces. “This is our version of a 130-year-old design,” remarks Pat. →

**THE MULTIMACHINE IS  
CONSTRUCTED FROM  
ENGINE BLOCKS**

## BUILDING BLOCKS

Due to their high strength-to-weight ratio, zinc, aluminium, and zinc-aluminium alloys are commonly used to manufacture die-cast parts for vehicles. These parts include wheels, cylinder blocks, cylinder heads, pistons, brake cylinders, and suspension arms. Despite the name, one of the most common zinc-aluminium alloys, ZA-12, also contains small quantities of copper and magnesium, which provides even greater strength, toughness, and rigidity.

# Pat's

## Yeomans Wartime Concrete Lathe

**Below** ♦  
Curt Filipowski was the first Open Source Machine Tools community member to build the concrete lathe. Pat sent him a large three-jaw chuck as a prize

**Images**  
Curt Filipowski

**P**at's next big project was to design a low-cost general-purpose lathe that was large and robust enough to resurface truck brakes and clutches – a task that wasn't possible on the Multimachine due to the limited size of readily available engine blocks.

Two years after the project was announced, a member of the Open Source Machine Tools discussion group, named Shannon DeWolfe, introduced Pat to the work of Lucien Ingraham

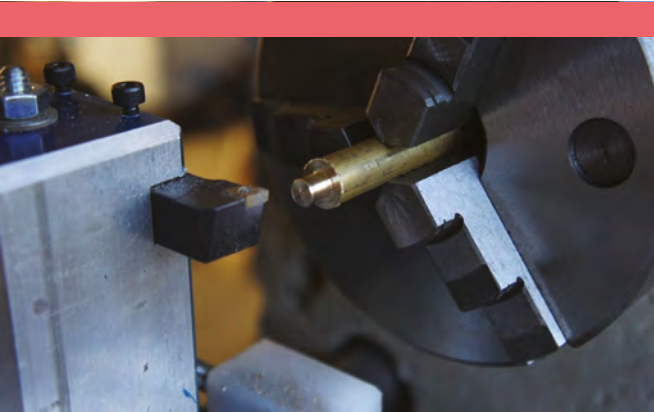
Yeomans, the first person to patent the idea of using concrete to reduce the time and cost of constructing machine tools.

### LUCIEN'S CONCRETE CONTRAPTION

Born in 1878, Lucien developed a novel method to quickly produce machine tools for the United States during World War I. The Allies were struggling with conventional cast iron framed methods of constructing machine tools, which were labour-intensive and time-consuming. In comparison, Lucien's method of using concrete, instead of cast iron, reduced construction time by around 180 days and the work could be carried out by semi-skilled general labourers rather than experienced metalworkers. Lucien's 'concrete machine tools' were so successful that he was honoured with America's highest engineering award, The Franklin Prize, in 1917.

"It is not too far afield to say that Lucien Yeomans' invention of concrete machine tools tipped the balance of power in World War I," says Shannon. "At the very least, his tools allowed the USA to have a more effective role in the European war, ending the stalemate of trench warfare."





## INSPIRE THE NEXT GENERATION OF GARAGE TINKERERS

original poured, non-shrinking metal with cement grout," says Pat.

With the support of more discussion group members, including Tyler Disney, technical drawings, 3D models, assembly guides, and other useful resources were shared online, enabling Pat to more effectively share his ideas and inspire the next generation of garage tinkerers.

If you'd like to find out more about the project, visit Pat's site: [opensourcemachinetools.org](http://opensourcemachinetools.org). □



**Left** ♦ The concrete lathe in action, here machining a round brass bar

**Above** ♦ Open Source Machine Tools community member Curt Filipowski pours concrete into his carriage mould

Thanks to Shannon's detective work, Pat had access to Lucien's century-old concrete machine tool patent, US1154155A, which would form the basis of his own low-cost general-purpose concrete lathe design.

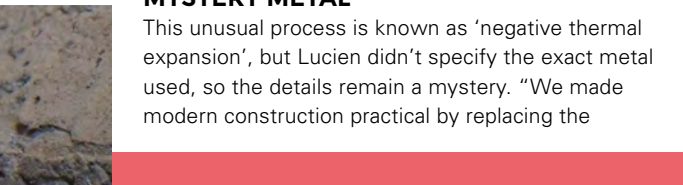
"It is well known that concrete shrinks as it sets," Pat explains. "This is not important when you pour your sidewalk, but this shrinkage would force a concrete machine tool out of alignment." To solve this problem, Lucien's method of production involves casting a concrete frame with oversized cavities for the machine tool's working parts and bearings. After the concrete has been left to set for around 28 days, the parts are secured in place by filling the oversized cavities with molten metal that expands as it cools.

### MYSTERY METAL

This unusual process is known as 'negative thermal expansion', but Lucien didn't specify the exact metal used, so the details remain a mystery. "We made modern construction practical by replacing the

## THE ORIGINAL METAL LATHE

The first fully documented metal lathe capable of manufacturing standard screw thread sizes was invented by Jacques de Vaucanson around 1751. Jacques' invention allowed the concept of interchangeable parts to be practically applied to nuts and bolts, which led to the Industrial Revolution, and the development of mass production.



**Left** ♦ The aluminium plates bolted to the concrete base seen here are used to centre the round pipe for the lathe's DIY linear rail

**Above** ♦ Curt Filipowski cast his own handwheel from scrap aluminium

HackSpace magazine meets...


# Adrian Bowyer

Guru of 3D printing, and non-expert of the future

**Y**ou might have heard of **RepRap**. It's the project that began at the University of Bath in 2005 with the aim of creating a self-replicating, open-source 3D printer.

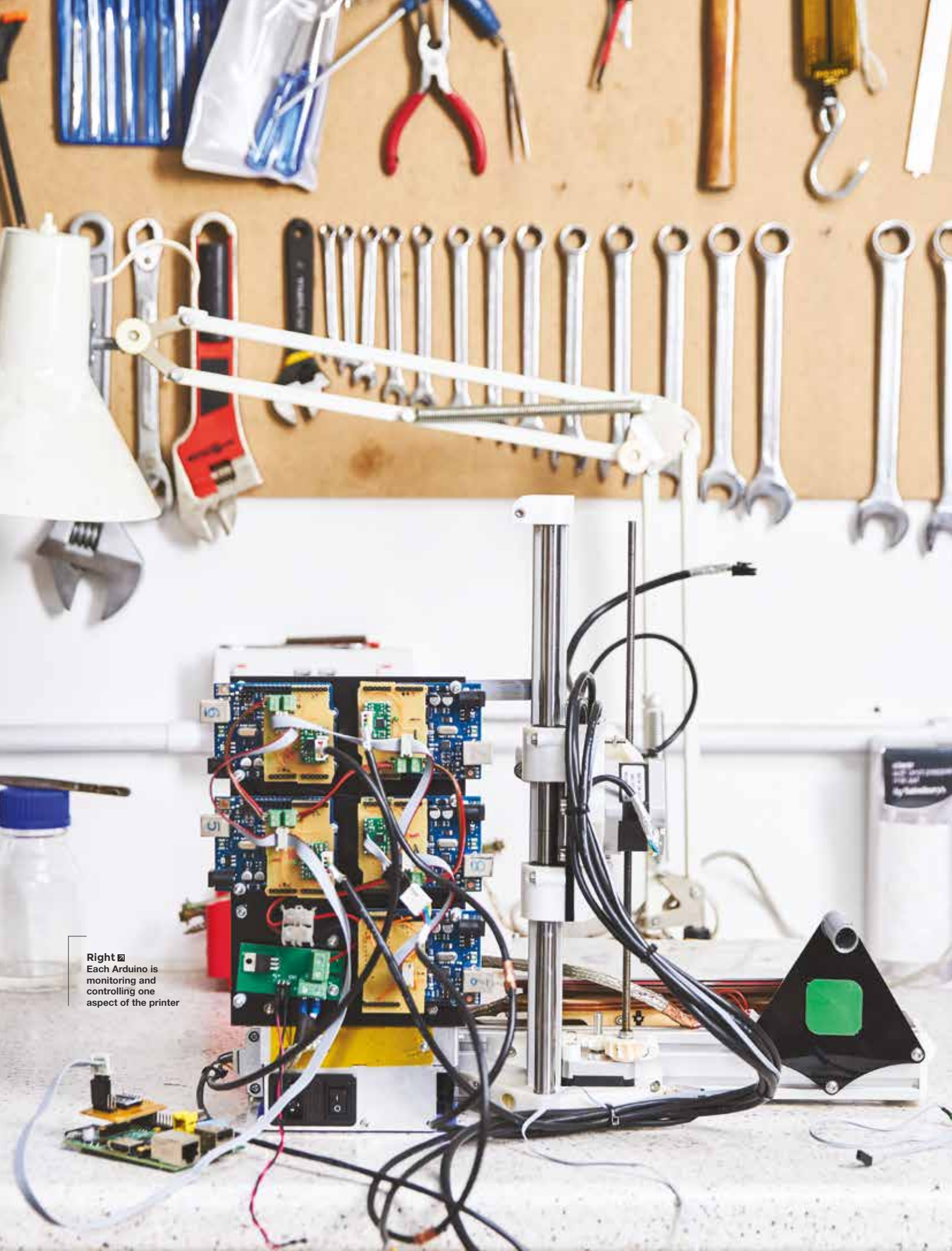
As is the nature of open source, many other projects have spun off from RepRap, including the Prusa i3. Without it, the field of 3D printing would be much smaller, less advanced, and a lot less open.

We drove many miles through wind and rain to meet Dr Adrian Bowyer, co-founder of the RepRap project who now, along with his daughter Sally, runs RepRap Ltd. The two of them are still pushing boundaries, raising standards, and lowering prices, so we sat down to talk about RepRap, where the 3D printing industry is heading, and an unlikely way to combat climate change. →

**Right**   
Adrian was made an MBE in the New Year Honours list, for services to 3D printing

**Images**  
Adam Gasson





Right  Each Arduino is monitoring and controlling one aspect of the printer

**HackSpace** It may be an obvious question, but why did you start the RepRap project?

**Adrian Bowyer** Curiosity. I have always been interested in the idea of self-replicating machines, ever since I was a child. When my university acquired some commercial 3D printers, as soon as they arrived I thought, ah, we've got a technology here that is sufficiently versatile that it stands a chance of being able to copy itself. Having had that idea, the very next question that occurs to your brain is: will this work? And that was the genesis of the project. I wanted to find out if we could make a machine that could print a significant fraction of its own parts and self-replicate.

It was literally the case that, at the height of development of RepRap in Bath 2008/2009, I was effectively running in terms of numbers of staff, the biggest research project in any UK university. I wasn't paying any of them of course, and they were distributed all over the world, but if you counted them up, there were more of them working with me than were working in any other single research project in any other university in the UK.

**HS** That's the dream isn't it, for an academic? To have a team all around the world working for nothing, and constantly giving you feedback and data?

**AB** Yes! And constantly enhancing my reputation with their work! It was perfect really. The way that academics are gauged on their performance is the number of papers they publish. And, of course, we got quite a few papers out of RepRap. Quite a few of those volunteers all over the world were authors on the papers – now, we couldn't have 100 authors on the papers – but the people who mainly contributed did, which was interesting because, for the first time I've ever seen in a journal, there were, you know, the author was John Smith or Sue Jones, and the affiliation was 3 Railway Cuttings, Crewe, rather than a university. It was a whole load of private

individuals, and the journals didn't know how to handle these as authors. I'd never seen that before.

That's really how science used to be. If you imagine Charles Darwin, in his study in Down House, outside Orpington, a century and a half ago writing letters to the Royal Society, or whatever, who got published, he was just – well, wasn't 'just' anything, he was one of the most important humans who've ever lived – but in the context of academia, he was just a bloke sitting in his study, writing letters.

**HS** What are you doing with RepRap at the moment?

**AB** We're developing little robots that are 3D-printed for use in schools; we've got one that uses the same technology in its

I wanted to find out if we could make a machine that could print a significant fraction of its own parts and self-replicate

sensors that self-driving cars do. And so the idea there is that you have a little robot, 200 mm in diameter, that you can program using a web interface, and which you can cause to drive around your classroom, using the same sorts of algorithms that self-driving cars use, without the obvious danger of pupils running each other over in real cars.

We're looking at distributed processor RepRaps, so instead of having a single CPU, we put a single CPU on each device in the machine, such as the heaters, the motors, and so on. This isn't a new idea; other people have tried this in the past. From the perspective of Raspberry Pi, that's interesting because such a machine wouldn't need real-time response from the processor that's at the heart of the machine.

If you've got a Linux system running on something, it's not great for real-time control, because of interrupts. Whereas the sort of system we're working on would have a Raspberry Pi in the middle, with a load of Arduinos around it. You can hand over the hardware timing to the Arduino, which, being dedicated, can be guaranteed to generate a poll every 20 microseconds or whatever it is. Whereas the thing sitting in the middle, doing the control, just has to be able to respond every few milliseconds. That's something we're putting together with Raspberry Pis and Arduinos.

One of the reasons that we want to do it is that we're looking at making larger machines, and also a machine that, not only is a 3D printer, but also incorporates a plasma cutter. Now, the thing about a plasma cutter is that it generates an enormous amount of electronic noise. You get lots of interference from it. So the ideal way to send electrical signals around the machine is not using electricity, but optically. So what we would be doing would be setting up a machine with optical communication between each of its component parts and the controller, so that electrical interference isn't a problem and, in order to do that, it has to be distributed in the way that I've just described.

We're developing a multilateral drive for machines, which should be retrofittable. If you've got a 3D printer that works with a single material, this would be a bolt-on that would allow you to put, say, four different colours through the same print head. That is the print head that the machine originally came with, and all it would require from the controlling computer in the existing machine would be that controlling computer driving one plastic filament drive, which it would do already, because [the first one] is one, plus the ability to send I<sup>2</sup>C signals to that device that we're talking about. That's something we've currently got in development.

**HS** And is this all open source, like the original RepRap project? →

**AB** Everything we do is open source. We have a discounting system, whereby if a company approaches us and wants some work done, we'll give them a discount if they're prepared to open source it, otherwise, we charge the full rate. It seems fair to us, and it works.

We're developing a scanner for three-dimensional objects. Again, this is nothing new in the way of technology – there are plenty of scanners available, including some on phones where you take a few pictures of an object and it will reconstruct it. We have a cunning plan to allow us to increase the accuracy of it using statistical techniques. What you've got is basically a machine in which you can put a three-dimensional object, and you can scan that object and it will produce a representation of it that you can then print. If you just let it run, the sort of accuracy you get is plus or minus about half a millimetre to a millimetre.

Using the same sorts of techniques that astronomers use to improve astronomical images – a thing called star stacking, where you take a series of images of the same piece of sky, and you use statistical analysis to remove the noise – we think we can use that effectively in three dimensions to remove noise from a 3D scan of an object. The calculations say we should be able to get down to an accuracy of 100 microns. Calculations are one thing – the experiment is key. You do the experiment and believe the calculations when you've done the experiment, or not, depending on what comes out.

We try to have so many projects on the go at once that we don't have time for them all. The important thing when you get stuck on a problem... sure, you shouldn't give up, you should think about it for ten minutes, but what you shouldn't do is think about it for two hours. That's a complete waste of time. You think hard, and if you cannot solve it, you go and do something else. It's nice to have a stack of something-elses to go and do when you get stuck on one of them.

**HS** Where, in general, do you think 3D printing is heading?

**AB** The analogy I often draw is with washing clothes, which went through three stages: it started off with us washing our own clothes. In the scullery or the kitchen, you'd wash your clothes once a week. And then in Victorian times, as economies of scale kicked in, there would be a town laundry, where you would send your clothes and they'd come back clean. But now we have a robot in the kitchen that can wash our clothes. It's come back to us, this time automated.

Making stuff in general, it seems to me, is going through that progression, just 100

“ We try to have so many projects on the go at once that we don't have time for them all ”

years later. It started off that, if you needed a gate hinge, you went to the blacksmith in your village. He would make you a gate hinge. Now if you want a gate hinge, you go to the shop and buy one, and it was made halfway around the world. But if we bring some of that back into our cities, it's like bringing our washing back from the town laundry into our homes. As long as it's automated – the rule seems to be that if something is automatable, such that people don't have to pay a lot of attention, and it's low-cost enough, people can take it back to themselves, and economies of scale get reversed.

**HS** The difference being that the town laundry is in Shenzhen nowadays.

**AB** Nobody is sending their laundry intercontinentally to be washed! It never has made economic sense for your shirt to go on a 747, and come back two weeks later. All these things depend, to a certain

extent, on the economics, obviously, and much more fundamentally the physics of what is being done.

For example, look at metal fabrication; at the moment the vast majority of 3D printers work with plastic, because it melts at 150°C. There are commercial machines that will work with metal, but there are comparatively few of those that work in the home, because the temperatures required are so much higher – it's more difficult in terms of the physics, and also less safe.

Physics constrains things, economics constrains things. But the world doesn't always head in one direction, which is the Adam Smith-type direction of everybody specialising, and economies of scale

mean that we can make a gate hinge cheaper if we make a million of them 100 miles away.


**HS** That's where I think there's a moral imperative to making closer to home – even if every village has a 3D printer, never mind every home, then there are so many fewer molecules of CO<sub>2</sub> in the atmosphere from all the ship tons that are saved.

**AB** Well, yes-ish. For example, 3D printing uses more energy than injection moulding. You save carbon in transport, and that's significant. But you spend more carbon in energy, because you've got a process that is not quite so energy-efficient.

The other aspect to that, of course, is what plastic do you use, assuming you're working in plastic? If you're injection-moulding Lego bricks – which are made out of ABS, that's an oil-derived plastic – you're not contributing carbon to the environment, because it's a solid object, except in the energy used to make the Lego brick. If you're 3D-printing things in polylactic acid, which is the most common 3D printing material, then that isn't an oil-derived plastic; all the carbon in polylactic acid came from a plant, which means that it came from the atmosphere. Now, the interesting calculation is that, when you're running a 3D printer with PLA, you're sending carbon up the smoke-stack in the power station in the next county, →





**Left**  This ukulele was printed in two parts. It's playable, and sounds great

but you're actually laying down more carbon out of the atmosphere than you're sending up out of the smoke-stack. It's about twice as much. In other words, assuming you're using conventional electricity generation, which is carbon-intensive (and fortunately, increasingly we are not), you're sending half as much carbon out of the power station smoke-stack as you're taking out of the atmosphere in the form of the solid object you're printing.

There's an argument that it may make economic sense and, more importantly, environmental sense to encourage people to make junk and throw it away. Every piece of junk they make is carbon out of the air. And when they throw it away, it goes into landfill and stays there. [NB – Adrian is not making this argument.]

Basically, human beings making red toy dogs for their children that get thrown away after six months and end up in landfill are actually taking carbon out of the air by doing that. So a throwaway culture is good for the environment.

**HS As long as they don't end up in the sea clogging up fish gills.**

**AB** Actually, the sea isn't a big deal for bio-derived plastic, for PLA. Polylactic acid in the sea is biodegradable, so it goes away unlike, for example, polythene. The problem with landfill is that polylactic acid, as I've just mentioned, is biodegradable, so it doesn't just sit in a landfill. It will compost to CO<sub>2</sub> and methane, so what I said is not necessarily true; just chucking it in a landfill is not a good idea. I'm not advocating this, I don't think it's a good idea for people to make things and throw them away. But the mathematics of the carbon in that process is interesting with 3D printing and plant-derived plastic.

**HS Do you think we'll reach the point where everyone has a 3D printer in their home any time soon?**

**AB** I suspect probably not. Pretty much everyone has a computer; not everyone

has a conventional printer for documents. Most people don't bother. I'm guessing about 20% of people who have a computer have a printer to go with it, and that would be the maximum level at which we'd see home usage of 3D printers in the long-term, for much the same reasons.

If our neighbour wants a document printed, they come round with a memory stick and get me to do it. It'll be the same with 3D printing. When laser printers were first introduced they were damned expensive, they cost three or four thousand pounds each. And so, there were printing shops. Already there were conventional printers in every town. They acquired a laser printer, and then people would go to the printing shop to get things printed. But that was a stage – those shops are not so prevalent now, because you can get a really high-quality inkjet printer for £100. That idea of a village blacksmith or a town 3D printing hub is a transition stage, but not an end point, I don't think.

Having said all that, there's a really important piece of psychological work that should give us pause here: it's been well established that if you ask experts in virtually any field what the future of their field will be like, they are no better at answering that question than someone plucked randomly off the street. This is easy to study of course; you just ask a bunch of random people and experts, wait ten years, and you've got the answer to your study. This has been done, and it's been discovered that experts know very little about the future of their own topic. The one exception is people who have to do predictions for a living. People like weather forecasters, anaesthetists, for example. They really can tell what's going to happen next when they do something. In other words, people who get constant immediate feedback on their predictions get good at it, but people who are just experts in a subject know no more about its future than anybody else. That includes me.



**Left** This RepRap model, Ormerod 2, is the basic model on which the prototype Arduino-controlled printer is built

**HS** We've seen some brilliant uses of 3D printing over the last year – resin and 3D printing for example. Have you branched out into other fields or is RepRap exclusively hot plastic?

**AB** The type of 3D printer that RepRap is... the trademarked term is an FDM printer, but the open-source term is FFF – fused filament fabrication printer. The vast majority of 3D printers in the world use that technology because of RepRap, if I may say so. But there are half a dozen other technologies which are much, much rarer and more specialised.

There's a brilliant new one from UC Berkeley, Lawrence Livermore and NC Chapel Hill, which is basically a reverse CT scan. If you imagine yourself in a CT scanner and they run an X-ray source around you in a loop, and they have an X-ray receiver on the other side, and they can build a cross-section of you in that way with standard CT-scanning technology.

Now imagine the same thing with a light projector that projects a beam of light, the intensity of which is what you would need to reconstruct the X-ray. And then you rotate the object in the beam of light (it's not just a beam of light, we'll come to that in a moment); it constructs a 3D object in one rotation. But it's not just a single beam of light: it's actually more like a data projector. So it's projecting an entire three-dimensional image, the light pattern that you need to create an object. You have a cylindrical vat of resin that sets solid, once the amount of light falling on it integrated over time is above a certain point. As the cylinder of resin rotates, all the bits you wanted solid are the only bits that have had enough light going through them to turn solid. Which is a straightforward calculation to do: it's actually the reverse of the mathematical transform they use in the CT scanner. The only other trick is that the cylinder they use is useless for projecting light through because it acts like a lens. So,

they put the cylinder in a rectangular bath of liquid with the same refractive index, so it's got a flat face on it and so you can project through a flat face and it comes out spot on.

That's the cleverest idea in 3D printing I've seen in five years. Really nice. A beautiful, beautiful piece of work.

**HS** I was up at Sheffield University talking to a PhD researcher last month. He was looking at ways to monitor the heat patterns within large 3D-printed metal structures, to predict failures and make the product more consistent.

**AB** Forget about 3D printing completely: suppose you're casting a block of metal in a mould. Real Victorian traditional casting, with a sand mould. And suppose you put a load of cast iron into this sand mould and there's a big lumpy bit and a thin bit. As you would imagine, the thin bit sets before the big lumpy bit, and what happens is you get differential contraction as a consequence. And so, first of all, it has internal stresses. And secondly, it will distort, even though the sand mould was the correct shape.

Have you ever noticed that cast wheels with spokes always have an odd number of spokes?

That's the reason. If you cast the wheel with spokes opposite each other they set up a force opposing each other. Whereas if you've got five spokes, or seven, or three, then there are no two opposites, and you end up with much lower residual stresses in the wheel. The interesting thing about that is that, if you look at loads of modern wheels that are not made by that process at all, that's retained the odd number tradition. And nobody can remember why. 3D printing, metal casting – a lot of the physics is the same.

**HS** Finally, congratulations on your MBE!

**AB** That's very kind! The certificate is an impressive thing. Signed by Her Majesty the Queen and by Prince Philip, as the person who is in charge of knighthoods and such.

I'm going up in May to Buckingham Palace to have it pinned on my chest, so that'll be interesting. The commendation says: "Inventor: for services to 3D printing." Short and to the point. □



**Right** □

The clock was printed on the first RepRap printer – the MDF board is that printer's build base



# SPONGE



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.

## Scrub the blueprints for your builds with this versatile soaker

**Learning tools are a modern-day necessity.** They come in all shapes and sizes, but perhaps the most inconspicuous, yet indispensable, is the kitchen sponge. This cleanser can be traced back to the hygiene-conscious ancient Greeks, and has an interesting evolution that dates back to the beginning of life on the planet. The earliest Olympic participants bathed with handfuls of specially treated aquatic animals, known as sea sponges! The use of multicellular marine life as a cleaning tool was very popular through the Middle Ages, and has been documented in ancient Roman text as well. Sea sponges were thought to have therapeutic properties, which is why they were used by the physicians of the day to clean and treat injuries and wounds. Recent research by MIT has confirmed that sea sponges were, most likely, the first animals on Earth.

The sponge didn't enter the kitchen until the 1940s. Until then, everything from sand, soda, and pieces of rag were used to wash utensils. Sometime between the late 1930s and early 1940s, German chemist Dr. Otto Bayer managed to synthesise polyurethane foam in his lab. Following his lead, in their bid to improve polyurethane foam for commercial viability, some German scientists ended up with a defective batch that was full of air

bubbles. While analysing the failed experiment, the scientists took a closer look at the spongy material and soon realised its potential, primarily because of its resemblance to the sea sponge.

It wasn't long before these polyurethane sponges were put to use as cleaning agents. But, like most discoveries, the earliest polyurethane sponges were very fragile and disintegrated easily when used in the kitchen for washing dishes. Also, while they produced good foam, they weren't suitable for heavily stained utensils, and would roll up and crumble quickly. The durability of the modern-day sponge was a result of refinement to its manufacturing process.

These days, you can find various types of sponges at the supermarket. Hand-sized cellulose sponges are the most common; inexpensive and providing good absorbency, they are suitable for all kinds of chores. Some cellulose sponges have an abrasive material on one side to help scrub patios, grills, and ovens. On the other end of the spectrum are nylon sponges that are meant for use on surfaces that scratch easily, like glass, porcelain, and kitchen counter-tops. They aren't as absorbent as cellulose sponges, but they can hold a good amount of liquid soap for effective cleaning. And, just as you can find sponges for all kinds of surfaces, our spirited makers have squeezed them for all kinds of hacks.

