

NEW

HACK | MAKE | BUILD | CREATE

HackSpace

TECHNOLOGY IN YOUR HANDS

hsmag.cc

January 2018

Issue #02

LASER CUTTING

Make flashing lights with
flashing light

CHOOSE A PRINTER

USE THE RIGHT FILAMENT

MUSIC BOX

When clockwork
meets electronics

BAD DOG DESIGNS

Transforming junk to Nixie clocks



MASTER SOFTWARE

MAKE A KNIFE

Unleash
Iron Age
technology

TROUBLE- SHOOT ISSUES

RUBBER BANDS

Build a thermodynamic engine



SINE WAVES

CHEESE MAKING

DIGITAL BRAILLE

CROCHET

CanaKit Raspberry Pi 3 Ultimate Starter Kit

Model B | 1 GB RAM | 1.2 GHz | Quad-Core CPU



- > Learn to Code
- > Explore Computing
- > Get started with Electronics

KIT INCLUDES RASPBERRY PI 3 AND ...

PREMIUM CASE &
HEAT SINKS



2.5A
POWER ADAPTER



32 GB CLASS 10
MICROSD CARD



PRE-LOADED
WITH OPERATING SYSTEM

USB MICROSD
CARD READER



PREMIUM
HDMI CABLE



QUICK-START GUIDE



GPIO TO BREADBOARD
INTERFACE BOARD



RIBBON CABLE



FULL-SIZE
BREADBOARD



JUMPERS



MALE TO MALE &
MALE TO FEMALE

LEDs



RESISTORS &
PUSH-BUTTONS



Available for worldwide shipping at:

WWW.CANAKIT.COM



Raspberry Pi Zero W

Available at CanaKit!



ELECTRONIC KITS • ELECTRONIC PARTS • RASPBERRY PI • ARDUINO



Welcome to HackSpace magazine

Automatically fabricating 3D objects has been the stuff of science fiction almost as long as there has been science fiction, and now – thanks to 3D printing – it's slowly becoming real.



There have been some fantastic advancements in the field in recent years and it's more accessible than it's ever been. Here at HackSpace magazine, we're fantastically excited about the new possibilities that 3D printing is opening up, such as small-scale manufacturing, hyper-customisable products and an open-source approach to physical things.

This technology is still in its infancy and it's too early to say for sure what direction this will take, but we do know that it's up to hackers like you to determine the future. That's why we've picked 3D printing for this month's cover feature and that's why we'll continue to feature great 3D printing content.

BEN EVERARD

Editor ben.everard@raspberrypi.org

GET IN TOUCH

- hackspace@raspberrypi.org
- <https://www.facebook.com/hackspacemag>
- <https://www.instagram.com/hackspacemag/>

ONLINE

- hsmag.cc



PAGE **50**
SUBSCRIBE TODAY

EDITORIAL

Editor

Ben Everard
ben.everard@raspberrypi.org

Features Editor

Andrew Gregory
andrew.gregory@raspberrypi.org

Sub Editors

Jem Roberts, Nicola King

DESIGN

Critical Media

criticalmedia.co.uk

Head of Design

Dougal Matthews

Designer

Lee Allen

Photography

Brian O'Halloran

Illustrations

Sam Alder

CONTRIBUTORS

Lucy Rogers, Andrew Huang, Cameron Coward, Jenny List, Liam Smyth, Ed Rogers, Mayank Sharma, Will Kalif, John Wargo, Eric Coates, Sophie Wong, John Teel, Andy Clark, Inderpreet Singh, Graham Morrison, Martin O'Hanlon, Phil King, Richard Smedley

PUBLISHING

Publishing Director:

Russell Barnes
russell@raspberrypi.org

DISTRIBUTION

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

SUBSCRIPTIONS

Select Publisher Services
Ltd, PO Box 6337, BH1 9EH
 +44 (0)1202 586 848

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

06 SPARK

- 06 Top Projects**
Brilliant builds to inspire and invigorate
- 14 Manchester Robots**
Explore 500 years of android history
- 18 Meet the Maker**
Paul Parry of Bad Dog Designs
- 22 Columns**
Learn to stop worrying and embrace failure
- 24 Letters**
In which you, the public, demand more robots
- 25 Crowdfunding now**
Gadgets to help you work better, smarter, faster
- 26 Hackspace of the month**
Say hello to Birmingham's fizzPOP makerspace

31 LENS

- 32 3D Printing**
What it is, why you should do it and how to get started
- 40 Badgelife**
The unlikely hacker culture art form
- 46 We Learn Crochet**
A new old skill with the help of Swansea Hackspace
- 52 Canute**
A massive step forward for Braille readers
- 56 Interview Hacker House**
Drones, cybersecurity, and baby monitors
- 62 Improviser's Toolbox** Rubber bands
Who knew you could build a fridge out of rubber bands?



18



86





40

69

FORGE

- 70 School of Making** Make a knife
Craft metal tools without any expensive specialised kit
- 74 School of Making** Laser-cut LED clock
Bring designs to life with rasters and vectors
- 78 School of Making** Digital data – Arduino
How to handle digital inputs on your microcontroller
- 82 School of Making** Sine wave Stylophone
Create musical sine waves, then write a top-ten single
- 86 LED goggles**
Embellish a pair of goggles with some Adafruit NeoPixels
- 92 Custom Arduino**
Arduino not doing enough for you? Design your own!
- 98 Music box**
Resurrect old clockwork tech with a pair of 555 timers
- 102 Build a cheese press**
Turn milk, time, and pressure into cheesy goodness
- 106 OpenHAB 2**
Set up an open-source home automation system



52



56



26

113

FIELD TEST

- 114 Review** Hero101 3D printer
The ultimate in affordable printing?
- 116 Best of Breed** Arduino IDEs
Find the best coding environment for you
- 120 Battle Tested** K40 laser cutter
The K40 laser cutter, tested to destruction
- 124 AIY Vision Kit**
Google shares its latest AI trinket with us
- 125 Raspberry Pi Power Switch**
Turn your Pi on and off remotely
- 126 Adafruit Grid-EYE**
A thermal camera on a tiny board
- 128 Bearables**
Programmable fun for kids
- 129 Books: The Case for Working with your Hands**
It's not just fun: turns out that making things is good for you!



128



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.



Wimborne Model Railway

By Terry Coles

 @WimborneModTown

The modern use of the word 'hack' comes from the model railway world (specifically the Tech Model Railway Club at MIT). It is, therefore, perhaps appropriate for HackSpace magazine to feature a model railway. In an ongoing push towards greater realism, the Wimborne Model Railway has received a new lighting model courtesy of a Raspberry Pi Zero controlling a system of LEDs. The lighting system had to be able to simulate a day-and-night cycle, showing the house and street lights gradually coming on as the sun goes down, and going off again in the morning.

A major requirement of the project is that it could be both developed and operated by volunteers. The team selected the Pi Zero to run the system, as this made it easy for developers to work on the system. To make it easy to work in an embedded system, the Pi runs on the piCore distribution, which is a stripped-down Linux that runs entirely in memory. As it boots up, the root file system is loaded into RAM rather than running off the SD card.

The LEDs are controlled by a combination of relays and MOSFETs. Although off-the-shelf modules were used, the driver boards came with IRF520, which don't work particularly well with the 3.3V GPIO pins on the Pi, so the team de-soldered these and replaced them with IRLZ44NPBF MOSFETs, which worked better.

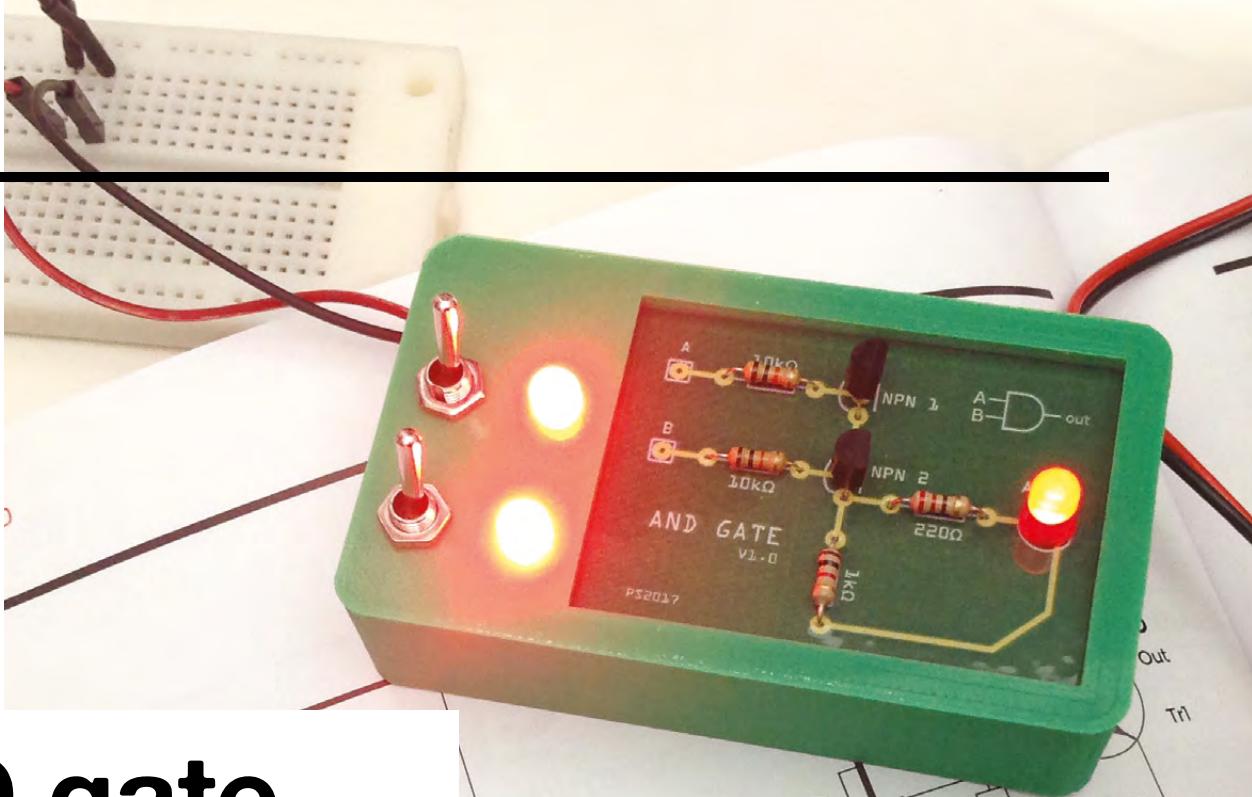
You can see all this operating for yourself as the 00-gauge railway is open for visitors at Wimborne Model Town and Gardens in Dorset, UK. □