# SPONGE BOT

**P**rolific maker and Instructables.com's Senior Community Manager, Randy Sarafan loves making robots from easily accessible knick-knacks. He's built the Sponge Bot using a couple of rulers as the robot's frame, a jar lid as its crank, and a paintbrush as the connecting rod. The robot was initially conceptualised as a rope-climbing robot: "I'd like to say there was some sort of grand inspiration for selecting sponges, but really it was just trial and error. Of all the things I have lying around, those

**" A SINGLE ROTATING MOTOR CREATES A COMPLEX SCISSOR-LIKE BACK AND FORTH MOTION "**

seemed to work the best. As I often tell my sceptical wife, building robots isn't really an exact science," Randy opines. When he couldn't get the rope-climbing robot to climb, he tossed the lamp cord locking mechanism he'd designed to help it climb, and replaced it with a swivel ball caster instead, and began attaching different things to the rulers. In the end, he ended up with a robot that uses a single rotating motor to create a complex scissor-like back and forth motion. As with all his Instructables, Randy has posted detailed build instructions that illustrate each and every step of the construction. □



#### Project Maker

**RANDY SARAFAN**

Project Link  
[hsmag.cc/eFAIbc](https://hsmag.cc/eFAIbc)

#### Above

If you haven't built a robot yet, take Randy's free online Robot Class ([hsmag.cc/oaznBO](https://hsmag.cc/oaznBO)) to get started

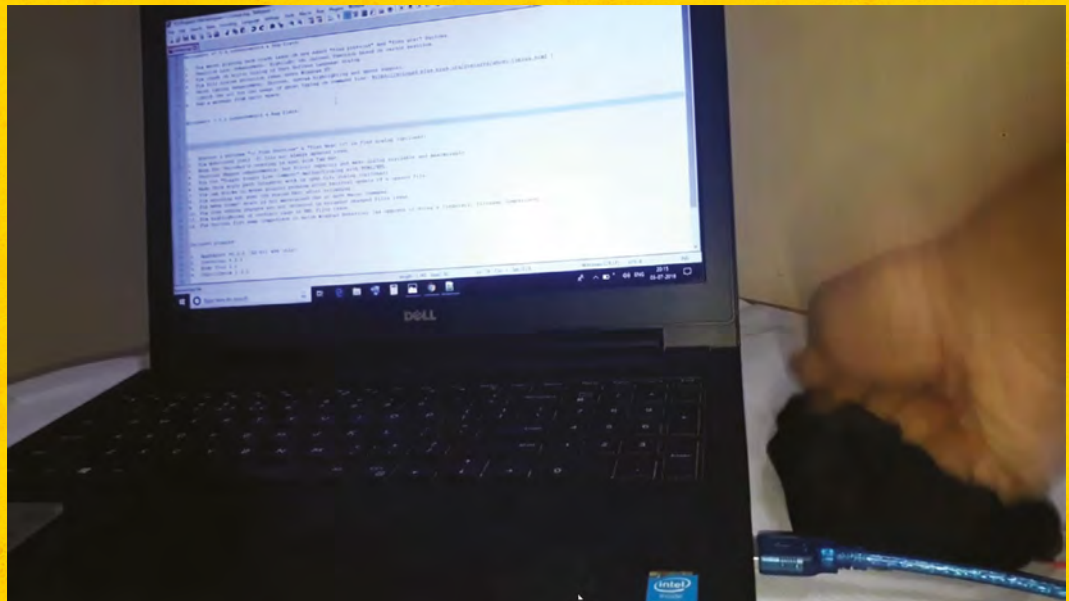
# PUNCHABLE KEYBOARD BUTTON

Project Maker

**AMAL  
MATHEW**

Project Link

[hsmag.cc/epJCNj](http://hsmag.cc/epJCNj)



**Right** ♦

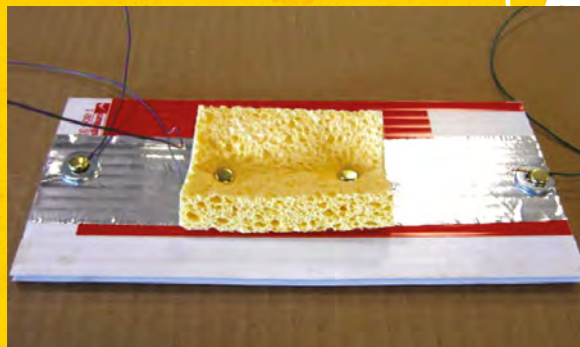
A Facebook video on how to deal with stress at work, which featured a punchable keyboard, inspired Amal to hack one of his own

**I** f you've worked with computers for a long time, there have probably been several times when you've felt like punching the keyboard in frustration.

Amal has put a sponge's elasticity to good use to help you vent your emotions without any monetary loss. He's used a piezo element, a high-voltage 1 megohm resistor, and an Arduino-based ATmega32U4 board, with a built-in USB that helps identify it as a keyboard. When the piezo element records the voltage generated by a punch, it's read through the analogue pin of the microcontroller,

which translates it into a key press. "I used two sponges and placed the circuit in-between it," shares Amal, whose mother helped stitch a cloth enclosure for the venting device. Follow his detailed instructions to build and code the circuit. Just enter the ASCII value of the key you want pressed when you punch the keyboard, and you're all set. Besides being an outlet for your emotions, since the punchable button is completely customisable, Amal suggests you can use the device as a short cut for entering a complex password, or for some other action you perform regularly. □

# SPONGE MUSIC



**K** eith is the Director of Learning Technologies Center, at the Science Museum of Minnesota. A Scratch veteran, who has tinkered with the block-based visual programming language for over a decade, he came up with the Sponge Music project as a means to demonstrate variable resistance using Scratch. Here, he used a PicoBoard that has a bunch of sensors, such as a light sensor and a

sound sensor, and can interface with Scratch. The Instructables page has an illustrated guide to help you build the music-oozing sponge sensor. You'll also need to download Scratch, and import Keith's Scratch project ([hsmag.cc/CsQMxt](http://hsmag.cc/CsQMxt)). Depending on how much force you exert to squish the sponge, the contraption will pass electricity from one connector to the other that turns it into a number, thanks to the PicoBoard, which then plays the associated note as defined in Scratch. □

#### Project Maker

**KEITH  
BRAAFLADT**

Project Link  
[hsmag.cc/ySWoGd](http://hsmag.cc/ySWoGd)

**Above** ♦ Although we like to minimise the use of plastics, you'll need it for this build, since the sheet will constantly be wet, due to the sponges

# SPROUTS ON A SPONGE



#### Project Maker

**TAMMY  
DUBE**

Project Link  
[hsmag.cc/luEpjR](http://hsmag.cc/luEpjR)

**Left** ♦ If you think growing a sponge garden is an interesting idea, head to Tammy's website for all kinds of kid-friendly projects and experiments

**T** ammy was looking for a child-friendly activity to mark St Patrick's Day, when she came across an article on how to grow sprouts on a sponge. So, she set about cutting some pieces of green sponge into the shape of a shamrock. The cut sponges were then soaked in water to make them damp. She then sprinkled lettuce, spinach, and broccoli seeds onto the sponges, and pressed them into the holes in the sponge. The planted sponges were then placed on a windowsill, and her kids used a water sprayer to keep them moist. "We found it helpful to turn a clear plastic container over the plate at night, to keep the moisture in," writes Tammy. They had themselves a sprout garden in a week, and the kids were amazed by the fact that the seeds didn't require soil to sprout. □

# HackSpace

TECHNOLOGY IN YOUR HANDS

## Download the app

Out now for smartphones & tablets



**£2.29**

rolling subscription

or

**£26.99**

subscribe for a year





# FORGE

HACK | MAKE | BUILD | CREATE

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

PG  
90

## MAKING VINEGAR

Ferment your own delicious apple-based condiment



PG  
94

## POLYPHONIC SYNTH

Wire up your own music-making machine



PG  
100

## GAMES CONTROLLER

Turn your hoodie into the ultimate games-playing device

PG  
74

## SCHOOL OF MAKING

Start your journey to craftsmanship with these essential skills

- 74 Integrated Circuits
- 80 PCB Layout
- 84 Arduino Debugging

PG  
106



## E-INK DISPLAY

Keep tabs on your smart home without blowing the batteries

# Electronics 101.9: Integrated Circuits

Making things smaller



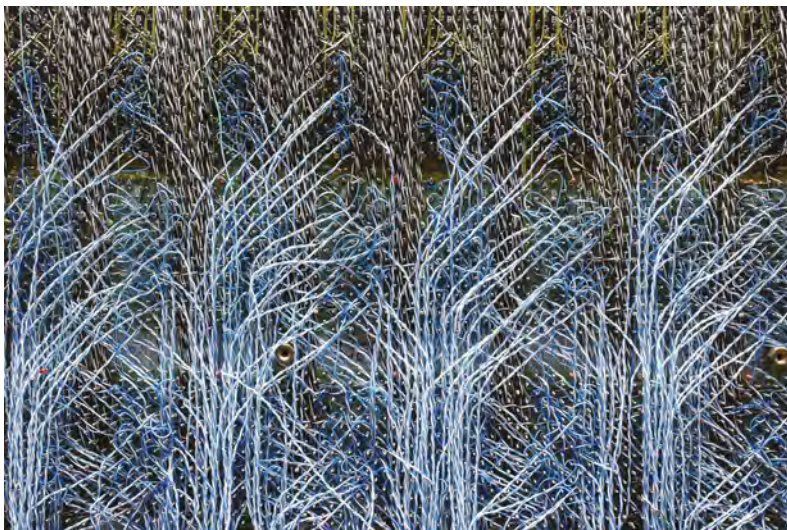
Dave Astels

[daveastels.com](http://daveastels.com)

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

**Figure 2** ♦  
A wire-wrapped backplane that connects many modules

**Credit**  
Dave Fischer CCA-SA 3.0

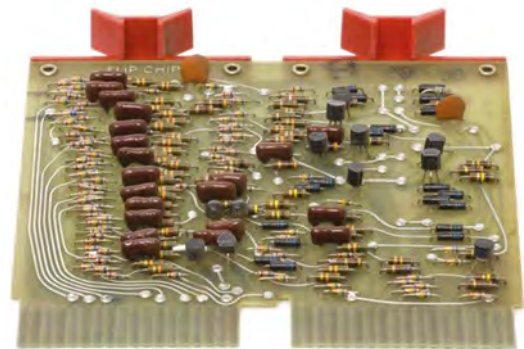


**W**e've seen several kinds of components so far. All of them are fairly big. Even if we consider the smallest surface mount resistors, capacitors, and so on, they're still quite large.

This wasn't a 'big' problem back in the day, when circuits were fairly simple. As we started designing and building more complex circuits, they got physically larger, and connections got more complex. At the same time, demand for electronic devices grew. Building devices was time-consuming and error prone. **Figure 1** shows a simple logic circuit built from discrete components. Many modules must be connected, typically by using a wire-wrapped backplane – see **Figure 2**. There had to be a better way. There was: the integrated circuit [IC].

## THE CHIPS ARE DOWN

Jack Kilby, at Texas Instruments (TI), developed a way of putting an entire circuit on a single chip in the autumn of 1958. The firm announced it in March 1959. However, Robert Noyce (then at Fairchild)



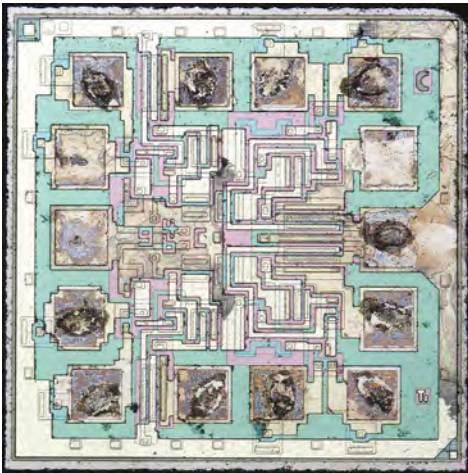
**Figure 1** ♦  
A transistor/diode-based logic module

had previously sketched out ideas for doing this (which shows the value of keeping an engineering notebook). Kilby's approach used actual wires to connect components on the chip, while Noyce came up with the idea of depositing metal traces onto the chip to connect things, much like the traces on a printed circuit board. Two years later, Fairchild released its first commercial ICs, which were heavily used by the Apollo spacecraft guidance computer. TI sued Fairchild, since it had patented first. However, Fairchild's idea actually worked. Before long, TI was building chips using Fairchild's process.

One of the first uses of integrated circuits was to take logic circuitry, like that shown in **Figure 1**, and put it on a single piece of silicon. **Figure 3** shows a single two-input NAND gate, which is one quarter of the classic 7400 IC. That takes a total of 16 transistors, four diodes, and 16 resistors, which would take up close to 16 cm<sup>2</sup> of circuit board, and puts it on a single piece of silicon that's only several square millimetres. This is shown in **Figure 4**. In order to be useful, this needs to be mounted in a package with leads that can be connected to it. **Figure 5** shows an example.

Logic ICs made several things possible:

1. High-quality mass production of circuits (although that took a while to get right).



**Figure 4** ♦ The 54HC00 quad-NAND gate die

**Credit**  
Robert.Baruch CC-BY

2. Along with mass production came lower cost.
3. Taking those discrete components, and placing them on a single chip, drastically reduced the size of circuitry. Replacing vacuum tubes with transistors let a room-sized computer fit inside a half dozen or so refrigerators. Moving to ICs let the same computer fit in a bar-fridge. With each advance, power requirements were reduced as well, which meant less heat was generated. That means less space required for power circuitry and cooling equipment.

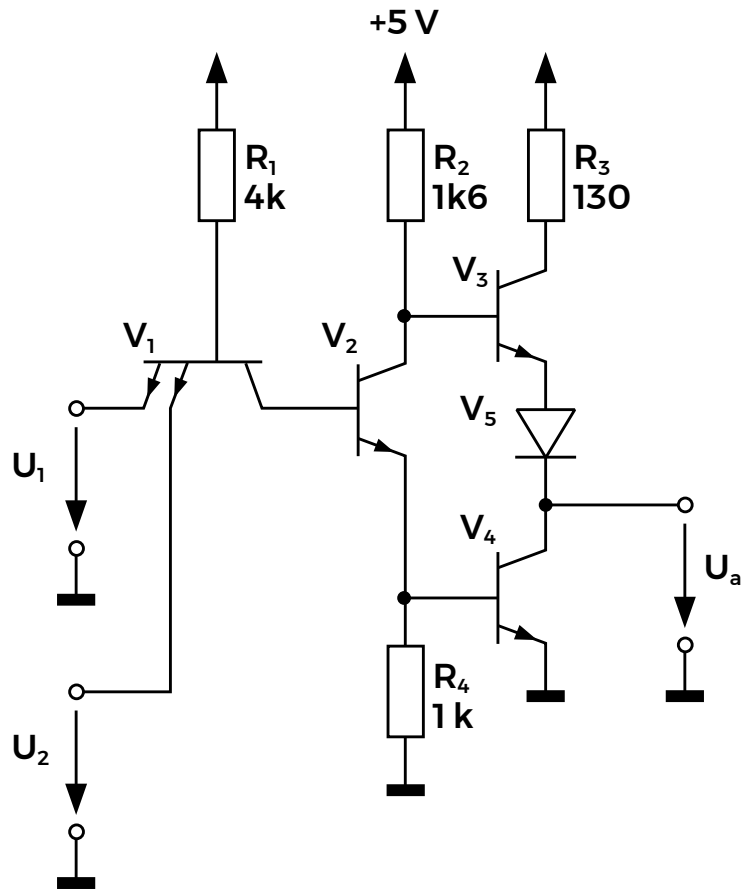
Of course, things didn't stop there. We figured out how to make the circuitry on silicon chips smaller, fitting more in the same space. At the same time, we figured out how to reliably make those chips larger.

**// We figured out how to make the circuitry on silicon chips smaller //**

As progress was made on each front, the power per square millimetre grew. Eventually, the entire CPU of a computer could be fitted onto a single chip: the microprocessor. That changed everything.

**SILICON BRAINS**

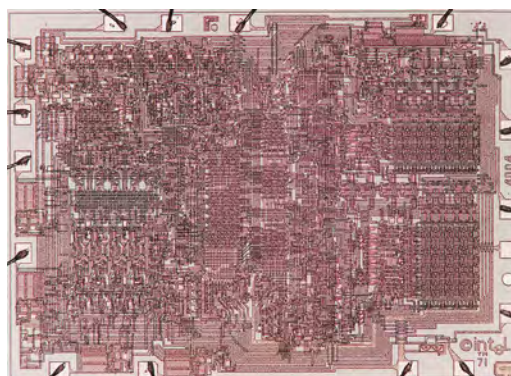
The first microprocessor was the 4-bit 4004, released by Intel in 1971. It was designed for making calculators. Having reprogrammable circuits meant not having to design new circuitry for each →



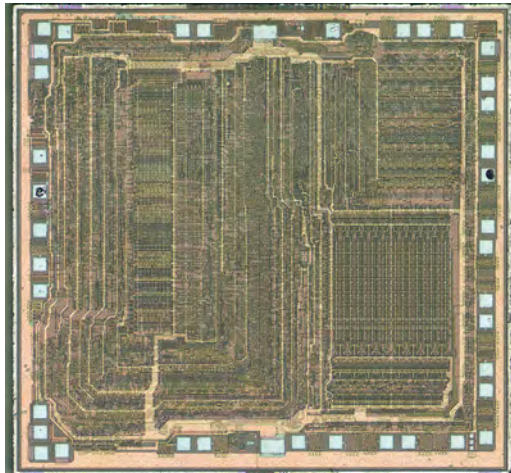
**Figure 3** ♦ The circuit for a single two-input NAND gate

**Figure 5** ♦ A chip mounted in a package, and connected to external leads

**Figure 6** ▣ The 4004 die




## SCHOOL OF MAKING



**Figure 7**   
The Z80 die

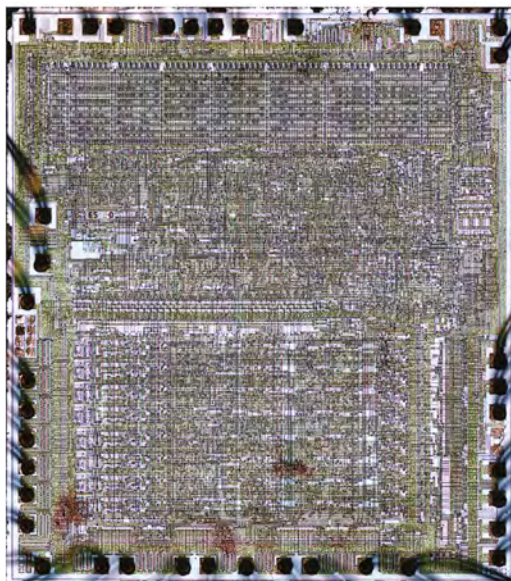
**Credit**  
Zeptobars CC-BY 3.0

**Figure 8**   
The 6502 die

**Figure 9**   
Comparison of 40- and 64-pin DIP packages

**Figure 10**   
The 68000 die

**Credit**  
Pauli Rautakorpi CC-BY-3.0



new product, you could simply write a new program instead. The 4004 die is shown in **Figure 6**, and contains 2250 transistors in 12mm<sup>2</sup>.

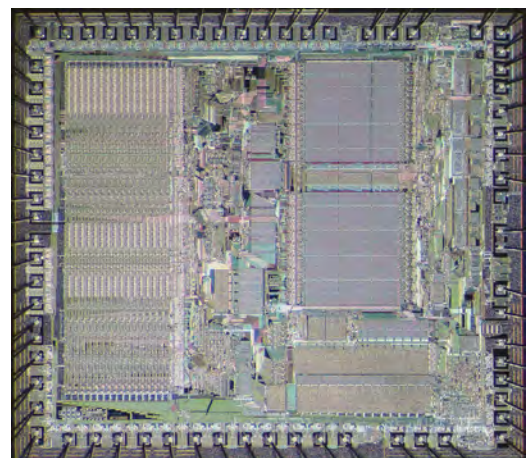
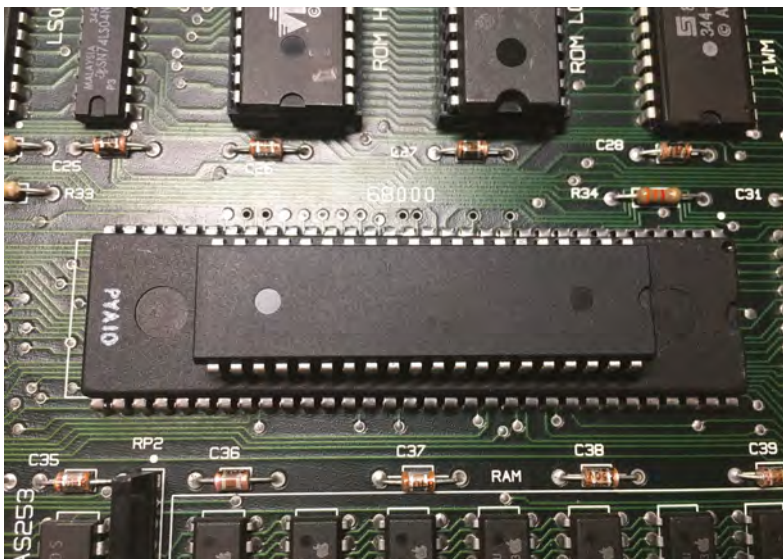
Chips kept growing in density and capability. Eventually, we had several families of 8-bit microprocessors. Intel's own 8080 became very popular. It contained 6000 transistors on a 20mm<sup>2</sup> die. Another alumnus of Fairchild and Intel (where he led the work on the 4004 and 8080), Federico Faggin started Zilog and developed the Z80, which went on to be incredibly popular. It was used in the Sinclair ZX80, ZX81, and Spectrum computers. The Z80 die is shown in **Figure 7**. It had 8500 transistors in a die that was 18mm<sup>2</sup>.

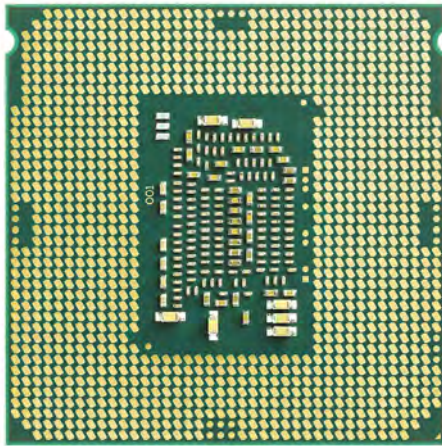
Another extremely popular microprocessor was the 6502, shown in **Figure 8**. With 3510 transistors on a 21mm<sup>2</sup> die, it was simple and spacious, in comparison to some. As a result, it was relatively easy to work with, and low in price. It was used in the Apple II line, Commodore's early computers (most famously, the C64), and Acorn's BBC Micro.

Eventually, 8-bit processors gave way to 16-bit ones. An early example is the 68000, famously used in the pre-PowerPC Macintosh models, the Commodore Amiga, and Atari ST line. A version of the 68000, the 68008, was used in the Sinclair QL. This was a much bigger chip: a 64-pin DIP package instead of the 8-bit processors' 40-pin (see **Figure 9**). The die is shown in **Figure 10**. The chip had 68,000 transistors on a 44mm<sup>2</sup> die.

### MODERN DESIGNS

As processors got more complex, moving to 32 and 64 bits, they started bundling more peripheral functions on the chip itself. For example, processors meant for desktop and mobile applications started incorporating





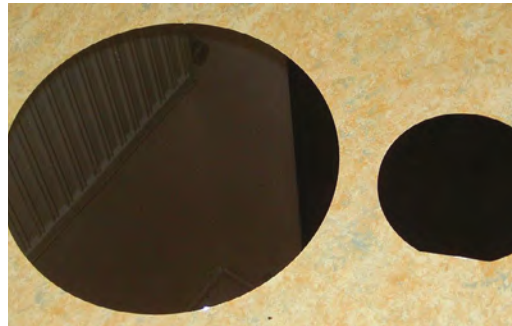
**Figure 11** ♦  
The bottom of an Intel Core i7 Skylake processor

**Credit**  
Eric Gaba CC-BY 3.0



**Figure 12** ♦  
A SAMD51 microcontroller

graphics hardware. Floating point and cryptographic hardware became standard. Processors for embedded applications bundle memory and assorted input/output components (I/O ports, serial ports, and so on). All this meant that more external connections were required. The DIP package was no longer adequate. At the same time, printed circuit board production and assembly techniques advanced. Package variety blossomed: square chips with pins on all four sides, grids of pins under the chips (**Figure 11** shows the underside of



an Intel Core i7). Surface-mount technology became standard as well, increasing board density even more as the physical chip packages became smaller. For example, **Figure 12** shows an Atmel SAMD51: a 32-bit ARM CPU core, 1MB of flash memory (for code), 256kB of RAM (for data), and an abundance of interface hardware.

photo-mask is made, which has transparent areas and opaque areas. The disc is coated with a photo-resist, the mask is placed on the disc, and it is exposed to ultraviolet light. This cures the resist where light reaches it. The mask is removed, and the uncured resist is dissolved, leaving a pattern of cured resist on the disc while the rest of the surface is exposed. Depending on what is needed, a small amount of the exposed surface can be etched away chemically, to become N- or P-type silicon, or can have a metallic layer deposited. The cured resist is then removed, and the process continues for the next mark and layer. Insulating layers of silicon dioxide are deposited at times, as required.

**// An integrated circuit starts out as a cylinder of very pure silicon //**

an Intel Core i7). Surface-mount technology became standard as well, increasing board density even more as the physical chip packages became smaller. For example, **Figure 12** shows an Atmel SAMD51: a 32-bit ARM CPU core, 1MB of flash memory (for code), 256kB of RAM (for data), and an abundance of interface hardware.

**GRAINS OF SAND**

An integrated circuit starts out as a cylinder of very pure silicon (see **Figure 13**) which is sliced into thin (no more than a tenth of a millimetre thick) discs. One side of these (where the circuitry will be built) are then polished (**Figure 14**), and coated with an insulating layer of silicon dioxide.

Now, they are ready to have components added. This is done in a similar way to the photographic process of making printed circuit boards. First, a

A final layer of silicon dioxide is added to seal and protect the circuit. This has contact areas etched away, to which metal contact pads are added which are used to connect to external pins via very fine wires.

This layering process builds a three-dimensional circuit. **Figure 15** (overleaf) shows a visualisation of this structure, while **Figure 16** shows a cross-section of a chip.

**AN INTEGRATED CIRCUIT IS A THREE-DIMENSIONAL CIRCUIT**

Each slice of silicon contains hundreds of chips, shown in **Figure 17**. Once the above process is complete, the individual chips are tested, and →

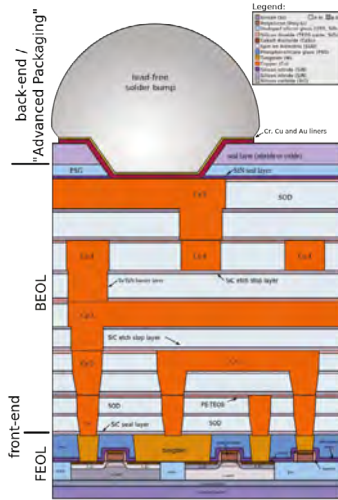
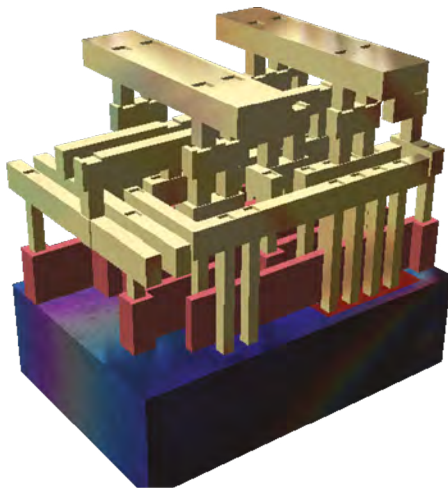


**Figure 13** ♦  
A pure silicon cylinder ready to be sliced

**Credit**  
Stahlkocher CC-BY 3.0

**Figure 14** ♦  
Polished wafers

SCHOOL OF MAKING



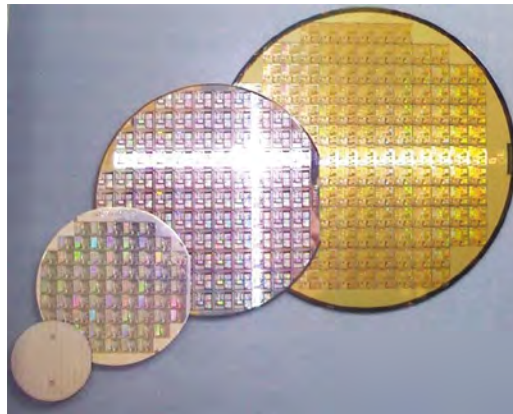
**Figure 15** A visualisation of the 3D structure of a portion of an integrated circuit. Silicon is removed for illustrative purposes

**Figure 16** Cross-section of an IC

**Credit**  
Cepheiden CC-BY 2.5

**Figure 17** Several wafers, ready to be separated into chips

**Credit**  
Guillem CC-BY-3.0



EXPLORING FURTHER

If you want to explore ICs further, check out the blog of Ken Shirriff who delights in exploring (and writing about) the inner workings of chips: [righto.com](http://righto.com).

A great documentary on the early semiconductor industry in the US, including the initial development of the integrated circuit and microprocessor, is the 'Silicon Valley' episode of the PBS series *American Experience*: [hsmag.cc/ouOUPG](http://hsmag.cc/ouOUPG).

*Micro Men* is a British docudrama, showing the early years of Sinclair and Acorn.

non-functional ones are noted, to be discarded later. A diamond cutter is used to score the disc between chips, which are then separated by, essentially, snapping them apart. The non-functional ones are discarded, as are any that were damaged by the separation process.

MORE POWER

In 1965, Gordon Moore (of Shockley Labs, Fairchild Semiconductor, and Intel) observed that the number of transistors in a dense, integrated circuit doubled about every year. In 1975, he revised this to a doubling every two years. See **Figure 18**.

Moore's law is reaching its limit. Moore, himself, commented in 2015 that it would only continue to about 2025. To pack more transistors in the same space, everything needs to be smaller. And therein lies the problem: how small can components be made and still function? As they get smaller and smaller, quantum-level effects can start creeping in and disrupting a transistor's operation. There are many research efforts underway to discover ways to make transistors smaller. Some include: a junctionless transistor, at Tyndall National Institute in Cork, Ireland; a single-electron transistor (that routes single electrons rather than electron flow) at the University of Pittsburgh; and a single-atom transistor from the University of New South Wales. Other research concerns itself with alternatives to silicon, quantum computing, and biological computing.