Right □

We don't want to alarm the people of Wimborne, but that police box doesn't bode well





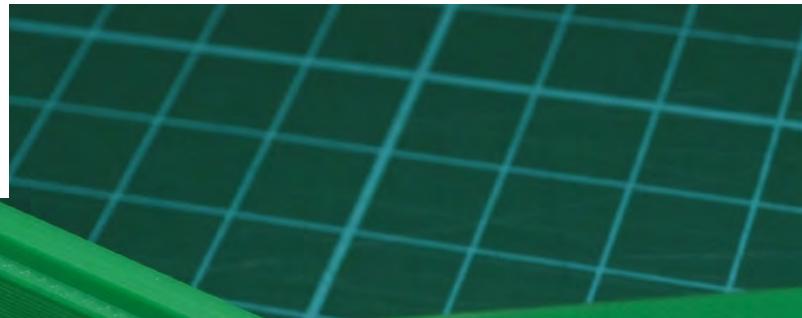
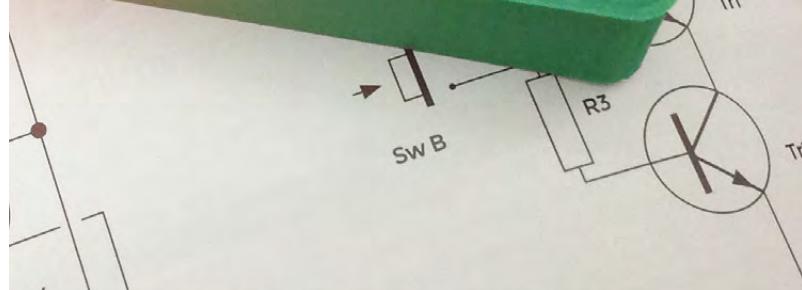
AND gate

By Paul Smith

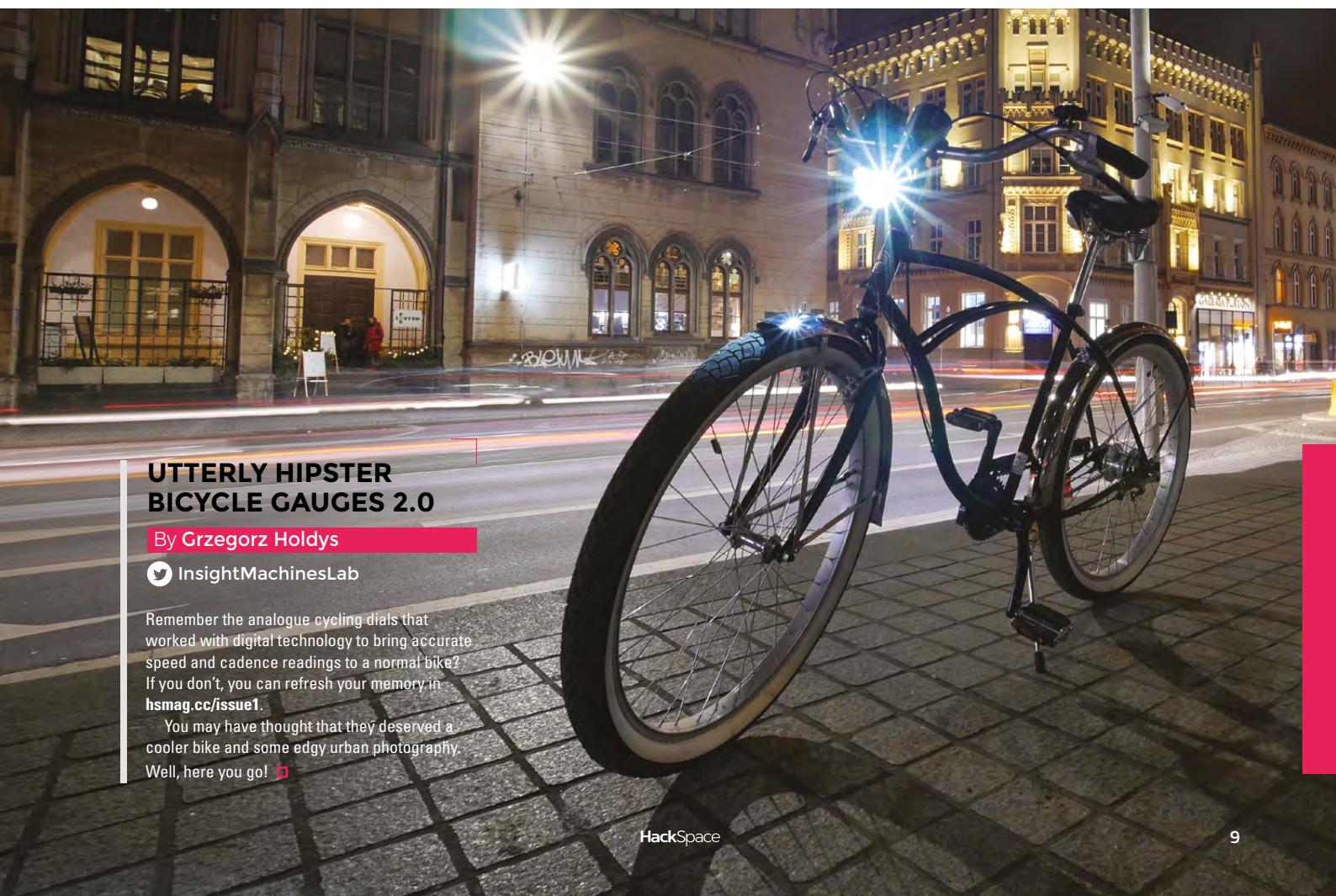
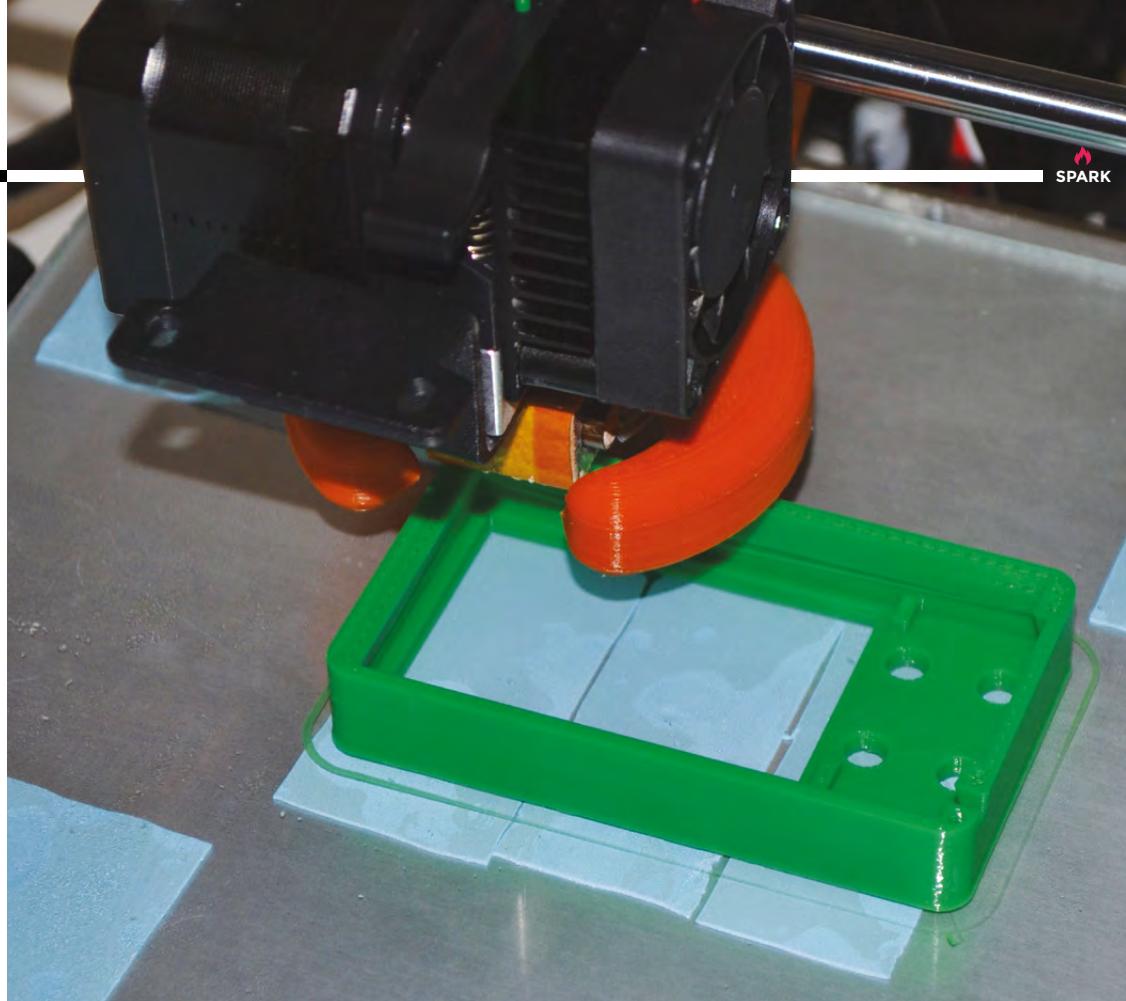
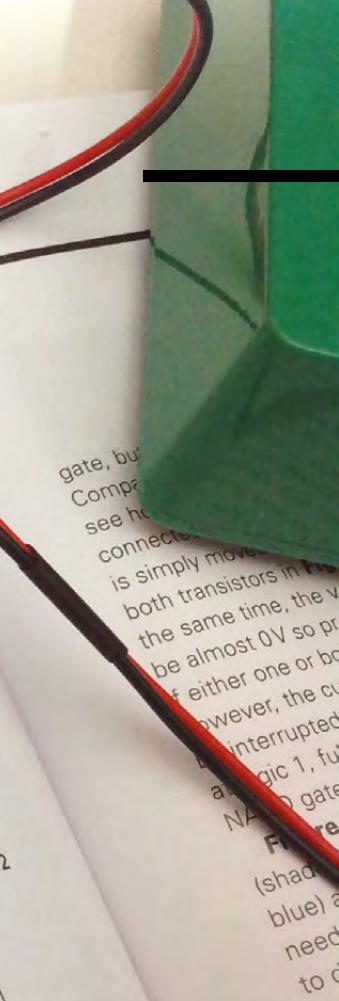
 @NncPSmith

We featured this logic gate in issue 1 of HackSpace magazine – you can read all about it here: hsmag.cc/issue1.