We've explored the basic ideas of electricity and electronics, basic electronic components, and now we've discussed taking those ideas and making them tiny, enabling entire circuits to become a single component. Next issue, we'll look at a very common and useful type of integrated circuit: the operational amplifier. It's time to get the breadboard out again!

**The number of transistors in a dense IC doubled about every year**



**Figure 18** You can explore how chip density has changed over time at [hsmag.cc/SCCGGA](http://hsmag.cc/SCCGGA)

# THE OFFICIAL Raspberry Pi Beginner's Guide

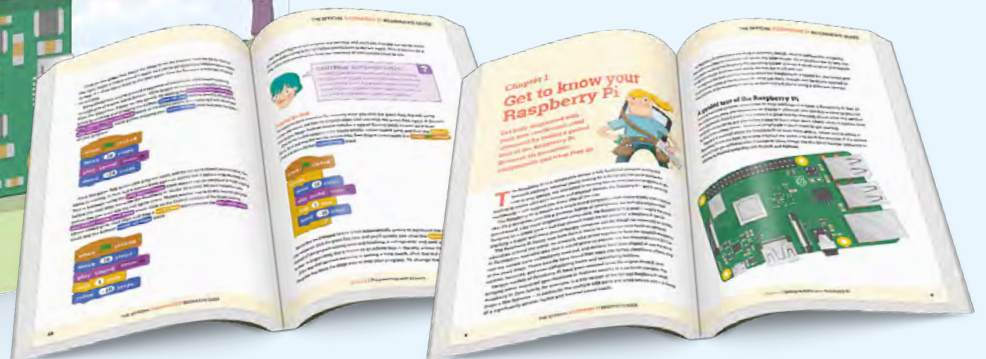
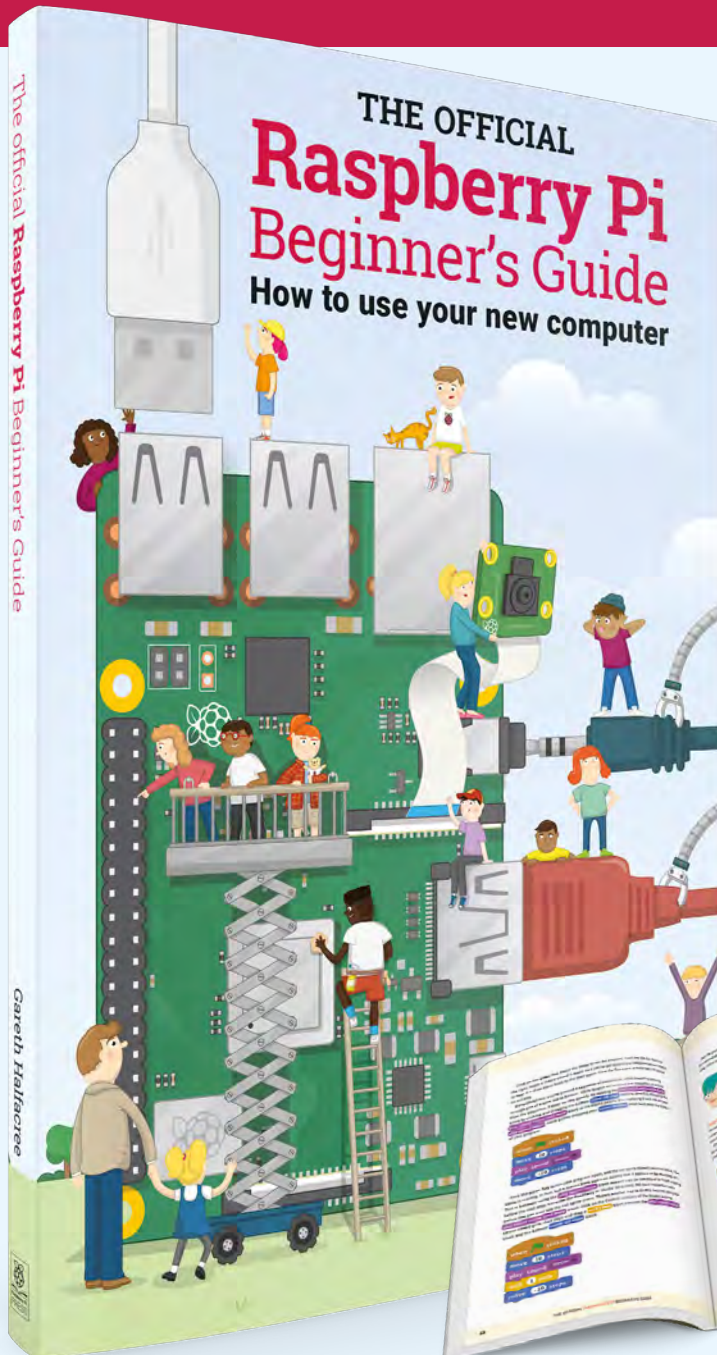
The only guide you  
need to get started  
with Raspberry Pi

## Inside:

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

**Plus much, much more!**

**£10 with FREE  
worldwide delivery**



Buy online: [hsmag.cc/BGbook](http://hsmag.cc/BGbook)

# Make a PCB in KiCad: Layout

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



Jo Hinchliffe

@concreted0g

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

## YOU'LL NEED

A computer with KiCad installed

## Right

The end result we are aiming for: a small simple PCB laid out well and ready to be fabricated

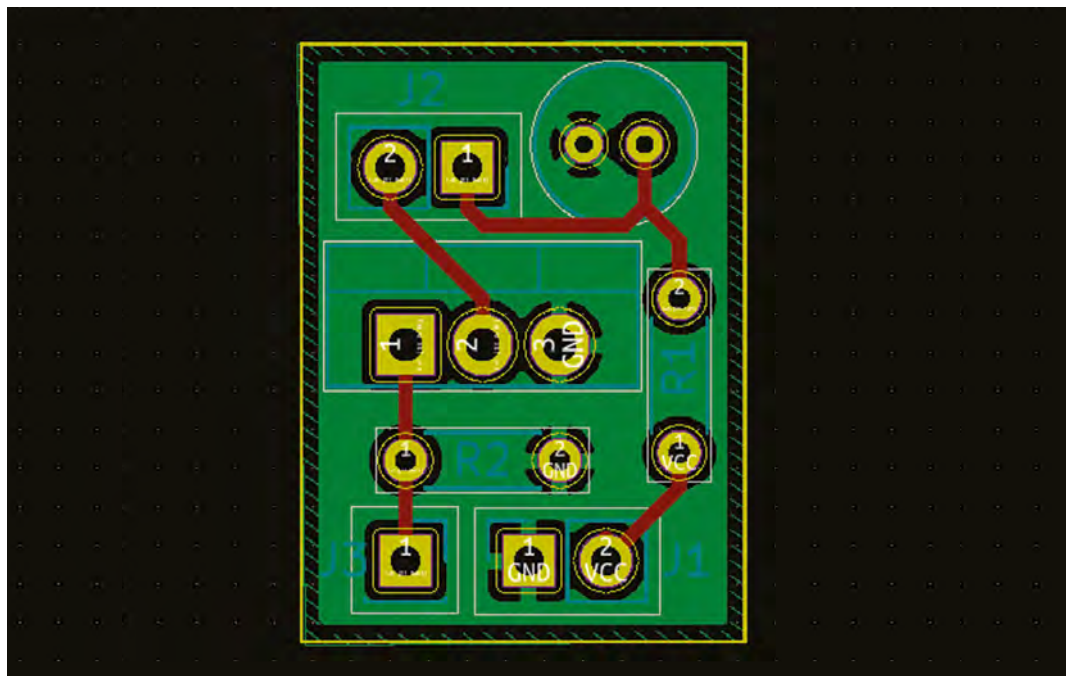
**T**he ability to lay out simple PCBs and get them manufactured is a fabulous skill to have as a maker, and modern software tools, like the free open-source KiCad, make this evermore within reach of the average maker.

In this two-part series, we will walk through creating a simple PCB using KiCad, to the point where it is ready to be sent for fabrication. This first part deals with creating a schematic representation of the circuit, and the second part will deal with the laying out of the actual PCB design. The PCB made here is designed to control the ignition of an electronic match but the process is similar for all small projects, and we'll go through the full life cycle of a KiCad project.

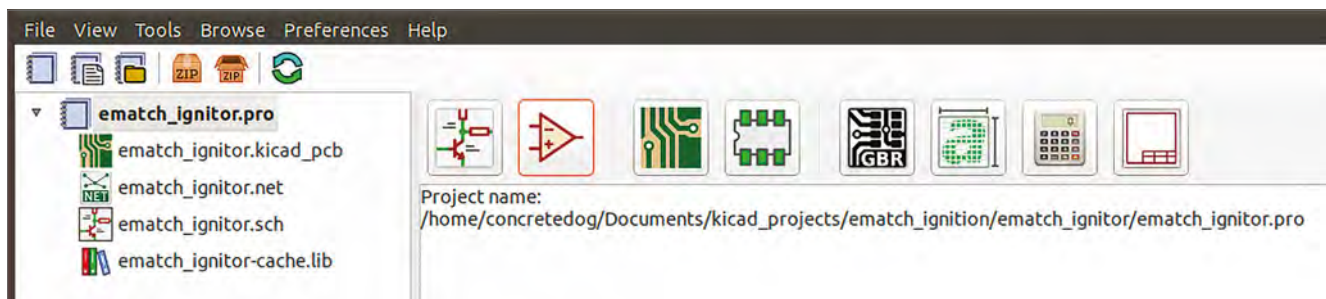
So to begin, download and install KiCad onto your machine and launch it. There are instructions for installing on this page: [hsmag.cc/VbCROd](http://hsmag.cc/VbCROd). Once installed, you will see, as in **Figure 1**, that KiCad is actually a collection of applications that work together to create a full suite of electronics design tools. For now, click File > New and create a new project in which we are going to work.

## GETTING INTO THE FLOW

The workflow of KiCad may seem initially complex, but actually makes a lot of sense. Broadly speaking, we are going to do the following over the two articles: first we create a schematic that represents our circuit in Eeschema, annotate the schematic so each component has a separate 'address', and run







**Figure 1** ♦  
The main project page of KiCad with the various program icons across the top, and the various files we will produce listed down the sidebar

a program that allows us to select which physical package a component on the schematic will use on the PCB layout. After choosing these ‘footprints’, we will then create a list of these called a netlist, which contains the info about the components’ footprints and also the circuit’s connections. Finally, we will import the footprints into the Pcbnew tool, and lay out the connecting tracks and other things that a PCB uses.

Firstly, we are going to launch Eeschema, which is the leftmost of the eight main icons.

In Eeschema, we are going to create a schematic of the circuit we are making. In **Figure 2** (overleaf) you can see the completed circuit we are aiming to reproduce. On the blank schematic page you will see, down the right-hand side, a toolbar with icons for tools we are going to use. Click the fourth icon down which, when you hover over it, should say ‘place power port’. Next, click anywhere on the empty schematic and a dialog box should appear where you will choose what power port you are going to place. There are numerous ways to search and choose, but simply type ‘gnd’ for ground into the search input box. We should see various ‘GND’ options appear underneath. Click the top item that appears: ‘GND power flag, ground’. You should have a ground symbol attached to your cursor which you can place anywhere by left-clicking. If you place it

incorrectly or want to move it later, hover over the object and then press **M** to move the item. Use the ‘place power port’ tool again, but this time let’s place a positive voltage port or ‘VCC’; use the letters VCC to search for a positive port symbol. As the circuit design is optimised for 7.4 volts, add a 7.4V VCC

// **In Eeschema we are going to create a schematic of the circuit we are making** //

power port from the search results. Finally, note that these items just indicate that there is power and ground in the board, but they don’t actually create a physical connector to connect to; those will be added later.

Next, we are going to add the rest of the components: the resistors, the capacitor, and the IRF540N MOSFET. To add these, use the third icon down on the right, which should read ‘place symbol’ when you hover over it. Similar to the power port tool, when you then click in the schematic you get a search dialog box to define what component symbol you wish to place. For the resistors, type **R** and select the first item in the list. Then click to place a resistor symbol. As we need to place two, we can click again and, instead of searching again, the place symbol tool handily remembers the last item you placed so you can simply press **ENTER** to place another. Having just placed the resistors, let’s hover over each of them in turn and press **E**; in the dialog box, we’ll add a field value to one of 4.7r and 10k to the other so we can distinguish them correctly.

Next, let’s add the capacitor and the MOSFET. For the capacitor, search using the letter ‘C’. Typing in the name IRF540N for the MOSFET brings up the correct symbol to be able to place. This circuit requires one input from a GPIO on a microcontroller that controls the MOSFET. Also, it requires a physical place to connect 7.4V and GND and two more connections to the electronic match the circuit is designed to drive. The simplest way to add these →

## PACKAGE SIZES

As you will know, some electronic components come in different physical packages. For example, considering resistors – some may be through-hole types that can vary in length from 2–3mm to 3–5cm, they may be mounted vertically or horizontally, and there are also surface mount resistors which again come in a range of sizes. Many PCB environments ask you to make all those choices about a component when you begin to lay out your schematic. KiCad differs in that it asks you to do this separately. The beauty of this is it enables schematics to be created quickly and if, down the line, you discover a component is not available in a certain package, you only have to change the footprint, rather than redoing the schematic.

## QUICK TIP

Adding field values to component symbols as early as possible can help you keep track of what is what.

### QUICK TIP

Hit Save if you haven't already done so.

is to add symbols for connectors – for the single GPIO input we add a symbol called 'Conn\_01x01\_male', which you can find by searching for 'conn' in the place symbol dialog box. Similarly, the other connectors are both 'Conn\_01x02\_Female'; however, it's useful to edit one of them to say 'ematch connector' for its field value for ease of identification, as can be seen in **Figure 2**. The other could be named as 'VCC and Ground' for example.

### LET'S GET CONNECTED

Now we have all the component symbols in place, it's time to make the connections. To do this, we'll use the fourth icon down on the right-hand side which, when you hover over, should say 'place wire'. To place a wire, start on the connector of a symbol and left-click. As you move the cursor, you should now see a green wire forming. To make a turn without completing the wire, perform another single left-click. Then, to finish a wire (either connecting it to another item or to create a junction on another wire), double-click. A connected wire should either run seamlessly into a component symbol or terminate in a green circle where it creates a junction. Anything else indicates you aren't connected properly. Note in the schematic that we have a point where wires cross without connecting, where the wire from pin 3 of the MOSFET crosses

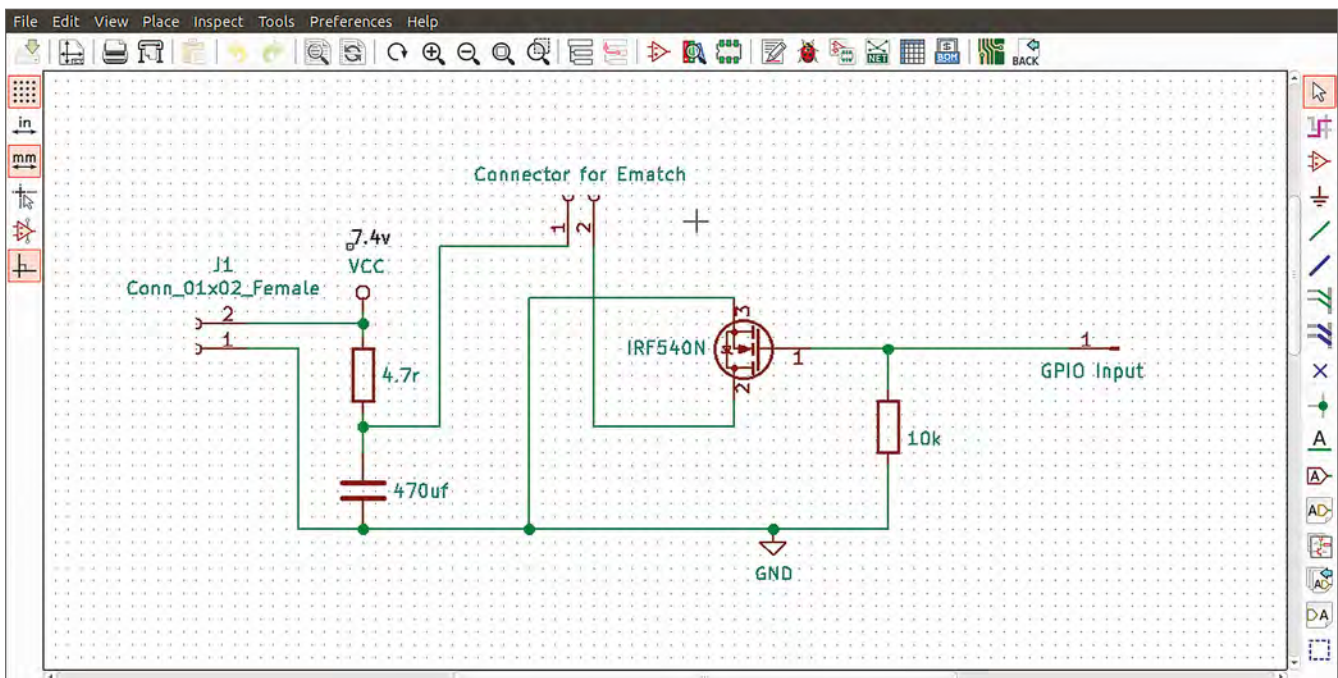
the output connector wire to the electronic match. This is correct, but take care not to form junctions or connections when you mean for things to cross.

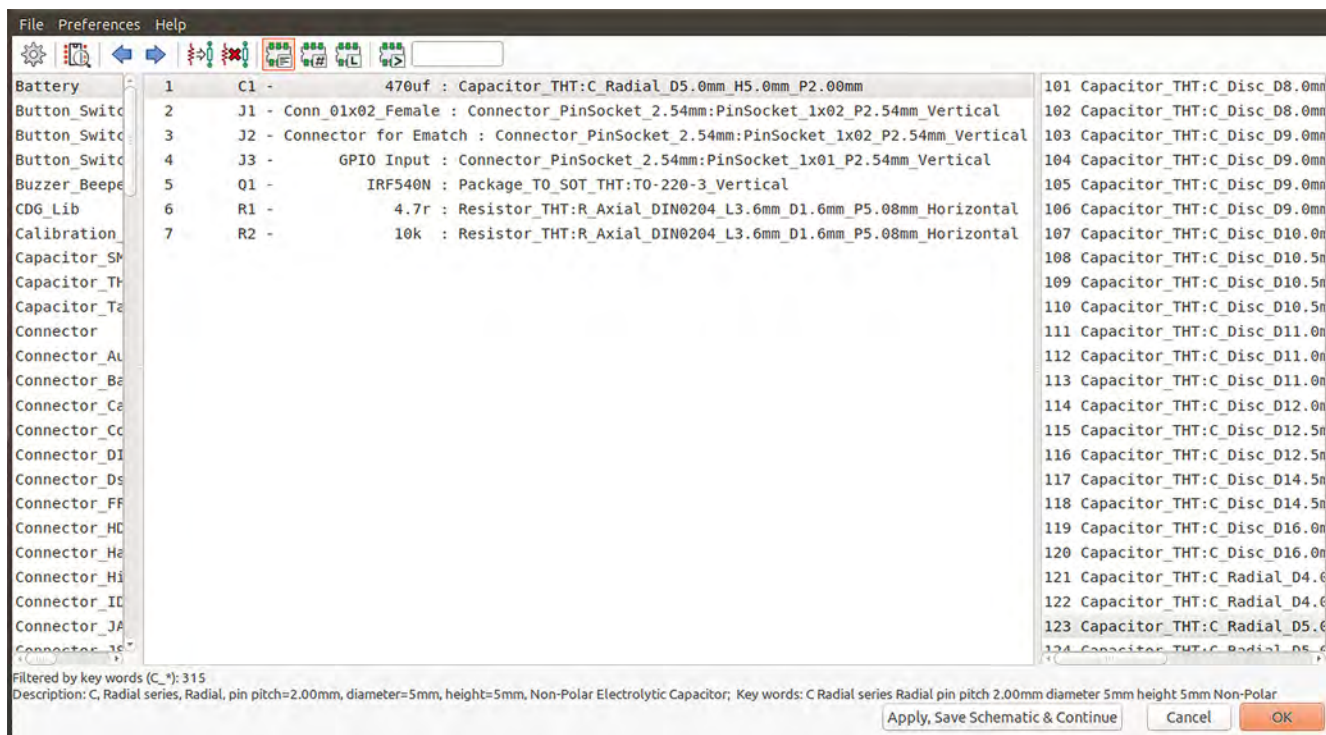
Continue with the schematic, connecting everything up to match the schematic as shown in **Figure 2**. When complete, the next thing we need to do is annotate the schematic so that each component has an individual identifier. We don't need to do this manually – KiCad does

**// A connected wire should either run seamlessly into a component symbol or terminate in a green circle**

this automatically. Click the 'annotate schematic symbols' icon, which is eighth from the right along the top toolbar and looks like a paper and pencil. A dialog box will appear; click 'Annotate'. On the first run of this in a new schematic, you will get a warning that it 'cannot be undone' – this is fine and in fact, although you cannot undo the annotation, we can re-annotate at any time if, for example,

**Figure 2** ◆ Eeschema, with the fully finished schematic for reference





you add any extra symbols. Having applied the annotation, all components should have an identifying reference number (for example, we have R1 and R2 in **Figure 2**).

### A GAME OF FOOTIE

The next process is to assign what physical footprint we want each of these components to have when we lay them out as a PCB. So click 'Assign PCB footprints to schematic symbols', which is the sixth icon from the right on the top toolbar.

You will get a dialog box with three columns, the centre one of which should contain a list of the components in your schematic that require a footprint. On the top toolbar you get some different options as to how to filter the footprints, and hovering over the icons will show descriptions. Highlight the 'Filter footprint list by schematic symbol keywords' icon. Now, click on each item in the centre column and the right-hand column should populate with results constrained to those that are useful for the component. We've selected hand-solderable, through-hole type component footprints, for this PCB. For each component symbol on the list, scroll through the lists and select the same component footprints as shown in the image above.

Once you have selected all your footprints, click the 'Apply, save schematic and continue' button. Then click OK.

The final task for this first part of the tutorial is to generate a netlist. A netlist is a collection of information about the circuit, which includes the footprints assigned to symbol components, but also contains the connectivity of the circuit. To generate one, simply click the 'Generate netlist' icon (fifth from the right on top toolbar) and then click 'Generate netlist'. KiCad will assign it the same name as the project title, and so there is no need to change anything, just generate the list.

Everything is now in place to create the PCB layout in the second part of this tutorial, next issue. Make sure you have saved all your hard work and we will get this finished next month! ■

**Above** ■ You need to link your schematic items with the correct physical components

## KEYBOARD SHORTCUTS

In KiCad you quickly realise there are some really useful keyboard shortcuts, and the following ones are all available in both the Eeschema and Pcbnew parts of KiCad:

- F1** zooms in centred around the cursor
- F2** zooms out around the cursor

Hover over a symbol/footprint and;

- Press **E** to bring up edit dialog
- Press **R** to rotate the object
- Press **M** to move the object

## QUICK TIP

You can use the 'View footprint' icon (second icon from the left) to see a plan of the footprint you are considering using at any time you have it selected.

# Arduino programming: Debugging

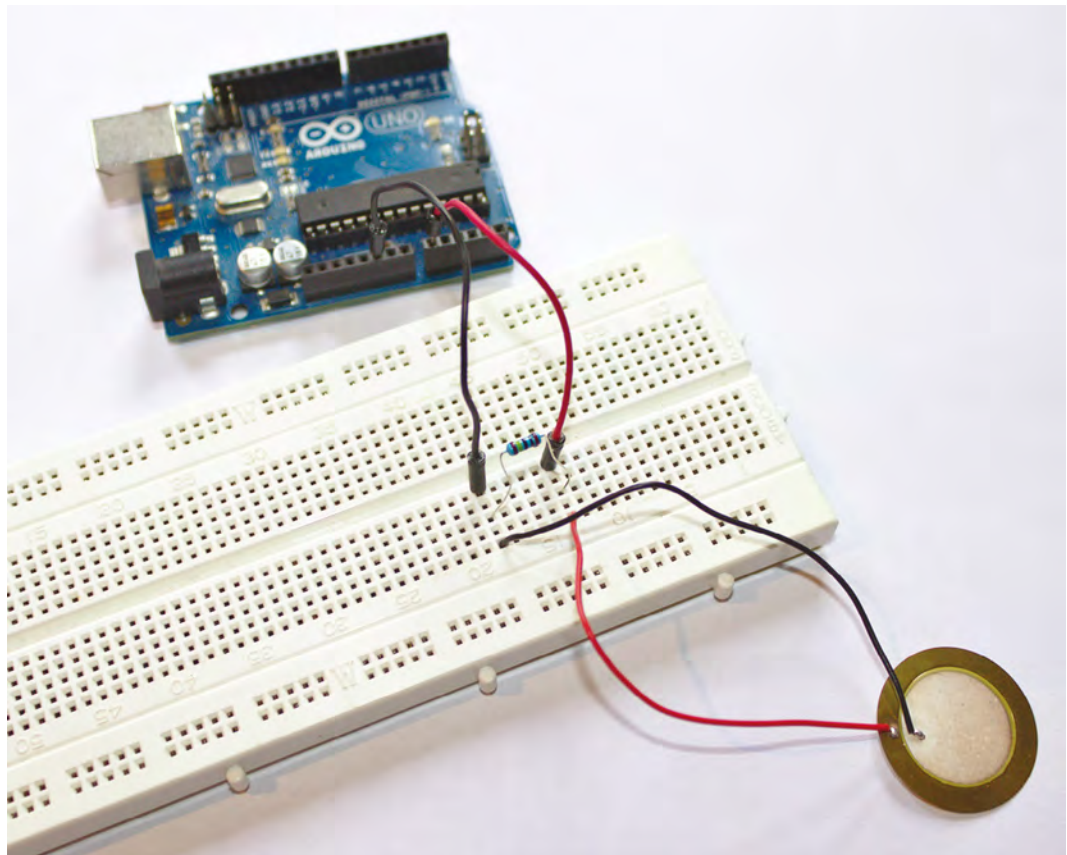
Delve into the dark art of troubleshooting and work out where things are going wrong



Graham Morrison

[@degville](#)

Graham is a veteran Linux journalist who is on a life-long quest to find music in the perfect arrangement of silicon.



**One aspect of Arduino programming we've barely touched on is the careful, cautious, and necessary art of debugging.** Debugging is a

very general term that covers a huge variety of processes that range from simply trying to find out why your code doesn't work, or why it's producing unexpected output, through to performance monitoring, profiling, and optimisation. The majority of modern development environments

and frameworks will offer tools to help with these debugging processes, commonly allowing you to step through your code line-for-line, while monitoring the state of your hardware (as with a debugger such as 'gdb'), or generating profile information from the code execution, such as the length of time spent in a function, or the amount of memory being used. But we don't have the same level of luxury on an Arduino.

With the Arduino, there's no graphical tool for monitoring the memory, no performance profiling

**Above**

Creating a circuit with a piezo sensor couldn't be simpler. Wire up positive to an analogue pin, negative to ground, and bridge the two with a 1 MΩ resistor

or graphical debugger. In this way, debugging your Arduino project can feel very similar to debugging a 1980s-era home computer project, because you need to come up with your own tests, and write your own code directly into your projects. This isn't necessarily a bad thing because you're learning about your code and learning how best to avoid mistakes through trial and error. But there's a lot you can do to make the process easier, and a lot you can do to make your code faster – both of which we're going to tackle by using the serial monitor and some lovely piezo sensors.

### SERIAL SCAFFOLDING

In writing any Arduino code, there's one element that's always required and yet nearly always cut prior to publication or release. This is the code used to debug the program, and it can be a little like the scaffolding around a building construction site. It performs an essential role that you seldom see

// Debugging your Arduino project can feel very similar to debugging a 1980s-era home computer

after the project is completed. You seldom (never!) get code working on the first write, and you often need to go back through what you've written and test your expectations against what is actually happening. The difficulty comes in trying to find out what is happening. This is actually very close to how professional development works, because you often have to write tests at the same time that measure those expectations against what can be shown to be happening. Those tests are then run whenever the code is updated, to make sure nothing added changes the behaviour of the older code; it's a process known as QA – quality assurance.