It's one of the fundamental building blocks of making a computer, and you can make one yourself with just some switches, wire, and a few cheap components. □



Right  The same enclosure could be used to create any other two-input logic gate



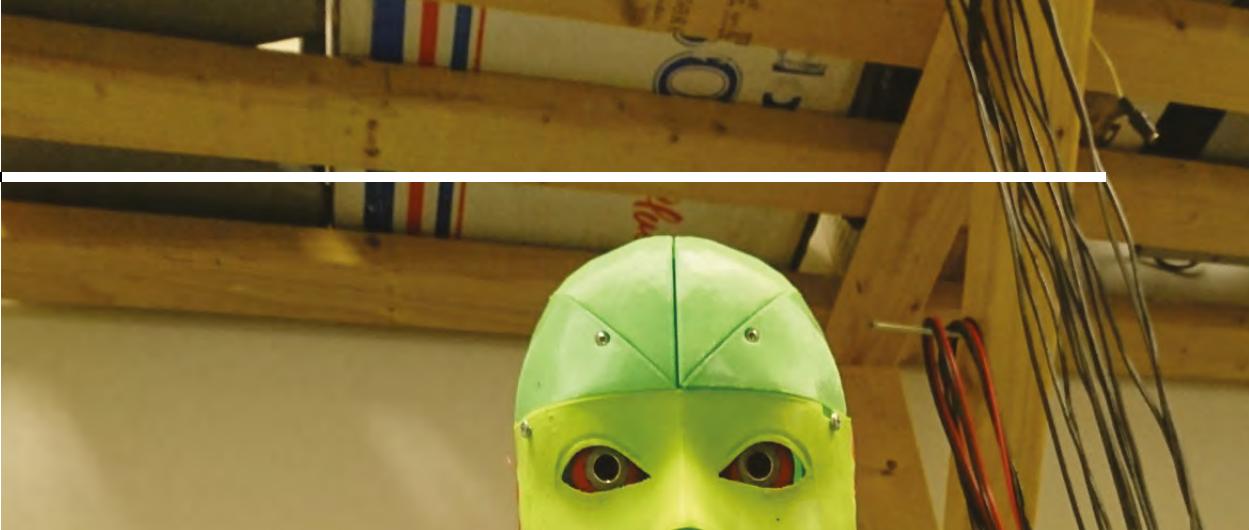
UTTERLY HIPSTER BICYCLE GAUGES 2.0

By Grzegorz Holdys

InsightMachinesLab

Remember the analogue cycling dials that worked with digital technology to bring accurate speed and cadence readings to a normal bike? If you don't, you can refresh your memory in hsmag.cc/issue1.

You may have thought that they deserved a cooler bike and some edgy urban photography. Well, here you go! □



Kermit

By Muhammad Badr

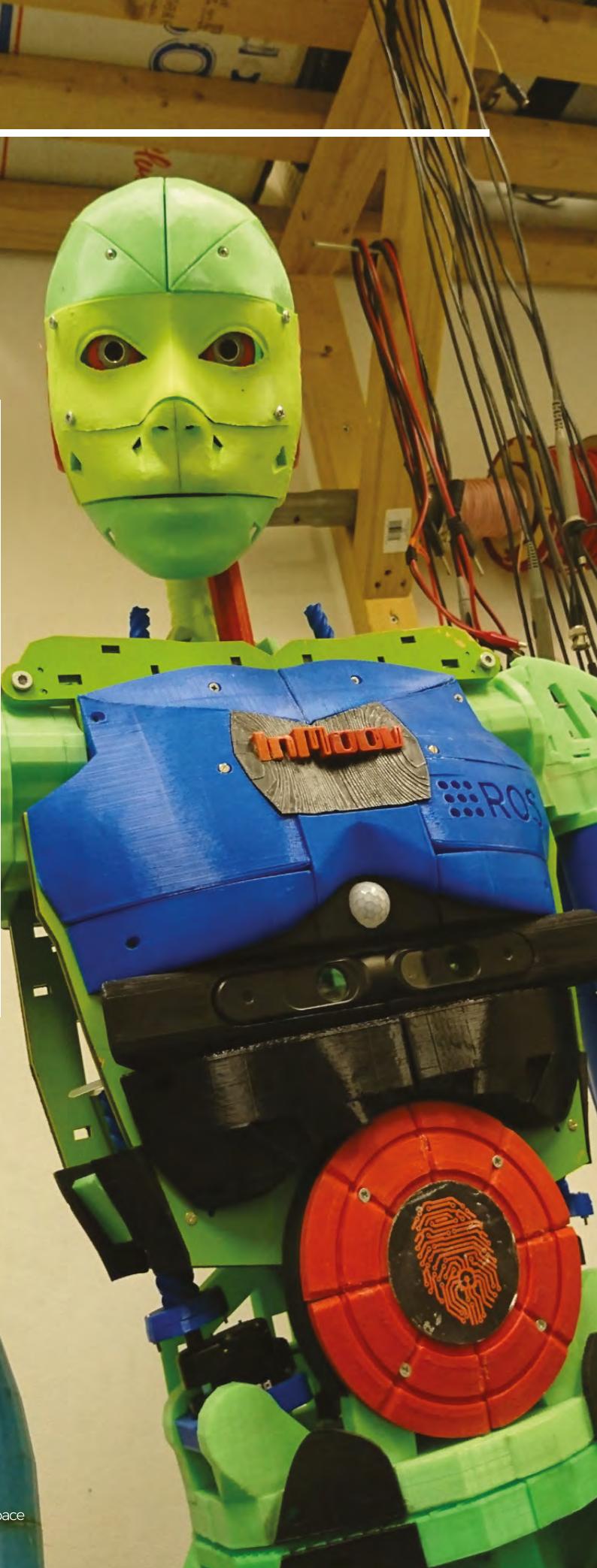
 @mobadr

Kermit InMoov is a humanoid robot that lives at [fizzPOP](#). It came to existence in 2014 as a 3D-printed right hand which was based on the InMoov designs published by Gael Langevin (see [inmoov.fr](#) if you want one!). With time, it grew into two 5 DOF arms with a pan-and-tilt head, a tilting stomach, a rotating waist, and a differential-steering base from an old wheelchair (though it can't wait for the day it walks!). Kermit is powered by ROS, and uses 17 digital servos to move its head and arms. A dozen more motors will soon give it the ability to grasp and manipulate stuff. It sees the world using a stereo camera in its eyes, a PIR sensor in its chest, and an Asus Xtion in its tummy. Kermit had been suffering from an identity crisis since its first day; it never knew what it wanted to be. It found a meaning for its plastic-based life when it came to fizzPOP. It now knows what it wants to be: a butler. After all, no modern makerspace is complete without a robot butler, right? □



Right □

You need a 3D printer with just 12cm x 12cm x 12cm volume to build your own InMoov humanoid







Above ↑
Hacking a set of scales gave both sensors and a frame to hold them



Smart dog bed

By Xavi Rigau

 @xrigau

built the project for the Android Things competition that Google was running on Hackster.io: hsmag.cc/EOZfwJ. I wanted to see if I'd be able to build an IoT device that could help me understand some of my dog's habits, in this case sleeping patterns. We had been playing with Android Things and some sensors for a few weeks with a colleague at work (Paul Blundell), so I decided I'd try to make a device that can be put under a dog bed and track how long my dog spends resting on the bed. The goal would be to understand if my dog gets enough hours of sleep and, if not, or if she gets too much sleep, understand why. Is it because she needs to exercise more? Or if the bed is not comfortable enough? Or is it too warm?

For my initial tests, I used an ESP32 board to make sure I had all the pieces needed, and that all the components were working as expected. Then, in order to build the actual device, I used a Raspberry Pi 3 running Android Things, connected to an analogue-to-digital converter which gets the signal from a feedback amplifier. This amplifier is used to be able to read the voltage differences that the load cell (weight sensor) outputs when some weight is applied to it. Then, on the software side of things, the Android Things device gets a reading every minute from the load cell sensor and uses a threshold to determine if the state has changed (the state can be either idle to busy); if so, then we save the new state to the Firebase database, along with the current timestamp. Lastly, there's a very simple companion app for Android that reads the data from Firebase, and does some basic calculations to figure out how many hours the dog has spent on the bed. I ended up buying a cheap electronic scale and removing the stuff I didn't need from it, so I was able to simply use the structure of the scale, connect the sensors to my circuit, and put the scale underneath my dog's bed.

I'd like to explore making the device smarter and enhancing the companion app to give more insights, but I wanted to keep it simple to be able to finish an MVP (minimum viable product) for the Hackster.io competition. The most obvious thing I'd like to do for the future is add a dashboard in the companion app so that you'd be able to see at what times of the day the dog is using the bed. The Android Things app could also be improved by adding more fine-grained sensing so, rather than only recording 'idle' and 'busy', it could convert the reading from the sensor to actual weight. If we want to take it to the next level, we could run some machine learning to detect anomalies. □

Robots: 500 years in the making

We welcome our new robotic overlords at the Museum of Science and Industry

We live in a world where robots are everywhere. They're in theme-parks, in movies like *Star Wars*, *Wall-E*, *Transformers*, *Robot Chicken*, *Cloudy with a Chance of Meatballs*, *Short Circuit*, *Short Circuit 2*, *Terminator*, *Kryten* (and the *Skutters*). Robots are a staple of all kinds of fiction. We play with toy robots when we're children, we build robots as soon as we can stick LEGO bricks together, and we probably own objects that themselves were made by robots.

But why are robots such a big deal for us puny humans? Ben Russell of the Science Museum Group might know. He's curated an exhibition of robot curiosities called *Robots: 500 years in the making*, which is currently on show at the Museum of Science and Industry in Castlefield, Manchester.

"We know our visitors are fascinated by their bodies," Ben suggests when asked. "Visitors spend ages going around the medical science galleries, looking at all the things that can go wrong with themselves and how to fix them when they do. We also have a natural tendency to anthropomorphise... the more human-looking robots make you think about yourself."

This natural interest in ourselves means that the robots in the exhibition hold up a mirror to us. Two of the more modern robots are the Kodomoroid, a Japanese robot used for reading weather reports, and the Robothespian. Both of them are very modern creations, but the Kodomoroid is almost too lifelike. In contrast, the Robothespian has its insides on display. It's not trying to suspend your disbelief that you're looking at a real person, and that arguably makes it easier to relate to, despite the fact that it's very obviously not a human. →

Right ↗
The Kodomoroid gets its name from the Japanese word for child (komodo), and was developed to read news and weather reports



Robots: 500 years in the making

EXHIBITION



Above Robots get to do the boring or dangerous jobs, which should free humans to do more rewarding things with our time

Ben thinks this indicates a bit of an identity crisis – if the things that make us uniquely human are being taken away from us, is it time for the robots to take a step back?

I WORK, THEREFORE I AM

The word 'robot' didn't exist until the 1920s, but the things themselves have been around a lot longer than that. The robots in the exhibition are all products of their time. The earliest are simple wooden figures that re-enact scenes from the Bible, to show an illiterate public the meaning of the crucifixion.

These animated scenes eventually spread into the homes of the wealthy and to exhibitions of curiosity for the paying public.

Machines in this age could write, play music, and produce woodwork but, without Arduino and

Raspberry Pis, they were still very much single-use creations, with no intelligence. We're extremely fortunate that whatever we make today can be repurposed with a few lines of code.

THE HUMAN ELEMENT

But it's not until the age of Fritz Lang's *Metropolis* that we start to see the mechanical humans that came to define the word 'robot' for most people. These are the classic science-fiction robots, which can mimic actions programmed into them by a human. Where it gets exciting is the point at which the robots go from mimicking the actions of the body to those of the mind. But how comfortable are we with having an intelligent mechanical servant? It's odd enough to issue commands to an Alexa, or something similar, without saying please; it's going to be weirder still when Alexa has a face.

Ben is optimistic about the rise of the robots. "Pepper, the robot created by SoftBank, is one of the first to cross the uncanny valley, the gap created when robots get a little too close to us for comfort," he says. "It's friendly. It can give you a fist bump and truly interact with you."

"99-point-something percent of all robots are not humanoid; they're designed to do a specific job. But, when they do finally escape from the factory into the human world, will it be helpful for them to have a human form? Where we go next is the big question. Let's see what happens." □



Right Our only worry is that the robots will remember being kept in these glass cases, and in time, it'll be us inside there

Right ▶

It's no coincidence so many of the 1920s and '30s robots are made of aluminium: it was an incredibly futuristic material at the time

ON TOUR

Robots: 500 years in the making is on at the Museum of Science and Industry until 15 April 2018. After that, it's going to the National Museum of Scotland, the Life Science Centre in Newcastle, and will then go on a world tour until 2021, including Stockholm and Tokyo.



Meet The Maker: Paul Parry

"I love things that light up and glow"

The orange-red glow of the numbers in a Nixie tube can be a strangely addictive thing. The Nixie bug first bit Paul Parry in his young days when – at the age of four or five – he played with a Nixie calculator. Although he didn't know what these glowing number tubes were called, he hasn't forgotten their enchanting look.

After a career as an electronics engineer, Paul found himself in management, and removed from the actual engineering that he enjoyed. Looking to re-engage with practical electronics, and having rediscovered Nixie tubes, he bought a frequency counter from eBay to convert into a clock.

Wanting to make another, he put this first clock on eBay for the (according to Paul) cheeky price of £250. When that sold within four hours, he realised that there was demand for this work, so he re-invested the £250 in a set of six more frequency counters. These six sold and the business was born.

Paul has slowly left his old job, first dropping to four days a week, then three, and now he spends all his time working on these clocks. In 2017 he became a full time clockmaker. ➤





Left The Bombe clocks are built from the original designs from the Bombe machines used to crack German cyphers at Bletchley Park

"I see myself now as retired, but I get to go down the workshop every day and play with Nixie tubes, electronics, and God knows what I've got there."

Some of Paul's clock enclosures are built from scratch, but many are upcycled:

"I love old bits of electrical equipment. Rather than copy it, I found it much easier to just use an original piece of equipment ... so many people throw them away. There's a mountain of voltmeters from the 1940s and 1950s. They're no use to anybody now – you can't calibrate them ... they're beautifully made out of solid oak and mahogany and dovetailed. I use them because they're there. I appreciate the work that went into them when they were built – all I do is put a clock movement in and maybe a little bit of steampunk decoration."

The Bad Dog store rooms are overflowing with equipment waiting to become clocks:

"I've got a stock room full of just about every bit of test equipment from the 1890s onwards. People have a look through and choose a picture and ask me to convert it into a clock ... between us we turn it into a design." □

UPCYCLED HOUR

Upcycling old bits and bobs is a great way to give your makes a vintage charm while keeping stuff out of landfill. If you're looking for inspiration, you can join Paul and other people who give old things a new lease of life on Twitter every Tuesday for Upcycled Hour. It's a chance to see what other people are up to and chat with other makers. Join in on Twitter by following @UpcycledHour.



NIXIE CLOCK KITS

Paul uses the gubbins available from pvelectronics.co.uk to make the internals of his clocks, so if you want to turn some old electrical equipment (or anything else for that matter) into a clock of your own, you can do it using exactly the same electronics as Paul.

The Nixies run on quite a high voltage (around 170 volts), so you do have to be a little careful when putting everything together, but provided you take appropriate precautions, most people with some electronics experience should be able to build one of these fantastic-looking clocks.



Above ↗
The man making
clocks glow

Right ↘
Old test equipment, like
this ammeter, forms
the basis of many Bad
Dog clocks

Left ↙
A clock in progress: the
Gemini Clock includes
two sets of Nixie tubes
showing the time in
Singapore and London



Below You can view more of Paul's designs on his website, bad-dog-designs.co.uk



Embrace failure?

Failure isn't always a bad thing

– it can often be helpful in the learning process



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

Do you embrace your failures? I don't mean the life-threatening catastrophic ones (which I hope are few!), I mean the niggly, annoying, ego-killing ones. The ones where you later think, 'I should have known better' or 'I'll remember not to do that again' or even 'ooh that's interesting!' The pressure of time constraints, tight budgets, and having to 'look professional' means making mistakes is something we don't usually like to admit. But for me, embracing these annoying failures has become one of my greatest safety nets. They allow me to try things I would otherwise be too scared to try. They also remind me not to become too complacent and to leave time for experimentation. Hours before something is required is not the time to realise you have just made your last LED go pfffft. (Yes, they do make that noise).

It has taken me a long time to reach this stage of acceptance with my failures. And, when they happen, my first response often sounds similar to the blown LED. Failure does dampen the 'fun' of making.

Embracing failure has helped me manage expectations – of both myself and of clients. This prevents 'anticipointment' – the expectation or anticipation of one thing, only to be disappointed by the final

result. Although I aim for 100% perfection in what I make, I know that achieving that first time round is highly unlikely, but 80% of perfect is still pretty good.

As most of my projects are bespoke, I make it clear that I need some trial and error time: that the item will solve the problem, but the final form factor – shape, colour, number and location of blinky lights etc. – is not fixed.

Failure helps me to find systems of best practice – the way to do things that will reduce the possible number of ways

in which I can muck it up. Failure has also helped me anticipate mistakes – following instructions on a 'how-to' is all very well, but what happens when something goes wrong? When

The pressure of time constraints, tight budgets, and having to 'look professional' means making mistakes is something we don't usually like to admit

something doesn't quite go the way you expected? When you don't quite have the correct parts?

The community of makers on the internet such as forums, Slack channels, Twitter etc. are brilliant for help with debugging, but it is down to me to understand and actually fix the problem. But I have found the more mistakes I make, the better I am at finding out what went wrong. The better I get at fixing mistakes, the better my making is, the more things I try, and the more fun I have.

Failure is a valid way of learning. Maybe you should try it? □

Be a good engineer and tackle prototype bias

Applying engineering principles to sociological phenomena



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.

Much of engineering is about compensating, trimming, and equalizing imperfections out of real systems: wrap a feedback loop around the system, and force the error function to zero. This thinking can be readily extended to sociological phenomena.

Consider this thought experiment: if someone asked you to draw a picture of an engineer, who would you draw? As you draw the figure, the gender and race assigned is a reflection of your mental prototype of an engineer – your own prototype bias. In the US and EU, most will draw a white male figure. Academic studies (hsmag.cc/hoCJZs) have shown

that these societies are biased to assign high-level intellectual ability to males, and this bias starts as early as six years old – over a decade before these children are confronted with picking a major in college. The integration of these biases over time seems to be reflected in STEM-major gender ratios.

Prototype bias is real and pervasive, and continues beyond school. For example, my co-founder in Chibitronics, Dr Jie Qi, is female while I am male. The company is founded on technology that she created for her MIT Media Lab doctoral dissertation. She is the inventor of paper electronics.

Any engineer who observes a bias in a system and chooses not to proactively correct it either stands to benefit from the bias, or they are a bad engineer

I am a supporting actor in her show. Despite laying this fact out repeatedly, she still receives comments and innuendo implying that I am the inventor or more influential than I really am in the development process.

Any engineer who observes a bias in a system and chooses not to proactively correct it either stands to benefit from the bias, or they are a bad engineer. Given two pieces of metal with different coefficients of thermal expansion, do we trivially bond them together and expect the resulting assembly to stay straight?

No; good engineers either call it out for what it is – a bimetallic thermal sensor – or compensate for the bias through clever design. Thus aware that our sociological channels are biased, what do you do? Compensate for the biases, or take advantage of them?

If you design to use biases in your favour, then document it and call it out. Good engineering requires routinely documenting and informing collaborators of decisions.

If you design to eliminate biases, then the challenge is to figure out appropriate corrections for the biases. Good engineers consider all the tools available to tackle hard problems, from adjusting salaries and titles to simply highlighting a collaborator’s contributions more often. □

Letters



Left Raspberry Pis are on their way to Glasgow

HELPING HACKERS KEEP ON HACKING

MAKLab, Glasgow's only public makerspace, shut its doors earlier this year leaving the local hacking community without their hub or equipment. Martin Goodfellow started the group Glasgow Makers and Hardware Hackers to keep the community that built up around the space alive. While they don't have a permanent venue, they are running regular meetups and workshops. You can always keep up to date with what's been going on by following @glasgow_makers on Twitter.