Programming for the Arduino presents several unique challenges. The biggest to overcome is that your code isn't running on the same system, or the same architecture, that you're writing your code on. An Arduino is really just a microcontroller. This is why there aren't any readily available native debugging tools, as these usually need to run and interpret the compiled output of your code on the system it's been built for. Instead, you only execute live Arduino code when it's been uploaded onto your device, and apart from a flashing LED, there's no →

## BOOST SPEED WITH REGISTERS

The Arduino platform has been designed to be as broadly compatible as possible. This enables it to work across many different kinds of devices and in many different kinds of environments. But this flexibility sometimes comes at the cost of performance, especially for specific devices. And one of the best examples of this is the `digitalWrite` function, used by nearly every Arduino project to send a signal to a pin. The Arduino documentation admits the code for `digitalWrite` is a dozen lines long, compiled into a multiple of machine-specific instructions, one of which is executed per 16MHz clock cycle. This takes time. But it's possible to do without `digitalWrite` completely, and instead write directly to the pin in question using what is known as a 'register'. And what's more, it can be done with a single command:

```
PORTD &= ~_BV(PD2);
```

A register is a special kind of storage that's tied to a specific hardware location, which is then read directly by the hardware when a certain function is performed. The chips on an Arduino have three different kinds of registers to cover all the analogue and digital pins, including PORTD for read/write access to digital pins 0–7, as shown above. The `&=` chicanery is because we're working at a low hardware level, and this is a bitwise AND assignment operator. This is followed by a bitwise NOT for the tilde (~), effectively allowing you to switch the state pin 2 (PD2 on the Uno) with the `_BV` macro for convenience. A longer way to write the same thing is the equivalent of `PORTD = PORTD & (~_BV(0b00000100))`. But it doesn't need to make sense for it to just work. In our experiments, the above code takes around two CPU cycles, whereas `digitalWrite` takes around 36, at least on our Uno.

**Below** ♦  
The best thing about using the serial monitor is that you don't need any extra hardware, such as a screen, to get meaningful information back from your Arduino





**Above** ♦  
Without feedback, it's very difficult to tell which values occur when on a sensor, so you can then generate functions

way for the device to communicate its running state, or whether it's encountered any problems, unless you specifically add that feedback into your code. What's more, while you can obviously create the code to send messages to attached LEDs, screen, and sound emitters, you can't then debug the output to those devices if even they don't work. The answer is to use the serial port.

**Below** ♦  
Most Arduinos have more than one serial connection, and this extra connection can be used to communicate with other hardware. The multiSerial example sketch shows how to do this

The 'S' in the USB umbilical cord we use to upload our code to the Arduino is for 'serial', and even modern USB is descended from this very early form of cross-device communication, where bits bounce from one hardware pin to another, one bit at a time. These pins were simply for 'transmit' and 'receive', and even on many modern devices, such as Amazon's Echo or your ISP's router, hackers can often locate TX and RX pads or pins on the

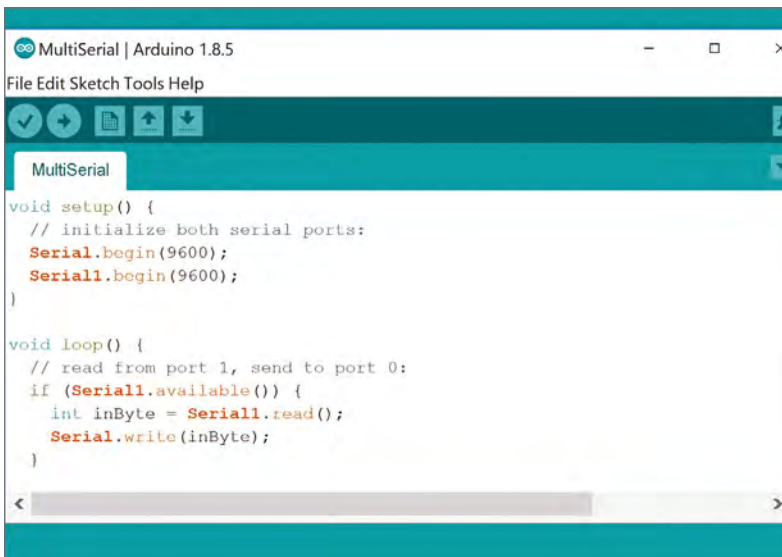
motherboard. The Raspberry Pi also has these two pins and makes a convenient testing platform for working or hacking with other boards. These connections are more widely known as UART (universal asynchronous receiver/transmitter) when the pins are used for a serial connection in this way, which is reflected in the Linux device name on the Raspberry Pi. But UART is also widely on the Arduino too, both manually via its pin connection, and via the USB connection to send data back from your code to a host system.

For a serial connection to work, both the sender and the receiver need to know how quickly the data is travelling. This was the baud rate in old modem terminology, and it corresponds directly to the number of binary bits being sent across a wire per second. To set the baud rate for the serial connection to the Arduino, add `Serial.begin(9600);` to the `setup` function. With that done, you can now send data from your code running on the Arduino back to the host computer using the `Serial.println`:

```
void setup() {  
  Serial.begin(9600);  
}  
  
void loop()  
{  
  Serial.println("Hello world.");  
  delay (500);  
}
```

**Both the sender and the receiver need to know how quickly the data is travelling**

115,200 bits per second is often the fastest serial speed you'll manage on Arduinos, and also with many devices using RX and TX pins. If you do experience problems, try slower speeds such as 57,600, 38,400, 19,200, or 9600. As the above code shows, we're starting at the slowest speed as this is always most likely to work. To test the above code, send it to your Arduino and open the 'Serial Monitor' from the IDE's 'Tools' window. This is the IDE's equivalent to those old pieces of terminal software that would help old computers connect to remote



bulletin board systems. The main window shows the output received from the connection, and the small 'Send' field lets you send data back across the serial connection. But before you can do that, you need to sync the monitor speed with the baud rate of the connection, which you can do with the drop-down 'baud' menu in the bottom right. If this is set wrong, you'll get a screen full of gibberish. When selected correctly, you should see a new 'Hello world' message every 500 milliseconds, or half a second.

### DEBUGGING

Of course, printing out a single message is no help at all. But you can now use the serial connection to troubleshoot all kinds of otherwise difficult to solve problems by using the same `Serial.println` command to indicate when your code reaches a specific section, or to see the value of a specific variable, or when a specific event has triggered a function. And to give these examples more solidity by showing `Serial.println` in action, we're going to create a specific example using a single component – a piezoelectric knock or vibration sensor. They're cheap and incredibly versatile and, as their name suggests, they can be used to create anything from motion detectors and door monitors, to drum pads and pressure gauges.

```

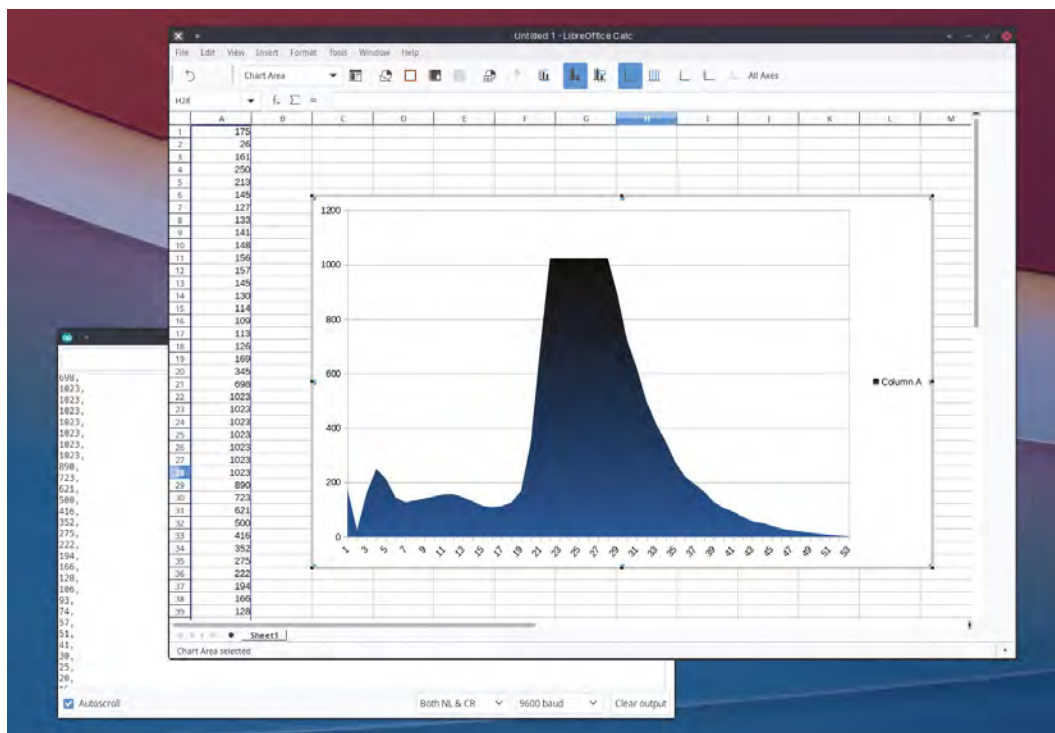
ReadASCIIString | Arduino 1.8.5
File Edit Sketch Tools Help
ReadASCIIString
/*
  Reading a serial ASCII-encoded string.

  This sketch demonstrates the Serial.parseInt() function.
  It looks for an ASCII string of comma-separated values.
  It parses them into ints, and uses those to fade an RGR LED.

  Circuit: Common-Cathode RGB LED wired like so:
  - red anode: digital pin 3
  - green anode: digital pin 5
  - blue anode: digital pin 6
  - cathode: GND
  */
  
```

A piezoelectric knock sensor is closely related to the 'piezo' we used to generate sound in a previous tutorial, as well as the 'piezo' used within electric guitar pickups and, ultimately, many microphones. They generate voltages from bending forces and changes in pressure. They can be easily wired up to your Arduino with the positive (red wire) connected to analogue input 1 and the negative (black wire) to ground, with a 1MΩ (megohm) resistor bridging the →

**Above** ♦ You can send data to your Arduino over serial as well. The example Communication > ReadASCIIString shows how to do this



**Left** ♦ Generating CSV data from your sensors is a brilliant way to conduct experiments and visualise the output, such as the response curve from a piezo knock sensor

**Right** 

Piezo sensors are cheap, easy to integrate, and incredibly flexible. It's always worth having a few around



two to dampen potential voltage from the sensor. But these sensors are also unpredictable, and you often have no idea of what kind of analogue values they're going to generate until you start generating them. This is important if you want to trigger something at a certain threshold, for example, or make sure that the threshold doesn't change under different conditions. And that means you need to get the data back from your Arduino across the serial connection.

For this simple example, add the following to the top of your code to set the analogue pin we're using and the integer we'll use to store the reading:

```
const int PIEZO = A0;  
  
int piezo_value = 0;
```

The main loop can then be updated with the following:

```
void loop()  
{  
  piezo_value = analogRead(PIEZO);  
  Serial.print("Current value: ");  
  Serial.println(piezo_value);  
  delay (500);  
}
```

There are two slight differences in the 'print' code above. The first is that we use **print**, as opposed to **println**, because we don't want a carriage return after the 'Current value: ' text, which is handled by the next **println** statement, although there are control codes that can do this within the text itself. But breaking this up into two lines makes it easy to see we're outputting the **piezo\_value** with the second line.

When you now upload this, run the code, and open up the serial monitor as we did before, you should see the following output:



```
Current value: 0
```

```
Current value: 0
```

Now try pressing down on the piezo sensor. You should see this value jump, although not always in a predictable way. The maximum value for the analogue/digital converter on the input is 1023,

**// You may wonder at which point you'd want the trigger to start, and this is a complicated problem**

and this can sometimes be achieved with a soft press rather than a hard strike, but as long as the untriggered value is 0, you can work with the sensor as a trigger.

If you wanted to use the piezo as a drum trigger, you may wonder at which point you'd want the trigger to start, and this is a complicated problem. You could use the transition from zero to a non-zero value, for example, but devices like these and momentary buttons will often include multiple transitions from zero in a single hit, and it's not always easy to tell when the main trigger should occur. This is a great example of when you might want to look deeper into the debugging aspect of your code by mapping out the typical values a sensor has during the course of an event, such as a trigger. We can do this easily with our own code by making only a few modifications:

```
void loop()
{
  piezo_value = analogRead(PIEZ0);
  if (piezo_value){
    Serial.print(piezo_value);
    Serial.println(", ");
  }
  delay (10);
}
```

The above code replaces the `loop` function with an `if` statement that's only triggered when the `piezo_value` isn't 0. It then prints out this value followed by a single comma, before waiting ten milliseconds and trying again. What this is actually doing is outputting a comma-separated list in the

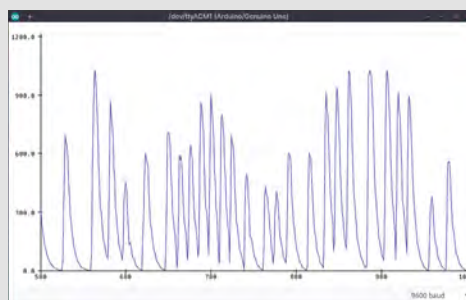
format typically known as CSV (comma-separated values). This is a very simple format that's supported by many different types of visualisation tools, both online and offline, and you can copy and paste those values directly from the serial monitor window into one of these – such as LibreOffice Calc – and, from there, generate a chart of the values. You can then analyse the chart to see what the typical sensor response might be, especially if you combine multiple triggers. You should then be able to derive a series of values that constitute a proper event without bounce or repetition, and you can only do this because of the debugging output from the serial monitor. □

## SERIAL PLOTTER

In the main text, we finish by outputting a CSV-formatted dataset that can be analysed from any one of the dozens of applications and web services that support CSV. But there's also a little-known feature in the Arduino IDE that lets you get real-time feedback from your sensors, without having to export your data at all. This feature is the 'Serial Plotter', found just beneath the 'Serial Monitor' in the Tools menu. It needs to be opened on its own, and it also requires the same baud rate setting as the monitor. But, most importantly, it requires a specific data format when sending values from your code. This is almost identical to the CSV format used in our original code, but replaces the comma with a single space. For example, our own code would look like the following:

```
if (piezo_value){
  Serial.print(piezo_value);
  Serial.println(" ");
}
```

With that small change and the code uploaded to your Arduino, you now simply have to open the plotter and start touching the piezo. You'll see the chart drawn almost in real-time in the plotter, which is a great way to both visualise sensors and the data they're generating, and create a model of how you might want to use specific value ranges within the data.



# Making vinegar

Acid makes everything taste better



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.

**V**inegar is one of those mystical household substances. It's there. We use loads of it. If you leave a bottle of wine open too long, it starts to taste like it, but where does it all come from? It's actually really easy to make high-quality vinegar, and it's far cheaper to make than to buy when compared to commercial raw vinegars.

The starting point for vinegar is ethanol – plain old alcohol that's found in beer (which makes malt vinegar), wine (which makes wine vinegar) and whiskey (which is sacrilege to turn into vinegar). You can make vinegar from alcohol you've bought, but here in the UK (as in many countries) the tax policy makes it expensive to buy alcohol to turn into vinegar, so let's go back a step. The raw ingredient for alcohol is sugar-water (or at least, some liquid with both sugar and water in it), and one of the easiest to work with is apple juice. Fermented all the way through, this produces apple cider vinegar.

Regular supermarket apple juice – the stuff that comes in one-litre cartons – works perfectly well for this. It's fine to use cartons of apple juice from concentrate, but make sure that it's only apple juice (a little added citric or ascorbic acid is fine) – some brands of 'apple juice drink' contain a whole host of other additives, including sugar and other fruit juices. These might be fine or they might end up tasting strange.

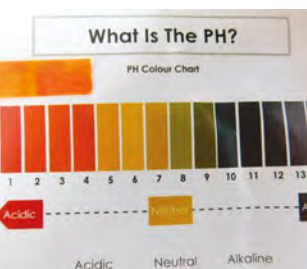
In principle, you can use any fruit juice for your vinegar making, but it will affect the end flavours for better or worse – it's up to you how experimental you feel. Most commercial fruit vinegars, such as raspberry, aren't fermented from these fruits, instead they're regular vinegars that are then combined with these fruits after fermenting.

Whatever juice you start with, the key thing you need to check is the amount of sugar in it. The amount of sugar will affect the amount of alcohol, and this will affect the amount of acetic acid in the vinegar. The amount of acid you want depends on what you want it for. Regular table



**Right** ♦  
A bottle of our finished vinegar, ready for drizzling on cod and chips

**Below** ♦  
You should end up with a pH around 3 or 4. However, it's better to be guided by taste than numbers on a strip



vinegar is usually about 5% acid, while pickling vinegars can be 10%. As a rule of thumb, you'll end up with 20% less acid than the amount of starting alcohol (it depends a bit on the particular microflora that gets to work in the ferment).

### SWEETEN THE DEAL

Approximately 20 grams of sugar per litre will equate to 1% alcohol. We wanted our ferment to end up around 8% acid, so we wanted about 10% ethanol, so we needed 200g of sugar per litre. Our apple juice had 114 grams of sugar per litre, so we needed to add an additional 85 grams per litre of ordinary caster sugar. All these calculations are subject to quite a bit of error, but they give us an indication of where to start.

The only additional things we need are yeast and yeast nutrient (in principle, you don't need yeast nutrient for cider, as the apples should contain all the nutrients the yeast need, but we add some to help ensure a clean ferment). We used sparkling wine yeast which is available as freeze-dried pellets in a sachet, but almost any brewing yeast should work.

That's the ingredients, now to the equipment. For the first ferment (sugar to ethanol), you need an oxygen-free environment. We used a glass five-litre glass demijohn (or carboy) fitted with a brewer's airlock, but anything that you can fit an airlock on should be fine.

**// We used sparkling wine yeast which is available as freeze-dried pellets in a sachet**

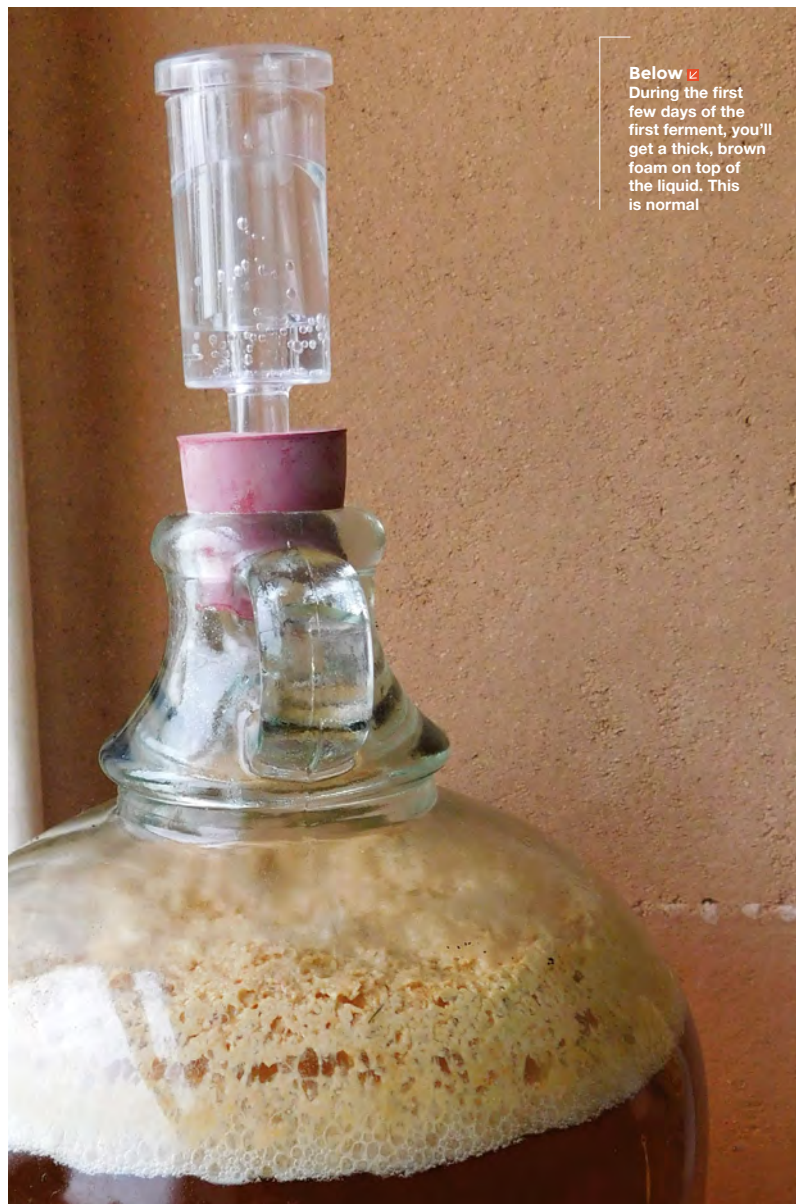
You need to make sure all this equipment is clean, but we don't sterilise our brewing equipment. Some authorities on home-brewing recommend it, but after several years of brewing without losing a batch, we're no longer convinced it's necessary, provided you're brewing to a high percentage and with an acidic liquid (as apple juice is, even before it's fermented). Beer brewers with the lower alcohol and acidic content may find sterility more important at this point.


We made four litres. First, add three litres of apple juice to the brewing container, then add all the sugar, put your hand over the demijohn opening and shake. This does two things – it helps dissolve the sugar, and it helps bring oxygen into the fruit juice. →

## HAVING A DRINK

We started by making cider that's quite similar to regular drinking cider, however, we haven't designed this to be a good drink. The apples used in commercial apple juice are dessert apples, and lack the acidity and tannins needed to make a good drinking cider. In vinegar, this doesn't matter because the acid level is strong enough to give it the body and character that it will lack before the second ferment.

If you like the style of West Country scrumpy ciders, it's a quite nice drink a couple of days into the aerated second ferment. It's got the acidic tang of scrumpy, but not the funk. However, if you plan to drink it this way, you might want to tone down the sugar to make it a more manageable percentage.



**Below**  During the first few days of the first ferment, you'll get a thick, brown foam on top of the liquid. This is normal

Although you want an oxygen-free environment for the main part of the ferment, having oxygen at the start helps the yeast colonise the liquid. Really, you should start with an open container with more airflow than a demijohn has. However, we've found that as long as you give it a good shake, you get enough oxygen to start your ferment.

You don't need to shake until all the sugar is dissolved, but once most of it has gone, top up with the remaining litre of apple juice. Mix the yeast with a small amount of lukewarm water and a pinch of sugar. It should start to froth and smell like baking bread within about 20 minutes. Once it does, add this to the apple juice and pop the airlock on top.

You'll need to keep the demijohn between about 18 and 33 °C. Much lower than this and the ferment will stall. Much higher and you will kill the yeast. Around 20–24 °C is ideal.

In a couple of days, the liquid should start to froth with a brown foam. If there's no signs of life

(bubbles) in the first three days, take the airlock out and give it a good shake. If there's still nothing a couple of days later, shake again, and add more yeast.

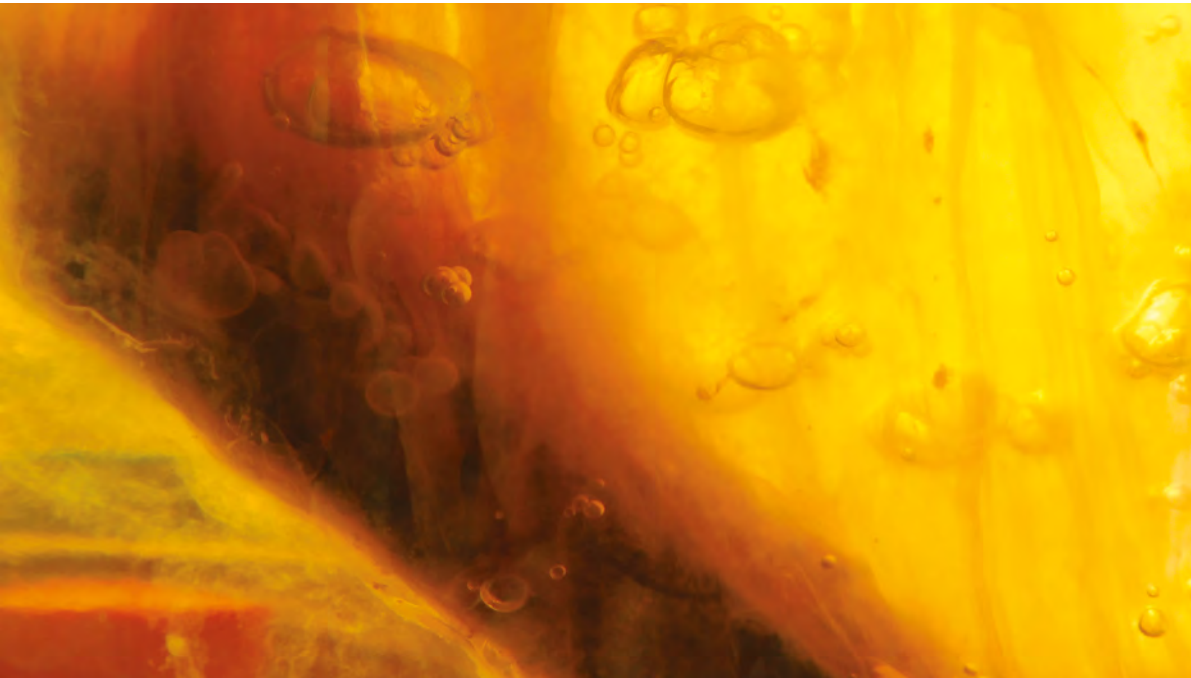
About a week later, the brown foam should disappear and it will be replaced by a much smaller covering of white bubbles. It'll keep happily fermenting like this for a couple of weeks depending on temperature – in cold weather it'll take longer. Once it's finished bubbling, you've got cider (or hard cider as it's known in America). If you drink alcohol, take a taste to make sure it's no longer sweet.

### A SECOND FERMENT

The next step is to turn this cider into vinegar, which requires acetic acid bacteria. Unlike the yeast, these require an oxygen-rich environment all the way through the ferment. We achieved this using an aquarium pump and (clean) air-stone, but you can also get there by just leaving the liquid open to the



**Right** ♦  
During the second ferment, secure a cloth over the top of the fermenting vessel to protect the vinegar from insects



**Left** ♦  
The acetic acid bacteria form mats around the air-stone that trap bubbles

air if you're willing to wait long enough (it might take six months or so). Whichever option you choose, you'll need to protect the liquid from flies as these can lay maggots in the proto-vinegar. The advantage of pumping air in is that you only need a small opening in the top of your container, and this is easier to protect. We continued to use the same demijohn that we'd done the first ferment in. It's best to make sure that whatever container you use is safe in acidic environments. Metals tend to corrode and plastics can leach chemicals, so glass is a good choice.

## // Acetic acid bacteria join together to form colonies

At this point, there should be a build-up on the base of the demijohn. This is dead yeast known as 'lees'. It's best to remove this, as it can lead to off flavours. This just means you need to pour the cider out into a bowl leaving the last centimetre or two, then rinse out the demijohn, and pour the cider back in.

### ADDING BACTERIA

This also has the secondary effect of freeing up a bit of space at the top of the demijohn. We need to introduce acetic acid bacteria. These are fairly common, so if you just proceed without adding anything, there's a good chance it will work, but to

speed everything up, and reduce the chances of something going wrong, it's a good idea to add some raw cider vinegar 'with the mother' – you can buy this from health food shops or larger supermarkets. The mother is a common term for the live acetic acid bacteria that we want. There's no hard and fast rules for how much to add – some sources recommend 20%, but that's expensive for your first batch, and you can get away with far less. We use about 100ml per four litres, but if you've got more, add it.

Pop the air-stone in and cover with cloth (we used a flannel). You'll need to secure this quite well with a rubber band or string to prevent anything getting into the vinegar that shouldn't.

Once the air-stone is bubbling, leave it to ferment. Acetic acid bacteria join together to form colonies. These can look a little dramatic as they grow off the air-stone or float on the surface. They can have the appearance of a cross between pondweeds and aliens, but it's a natural part of vinegar fermenting. Taste once a week or so – it should start to taste a bit vinegary after a few days, but might take up to a month to go all the way. Once it's fermented, decant into bottles and discard this growth (technically, this is the mother, but you never see it in bottles of vinegar sold 'with the mother').