Martin got in touch to say that without access to more equipment, they were struggling to keep the workshops running. We're happy to help out the hacking community in Glasgow and we're organising a box of Shrimping kits and Raspberry Pis to help keep the workshops running until Glasgow gets another space.

If you know of a making or hacking community in need of help, get in touch at hackspace@raspberrypi.org.

MAGS FOR MAKER SPACES

We received the box of volume 1 magazines and have been sharing them with our members, students and employees. They have been very well received – thank you!

I would like to inquire about receiving them in the future. I like having them around the facility, both in the office/customer areas, as well as a few out in the makerspace, for people to browse through, as I believe magazines like this are a good way to provide inspiration for folks to work on their own projects.

Thanks!

John Sheehan

Chief Operating Officer, Vocademy, Riverside, California

We're delighted to hear that John, thanks for writing. If anyone else out there knows a hackspace that would like some copies of the magazine for cost price, plus shipping, get in touch at hackspace@raspberrypi.org and we'll make it happen.

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

PAINTING PICTURES

Hi Hackspace,

Just wanted to say I loved the first issue of HackSpace magazine!

I found the Head 2 Head article very useful. I'm still very much a beginner when it comes to electronics, and one thing I want to improve on is recording schematics of the projects I'm making. Looking forward to visiting both Tinkercad Circuits and Fritzing.

Thanks,

Kaylee

Dorset

Kaylee – cheers, we're glad you found it useful. Everyone else – if there's anything in the future you want to us to take a look at, just let us know.

ROBOT REQUEST

Two words: more robots. Oh, and some wearables projects would be nice.

Adam

North Wales

Simple and to the point. We like it. Turn to page 86 for a literally and metaphorically brilliant wearables project; watch this space for some massive robotics builds.

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

Pokit

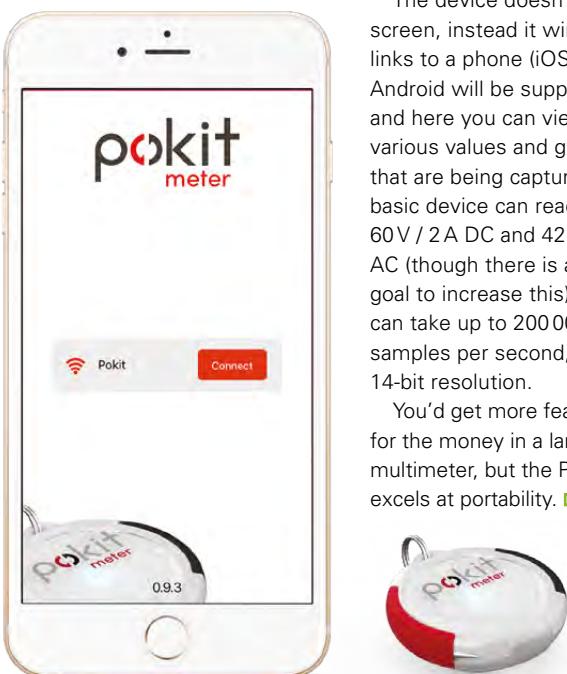
A multimeter, oscilloscope and logger, designed for portability

AU \$90 | hsmag.cc/rVAJfZ | Delivery: April 2018

Sometimes, the best tool isn't the one with the most features to boast about, but the one you actually have with you when you need it. If you're on the go and some electronics aren't working, it doesn't matter what you've got on your workshop bench: what really matters is the stuff you're carrying with you. That's where the Pokit comes in. It's a tiny multimeter, oscilloscope and logger that's small enough to fit on your key-ring. The leads are integrated into this small package so there's thankfully nothing to get lost or tangled.

The device doesn't have a screen, instead it wirelessly links to a phone (iOS and Android will be supported), and here you can view the various values and graphs that are being captured. The basic device can read up to 60V / 2A DC and 42V / 1.2A AC (though there is a stretch goal to increase this), and can take up to 200 000 samples per second, with a 14-bit resolution.

You'd get more features for the money in a larger multimeter, but the Pokit excels at portability. □



Twooolhead

Double your 3D printing capacity with a parallel printing head

\$599 | hsmag.cc/OnUFwC | Delivery: March 2018

There are many wonderful things about 3D printing, but at times it can be frustratingly slow, especially if you need multiple copies of a design. The Twooolhead is a print head for Lulzbot TAZ devices that simply prints out of two printheads. This means that two identical copies of the object are produced in the time it usually takes to make one.

You don't need to modify the whole printer: just attach the new printhead, update the firmware, and you're ready to go. The design of the Twooolhead is open source, so you're free to tinker with it and adapt it to your own needs.

The main downside of parallel printing is that it does decrease the size of the object that you can create, but there's still a print volume of 135 mm × 280 mm × 250 mm, which gives plenty of space for many items.

If you're looking to print large numbers of small objects, the Twooolhead could literally double your speed. □



Hackspace of the month: fizzPOP



Stuart Pearson

- fizzpop.org.uk
- [fzzPOP](#)
- @fzzpop
- hello@fizzpop.org.uk
- 0121 699 9989

S

Tuart Pearson got in touch to tell us about fizzPOP, a makerspace in Birmingham that describes itself as 'the garden shed of your dreams'.

Take it away, Stuart...

WHEN DID YOU START?

The origin story of fizzPOP started in 2009, when a group of makers and artists used to meet every month to talk about their projects. Over time, the need for a dedicated space for the members to work on those projects became obvious. Early in 2013 our hosts, at the time, were leaving Birmingham and therefore it seemed like the right moment to take the step. A group comprising of a student, university lecturer, consultant anaesthetist, locksmith, and a freelance IT guy, became the founding directors, and started hunting for a building to get the idea going. With a lot of support from the Custard Factory team and every friend / family member that we could get to volunteer some time (thanks again everyone), in July 2013 we moved into our first space; the following year, we moved to our current space and haven't looked back.

WHO IS IT FOR?

Rather than focus on a single aspect of making, we try to appeal to makers and artists of all disciplines and let their needs focus the direction the space goes in. So, as well as members interested in electronics, CNC, robotics, wood and metal work, we also have those working on costume design, paintings, and pottery, a direction we never originally thought things would take.

WHAT EQUIPMENT DO YOU HAVE?

Through the support of local businesses, local colleges and universities (see our thanks page for

details), we have a large collection of tools and equipment. We would also thank all the members who have donated equipment to the space, which allows the community to have use of it. There are also a few members who buy items: "Well, if I need it, so will someone else!" We ask all the members on sign-up, "What are we missing?" at that time, and we regularly poll everyone on what they would like the committee to hunt for next.

We are also not afraid (maybe we should be) of making our own tools. As a group, we have pulled on all our various skills to build a vacuum former from scratch, using various bits and pieces from our donation room. The design is on our Instructables for anyone to have a go at, and improve on what we made.

HOW DO YOU SHARE YOUR IDEAS?

We do workshops in the local area supporting other events with some of our tools/projects. As well as being avid users of Facebook and Twitter, we post projects on Instructables and have a strong Meetup group.

We have also gone along to meetings of other local spaces that are looking to start up, to share what worked, what didn't, and what kept us up at night. (First hint: People don't like the cold, even if it makes their laptops or Pis run very slightly faster.)

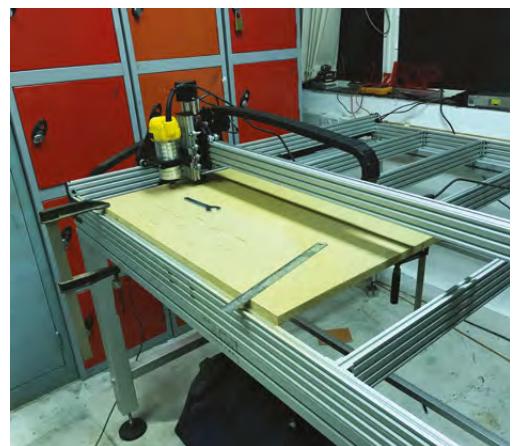
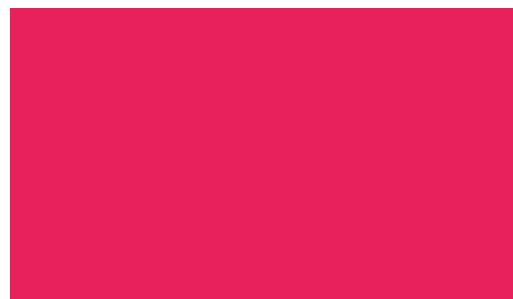
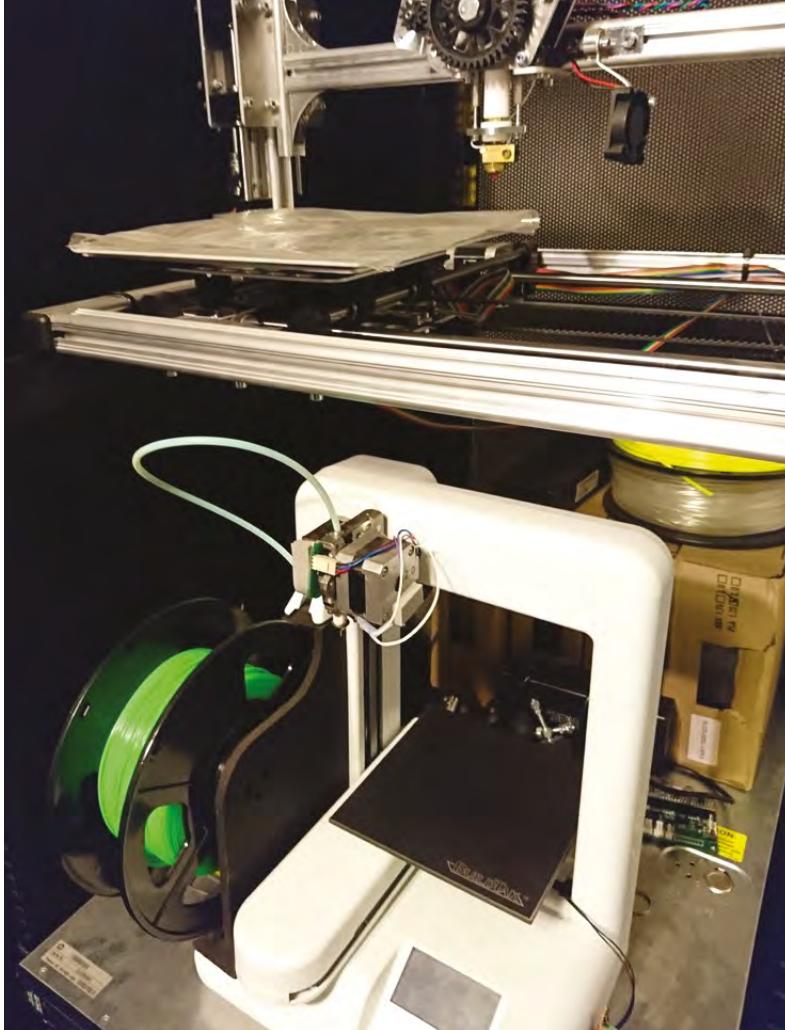
WHAT TRAINING DO YOU DO?

Rule zero... Do not be on fire! Okay, we may have stolen that from London Hackspace, but we think it works no matter where you happen to be... In 99... 95% of cases.

We offer regular training on the safe and productive use of our equipment; learn-a-skill nights, such as lock-picking or PCB design; and are about to start offering multi-week beginners' courses, in topics such as building a wooden box, Fusion 360, and ceramics.

WHAT HAVE YOU MADE RECENTLY?

As well as working on our own projects at fizzPOP, we love working together on group projects. Recent ones have included the following...



Above fizzPOP has loads of tools, including a potter's wheel!

Left fizzPOP's CNC router has had a few upgrades: they've added limit switches, eStop, dust extraction, a bed with threaded inserts, and a robust stand

Hackspace of the month

REGULAR



Gareth's Coffee Table ♦

I saw a project using resin and reclaimed wood somewhere on the internet, then got a flyer for a local charity (Woodshack) that sold reclaimed wood, so I decided to replace my boring glass coffee table with something I made myself. Recycled scaffolding board, recycled scaffolding poles, new clamps/scaffolding ends, resin/paint/glowing powder, and lots (LOTS) of sanding. Didn't flatten it completely, so the history of the wood was still visible.



Chris & Sam's Monome ♦

In collaboration with Birmingham Open Media and Leon, a local artist and member of fizzPOP, two of our members built a giant music-making Monome which was installed in the chill-out zone in James Brindley Dovedale Centre, a local special-needs school in Birmingham. The Dovedale Monome is made of a grid of 64 capacitive touch buttons and NeoPixel LEDs, and functions like a Monome using BEAP Polygome. The buttons were heat-pressed out of 3 mm acrylic in a form made on the CNC router, the frame was produced in the woodworking room out of iroko hardwood, and the bezel cut from aluminum laminate on the CNC router.



Chris's Bed ♦

A pile of old floor joists were planed by hand, before the arrival of the Scheppeach HMS 260 Planer Thicknesser, and joined using handmade dowel pegs in a traditional fashion to create this bespoke bed frame incorporating reclaimed desk drawers and IKEA spring slats.





Xmas Jumper ♦

Last Christmas, we were sponsored by Maplin to create a tech-fuelled Christmas jumper. This was a great project that brought together many different disciplines – textiles, soldering, 3D printing, coding – and, as a group project, it was fantastic to have so many people involved. The Christmas jumper got both ourselves and Maplin some great write-ups in the press, and our ‘how to make a Christmas jumper’ video can be found on Maplin’s YouTube channel.



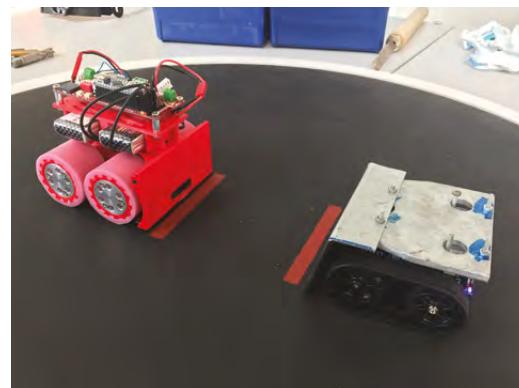
Stuart's Battery Tab Welder ♦

Having been cursed with failing power tool battery packs, I am currently working on a battery tab welder. Built from recycled capacitors, and an SCR to trigger the pulse, it will be used to weld tabs onto lithium-ion batteries.



Wesley's Knife Grinder ♦

What do you do when you’re trying to make some cutlery and need a large belt grinder? If you’re a normal person, you’d probably look for someone who has access to one, or buy one if you can afford it. But where’s the fun in that? If you’re a maker, you make one. One of our members, Wesley, designed and hand built a 2” x 72” belt grinder from an upcycled treadmill DC motor and a speed controller built from common, inexpensive components. It’s used for metalworking and especially handmade cutlery.



Robot Mini Sumo ♦

We are big fans of Robot Mini Sumo, with many members competing in regular monthly competitions. Some members have built their own, and others have based theirs on off-the-shelf kits which they have improved or modified.

Vacuum Former ♦

One of our older, but equally very popular, group projects was building a vacuum former based on GU10 halogen lamps. We published this one on Instructables, and it has now received over 700 likes on the platform.

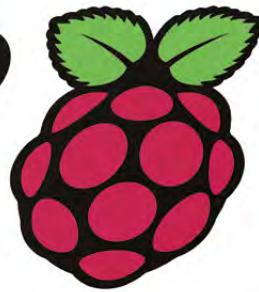
CONTACT US

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

DON'T MISS THE BRAND NEW ISSUE!

YOUR OFFICIAL RASPBERRY PI MAGAZINE

The MagPi



Issue 65 January 2018

raspberrypi.org/magpi

APHEX TWIN MIDIMUTANT
Megastar talks to us about new AI music tool

RASPBERRY PI FOR NEWBIES

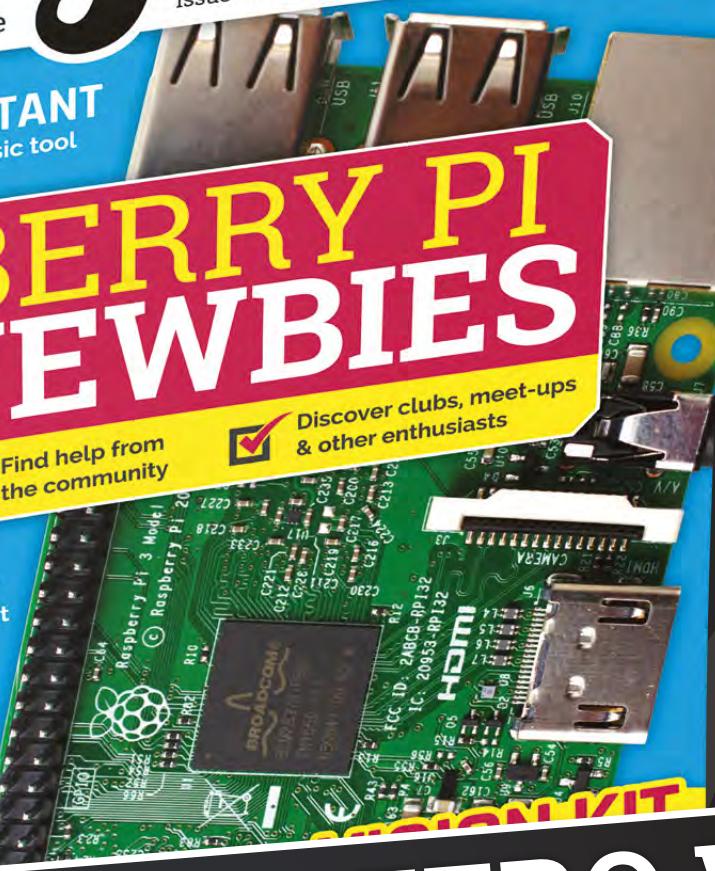
Set up and start using your Raspberry Pi

Find help from the community

Discover clubs, meet-ups & other enthusiasts

MAKE GAMES IN C++
Professional video game development

PI-TOP LAPTOP
REVIEWED!
The p



FREE PI ZERO W
With your 12-month subscription to the print magazine
magpi.cc/Subs1

PLUS! FREE
CASE, THREE
COVERS &
CABLES



Buy online: store.rpipress.cc

LENS

HACK | MAKE | BUILD | CREATE

Uncover the technology that's powering the future

PG
40



BADGELIFE

Charge your batteries and explore the unique art form of the hacker conference

PG
46

WE LEARN CROCHET

Blankets and cat toys: we tangle yarn the right way with Swansea Hackspace

PG
52

THE CANUTE

How Bristol Hackspace is making Braille cheaper, faster and all-round better

PG
32

3D PRINTING

Everything you need to know to set up your own desktop factory



PG
56



INTERVIEW HACKER HOUSE

Drones, security, and why the Internet of Things is the way it is

PG
62

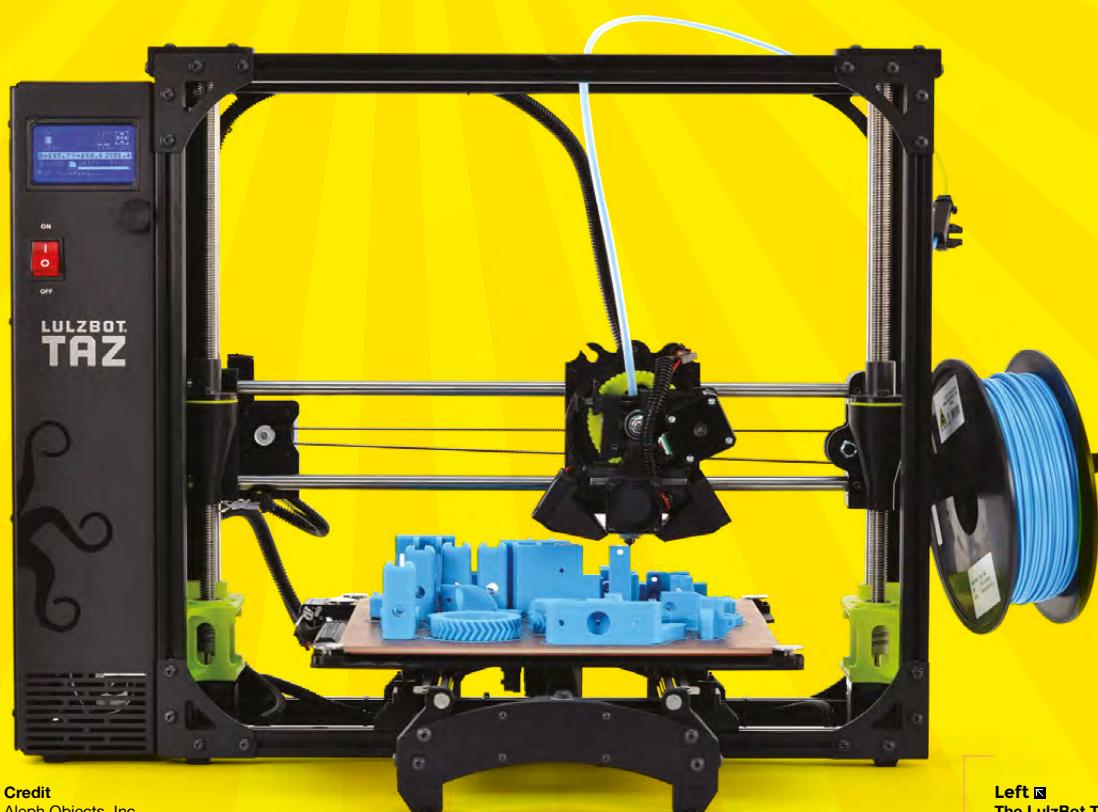
RUBBER BANDS

Find out why rubber bands are an engineering miracle

How to Start 3D Printing

Everything
you need to know
to buy your first
3D printer
and start making

By Cameron Coward



Credit
Aleph Objects, Inc.