The final result can be very powerful – you can water it down, or temper the acidity with a little sugar – but behind the sharpness, it should be fruity and delicious. You've now got four litres of raw, unfiltered, delicious cider vinegar to pickle, marinade, drizzle, and generally enjoy. How will you consume yours? □

## USING YOUR VINEGAR

Your four litres of vinegar will last almost indefinitely if kept in an airtight container, so you don't need to rush to use it all, but here are a few of our favourite ways of using vinegar:

**Pickling:** Mix your vinegar with a few spices, and perhaps some sugar, and pour over onions, cabbage, chillies, or any other vegetable for a pickled delight. If you plan on storing your pickles, you need to make sure that the pH is low enough to protect the food.

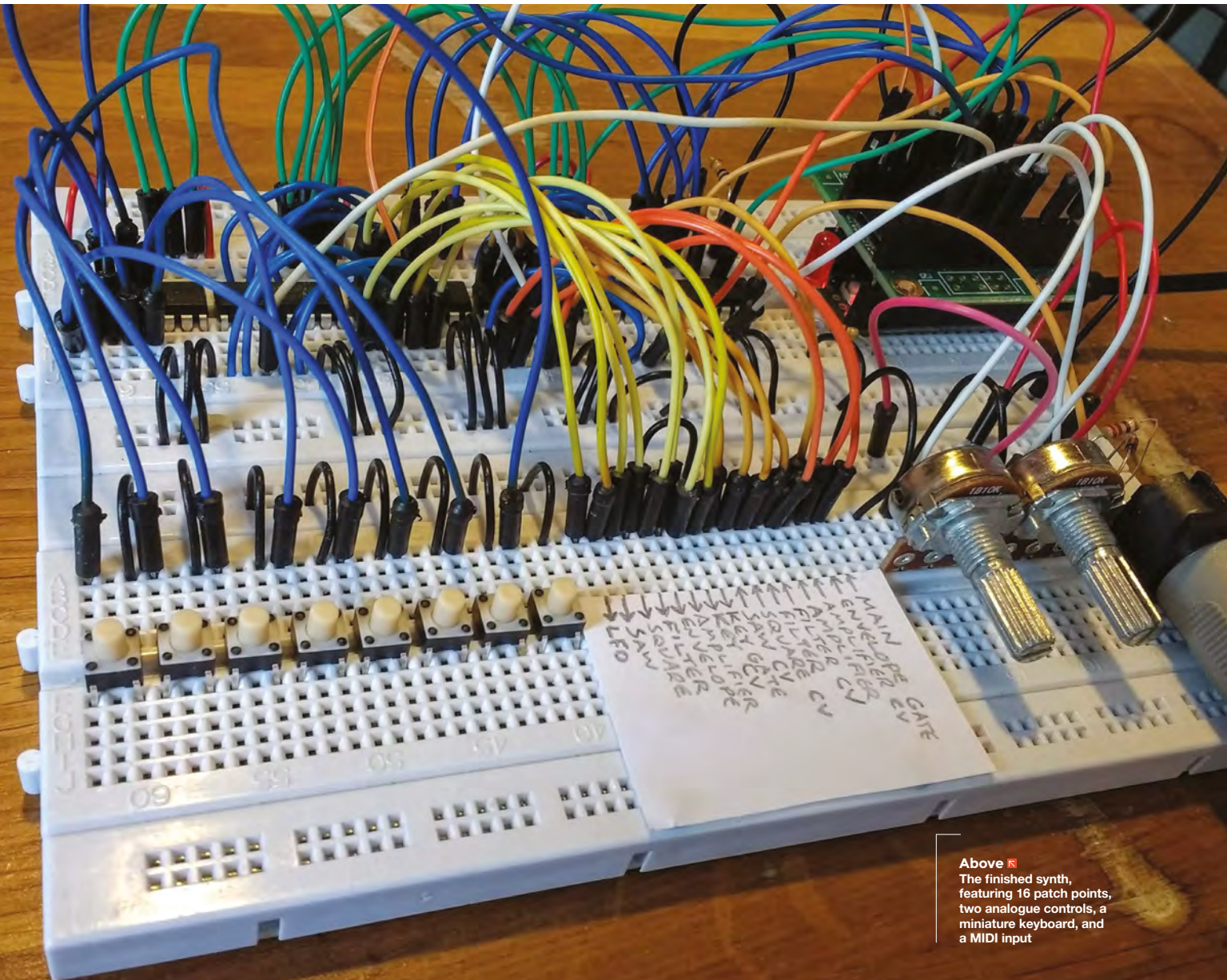
**Salad dressings:** Mix your vinegar with a good, flavoured oil, such as olive oil, (three parts oil to one part vinegar) and add a little salt and pepper for a classic vinaigrette.


**Drinking:** Dilute, to taste, with water for a refreshing sour drink.

**Marinades:** Combine with herbs or spices, and rub onto meat. The acid helps tenderise the meat, ensuring the flavour of the herbs and spices penetrate further into the meat.

# Polyphonic digital synthesizer Part two

The conclusion of our two-part guide to making a polyphonic digital synthesizer



**Above**  The finished synth, featuring 16 patch points, two analogue controls, a miniature keyboard, and a MIDI input



**Matt Bradshaw**

[mattbradshawdesign.com](http://mattbradshawdesign.com)

Matt Bradshaw is a programmer, maker, and musician from Oxford. He likes to build instruments to play with his band, Robot Swans. You can find more of his projects at [mattbradshawdesign.com](http://mattbradshawdesign.com)

**L**ast time, we built a digital synthesizer on a breadboard. It could make some fun noises, but it wasn't very useful for playing music. This time, we're going to rectify that by adding a simple keyboard, as well as a 'MIDI input' port so that you can control the synth with an external keyboard. We're also going to double the number of patch points so you can create more complex sounds. Finally, we're going to edit the code to allow the synth to play multiple notes simultaneously.

If you haven't already read part one, go back and start there – otherwise, let's get stuck in. We've already filled our first breadboard, so we need to add another one. We can leave a lot of our first synth in place: the Teensy, audio board, LED, resistor, and the two 4051 chips can remain untouched on the first breadboard. However, in order to make space for our awesome new features, you should remove the two potentiometers (and their wires), the row of eight

wires that connect the 'patch points' to the 4051 chips, and the label that showed what each patch point did.

The full new layout can be seen in **Figure 1** (overleaf). There is a 6N139 optocoupler (explained later) and three extra 4051 chips. Two of the 4051s perform the same function as in part one, detecting which patch points are connected to each other, but by adding another two chips, we are able to double the number of patch points.

The final (leftmost) 4051 chip acts as a multiplexer for the eight buttons of our mini-keyboard on the front breadboard. These eight buttons will play a simple major scale, although you can change this in the code if you'd prefer a more interesting set of notes.

The front breadboard also now contains the MIDI input, the 16 patch points, and the two potentiometers, so all of the 'hands-on' components (things you might want to access during a performance) are easily accessible.

### WHAT'S NEW?

Before we assemble everything, let's look at what new 'modules' we're adding. The synth already has two oscillators, a low-frequency oscillator (LFO), and →

#### Below

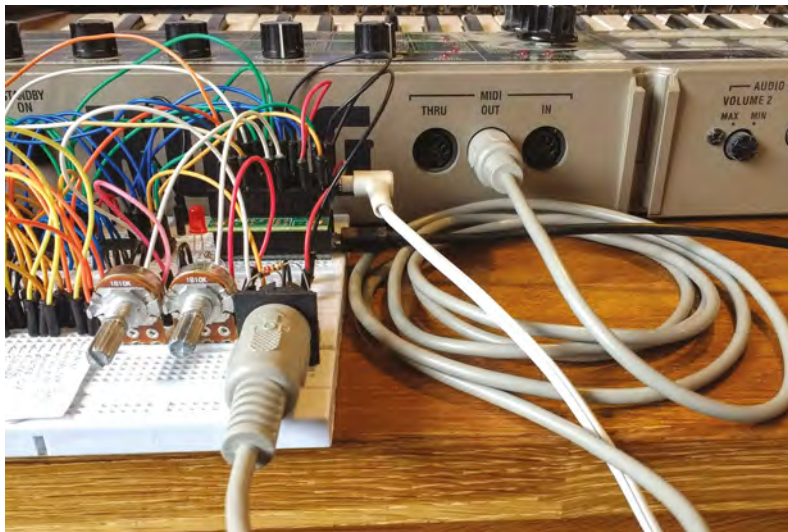
Taking the synth for a test drive – playing with a proper keyboard via MIDI opens up a wider range of notes than the breadboard buttons



### YOU'LL NEED

- ◆ Teensy 3
- ◆ Teensy audio adapter board
- ◆ 4 × 14-pin stackable male/female headers (2 kits)
- ◆ 2 × breadboards
- ◆ Jumper wires
- ◆ 2 × rotary potentiometers (10 kΩ, linear)
- ◆ 5 × 4051 multiplexer chips
- ◆ 8 × tactile buttons
- ◆ LED
- ◆ 6N139 optocoupler chip
- ◆ MIDI socket
- ◆ Capacitor (0.1 μF)
- ◆ Resistors (various)
- ◆ Micro USB cable
- ◆ Soldering equipment
- ◆ Headphones
- ◆ Computer
- ◆ 1N4148 diode

## TUTORIAL



**Above** ♦ Many MIDI devices have three ports: 'in', 'out', and 'thru' – make sure to connect the MIDI out port of your external device to the MIDI in port on the breadboard

### WHAT CONNECTIONS ARE ALLOWED?

In part one, we briefly discussed the 'bad connection' LED, which lights up if you make a connection other than input-to-output. This is a useful feature for diagnosing why your patch might not be working (perhaps you accidentally connected an oscillator to the filter output instead of the input). However, there are some valid patches which will also trigger the LED. If, for instance, you connect both the square and sawtooth oscillators to the main output, the synth will happily mix the two signals, but the LED will illuminate. This is because, electrically, the two oscillator outputs are now connected to each other in a circuit. If you would like an interesting little programming challenge, you could extend the LED code to detect valid connections such as this and disregard them.

a filter. This time we will add an amplifier, an envelope generator, and a MIDI-to-CV converter.

Briefly, an amplifier takes an audio signal and changes its volume. If you feed the module a high control signal, the audio will be loud, while a low control signal will quieten the audio. You can therefore use this module to make an oscillator 'turn on' when you hit a key and 'turn off' when you release it. However, notes that just turn on and off suddenly are not very interesting, which is where the envelope generator comes in.

An envelope generator (EG) mimics the sound of an acoustic instrument. When triggered by an input control signal, often from a keyboard key being pressed, the EG outputs a control signal which, when connected to an amplifier or filter, can evoke the sound of a guitar, a violin, or a piano (depending on the settings).

Finally, a MIDI-to-CV converter takes a MIDI signal from an external keyboard and converts it to a CV (control voltage) signal. This module outputs a 'note' signal (which communicates the last note to have been pressed), and a 'gate' signal (which is simply high or low depending on whether a key is currently being pressed).

Don't worry if these descriptions are new to you – YouTube has plenty of videos detailing how different synth modules work if you'd like to learn more, and we've provided some patching examples to get you started.

### WE WILL REBUILD

Now we've got an idea of the new modules, let's add some components. It makes sense to build the circuit step by step, so we can check for errors at each stage. Firstly, using **Figure 1** for reference, add the two potentiometers, as well as the two 4051 chips directly to the left of the existing ones, and wire them up as shown.

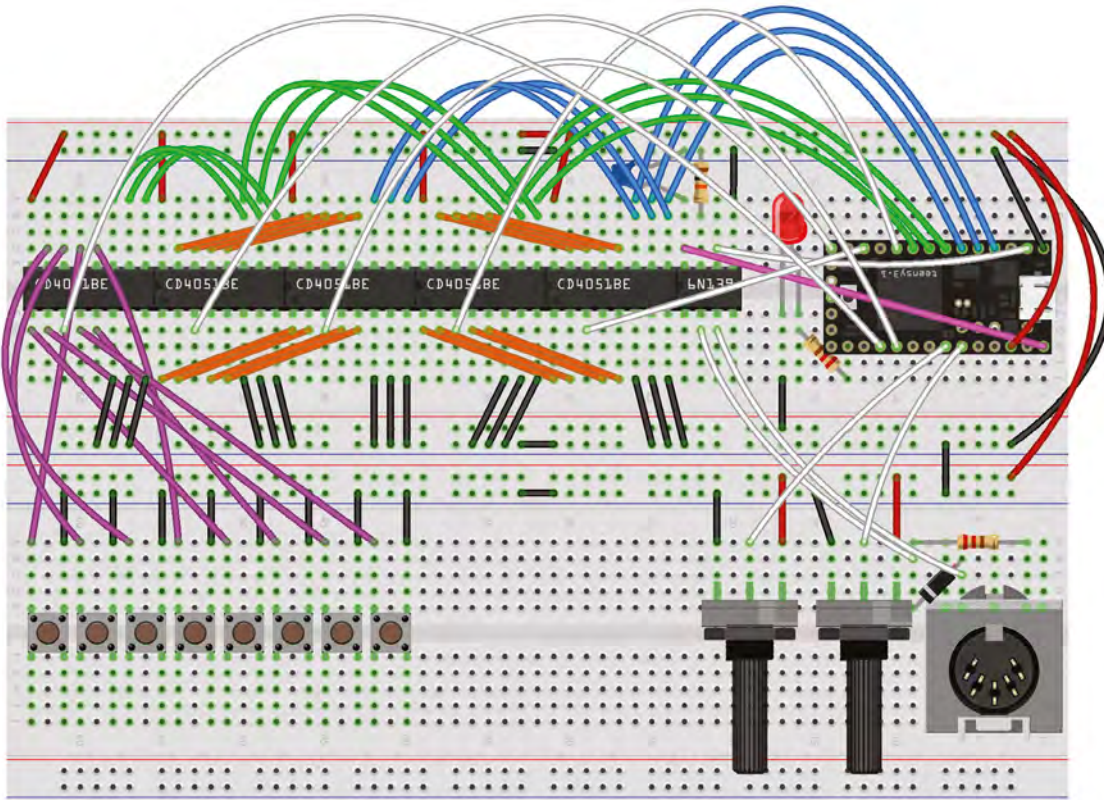
Remember that the original synth required two of these eight-channel chips to provide eight patch points: one chip sends a test signal while the other reads it. By adding another two chips, we can have 16 patch points.

You could patch directly between the chips but, like last time, it's a lot easier if we run a jumper wire from each patch point to a separate, labelled patching area. These wires are omitted on the breadboard diagram for clarity (there are already too many wires on there!), but there is a separate zoomed-in diagram (**Figure 2**) with the patch points labelled as follows:

- A)** LFO (out)
- B)** Sawtooth oscillator (out)
- C)** Square oscillator (out)
- D)** Filter (out)
- E)** Envelope generator (out)
- F)** Amplifier (out)
- G)** Keyboard CV (out)
- H)** Keyboard gate (out)
- I)** Sawtooth frequency (in)
- J)** Square frequency (in)
- K)** Filter (in)
- L)** Filter frequency (in)
- M)** Amplifier (in)
- N)** Amplifier CV (in)
- O)** Envelope gate (in)
- P)** Main output stage (in)

As before, make yourself a label and Blu Tack it to the breadboard.





**Figure 1** ♦ A diagram of the full synth (audio board and patch point wiring omitted for clarity) – note the diode, resistors, and capacitor required for the MIDI input

### SPOT THE DIFFERENCE

Download the code from [hsmag.cc/issue17](https://hsmag.cc/issue17) and have a look at it – there are quite a few differences from part one. Firstly, the audio connection code (generated by the online Teensy audio design tool) has been moved to a separate file. This is because there are a lot more virtual connections this time, so keeping them in their own file makes the main sketch look a lot tidier. The code that handles polyphonic note data from the keyboard has also

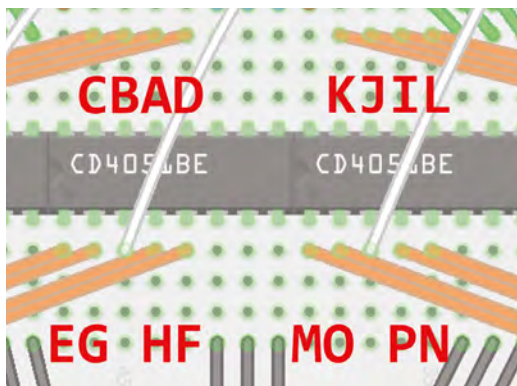
been moved to separate files. Another new element is the MIDI library, which is included and initialised at the top of the code.

The next change is that the `inputMixers` array is now a much more complicated, multidimensional array. Instead of being a simple list of references to four modules, it now contains two separate arrays, which we need because we are creating a polyphonic (multi-note) synth with two copies of every module.

The other most significant difference is that the main `for` loop is now more complex. Previously it was a nested `for` loop with two levels, which was fine because we only had one chip sending data and one chip reading it, but our new circuit →

**Figure 2** ♦

There are 16 'patch points' which connect to each other, creating the signal chain – run jumper wires from here to the second breadboard



### HOW DOES POLYPHONY WORK?

A lot of classic synths, and the vast majority of modern modular synths, only play one note at a time. When designing a synth that plays multiple notes at once, you have to consider what the maximum number of notes playable will be, and which notes should be silenced if you go beyond this maximum.

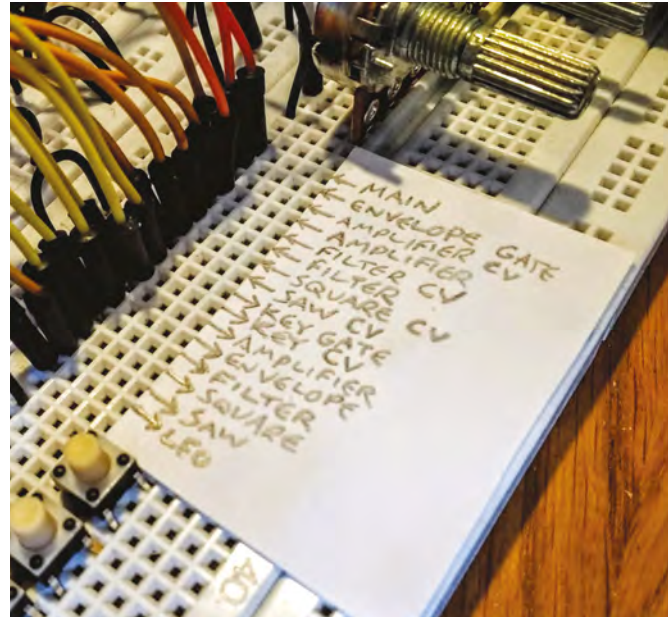
In this synth, we have created two copies of every virtual module in the code, giving two-note polyphony. Try holding down three or more notes and see what happens. If you want to change the current behaviour, for instance to prioritise the highest note, you can edit the `KeyboardHandler` class files.

This synth's polyphony has been kept at two notes to make the code easier to understand, but you should be able to increase it to four notes or even more by tweaking the code, without changing anything in the circuit.

```
Patch 1: "sci-fi" test
A (LFO)  —————> J (square CV)
C (square) —————> P (main output)

Patch 2: keyboard test
G (key CV) —————> I (saw CV)
H (key gate) —————> O (envelope gate)
E (envelope) —————> N (amplifier CV)
B (saw) —————> M (amplifier)
F (amplifier) —————> P (main output)

Patch 3: throbbing filter
G (key CV) —————> J (square CV)
A (LFO) —————> O (envelope gate)
E (envelope) —————> L (filter CV)
C (square) —————> K (filter)
D (filter) —————> P (main output)
```



**Above** Wondering what to do with your new synth? Here are some things to try

**Right** We've kept the patch points in order for this design (outputs on left, inputs on right), but you can easily rearrange them into a more convenient order

necessitates a four-level loop. The principle is the same, but we are having to alternate which chips are active at a given time, hence the extra levels.

Inside the `for` loop, we also check for incoming MIDI data, pass it to the `KeyboardHandler` class so that polyphony is handled correctly, then convert the notes to virtual CV and gate signals so that they can be used for patching.

### PLUG IN, BABY

Upload the code to the Teensy. If all has gone well, you should now have a synth that is very similar to part one, but with 16 patch points. Try some simple patches, such as the square wave going straight to the output – this should produce a simple tone. Now, referring to patch diagram, recreate patch

Finally, add the MIDI input components. Because a MIDI input allows us to connect to another device, we use an optoisolator, which turns the incoming data into a series of pulses of light, then back into a digital signal again. If you would like more detail or ideas for troubleshooting, go to [hsmag.cc/vTjPpc](http://hsmag.cc/vTjPpc) – the MIDI circuit for this synth was based on this design.

If everything seems to be working, congratulations! You have built a semi-modular polyphonic digital synthesizer, and you're ready to make the world a more musically interesting place.

## WHAT IS MIDI?

MIDI stands for 'musical instrument digital interface', and is a system whereby one instrument can control another via a special cable. The MIDI standard can deal with all sorts of musical information, such as tempo, pitch bend, and sustain pedal, but for this synth we're just going to implement the basic 'note on' and 'note off' commands.



**It makes sense to build the circuit step by step, so we can check for errors**



1 using jumper wires, and adjust the right-hand potentiometer – if it sounds like sci-fi effect, it's probably working. If not, check your connections.

Next, add and connect the final 4051 chip, plus the eight buttons that constitute our miniature keyboard. You should now be able to use patch point G (keyboard CV) to control the frequency of your oscillators, and patch point H (keyboard gate) to control the amplifier or envelope generator – try recreating patch 2 to see the keyboard in action. The sketch will allow the breadboard keyboard to function until a MIDI signal is detected, at which point the breadboard keyboard will be disabled.

## WHAT TO DO NEXT

There are loads of things you could do next with this synth. You could add code to make it recognise more MIDI commands, allowing MIDI control of the filter and envelope. You could change what the patch points do – perhaps you would like a white noise generator instead of a second oscillator? If so, have a look at [hsmag.cc/WzjFUw](http://hsmag.cc/WzjFUw) – there are lots of virtual synth building blocks for the Teensy detailed there.

Perhaps the most satisfying next step, though, would be to upgrade this design from a pretty mess of breadboard wiring to a more permanent form using stripboard. You could keep using jumper wires for patching, while soldering everything else in place, and make a sturdy enclosure from wood, metal, or 3D-printed plastic.

# Wireframe

Join us as we lift the lid  
on video games



Visit [wfmag.cc](http://wfmag.cc) to learn more


# Turn a hoodie into a wearable game controller

Play Flappy Bird by flapping like a bird

**A**re you ready to fly? In this project, we'll build a wearable game controller for a one-button game. What can you do with one button? Play Flappy Bird, of course! Our game controller is built into a hoodie. Flap like a bird to make your in-game bird soar!

While the original game is no longer available as a mobile app, you can find several Flappy Bird emulators built by fans with the free online programming language, Scratch. The game is played by tapping a button, the **SPACE** bar, to keep your bird in the air. Each tap gives your bird a boost, like a flap of its wings. The goal is to tap at the right time to guide your bird through as many gates as possible. But why tap when you can flap?

We can turn our flaps into taps by using the HID capability of a Circuit Playground Express (CPX). HID stands for Human Interface Device, and it means that the CPX can act like a keyboard or mouse for your computer. We'll program the Circuit Playground Express to send a **SPACE** bar press to the computer every time it detects an arm flap. Conductive fabric

Right   
The final controller, ready to be worn





Left ◀  
It really is as fun as it looks



Sophy Wong

🐦 @sophywong

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

patches on the arm and torso of the hoodie will detect when the wearer's arm is down. Each real-life arm flap (think chicken dance) will be translated into an in-game bird flap.

To create conductive surfaces on the hoodie, we are using conductive fabric. Conductive fabric is plated with a conductive metal like silver, and can be either woven or knit. To keep the coating from tarnishing or rubbing off, dry-cleaning is recommended for this material. We'll be hot-gluing feathers to this hoodie when we're done, so do not throw this hoodie into the washing machine – spot-clean only. When adding conductive fabric to a garment, match the stretchability of your garment: woven conductive fabric for non-stretch garments, and knit conductive fabric for stretch garments. Here, we are using knit conductive fabric because our hoodie is also made of knit fabric. Keep in mind that while knit fabrics stretch, straight seams of thread do not (such as the conductive thread traces we will be sewing). To allow for this, leave your stitches a little loose when sewing, or use a stretchable stitch, like a zigzag.

Making your own game controller is a great way to customise your gaming experience and experiment with inputs like buttons and sensors. Another great microcontroller option for this is the Makey Makey, a board designed specifically for turning bananas and other



Above ♦  
You can test out the fabric with crocodile clips

capacitive objects into human input devices. Visit the Makey Makey guide website, at [hsmag.cc/vpwwvT](http://hsmag.cc/vpwwvT), to see tons of creative input devices made with unconventional objects. Even a relatively boring game might be fun to play on bananas!

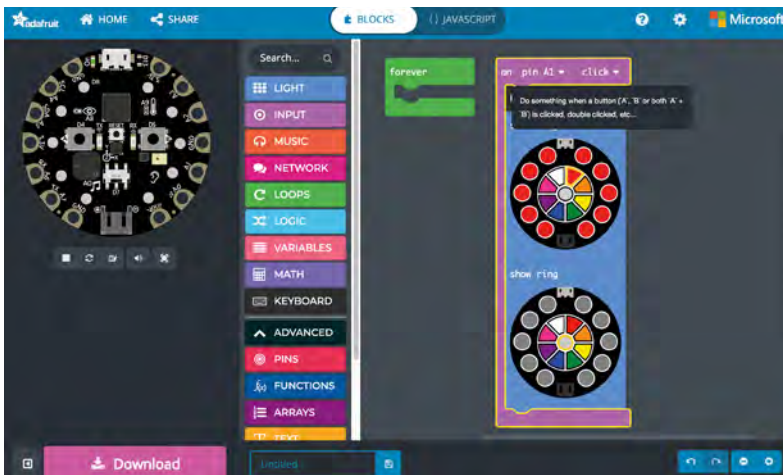
Adding movement to gaming is more than just fun, it's good for you too! In an effort to combine physical activity with coding, and promote STEAM subjects, educators John Lynch and Susan Klimczak, in Boston, Massachusetts, have created a workshop called Explode the Controller. The students use Makey Makey boards to build and hack physical game controllers, like jump-sensing platforms and slap switches. The students can also build their own games that can be played with movement using Scratch programming. For more innovative physical controllers and some inspiration, visit: [hsmag.cc/JfZoAE](http://hsmag.cc/JfZoAE). →

### YOU'LL NEED

- ♦ Circuit Playground Express
- ♦ Micro USB cable (long)
- ♦ Conductive fabric, knit (Adafruit part 1168)
- ♦ Conductive thread
- ♦ Crocodile clip test leads
- ♦ Hoodie
- ♦ Felt (coordinating colours)
- ♦ Fusible web (available from sewing and craft stores)
- ♦ Press cloth or Teflon pressing sheet
- ♦ Iron
- ♦ Sewing needle, thread, pins
- ♦ Glue gun and hot glue
- ♦ Fray Check or CA glue
- ♦ Removable pen or dressmaker's chalk

# Turn a hoodie into a wearable game controller

## TUTORIAL



**Above** ♦  
MakeCode makes programming easy

**Below** ♦  
Measure twice, cut once

In our project, we're using the Circuit Playground Express by Adafruit, which is perfect for wearables with its easy-to-sew-through pin holes.

The Circuit Playground Express has seven capacitive touch inputs, but we're only using one for this project. For a more complex game controller, you can add more conductive fabric patches to your hoodie, or use some of the other sensors onboard the CPX. Our hoodie only requires one wing to flap, but for a more accurate experience, add a second sensor to the other arm. In MakeCode, you can modify our program to require both arms to flap at the same time to give yourself more of a workout.

### PROTOTYPE AND PROGRAM

Start by making a quick prototype of the circuit. The Circuit Playground Express has seven capacitive touch inputs on pins A1 through A7. Snip a small piece of conductive fabric and clip it into one end of an alligator clip lead. Clip the other end of the lead onto the A1 pin on the Circuit Playground Express. This is a slightly

simplified version of the circuit we are going to build, and we can use it to work on our code.

Connect the Circuit Playground Express to your computer using a micro USB cable, open your browser, and navigate to [hsmag.cc/qJhlwD](https://hsmag.cc/qJhlwD).

If this is your first time working with Microsoft's MakeCode, head over to the Adafruit guide at [hsmag.cc/uKjuVC](https://hsmag.cc/uKjuVC) for a great introduction and more tutorials.