Left ▶
The LulzBot TAZ 6

C

ongratulations on making the decision to start your journey into 3D printing! You've probably seen online all of the amazing creations people are making, and are eager to dive right in yourself. But, like all hobbies, getting started with 3D

printing can seem overwhelming at first. There are hundreds of printer models to choose from, a wide variety of filament types to consider, and even the available software options can be a lot to think about.

Luckily, hobbyist 3D printing has been dramatically refined in recent years, and getting started has never been easier. 3D printers have become very affordable, and even consumer-grade models now rival the quality of professional printers from less than a decade ago. The various software options are stable and reliable, and are now easy to learn, even for a complete beginner. In this guide we'll show you what to look for when purchasing a 3D printer, and how to start using it.

The most exciting part of getting started with 3D printing is choosing the printer itself. But, with so many options out there, how do you know what to look for? While it's true that there are many models on the market, the vast majority of consumer 3D printers are actually quite similar. Virtually all entry-level hobby printers use a process called fused filament fabrication (FFF), and that's what you'll want to purchase.

3D PRINTERS ARE MORE AFFORDABLE THAN EVER

Some brands refer to this as fused deposition modelling (FDM), but the way they work is the same: rolls of plastic filament are fed into an extruder, which contains a nozzle assembly called the hot end. The hot end melts the filament, and the molten plastic is then laid down in layers to form the part being printed. There are other types of 3D printers on the market, but most of them are still too expensive for a beginner who is just getting their feet wet with the hobby.

Consumer FFF and FDM 3D printers can still vary dramatically in price, however. At the bottom end are models that can cost less than £150, while the more expensive prosumer printers can easily exceed £3000. The large differences in price generally come down to just a few factors: the size of the print area, the quality of the components used, and the non-essential tech goodies that are built in.

SIZE DOES MATTER

The most pressing consideration is the printer's size – specifically the build volume. This is a direct specification for how large a single printed part can be. Generally,



Above ♦
The Monoprice Select Mini V2

Credit
Monoprice

this is also directly correlated with the model's price. It can be tempting to buy something with a huge build volume, but bear in mind that print time increases exponentially with volume, and most people rarely need to utilise all of the available space. That said, we recommend buying a printer with at least a 4" × 4" × 4" (100mm × 100mm × 100mm) build volume.

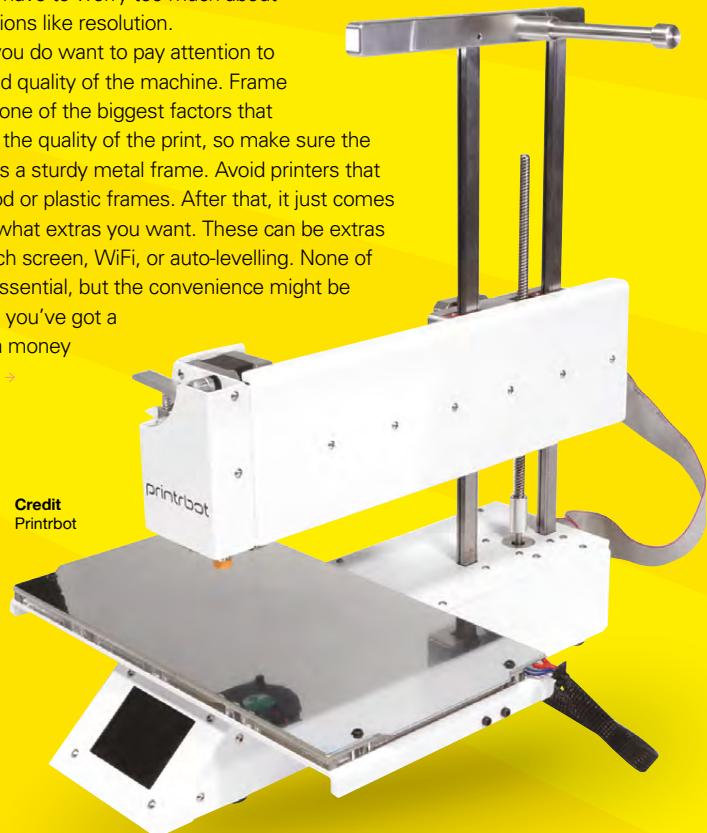
While most new 3D printers come with them now, it's still important that you purchase a model with a heated bed. Having a heated bed at your disposal will greatly expand the types of filament you can work with, and there is no reason not to get a printer that doesn't have one. Aside from that, the individual components of 3D printers these days are pretty comparable, and you shouldn't have to worry too much about specifications like resolution.

What you do want to pay attention to is the build quality of the machine. Frame rigidity is one of the biggest factors that goes into the quality of the print, so make sure the printer has a sturdy metal frame. Avoid printers that have wood or plastic frames. After that, it just comes down to what extras you want. These can be extras like a touch screen, WiFi, or auto-levelling. None of them is essential, but the convenience might be worth it if you've got a little extra money to spend. ↗

YOU'LL NEED

- ♦ Windows, Mac, or Linux computer
- ♦ An internet connection
- ♦ A 3D printer
- ♦ Filament
- ♦ Basic hand tools

Below ♦
The Printrbot Simple Pro



Credit
Printrbot

Filament Types Explained

The plastic material that your 3D printer uses is called filament, and all FFF 3D printers use some type of thermoplastic formulation. That means the filament is designed to melt when hot, but then return to a stable structure when it cools back down. Most filament comes in spools, commonly in amounts of 1 kg, which allows the material to be fed evenly into the printer's extruder.

The extruder contains a small stepper motor, and is capable of feeding the filament at very precise rates. The filament is then pushed into the hot end, where it's very quickly melted. Additional filament entering the hot end creates pressure, and forces the molten plastic out of the nozzle and onto the build plate. There, it cools and forms your part.

But, while all filament types follow this process, the specific type of thermoplastic being used can make a major difference in the resulting part. Some material is meant to be as strong as possible, some is intended to optimise print quality, and others are

designed to be well-rounded for general use. Most hobbyists keep a selection of a few types on hand for different scenarios. In the rest of this section, we'll cover some of the most popular kinds of filament so you can decide what to order.

BASICS AND BEYOND

PLA and ABS are great filaments to get started with (see next page) as they're easy to use and work for a wide range of objects. There are some more advanced options that are good for specific purposes. One thing to be careful of is many of these can leach unsafe chemicals into food or drink that they come into contact with. This isn't a problem for general makes, but if you're hacking any kitchen equipment or utensils, make sure that you get food- (and possibly dishwasher-) safe filament from a reliable supplier.

You may find that you waste quite a lot of filament in your 3D printing experiments. It's possible to get recycled filament, which may make you feel a bit better about the tangle of plastic in a failed print. ↗

Speciality filaments

There are many more filament materials on the market, but most are meant for special-use scenarios. These exotic materials are designed for either aesthetics or for unusual tasks. For cosmetic cases, there is filament you can buy that replicates the look of other materials like wood or metal. There may be times when you require something physically unusual, like flexible filament or support material that can be dissolved. These are harder to use, though, and sometimes need specific types of extruders; make sure you follow the filament provider's instructions when printing with them.

Below A standard 1 kg spool of ABS filament



Credit: MaterHackers

Nylon

Pros

Very strong, low coefficient of friction, resistant to chemicals, flexible when printed thinly

Cons

Expensive, difficult to finish, usually only available in white or 'natural' colour

For a long time, nylon has been used in the engineering world to reduce friction in mechanisms with moving parts. When it comes to 3D printing, it's useful for parts that need to have a low coefficient of friction, or that could be exposed to chemicals which break down other materials. However, it's much more expensive than both PLA and ABS, and only comes in its natural colour. Because of its low-friction nature, it doesn't handle sanding or painting well.

ABS

Pros

Strong, more resistant to high temperatures than PLA, readily available in a range of colours

Cons

Requires a heated bed, prone to warping, fine detail is often lost

ABS is only slightly less popular than PLA, but has a few drawbacks that should be considered. Most importantly, it can't be printed successfully without a heated bed, which means it's not an option if your printer isn't equipped with one. It's also very prone to warping, and special design considerations need to be made – particularly with large parts. But it's much stronger than PLA, and is more suitable for parts that will see physical use.

QUICK TIP

Make sure to purchase filament with the right diameter for your printer (either 1.75 mm or 3 mm)

PLA

Pros

Good for general printing, affordable, widely available, comes in many colours, a heated bed isn't required

Cons

Not very strong, can be difficult to sand, won't stand up to high temperatures

The most popular filament type on the market, and for good reason. PLA is very easy to work with, and can be printed without a heated bed. It's capable of capturing detail nicely, but doesn't accept post-print sanding very well.