In the MakeCode workspace, we first need to add the Keyboard module to our project. Open the Advanced module, then open the Extensions module. In the screen that appears, click on Keyboard. You will be transported back to the MakeCode workspace, with the Keyboard module added to the list below Math. Next, we'll set up the A1 pin as a capacitive touch input. From the Input module, find the 'on button click' block and drag it onto the workspace. Click on 'Button A' and change it to pin A1. Now, whenever the A1 pin senses a 'click', the code we enter inside this block will be triggered.

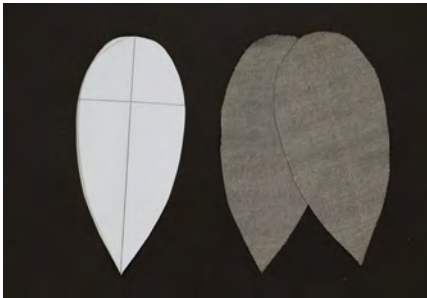
**// In the MakeCode workspace, we first need to add the Keyboard module //**

Think of 'click' like a mouse click – you press down and release. Selecting 'on click' means the event won't be triggered until the pin (or conductive patch) is released. There are other options in the button event module (long click, down, and up) but we're going to use 'click' so that each flap is like a tap of the **SPACE** bar.

To send the **SPACE** bar key press to the computer, open the Keyboard module and find the Keyboard type block. Drag this block onto the workspace and drop it into the button event block. Between the quotation marks, type a single space.

Now, without the game running, it may be hard to tell if your program is working. So let's add another action to this button event so that we have some feedback on the Circuit Playground Express when A1 is clicked. This feedback will also help if we need to do any debugging later.

From the Light module, grab the 'show ring' block and drop it in the button event block below the keyboard command. This will turn on the NeoPixel ring on the Circuit Playground Express and show red. To turn the NeoPixels off, place another 'show ring' block below the first one, and turn off all the NeoPixels in



this second block (click on the grey circle in the centre of the ring, then click on each of the NeoPixels to turn them grey).

Test your code by clicking on the A1 pad in the simulator window on the left. You won't see any representation of the keyboard command, but you should see the ring flash red each time you click. Once you've confirmed your code is working, remove the Show Ring blocks from the On Button Click block for better timing and more responsive gameplay. Click Download and follow the on-screen instructions to transfer the code to your board.

To see if the **SPACE** bar command is being sent, load up your game and give it a go. Touching the piece of conductive fabric should make your bird fly, just like pressing the **SPACE** bar on your keyboard.

That's all the code you need for a simple one-button flying bird game. But you don't have to stop there! Now that you know how to connect inputs to keyboard commands, you've got all the on-board inputs at your disposal. Add more conductive patches to your hoodie to build more complex game controllers for games that require more than one button. Or try using the accelerometer or microphone to control your game!

## DESIGN YOUR HOODIE

With the game controller circuit prototyped, now we can work on how to integrate it into a wearable form. A hoodie is a good choice for this project, with lots of surface area for capacitive sensor patches.

To achieve the 'flapping bird' motion, we'll place one conductive fabric patch on the inside of one arm, and the other patch on the side of the torso. The patches need to touch when the wearer's arm is down. With a tape measure, find the distance from your armpit to your elbow. Subtract about 5 cm from this measurement, so the patch will sit above your elbow. On the hoodie, mark this distance with a pin on both the inner seam of the sleeve and the side seam of the torso. Or, you can simply put the hoodie on and eyeball your marks.

Decide where to put the Circuit Playground Express, and remember that it will be connected via USB cable to your computer while you are playing the game. Choose a spot on the front of the hoodie



on the same side as the sensor patches. This will make it easy to sew a trace from the board to the torso patch.

## MAKE CONDUCTIVE PATCHES

In keeping with our bird theme, our patches are shaped like feathers. Use a template to cut two patches of conductive fabric and two matching pieces of fusible web. Follow the instructions on your fusible web to apply it to the underside of the conductive patches (do not apply steam). Use a press cloth or a Teflon sheet to make sure you do not melt the conductive fabric.

Place one patch in place on the sleeve, centring it on the side seam. Iron without steam, using your press cloth, to fuse it in place. Then repeat to affix the other patch to the side of the torso, making sure these patches will match up when worn. To prevent →

**Above** ♦  
You could use any shape, but we like feathers

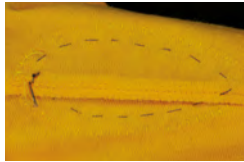
**Below** ♦  
Yellow cotton gives a feathered edge effect



# Turn a hoodie into a wearable game controller

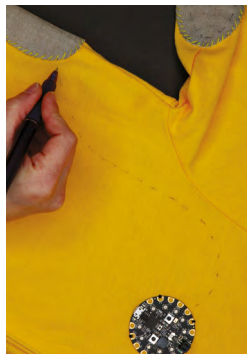
## TUTORIAL

**Right**  Conductive thread takes the place of wires in this circuit



**Right**  A few stitches hold the CPX in place

**Below**  The thread will be visible, so be neat!



the edges from peeling up, thread your needle with regular sewing thread and whip-stitch around the edges of each patch.

To make the arm patch work as a conductive signal, it needs to be connected to the inside of the sleeve so that it will be touching skin when the hoodie is worn. A line of conductive thread works perfectly for this. Thread your needle with conductive thread and sew a running stitch (under, over, under, over) around the inside perimeter of the arm patch. Tie a knot on the inside of the sleeve and dab a little Fray Check, or cyanoacrylate glue, on the knot to keep it tight. Cut the thread tail short.

### ATTACH THE CIRCUIT PLAYGROUND EXPRESS

Hold the Circuit Playground Express in place on the front of the hoodie, and make a mark through four of the pin holes: A0, A3, A4, and A7. Thread your needle with regular sewing thread. Using your marks as guides, sew a few times through each of these holes to attach the board securely to your hoodie. Tie a knot on the inside and trim the thread.

With the Circuit Playground in place, you're ready to sew a conductive trace from the A1 pin to the side torso patch. Use a removable marker or chalk to draw

out your stitching line first. Avoid sewing over any openings or pockets.

Thread your needle with a piece of conductive thread long enough to sew the whole stitch line you just drew in one piece. Tie a knot in the end, and dab some Fray Check or CA glue on the knot. Starting at the Circuit Playground Express, stitch several times through the A1 pin, pulling your stitches tight against the board for a good connection. Then start sewing a running stitch along your line toward the patch.

While sewing your running stitch, do not pull your stitches too tight. You just want them to lay flat against the fabric – if you pull too tight, you will gather the fabric and bunch it up. Remember that hoodies are made of stretchable fabric, but conductive thread does not stretch. When you get to the patch, take several stitches through the patch for a good connection and tie it off on the inside of the hoodie. Apply glue to the knot and trim.

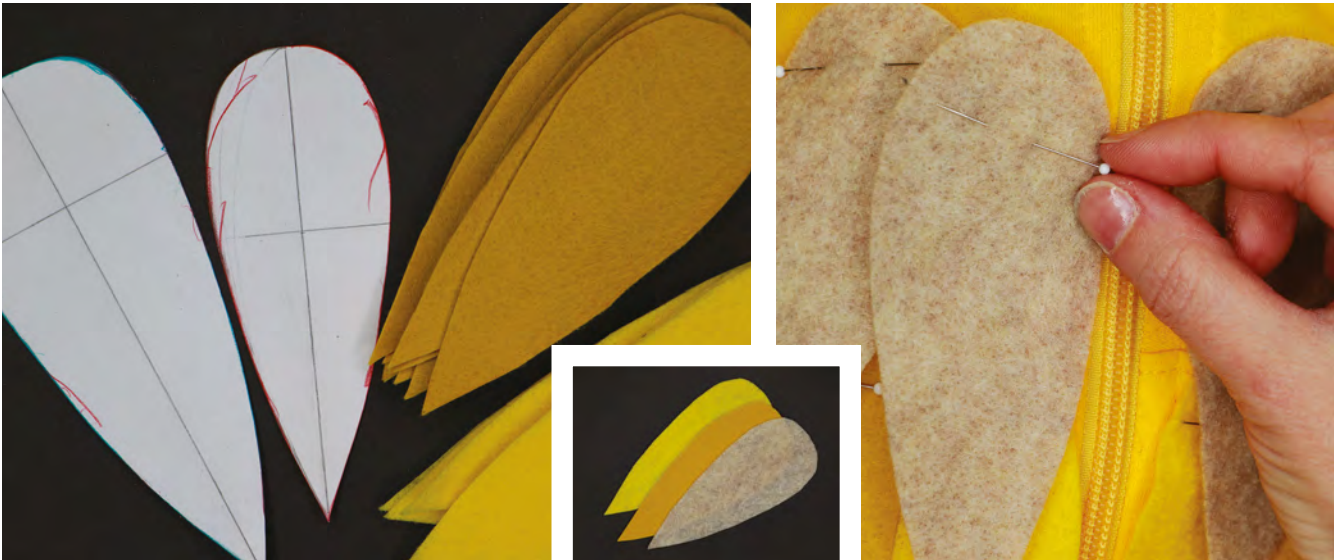
**While sewing your running stitch, do not pull your stitches too tight**

Put your hoodie on, and check to see if everything is working. Load up your game, plug in your CPX, and flap – does your bird fly? Hooray! If not, check your connections, and make adjustments until your flaps work as intended.

### BIRD ALERT

When you need a break from flapping, you can decorate your hoodie for more bird power. We can't guarantee it will make the game any easier, but it can't hurt!





Use a template to cut feathers out of felt in a few different colours and sizes. You don't need to cover every inch of the hoodie – you can hint at wings by just covering the shoulders. For the shoulders, hood, and tummy of our hoodie, we used about 130 feathers.

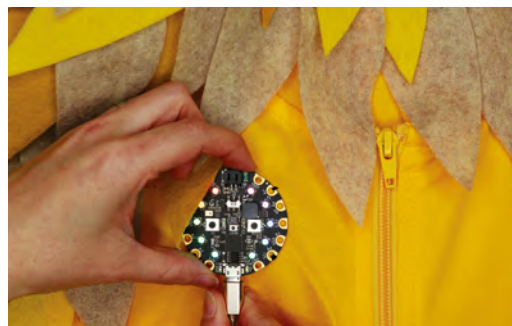
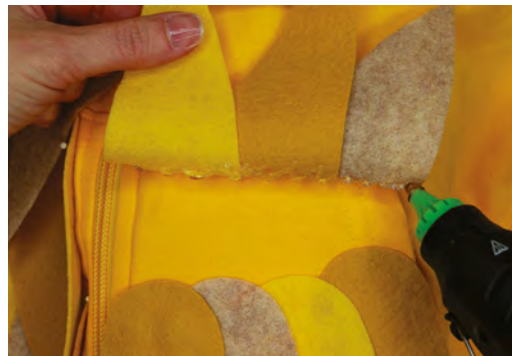
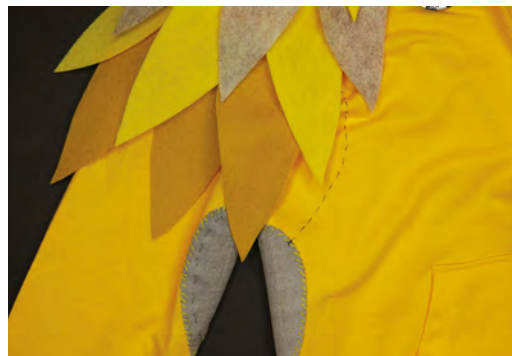
Lay your hoodie out flat on your work surface, and begin laying out your feathers. Pin the feathers in place, with one pin at the top of each feather. Mix up the colours for a mottled effect, or use them in layers for more drama. Don't worry too much about symmetry, just go for visual balance and have fun with your pattern. If you want to add more conductive feather patches, you can work them into your pattern and sew conductive thread traces to them from your Circuit Playground Express as before. Keep all the conductive traces separate from each other. Make sure that your felt feathers do not block the two conductive patches from touching each other when the arm is down.

When you are happy with your pattern, attach the feathers with hot glue. Starting at the bottom layer, lift up each feather and apply hot glue at the top only. Remove pins as you go. Continue gluing feathers down, layer by layer, bottom to top. Be sure not to cover any pocket openings and leave the zipper clear.

**LET'S PLAY!**

Keep adding feathers until you're happy with your hoodie. Now you're ready for both Halloween and Game Night! Remember that you are tethered to your computer when playing – be sure to unplug before walking away!

This fun project can be as simple or as complex as you make it, and we want to see how you customise it and make it your own. Show us your creative controllers at: [hackspace@raspberrypi.org](mailto:hackspace@raspberrypi.org)!



**Above** Use a stencil to get the feathers the same size

**Left** Birds don't have to be yellow – you can be creative!

**Left** Once you're finished, power up and play!

# Remote display for hackspace automation

A low-powered display device for connecting to your home (or hackspace) automation system



Andy Clark

workshopshed

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

**H**aving a phone app to control your home automation is fine when you are the only user. But in a shared space it can be a lot easier if there is a dedicated device to interact with. So that we don't need to keep the remote plugged in, we want to keep the power consumption really low, this is done using an e-ink display and an ESP32 microcontroller module. Communication is via MQTT, an open protocol, which means our device can be easily used with different systems. Here, we're building a display that reads a message from the MQTT queue containing the temperature information and displays it, but you can easily modify it to display other information. You'll need another device to publish messages to the queue in order to get information to display.

## COMMUNICATING WITH ESP

The ESP32 chip from Espressif can be programmed with the Arduino IDE. Although it is possible to buy

the module as a standalone, it is very small and hence can be tricky to solder. So we bought ours on a dev board, the 'ESP32 D1 Mini32 LOLIN32 Pro', which is also known as the TTGO MINI32. This board already contains a low-dropout regulator and a battery charger, if you don't have these on your board, then you'll need to purchase them as separate components.

The ESP32 does not have USB built in, and it communicates and is programmed via a serial protocol. Many of these ESP32 dev boards have a simple USB to serial chip such as the CH340 or CP210x. You'd need to buy a USB to TTL serial adapter if your board does not include one.

Windows PCs and Macs don't include drivers for these chips, so you'll also need to download and install these if you don't already have them: [hsmag.cc/lwdWEy](http://hsmag.cc/lwdWEy)

The Arduino IDE also needs to be configured to talk to this board; this step requires an internet connection. From the Arduino IDE, select the File menu, then Preferences. Add the following to the Additional Boards Manager URLs:

**`dl.espressif.com/dl/package_esp32_index.json`**

Then, from the Tools menu, select Board and Boards Manager. Search for 'ESP32' and install. If you plug in your board, you can select the COM port and board for the device.

## ON DISPLAY

E-ink displays are quite complex to control, but Jean-Marc Zingg has created a library that simplifies the process. This works with the Adafruit GFX graphics libraries, so we have a choice of fonts and some drawing capability.

From the Tools menu, select Manage Libraries, search for 'GxEPD', and install the GxEPD2 library.



**Left** The low-powered e-ink display shows temperature data



**Above** ♦ Download the tools and libraries to compile for the ESP32 module

Repeat these steps to install the Adafruit GFX Library and U8g2\_for\_Adafruit library for graphics and fonts. Also, add the PubSubClient from Nick O’Leary to talk to MQTT.

The e-ink module comes with an adapter that connects to the host board via a number of wires. Two of these are for power, four handle the serial communication (using the Serial Peripheral Interface protocol, SPI) to the display, one is for reset, and one is an output that tells us when the display is busy.

The GxEPD library has some suggested pins for the connections; the critical ones are the four SPI connections which need to map to your board’s SPI hardware – the others can be moved if you want to use the pins for something else. Normally we’d recommend using headers and connectors, but for this project space is a bit short, so we wired the connections directly.

We discovered the pin diagram supplied with the board had some errors, with pins 16 and 17 swapped. Luckily, the silkscreen did match the correct pins, and we spotted the problem before soldering.

To power the board, we used a 2000mAh battery. The connector on this was swapped for one that fitted the ESP32 board. Carefully cut each wire one by one so you don’t short the battery, and solder on the wire to the new connector. If you have to do this, make sure you get the polarity correct and insulate the joints (ideally with heat-shrink tubing). Be careful when dealing with LiPo batteries and double check before making any changes, as they are unstable.

### PLASTIC FANTASTIC

The case for this display is made from acrylic sheet. It’s a great material to work with once you understand its limitations. Acrylic can be cut with a saw, filed, drilled, sanded, and formed using heat. It is supplied in sheets so you can quickly create large items that would take a long time to 3D-print. It comes in a range of colours, and can have a high-gloss finish. Its main drawback is that it’s brittle, so applying too much force or heavy impacts can cause it to break.



**Above** ♦ A library to control your e-ink display

There are three main parts to our case: the front panel, back, and a wraparound side. It’s best to make that part first. Marking out on acrylic can be tricky, so cover the areas to be marked out with masking tape.

Using a coarse-tooth hacksaw (14 TPI), cut a strip 25mm by 300mm. The saw can leave a rough edge which can be smoothed with a scraper. A scraper is a sharp-edged metal tool. These are pulled along the surface of the material to cut a thin sliver from the material. They can be bought, but an old ruler or offcut of stainless steel sheet can also be used.

Once you have a smooth strip, it can be bent. We made a simple wooden former with rounded corners, then two blocks were glued and pinned to a small sheet to provide the negative part of the former. Heat the plastic at the point you wish to bend it. Heating both sides will speed the process. Once the plastic is floppy, kill the heat and quickly bend the plastic around the former. Repeat this for the other corners.

For the last side, you will need to trim the end to length before completing the last fold. Bend the corner so they overlap, trim the end, then →

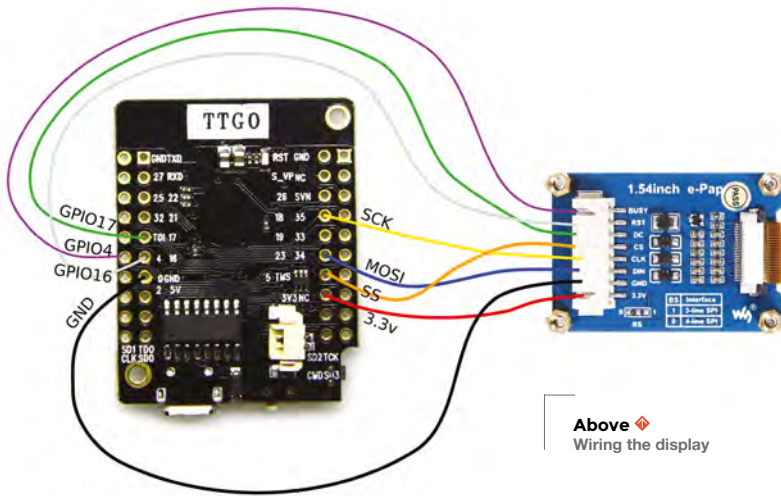
### YOU’LL NEED

- ♦ ESP32 dev board
- ♦ LiPo battery
- ♦ E-ink display
- ♦ Acrylic sheet
- ♦ Threaded inserts
- ♦ Adhesive
- ♦ Hot-air gun
- ♦ Heat-proof gloves
- ♦ Old steel ruler or scraper

**Below** ♦ Module wires

<b>VCC: 3.3 V</b> <b>GND: Ground</b>	<b>Power</b>
<b>DIN: SPI MOSI pin</b> <b>CLK: SPI SCK pin</b> <b>CS: SPI chip selection, active low</b> <b>DC: Data/Command selection</b> <b>(high for data, low for command)</b>	<b>Serial Peripheral Interface (SPI)</b>
<b>RST: External reset, active low</b> <b>BUSY: Busy status output, active low</b>	<b>Control (GPIO)</b>

## TUTORIAL



reheat and fold. Glue the ends together with some plastic adhesive.

Once you have the side walls formed up, you can cut the front and back faces to size. Mark out the cut-out for the display, and drill a hole in each corner. When drilling, ensure the sheet is well-supported and that you do not use too much pressure, or the plastic will crack. Use a coping saw to join the holes,

**// We also drilled holes in each of the corners of the support sheet – this is used as a template to drill the back panel**

and a file or scraper to smooth the edges. Use a fine-grit sanding paper with lots of water to ensure the edges are silky smooth. The front can now be glued to the side wall.

To support our display and the ESP32 board, we cut a smaller piece of acrylic to fit inside the case. We drilled holes so that the boards could be screwed in place.

We also drilled holes in each of the corners of the support sheet – this is used as a template to drill the back panel. We need some small pieces of acrylic cut and drilled to hold the threaded inserts; we used thicker sheet for this, but you could also make them with layers of material. Take care to leave enough space around these for the electronic components. Thread the bolts through the back panel and into the inserts. Glue the inserts into the front of the case.

### QUICK TIP

To check you have configured your IDE correctly, you can compile and upload the blink example.

### MESSAGES FROM THE CLOUD

That's the hardware of our build sorted – now let's take a look at the software side of things.

To test the code, we used a public MQTT broker at [hsmag.cc/zowAiY](https://hsmag.cc/zowAiY) – this comes with a test WebSocket client. A broker gives us a single place

to send messages to, and receive messages from. This way, each device in our home network only has to send messages to one place – the MQTT broker – and it can communicate with all the other devices.

It is also possible to test the using a Raspberry Pi and a suitable MQTT broker, such as Mosquitto, if you'd rather keep control of the whole system.

Our device listens for two messages: one for the temperature and one for status. These are:

**automation/mainroom/status**  
**automation/mainroom/temperature**

We can subscribe to both of these using:

**automation/mainroom/#**

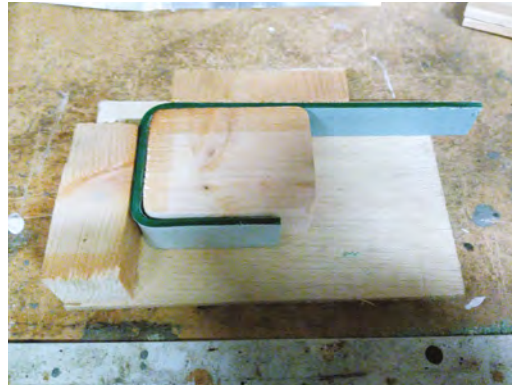
You can grab the code from [hsmag.cc/TsUiQy](https://hsmag.cc/TsUiQy). We won't go through the whole code, but let's take a look at the key parts.

We use the PubSubClient to connect to the server, and set a callback that runs the function `on_Msg` every time a new message arrives in the queue. This is done with the following bits of code:

```
#include <PubSubClient.h>
...
PubSubClient mqttClient (espClient);
...
void setup_MQTT() {
  mqttClient.setServer(mqtt_server, mqtt_port);
  mqttClient.setCallback(on_Msg);
}
...
void on_Msg(char* topic, byte* payload, unsigned
int length) {
  String sTopic;
  for (int i = 0; topic[i] != 0; i++) {
    sTopic += topic[i];
```

## INK BUT ELECTRONIC

Modern displays are designed to update quickly and show bright, moving images. The semiconductor technology which drives them uses a lot of energy, even if the screen is not changing. E-ink uses a different technology which is optimised for low power. An e-ink display uses microscopic charged particles, which are attracted to and from the surface of the display using an electrical field. The display only requires power to change the colour, and hence a static image can be displayed on the screen for a very long time with no power at all.



**Far Left** ◆  
Scraping produces less mess and can be faster than sanding

**Left** ◆  
A jig helps form the correct shape

```

}
String sPayload;
for (int i = 0; i < length; i++) {
  sPayload += char(payload[i]);
}
if (sTopic.endsWith("/temperature")) {
  temperature = sPayload;
}
if (sTopic.endsWith("/status")) {
  statusMsg = sPayload;
}
updateDisplay();

```

The **loop** function then just has to make sure that we stay connected to the MQTT broker, and run the internal **loop** function from the MQTT library. Every 30 seconds, we also send a 'heartbeat' to the broker so it stays connected.

```

void loop() {
  if (!mqttClient.connected()) {
    reconnect();
  }
  mqttClient.loop();

  long now = millis();
  if (now - lastSend > 30000) {
    lastSend = now;
    mqttClient.publish(clienttopic.c_str(),
    "Heartbeat");
  }
}

```

This gets the appropriate data on the device, but we still have to get it on the display. In the **on\_Msg** function we called **updateDisplay()**, so let's have a look at that now:

```

void updateDisplay()

```

```

{
  display.firstPage();
  do
  {
    int margin = 3;
    int lineHeight = 3;
    int linespace = 6;
    display.fillScreen(GxEPD_WHITE);

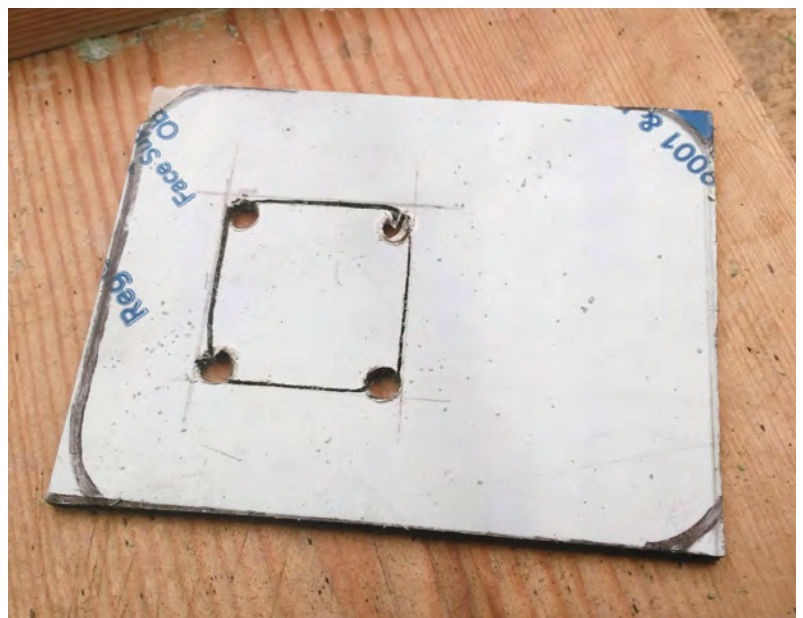
    u8g2Fonts.setFont(u8g2_font_logisoso20_tf); //
    select u8g2 font from here: https://github.com/olikraus/u8g2/wiki/fntlistall
    uint16_t y = u8g2Fonts.getFontAscent() +
    margin;
    printCentredMsg(y, roomName);
  }
}

```

**QUICK TIP**

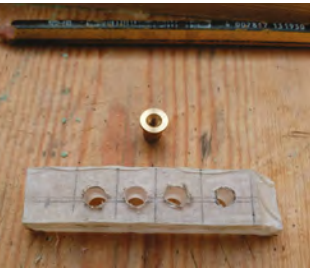
Pressing the button to the right of the additional boards box, then following the link takes you to a list of additional URLs.

**Below** ◆  
Cutting the window



# Remote display for hackspace automation

## TUTORIAL



**Above** ♦  
A solvent adhesive can be used to glue the parts together

```
drawLine(y + linespace,lineheight);

u8g2Fonts.setFont(u8g2_font_logisoso58_tf);
uint16_t yt = (display.height() + u8g2Fonts.
getFontAscent()) / 2;
printCentredMsg(yt,temperature);

u8g2Fonts.setFont(u8g2_font_logisoso20_tf);
uint16_t ys = display.height() - margin;
printCentredMsg(ys,statusMsg);

drawLine(ys - u8g2Fonts.getFontAscent() -
linespace - lineheight,lineheight);
}
while (display.nextPage());
}
```

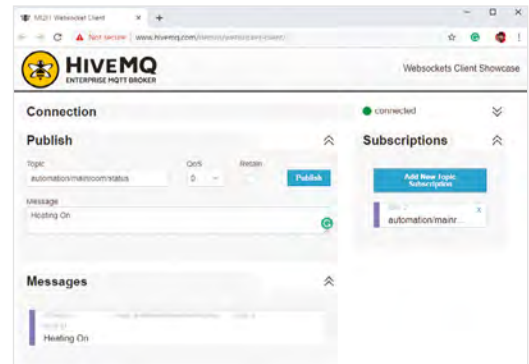
This is built around the GxEPD2 library, which in turn uses the U8g2\_for\_Adafruit\_GFX library, so you'll need to make sure that both of these are installed.

With the code uploaded, your device is ready to use. You can test the display by powering it up and publishing messages from the MQTT client. Send on topic 'automation/mainroom/status' for the message at the bottom of the screen and 'automation/mainroom/temperature' for the temperature in the

middle of the screen. If you use the 'retain' option then the display will get the values from when the topics were last published.

This display is just one example of what you can achieve based on your needs. You can send almost any data through MQTT, and display it however you like. If you want a weather display, thought of the day, calendar, or any other data displayer, you can customise this code for that. Let us see what you make @HackSpaceMag or [hackspace@raspberrypi.org](mailto:hackspace@raspberrypi.org).

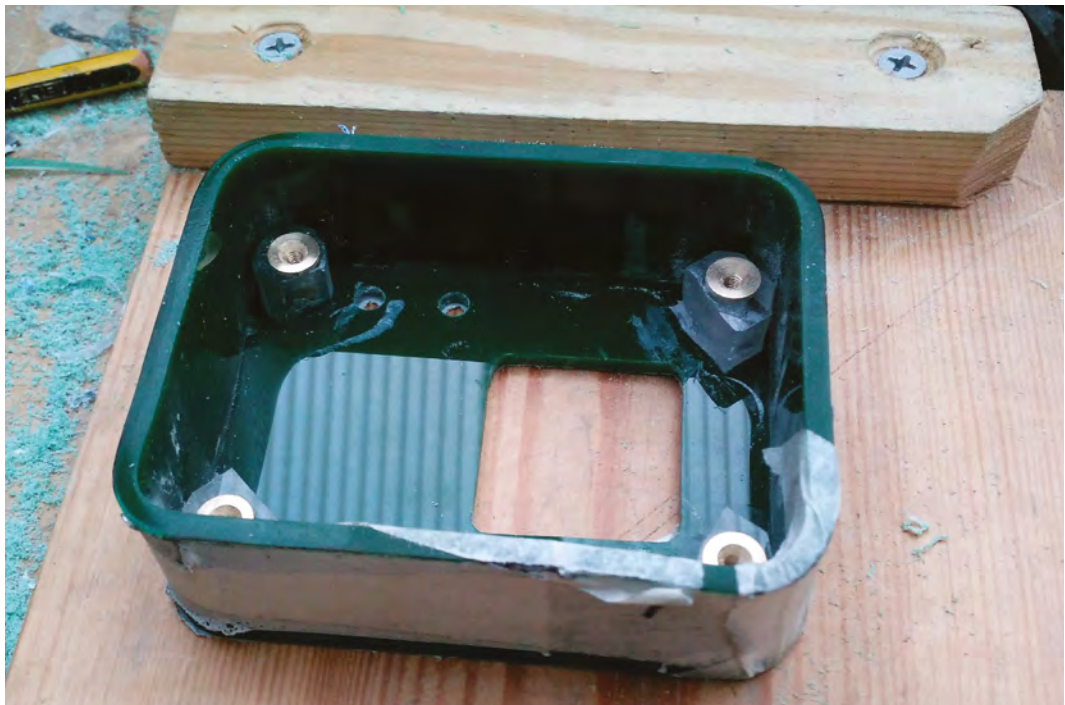
**Below** ♦  
Testing with status and temperature topics



### THREADED INSERTS

These are a metal tube with grips down the outside and a thread down the middle. They're great for use with soft materials that don't tap well, and can also be used for repairs. They can be designed to push fit, screw fit, or heat fit. The screw fit ones work best with wood, and push fit for acrylic.

**Right** ♦  
The metal inserts fitted into the box to mount it



£12.99  
200 pages of  
Raspberry Pi

THE *Official*

# RASPBERRY PI PROJECTS BOOK

VOLUME 4

Amazing hacking  
& making projects

from the creators of

*The MagPi* magazine

Inside:

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

Available Now

**FREE  
DELIVERY**

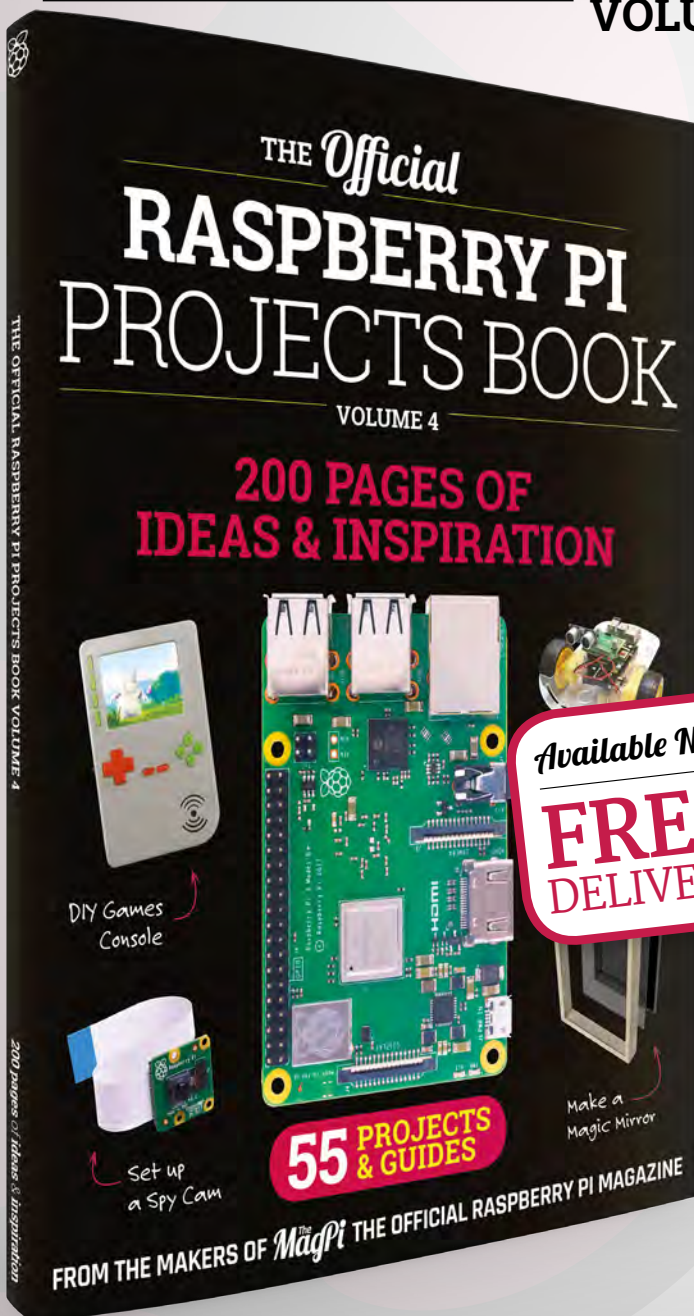
[store.rpiexpress.cc](http://store.rpiexpress.cc)

plus all good newsagents and:

WHSmith **BARNES & NOBLE**

Available on the  
App Store

GET IT ON  
Google Play

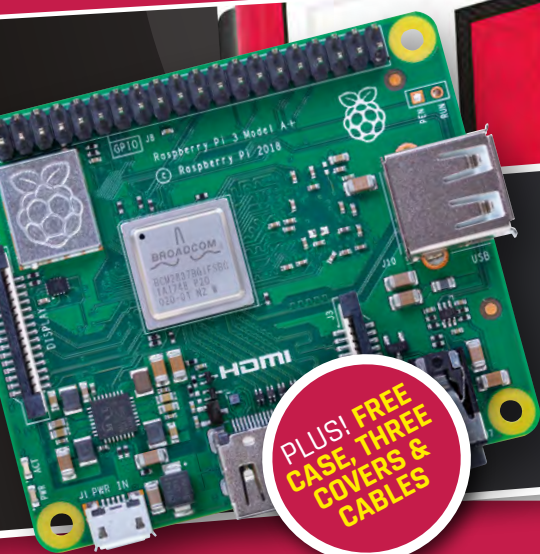


DON'T MISS THE **BRAND NEW** ISSUE!



**SUBSCRIBE FROM JUST £5**

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



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

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

\* While stocks last

**Buy online: store.rpipress.cc**



# FIELD TEST

HACK | MAKE | BUILD | CREATE

Hacker gear poked, prodded, taken apart, and investigated

PG  
114

## DIRECT FROM SHENZHEN: FUNCTION GENERATOR

Beeps, boops, and modulated  
signals in one easy kit

PG  
122

## CAN I HACK A PROJECTOR

Post your messages on a wall with  
no Blu Tack or sticky tape

## REVIEWS

124 **LumiDrive**  
APA102s and CircuitPython

126 **Sony Spresense**  
A microcontroller meets UNIX

PG  
116

## BEST OF BREED

Don't be bound by WiFi: get your makes  
on the internet at long range with these  
radio modules

128 **ESP Game Engine**  
Portable games for ESP hardware

129 **Self-Sufficiency**  
Take to the land



DIRECT FROM  
SHENZHEN

# Frequency generator module

The easy way to analogue wave forms

By Ben Everard

@ben\_everard



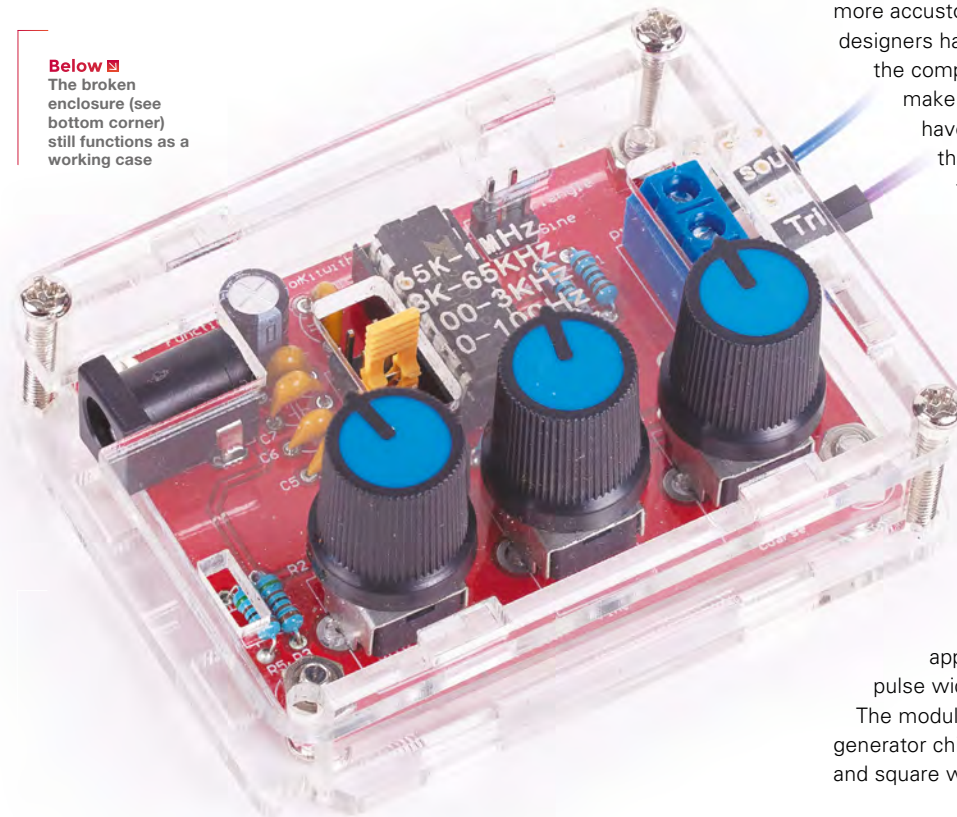
**oscillating circuits: they're useful for a whole range of things, from making music to modulating data before transmitting it across the ether.**

However, accurately creating these waves can require analogue electronics, which can induce panic in those of us more accustomed to digital circuits. Fortunately, chip designers have done the hard work and wrapped the complex bits up in easy-to-use chips. To make things even easier, module designers have taken these chips and created PCBs that do the remaining bits. All we have to do is solder it together, and twiddle the knobs to get the frequencies we want.

The module we've tested is the rather grandly titled 'XR2206 Signal Function Generator Synthesizer dds Frequency Pwm Pulse Generator Sine Gerador de Sinal Adjustable Module DIY' from the YaYa SweetLife Store on AliExpress. It cost £3.83, including delivery to the UK. Identical modules are available from many other direct-from-China sellers. Despite the name including PWM, there doesn't appear to be any ability to change the pulse widths using this module.

The module is based on the XR2206, a function generator chip capable of producing sine, triangular, and square waves at a range of frequencies. It's a

**Below** The broken enclosure (see bottom corner) still functions as a working case



fairly straightforward chip that just needs connecting to a range of RC timing circuits that control its output, and these circuits are all on the module. There's an adjustable capacitor bank, using a jumper that selects the frequency band you're working in (1–10Hz, 10–100Hz, 100–3000Hz, 3kHz–65kHz, and 65kHz–1MHz), and then two variable resistors that adjust the frequency with this band – one for course adjustment, and the other for fine-tuning, allowing you to pick your frequency fairly precisely.

**// This module is based on the XR2206, a chip capable of producing sine, triangular, and square waves**

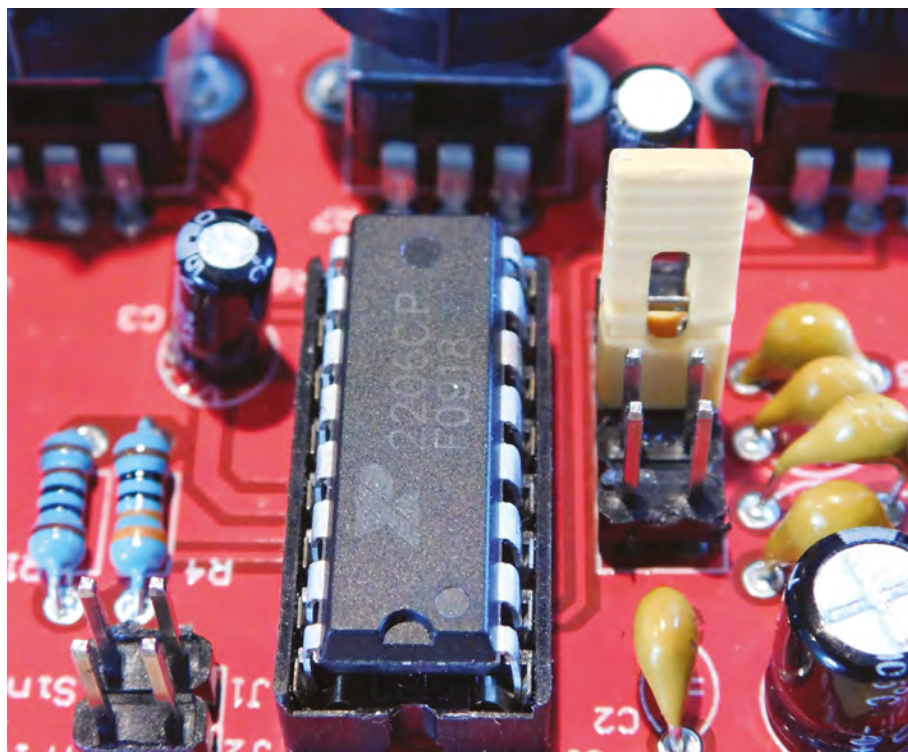
The module takes 9–18V, via a 2.1 mm jack – the same as on an Arduino Uno and many other microcontrollers. Output is via a set of three screw terminals: one for ground, one for square wave, and one selectable between triangle and sine.

The only instructions that came with the project were a list of the values for capacitors and resistors for the different locations on the board. This should be sufficient for anyone comfortable with basic electronics (such as identifying resistor values), as there are no particularly complex parts to the build.

All the components are through-hole and all easy to solder together. If you're after a cheap project for practising your soldering with, then this is certainly a decent choice (the only slight downside to this is that you'll need additional components – such as a speaker – to get any output from this).

The frequency adjusters worked well giving us the full range from infrasonic to ultrasonic and everything else along the way. The different waveforms are audibly different so you can test it out with just a speaker. Be aware that the 3V maximum output from this kit is enough to damage some headphones.

The acrylic enclosure left a little to be desired. The bolts screwed directly into the brittle acrylic, and we sheared one corner off in the process of assembling the kit. This left us with only three vertical sides on the box. A spot of glue would have held the side in place, but the enclosure was solid enough with a side missing, so this wasn't a huge problem and, for the price, we'd have been happy with just the bare board anyway.



**Above** ♦  
The beating heart of this module can be lifted out and used in other projects

### MORE FEATURES

The XR2206 has some more advanced functions that aren't broken out on this module – perhaps most usefully, it can use input voltages to adjust the amplitude or frequency, which can be used both for music production and modulating signals (such as for radio transmission). If you want to use the features of the XR2206 in larger projects, you might find yourself limited by the circuitry on this module. As so few additional components are required to make this chip work, it might be a better option to build your circuit from scratch. You may find that the cheapest way of getting all the necessary components is via one of these kits, even if you're not using the PCB and enclosure.

Overall, this kit is a fun little project – it's easy to solder together and can produce output that's useful for quite a few things. It's also an introduction to the XR2206 (which is held in a socket so can be removed for other projects if necessary). Add to this that it's a chance to practise your soldering, and it's pretty good value at £3.81, even taking into account the weak enclosure. We got the cheapest kit we could find, so we can't really complain if we got the lowest quality, but we still got a fun build and an interesting chip to begin exploring. □

DIRECT FROM SHENZHEN

ONLY THE BEST

# Long-range radio communications made easy with LoRa

Adding low-power communications to your project has never been easier

By Marc de Vinck

 @devinck

**T**his month is all about **communications**. And not just your run-of-the-mill WiFi or Bluetooth solutions – we’re talking about long-range radio communications, also known as LoRa.

LoRa is hoping to be the Internet of Things (IoT) standard, allowing multiple manufacturers to create a variety of different clients that can connect to a LoRa WAN, enabling global radio connections. Although you can also do point-to-point communications without connecting to a WAN, adding a gateway allows for true global communications.

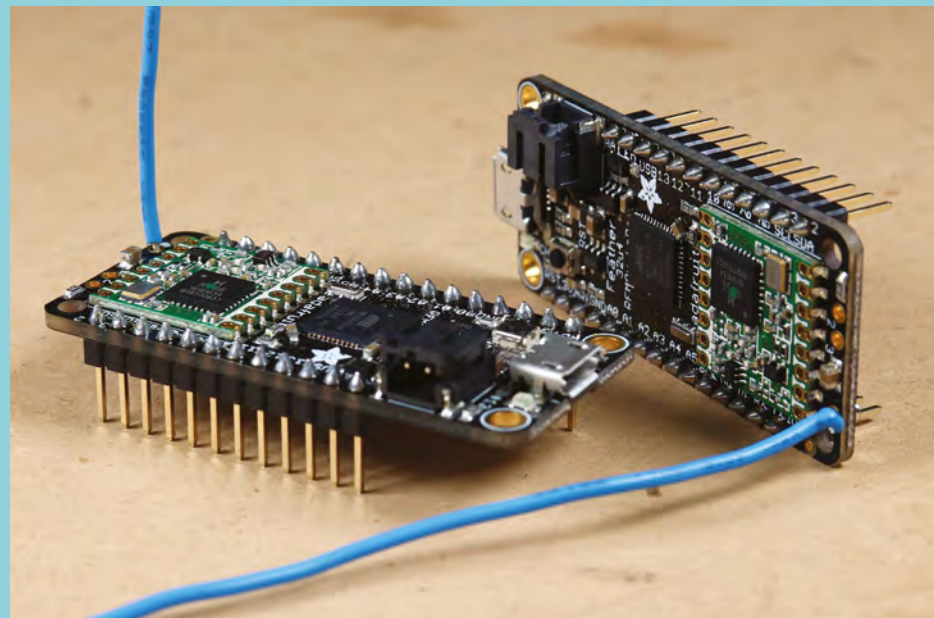
LoRa communications require very little power to operate in a low bandwidth capacity. The radio signals are transmitted in a chirped, multi-symbol format, which enables the encoding of information. You typically don’t need to know much about radio technology to easily transmit bits of data via LoRa. It’s simple, and reliable, radio communications.

Speaking of data, keep in mind that you aren’t going to get the data throughput of WiFi. Not even close to those numbers! However, you can send small bits of info over a long range with very little power, which is typically more of an issue with simple data gathering of remote sensors. Because

of the low power mode, some devices can operate for years on a small battery, and you can easily implement solar power for years of uninterrupted data transmission.

**Below** ♦

This author has created several different environmental monitoring experiments using these two Feather modules from Adafruit



# Arduino MKR WAN 1300 vs Adafruit Feather M0 RFM96 LoRa Radio - 433MHz - RadioFruit

Two ways of creating DIY weather sensors

**ARDUINO MKR WAN 1300** ◆ \$40 | [store.arduino.cc](https://store.arduino.cc)

**ADAFRUIT FEATHER M0 RFM96 LORA RADIO - 433MHZ - RADIOFRUIT** ◆ \$35 | [adafruit.com](https://adafruit.com)

**T**he Arduino team has developed a simple way to integrate LoRa communications in your next project.

The Arduino MKR WAN 1300 is based on the Atmel SAM D21, and is coupled to a Murata CMWX1ZZABZ LoRa module. Unlike other Arduino boards, you can power this one with just two AA or AAA batteries. Just like all other Arduino boards, you can use the Arduino IDE to program it with ease. It's hard to go wrong with this board since it's brought to you by the people who made the original Arduino ecosystem. □



**Left** ◆ From the creators of Arduino, the MKR WAN 1300, featuring LoRa

**Below** □ It's hard to go wrong with a Feather form factor LoRa board

## VERDICT

Arduino MKR WAN 1300

A robust board, designed by the official Arduino creators

9/10

Adafruit Feather M0 RFM96 LoRa Radio - 433MHz - RadioFruit

Packs a lot of power in a small form factor

10/10



**A**lthough the Adafruit Feather M0 RFM96 LoRa Radio microcontroller, also known as a RadioFruit, isn't an 'official' Arduino board, its innovative form factor and extensive tutorials and ecosystem make it the best choice when you want to work with Arduino and LoRa together.

The board features an ATSAM D21G18 ARM Cortex-M0 processor, clocked at 48MHz and at 3.3V logic. It also has a whopping 256kB of flash memory, and 32kB of RAM. It's also easy to connect to your computer with a built-in USB port for USB-to-Serial programming and debugging capability. And if you want to go portable, Adafruit added a connector for a 3.7V lithium polymer battery along with built-in battery charging and monitoring. □

# SparkFun Pro RF - LoRa, 915MHz

Bringing radio back down to earth

SPARKFUN ◆ \$30 | sparkfun.com

**T**he SparkFun Pro RF is a LoRa board that integrates a SAMD21 and a long-range RFM95W, to make a compact Arduino-compatible board. An interesting feature is the fact that every pin on the board is accompanied with a ground connection, which allows for hooking

up sensors and other components simply. We've all been in the middle of a build and have run out of GND connections, and SparkFun hopes to eliminate some of those pains with the Pro RF. □



**Left** ◆ A good LoRa solution for those who are willing to tinker with the code

**VERDICT**  
A good board that could use better documentation

8/10

**"** Every pin on the board is accompanied with a ground connection

## LoPy4

**"** Pick a radio, any radio

PYCOM ◆ \$40 | pycom.io



**T**he LoPy4 is a MicroPython-enabled development board that features LoRa, Sigfox, WiFi, and Bluetooth. It's the only board we're aware of that integrates all of those protocols in one package. At its core, it features the latest ESP32 controller, which is a favourite among the DIY electronics community.

The downside of this feature-rich board is, ironically, the lack of USB connectivity to a computer. You will need to purchase a separate Expansion Board 2.0, Pysense, or Pytrack to program the LoPy4. If you are programming one board, it's a costly addition, but if you are deploying dozens of these boards, it's not an issue, since you do save the additional required hardware on each LoPy4 board. □

**Above** □ A little bit of everything in this compact board

**VERDICT**  
A great choice when you need multiple communication options

7/10

## VINDUINO REMOTE SENSOR STATION BOARD

You may be wondering if people are really using LoRa commercially? The answer is a resounding yes! One of my favourite examples is the Vinduino. It's a purpose-built Arduino-compatible board that can be solar-powered, and uses LoRa technology to monitor and report soil moisture values, along with being able to control standard irrigation systems. Their goal is to save at least 25% water over an unmonitored vineyard.



# The Things Gateway

Linking to the internet

**THE THINGS** ◆ \$350 | [adafruit.com](http://adafruit.com)

**Once you have multiple LoRa sensors deployed, you will want some sort of gateway (LoRaWAN) to manage communications and connect them to the internet and, although you can roll your own, The Things Gateway is a well-designed turn-key solution.**

In addition to connecting your sensor nodes to the internet, you can allow the gateway to be used by other users, adding to the 6500+ gateways already deployed around the world. Each one of these gateways can reach up to 10km, and can scan up to eight channels simultaneously. You don't have to make it public, but the idea is to create a shared global network that benefits everyone. Since LoRa radios typically communicate at timed intervals, and transmit minimal data, sharing your gateway isn't going to slow you down. Now you can be a part of the IoT revolution, and create a LoRaWAN in your community. This model operates at 933MHz for use in the US. □



**Above** ◆  
A robust and well-designed LoRa gateway

## THE ARDUINO PRO GATEWAY

The newly announced Arduino Pro Gateway kit provides LoRa connectivity using the 868MHz radio band. It's designed around the SX1301 from Semtech, which enables a reliable connection between the gateway and your wireless end points. The Arduino Pro Gateway hasn't been released in the US yet, so we'll just have to wait to see how it all plays out before fully reviewing it. This is one device to keep on your radar if you're interested in LoRa technology.



**VERDICT**  
With high quality comes a higher price

**8**/10

# Adafruit LoRa Radio Bonnet

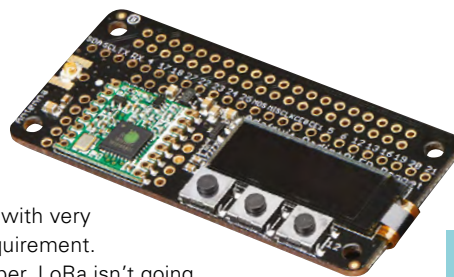
Pi in the sky

**ADAFRUIT** ◆ \$32 | [adafruit.com](http://adafruit.com)

**The newest Raspberry Pi computers feature both Bluetooth and WiFi and, thanks to Adafruit, you can easily add LoRa to your Raspberry Pi.** In addition to the bonnet being able to operate at 868MHz or 915MHz transmission/reception frequencies, it features an on-board 128x32 OLED display and three buttons. This bonnet will allow your Pi to communicate over

long distances with very little power requirement.

Just remember, LoRa isn't going to be as fast or be able to send as much data as WiFi, but good luck getting WiFi to travel up to 2km! Adafruit also offers several different variations of this bonnet, including one that operates at 433MHz, and they all come fully assembled, so you will be up and running fast. □



**Below** ◆  
LoRa and an OLED display for your Raspberry Pi

**VERDICT**  
The perfect solution for adding LoRa to your Raspberry Pi

**10**/10



50 PROJECTS FOR MAKERS & HACKERS

# BOOK OF MAKING

VOLUME 1

FLY YOUR OWN CUSTOM DRONE

BUILD AN A.I. WATER CANNON

CONTROL YOUR HOME. SMART TECH YOUR WAY

BUILD A TREBUCHET

MAKE MUSIC ON A SYNTH

& LOTS MORE

FROM THE MAKERS OF HackSpace

**£12.99**

**FREE** WORLDWIDE SHIPPING

BUY TODAY AT [STORE.RPIPRESS.CC](http://STORE.RPIPRESS.CC)



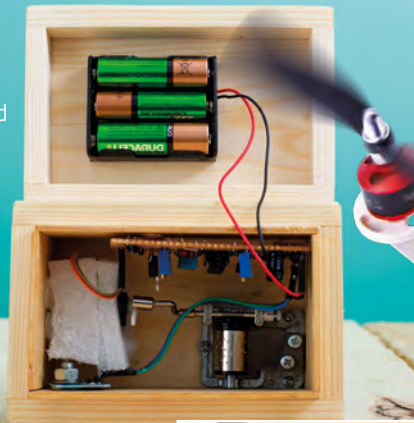
---

# THE BEST PROJECTS FROM HACKSPACE MAGAZINE

THE ULTIMATE SKILLS,  
TRICKS, AND MAKES

## MUSIC BOX

Build a touch-activated  
music box with no  
coding required



## BUILD A DRONE

The ultimate guide to making  
your own quadcopter



## LASER-CUT TURNTABLE

Create stunning 360° animated  
GIFs with this geared turntable



## Can I Hack It?

# A projector

Can we hack an LED projector to do more?



Les Pounder

@biglesp

Les Pounder loves taking things to pieces and seeing how they work. He teaches others how to be makers and tinkerers at events across the UK. He blogs at [bigl.es](http://bigl.es)

**T**his issue, we take a look at a projector that offers '1080p' resolution for around £34. It may not provide a true 1080p – rather, the image is 800×480 and can be scaled to resemble 1080p. This small projector measures 30×19.5×10.5 cm and weighs only 1.6 kg. Could we use this projector in a hack? Can we hack the projector with embedded tech? Only one way to find out – we need to take it apart!

The case is made from an easily workable plastic, but it does have variable thickness, depending on location. For example, the front panel, where the lens is located, has a thin plastic frame, and the top

and bottom of the projector have a thicker plastic for rigidity. There are five cross-head screws around the perimeter of the projector, these are self-tapping into plastic columns that connect the top of the projector to the base. Inside the projector we see three sections: the low-voltage electronics, the path that light takes to the lens, and the high-voltage area. The low-voltage area is located to one side, the light path in the centre, and the high-voltage on the other side. This limits our access somewhat, as the light path area is a big, open space, and it requires a clear path, otherwise our projection will have shadows. The light path starts in the low-voltage area, where our image is generated and shone via an ultra-bright LED that requires a 'squirrel cage' fan to keep it cool. This is then bounced off a mirror and then through the main lens, which uses a screw mechanism to enable the user to create a manual focus. An interesting mechanism is the keystone-effect correction dial. This provides correction for images that appear distorted due to the angle at which the projector is being used. Take a look at this simple yet brilliant mechanism, above right.

First, we shall address the high-voltage electronics. There is a direct connection to mains voltage so here we say as always: be extremely



**Above** It may just look like a typical projector, but this cost-effective product has plenty of scope for hacks!

### YOU'LL NEED

Apeman LED Video Projector LC250


### COST

£34 (approx.)

### WHERE

[hsmag.cc/SeDpRt](http://hsmag.cc/SeDpRt)



**Above**  The mechanism to control the perspective of the image is beautiful in its simplicity

careful with mains voltage, it can kill you! If you have never worked with mains voltage, seek the skills and advice of someone who has. From the figure 8 power input, we have a simple switch that makes or breaks the circuit. This is connected to a specially designed board that provides power for the LED light source, and provides power for the low-voltage electronics. There is a single mainboard for the low-voltage electronics, but from this mainboard there are two extensions. The first is for the control panel and is a simple keypad breakout with domed 'snap' buttons that provide tactile feedback. The circuit tracks are clear to see, and we can even interface with them as they operate at 3.3V, so connection to a Raspberry Pi is possible. The projector uses a shaped extension cable to connect to the mainboard and prevent reverse connection issues. The second extension is for VGA input, which is located deep in the chassis. This extension uses a flat flex cable, similar to that used for the Raspberry Pi camera to connect. The connector has a locking bar across the flat flex, and care is required to gently remove it. On the edge of the mainboard are the various inputs for HDMI and composite video for which an adapter is included in the box. These connections are accessible from the inside of the unit if your soldering skill is sufficient.


The mainboard can provide a 5V output to power a device and, using a UNI-T UT658 USB tester and a 1 A / 2 A load tester, we found that the starting voltage with no load was 5.12V, but as soon as we tried to pull 1 A we found that the supply went down to 4.98V at 900mA. This would just power a Raspberry Pi Zero W. We then tried to pull a 2 A load, and the voltage dropped to 4.95V, but was able to supply 1.76 A. So, we have a power source that can provide a modest amount of power for our hacks.

## LET THERE BE LIGHT

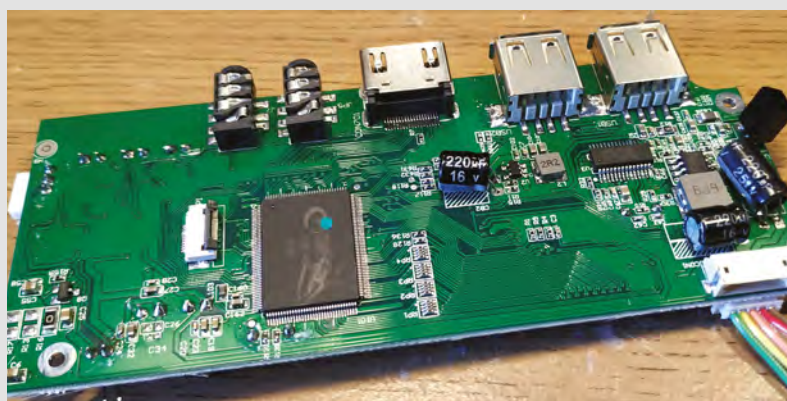
Projectors measure their power using 'lumen', which is a measure of the amount of light that can be output. For example, this projector has a 2200 lumen output via an LED bulb, which provides around 24 W of light output. This is why the projector requires a dark room, as the light source is too weak for general daylight use. Older projectors would use an incandescent bulb, similar to a light bulb, which was not energy-efficient and had a much shorter lifespan than newer LED bulbs. This also brings us on to this projector's bulb lifespan – there is a claimed 45,000 hours of life per bulb. This equates to 1875 continuous days of use, around five years. But, what happens when the bulb finally dies? Well, it can be replaced with another ultra-bright LED light source, as the LED is just a simple LED. A search of eBay is all you need to do when replacing the bulb, or perhaps you can hack your own solution?

## HOW HACKABLE?

This is a tough call. Internally there is limited space, but with something like a Pi Zero W, we can just squeeze a little extra tech. Tapping into the 5V from the USB port is possible, as is the HDMI connection, but that will require patience and skill with a soldering iron. If we were to do this, we could create a projector with built-in wireless interface, even taking this further to map the control panel to GPIO pins that we can remotely control from a mobile device/laptop. But how can we get media on to the projector? Well, using a Samba or NFS file server on the Pi Zero W, and over the WiFi network, we can send our videos and control them from one app.

This is a great projector that offers a cost-effective solution for small projects. Sure, it's not a real '1080p' projector, but then we are only paying around £34 for it, versus projectors that can cost significantly more. The light source for this projector is rather weak, and it requires a dark room for best effect. Use this projector as a learning tool, understand how the tech works and, when ready, purchase something that will give you better quality and results. Happy Hacking! 

**Below**  The mainboard is compact and contains some connections that we can hack. The USB ports can be used to power any internal hacks we create



# LumiDrive LED Driver

Python control for LEDs

SPARKFUN ♦ \$19.95 | Sparkfun.com

By Ben Everard

🐦 @ben\_everard

**T**he LumiDrive is a board for controlling APA102 (aka DotStar) RGB LEDs with CircuitPython.

It's powered by a SAMD21G-AU microcontroller, and has a built-in LiPo battery charger.

Slightly unusually, this board has USB-C for power. While this probably is the future, we're currently stuck in the past. We have drawers full of micro USB connectors, and precious few USB-C. This isn't a good reason to cling to the past, and in a few years we'll probably be complaining about antiquated micro USB connections. Alas, times move on and we'll begrudgingly accept that USB-C is probably a good choice now. This power passes through to the LEDs, so providing you're not using too many,

you won't need an additional power supply. Not too many is an annoyingly vague point, but fortunately, SparkFun has a handy chart for power draw of APA102s under different conditions. You can see the results here: [hsmag.cc/NJcfuF](https://hsmag.cc/NJcfuF). For example, you can power 100 LEDs on the rainbow cycle at 50% brightness and expect to draw under 1 amp, which should be possible from many USB power sources. However, if you want to go full brightness and display white, you'll get this current with about 20 LEDs.

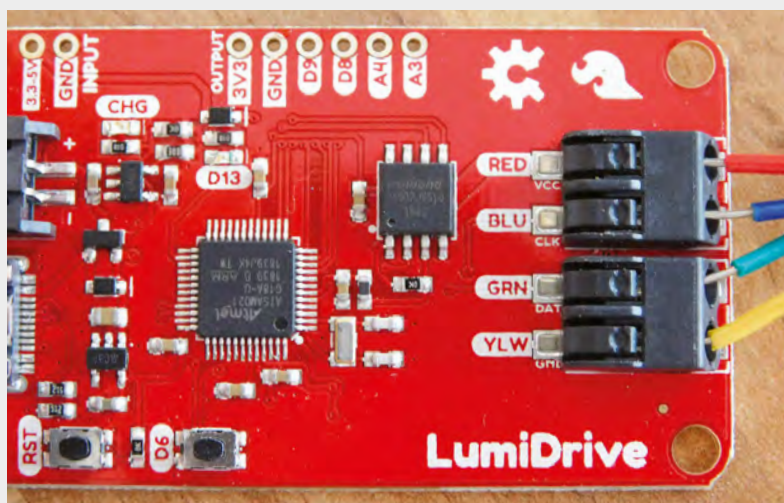
Unfortunately, the amount of power you can get from a USB port is a bit vague. Officially, on computer ports, USB 2.0 should support 0.5A while USB 3.1 (which has a blue tongue) should support up to 0.9A. In practice, most will supply more than this without any problems, but it's up to you whether you want to risk it, as it could damage your computer. On wall warts, you should see the current rating written on the adaptor, and you shouldn't exceed this.

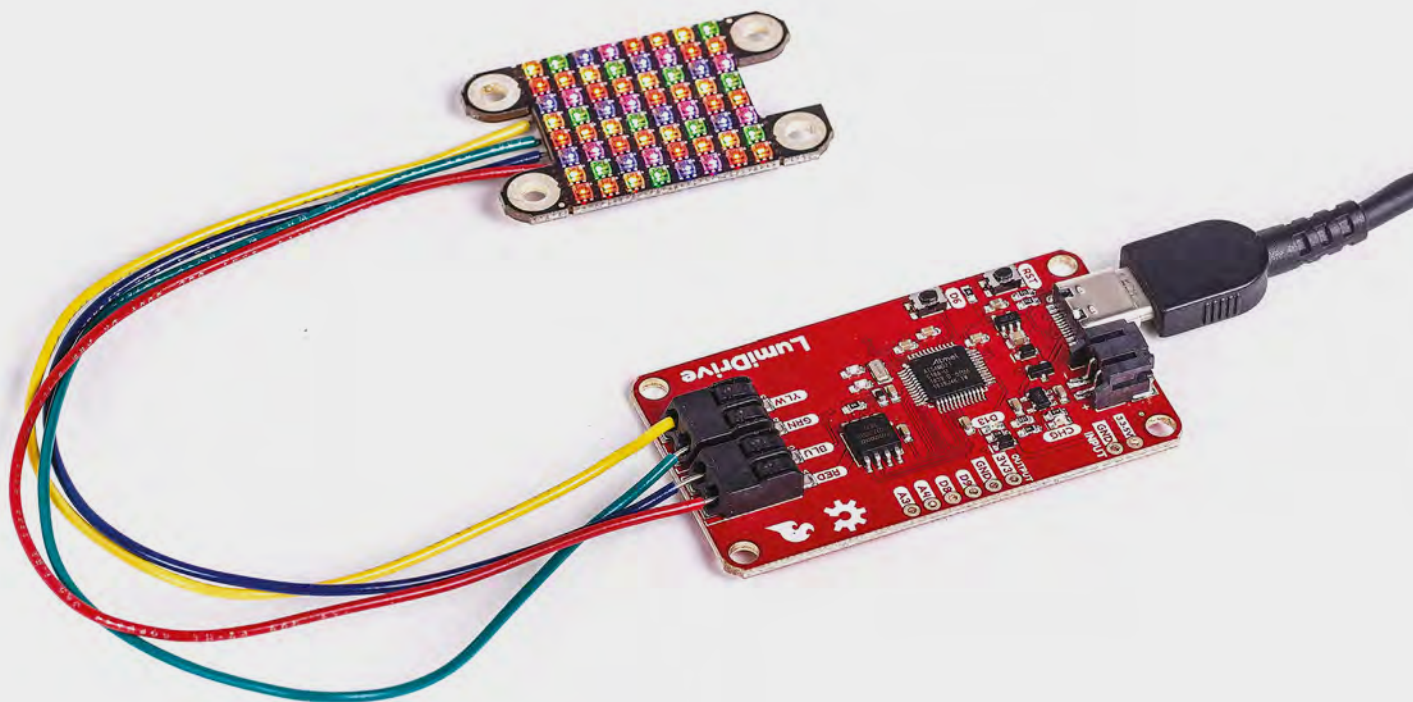
If in doubt, make the LEDs extra dim (don't be afraid to go down to 10% brightness when testing), and be conservative. You can always measure the current and boost the brightness later, but it's much harder to fix a broken USB port.

The APA102 LEDs that this board controls are chainable, so you can add almost as many as you like to the four outputs on the board (power management gets difficult when you've got lots, and you'll run out of memory eventually). There are poke-home connectors that are secured by pushing the wire in while pressing down on a button on the top. These provide a connection that's secure enough to use in projects, so there's no need to solder,

#### Below ♦

The connectors are labelled with both the function and the common colour of wire to make it easy to hook up





even for long-term projects. We're really impressed with how well these work – the wires don't come out accidentally, even with a concerted yank, and the button at the top is hard to press accidentally. This makes the board really easy to get started, prototype, and change around.

Of course, there are bits you can solder on to – there's two digital GPIO pins and two analogue inputs. This isn't a huge number, but it's enough to provide a little interactivity to a light-up project.

There's a button on the board connected to the D6 pin, so you can access the status of this in your code. It is a little close to the reset button, so you have to be careful to make sure you press the correct one. If this is a significant problem for your project, you could cover up the reset button, or wire another button on one of the other GPIO pins.

SparkFun makes a range of Lumi-branded LED boards, including a LuMini matrix (pictured) and LuMini LED rings. However, any APA102 products should work. You can get them in strips and other configurations from a range of manufacturers. They're not quite as common as WS2812 LEDs, and they are more expensive, but they have a high data rate so it's possible to drive them much faster.

### INTERACTIVE SOFTWARE

This board comes set up with CircuitPython, and there's currently no ability to program it in another language. There's a guide to getting started with the language on this board at: [hsmag.cc/NJcfuF](https://hsmag.cc/NJcfuF). CircuitPython is a great language for getting started,

and it allows you to really quickly add some interactive lights to your build. However, complex animations can require quite a bit of processing and, while the processor on this board is no slouch, you'll struggle to keep complex animations running smoothly with this language. Compiled languages (such as Arduino C++) are much faster, but at the moment, there's no option to use these on this board.

“

**This is a really well thought-out product, and an example of 'do one thing well'**

”

This is a really well thought-out product, and an example of 'do one thing well'. There are plenty of limitations on this board – for example, it only controls one type of LED, and there aren't many GPIOs. By not catering to everyone, SparkFun has made a board that suits a small subset of people really well. You can get up and running without soldering anything. The focus on Python, as the language of choice, will make it easier for beginners to get started. The on-board button means you don't have to worry about circuitry (even simple circuitry) to get some input. If you're an experienced maker, these might seem like trivial things, but to some people they're not. □

**Above** □ Any APA102 LEDs will work, but the LuMini matrix has 64 tiny (2020) LEDs, which are great for high-detail animations

### VERDICT

A great board that makes it easy to get started with APA102 LED projects.

9/10

# Sony Spresense

Hexa-core, GPS-enabled, AI-ready microcontroller

SONY ◆ From £49.19 | [sony-semicon.co.jp](http://sony-semicon.co.jp)

By Ben Everard

 @ben\_everard

**T**here are three Spresense parts from Sony. A base board (£49.19) that gives you a system on a chip (with GPS built-in), 17 digital GPIOs, two analogue inputs, and four LEDs. You can plug this into the expansion board (£32.39) which then gives you an additional 12 digital I/O pins, four analogue inputs, and a headphones connector. There's also a camera (£35.88) that slots into a connector on the base board.

The brains behind the base board is a six-core Cortex-M4F processor running at 156MHz. For anyone familiar with ARM processors, this will sound like a microcontroller – the M-series is found in boards such as the Adafruit Grand Central and Arduino Due, whereas the A-series is the type of ARM core found in more general-purpose computers such as the Raspberry Pi.

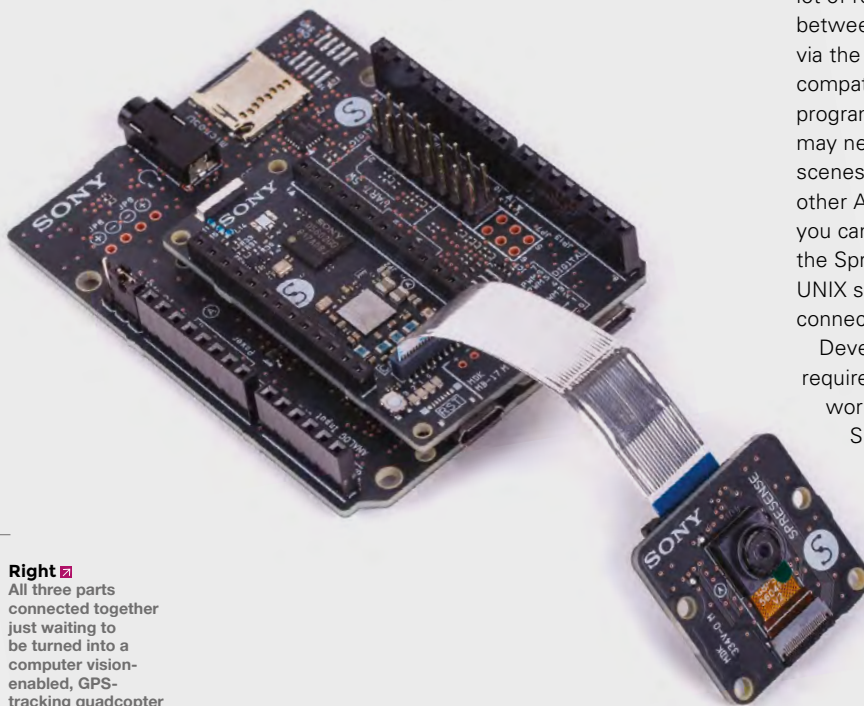
## A BIT OF THIS, A BIT OF THAT


This board, however, with its six cores has quite a lot of resources to manage, and sits somewhere between the two. Although it can be programmed via the Arduino IDE, it runs a stripped-down UNIX-compatible operating system called NuttX. If you program the Spresense using the Arduino IDE, you may never know that there's this OS behind the scenes managing things, as it works just as any other Arduino-compatible board does. However, you can delve right down into its internals using the Spresense SDK. You can even connect to the UNIX shell-like command-line interface over a serial connection, and interact with the board this way.

Development via the Spresense SDK officially requires Linux; however, we were able to get it working on Windows 10 using the Windows Subsystem for Linux. It's not a particularly straightforward process, though, and probably not worth it for casual users.

It does, however, let you unlock more performance due to the multi-core architecture.

Although the Spresense has six cores, only one will run your application. The remaining five can be used to run specific, additional tasks. Sony provides



**Right**  All three parts connected together just waiting to be turned into a computer vision-enabled, GPS-tracking quadcopter

some code for running audio code on these cores from within your Arduino sketches. You can find more details of these at: [hsmag.cc/luPaPI](http://hsmag.cc/luPaPI).

Sony supplies a set of Arduino libraries to help you get the best out of this board in a few areas, as well as audio. The DNNRT library allows you to use deep neural networks (DNN) created using the Neural Network Console – a training tool that lets you configure and train your neural networks on your main computer before transferring this trained brain to your Spresense. While this isn't completely straightforward, it doesn't require a lot of programming skill, just the knowledge of how to configure a network.

### BONUS HARDWARE

The module has built-in satellite positioning that uses GPS, GLONASS, and QZSS satellite positioning systems. We found that the antenna wasn't particularly sensitive and failed to get a fix inside a location our phone could get a fix at. That said, you're unlikely to be relying on satellite positioning inside.

The base board includes a camera connector, and Sony has released an official camera module. By microcontroller standards, this camera – at 5 megapixels – is really high resolution. While you might be used to cameras with far more pixels on your phone or Raspberry Pi, these cameras are rarely compatible with microcontrollers and the high data rate needed to operate such a camera. When combined with the AI capabilities on this board, this has the potential to add some really powerful vision capabilities to your makes.

While the Spresense is a very capable board, it's not a beginner board – its unusual hardware means that it works a bit differently to most microcontrollers. The libraries that unlock the power of this board are well-documented and do make it reasonably straightforward to use this particular hardware, but it is a bit different from other Arduino-compatible boards.

There are a few oddly disparate areas where the Spresense can really stand out. The ability to play or record sound in the background – without taking up processing time on the main CPU – could be a huge boon. The hi-res camera can be really useful,

as can the AI capabilities. Looking at these, they read like a set of scenarios more suited to small Linux-capable computers, such as the Raspberry Pi, than microcontrollers. Perhaps this isn't surprising, since the Spresense runs a UNIX-like OS – though one that much more stripped down than Linux.

The Spresense is more powerful than most microcontrollers (particularly when the extra cores are taken into consideration), yet it boots in under a second and runs real-time code, so can be used for precise hardware control. While it's not for every project, the particular capabilities of the Spresense make it an excellent addition to the arsenal of hardware available to makers. □



**Above** ♦  
The base board itself packs a huge amount of power into a very small footprint

### VERDICT

An unusual microcontroller with great potential for AI and vision projects.

9/10

# Esp little game engine

Build your own handheld games console

CORAX89 ◆ Free | [hsmag.cc/cRGMjj](https://hsmag.cc/cRGMjj)

By Ben Everard

@ben\_everard

**T**he ESP8266 is probably best known for its WiFi capabilities, but even for projects that don't need this, it's still cheap, and packs a bit of processing punch. Enough for some basic gaming, for example. GitHub user corax89

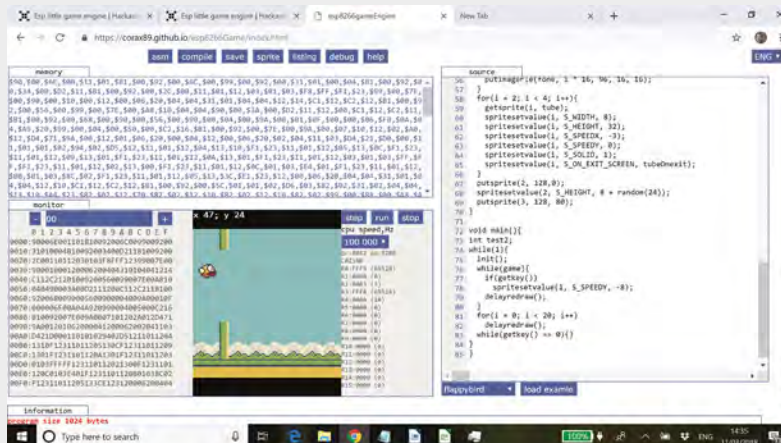
has catered exactly to that use with the ESP little game engine. It's a web-based C++ IDE for building simple games to run on ESP8266 boards with SPI TFT screens. Perhaps the most impressive part of this project is that it includes an emulator, so you can see how your code will run, and even play your games, without uploading onto the hardware. Once your game is complete, you can download a BIN file, and upload it to your ESP8266 for playing on the go.

There isn't a lot of documentation to help you get started, but there are nine examples, which range from a basic 'Hello World', to a version of Flappy

**Any child of the 1980s knows, you don't need good graphics to have a good game**

Bird, which should take you through most of what you need to know about developing games using this engine.

The IDE itself is quite stripped down. There's a box for entering the code, and once you've made any changes you need, hit Compile. This populates the 'memory' area of the screen with the HEX values of the memory. Hit Run, and your game will start playing, and you can use the keyboard to interact with it. The most useful part – from a programmer's perspective – of this virtual machine



is that it's far easier to see what's going on than when running on real hardware. You'll see that the 'currently executing' line is highlighted in the source box, and if you open the debug window, you can also see the global variables change in real time. You can also see the bounding boxes of any collisions when they take place. Both are features that will make it much easier to get your code to behave properly.

The Sprite editor makes it easy to draw your own game graphics. The ESP8266 isn't powerful enough for anything too fancy, but as any child of the 1980s knows, you don't need good graphics to have a good game.

Perhaps the most obvious missing feature is the ability to save code – the Save button downloads the compiled object code. This isn't as big a problem as it sounds, as the games are usually short (Flappy Bird, for example, fits into just 85 lines), and can be copied and pasted into another text editor for saving.

A few graphical niceties, such as syntax highlighting, would be a nice edition, but even without these, it's a fantastic microcontroller games development environment. What's more, you don't even have to wait for the hardware to arrive before you start tinkering. □

**Above** ◆ As well as Flappy Bird, there are also examples of Asteroids, a vertical scrolling space game, a platformer, snake, and a maze generator

## VERDICT

A great games engine to build software to run on cheap, homemade games consoles

9/10



# The New Complete Book of Self-Sufficiency

John Seymour ♦ £25 | [bit.ly/2EUWaE1](https://bit.ly/2EUWaE1)

By Richard Smedley

@RichardSmedley

**It's with some gratitude that this reviewer turns to review this new edition of Seymour's self-sufficiency classic, as all of his previous copies have been lent out, and never came back – yes, it's one of those books; too useful to let out of your sight, yet so inspiring you want to share it with everyone you think would benefit.**

*The Fat of the Land*, published in 1961, established Seymour's reputation as a self-

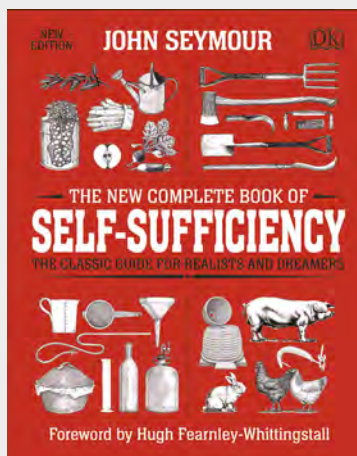
sufficiency guru, but it was the first edition of this book in the 1970s that got a generation dreaming of getting 'back to the land' and, for those who went beyond dreaming, gave them the tools to succeed.

Seymour gives practical advice on supplying food and energy, as well as teaching practical craft skills for a self-sufficient life, like the remarkable one that he himself led. As well as one-acre and five-acre plots, allotments and gardens are planned, and all the information you need – even if you have never really gardened before – is at your fingertips.

Clear illustrations and diagrams will help you trench asparagus, turn a bowl on a lathe, and thin your fruit crop for bigger, sweeter yields. The breadth of coverage is awe-inspiring – a whole village-worth of traditional skills, backed by experience and intelligent observation.

A warning to the squeamish: Seymour is not a vegan and so, not being a hypocrite, he reared and butchered his own animals. If you don't like reading about this, you'll have a few pages to skip. *The Self-Sufficient Gardener* (1978), if you can find it second-hand, covers half the material in this book but, as well as dropping the beef and lamb sections, you'll miss out on everything from basketry to building your own furnace, via thatching, and even spinning.

This is an uncompromising and comprehensive guide to just about every aspect of living off-grid and, as such, also provides plenty of inspiration to makers wishing to take back control of parts of their lives. But the lucid and engaging writing, and cornucopia of gardening tips, make it a great read for almost anyone. Buy a second copy, as you're sure to be lending it to someone! □



## VERDICT

Think global, act local, with the skills and knowledge for off-grid abundance.

9/10

next month

issue

#18

ON SALE  
18 APRIL

FEATURING

SPACE

ALSO

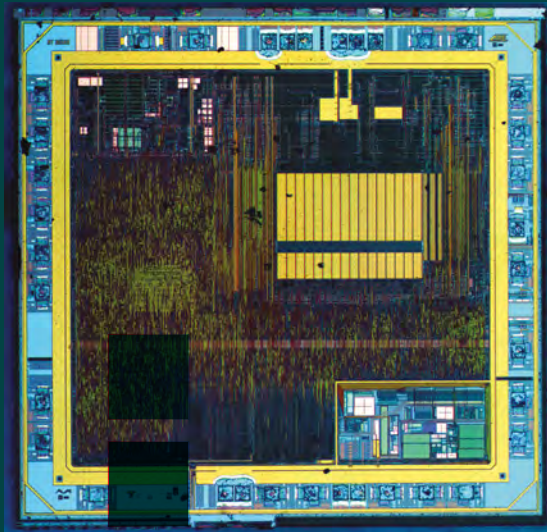
- SATELLITES
- THE FINAL FRONTIER
- THE DEPTHS OF THE UNIVERSE
- THE SKIES ABOVE US
- AND MUCH MORE

DON'T MISS OUT

[hsmag.cc/subscribe](https://hsmag.cc/subscribe)

# HackSpace

TECHNOLOGY IN YOUR HANDS



## ATMEGA8

The ATmega8 is a cheap, low-power, Arduino-compatible microcontroller shown here after it's been boiled in acid to expose the silicon die and put under a microscope.

— [hsmag.cc](http://hsmag.cc) —