PETG

Pros

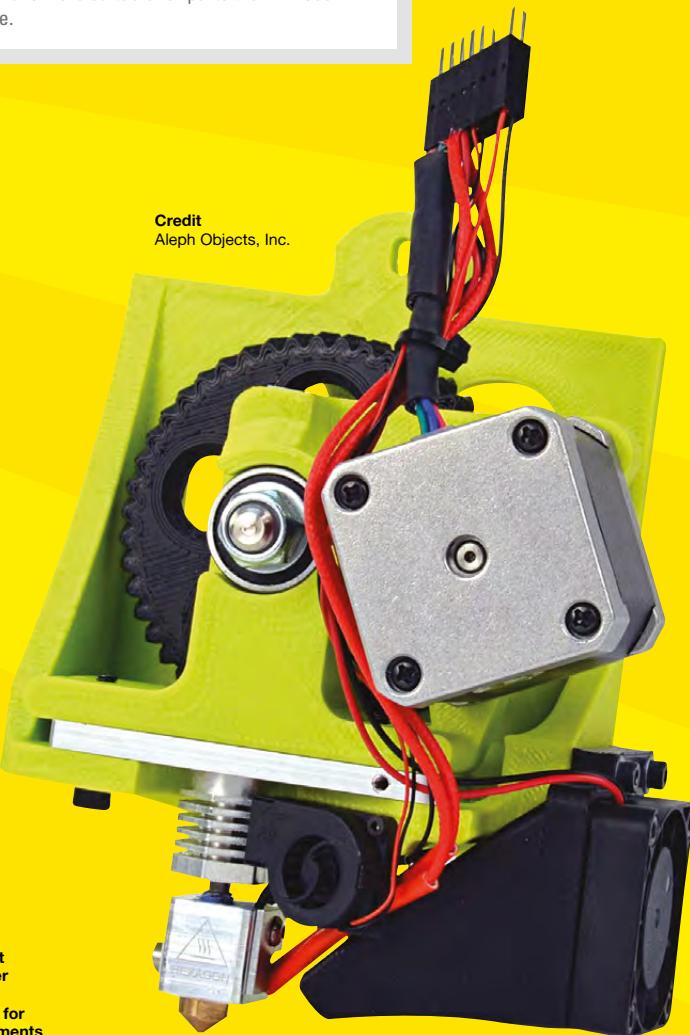
Extremely strong, flexes rather than breaks, decent detail

Cons

Expensive, opaque colours are usually unavailable

PETG is more expensive than PLA and ABS, but less so than nylon. Its primary benefit is its strength and durability, and unlike ABS, warping isn't a problem. The material is naturally translucent, and doesn't take dyes well, so opaque colours aren't often available.

Credit
Aleph Objects, Inc.



Right ↗
The LulzBot Flexystruder is meant specifically for flexible filaments

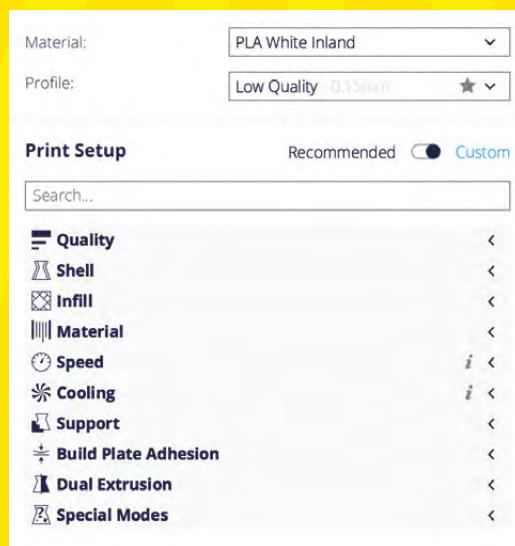
The Software You'll Need

Software for 3D printing can be broken up into four categories: **3D modelling, slicing, printer control, and printer firmware.** Unless you're building your own 3D printer, which is a topic for another article, you shouldn't have to worry about the firmware; it should be already installed and tested by the manufacturer. And, if you're just planning on printing designs modelled by other people, you don't need 3D modelling (CAD) software.

QUICK TIP

Try using different slicers to see which gives you the best results

Right Slicing software has a variety of options for refining your prints



DEFAULT SLICER SETTINGS WORK WELL

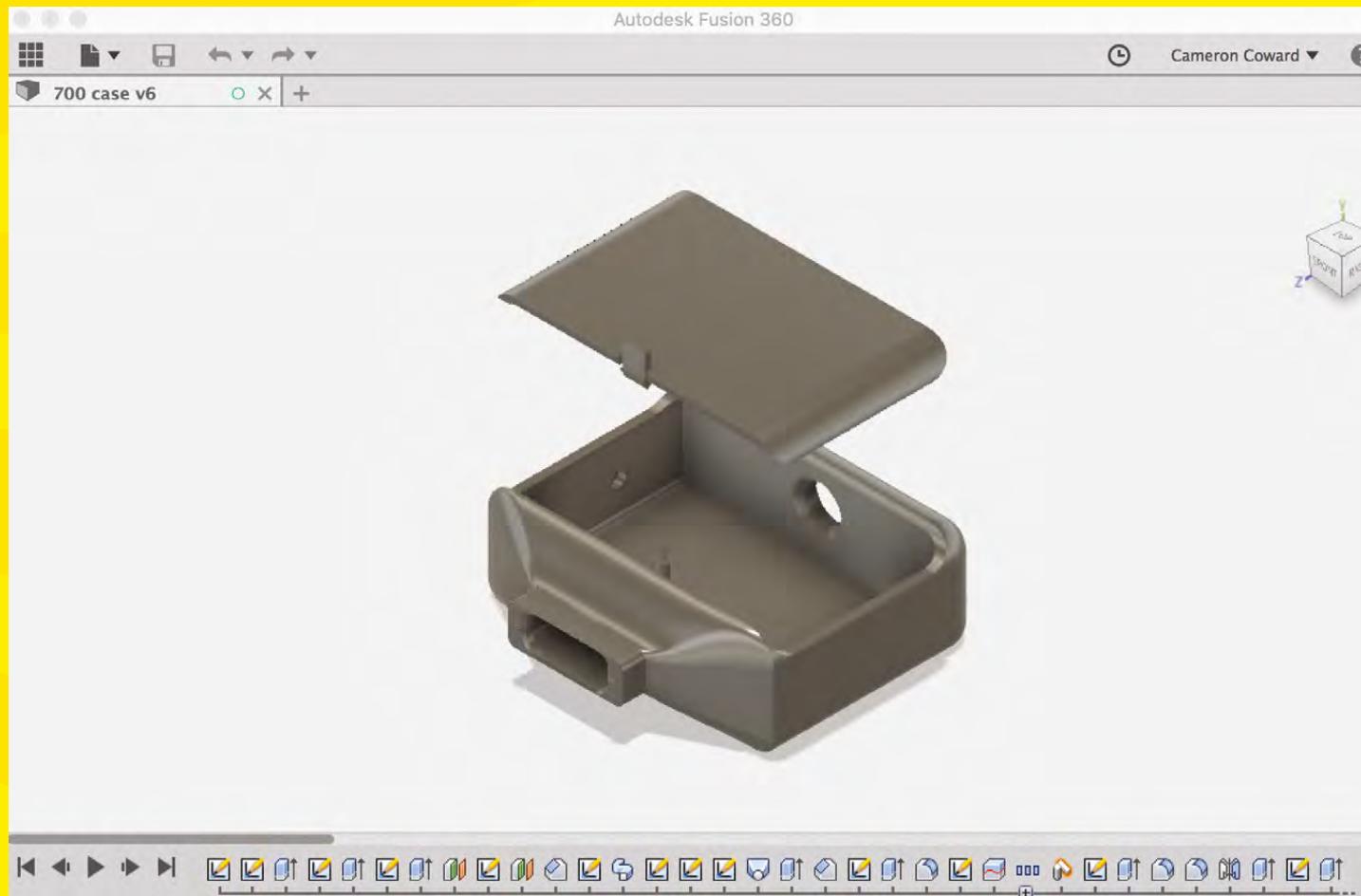
The slicer and the printer control are usually combined into a single software package. The job of the slicing software is to convert an STL file (either one you model yourself or which you find online), into G-code instructions that the printer's firmware can understand. This takes into account the printer's specifications, as well as the parameters that you set for individual prints.

Tweaking those parameters is a key part of getting good printing results. Factors like print speed, layer height, and hot end temperature can all dramatically impact how the part turns out. But, as important as these are, they can be difficult to understand. Extruder retraction, for example, temporarily reverses the direction of the filament flow. But, how do you know how far it should retract, for how long, or how quickly?

SO MANY SETTINGS...

Luckily, the 3D printing hobby has come a long way, and most slicing software provides default options that work well in most cases. There will still be settings that can benefit from experimentation and tweaking, but those default options are a good place to start. Your filament should also come with instructions on specifics related to the particular material.

The control software is much more straightforward. It simply sends the instructions that were created by the slicer to the printer itself.



Some printers will have this built into a control on the printer itself, but the function is the same: to interpret the G-code. The two most popular examples of software packages are Cura and Repetier (which uses Slic3r by default). Both are great options, and will work well for most people.

Finally, there are many choices when it comes to CAD software for making your own 3D models. There are plenty of professional options out there that are commonly used by engineering teams, such as Solidworks, Inventor, and Creo, but these cost thousands of dollars and are unnecessary for most hobbyists. As a direct result of 3D printing growing as a hobby, a number of free or low-cost CAD programs have become available.

One noticeable example is Autodesk Fusion 360, for which we had an in-depth tutorial in the first issue of HackSpace magazine, and there are other programs you might also want to consider, such

as TinkerCAD and FreeCAD. For those of you who are more comfortable thinking in terms of code, OpenSCAD is a popular choice that lets you program a 3D part in the same way that you might with something like CSS or even BASIC graphics. *

Above ▶ Autodesk Fusion 360 3D CAD software



Left ◁ Ultimaker's Cura software can be used with most FFF 3D printers

Print Quality and Common Problems

QUICK TIP

Actually measure the diameter of your filament, as it's rarely exactly as labelled.



While it's generally pretty easy to get your printer oozing plastic, many beginners find themselves frustrated when their prints come out looking sloppy. You've

probably seen photos online that showcase intricate and beautiful 3D printed parts, and it can certainly be disheartening when your first parts don't meet those expectations. Fortunately, most problems that cause poor print quality are well understood, and can be fixed with a few tweaks to your settings. We'll cover a few of the most glaring

examples here, and you'll find that the online printing community is friendly enough to provide you with solutions to any issues you might encounter.

Your first step when you encounter a problem is to verify that your slicer and printer control settings match the 3D printer model you purchased. This will include parameters like build plate size, steps per millimetre for each axis and the extruder, and nozzle size. The manufacturer of your printer should have these specifications readily available, along with instructions on how to set them. After checking that those are correct, you can move on to finding



Right ♦
Your first prints
probably won't look
this good, but you
can get there



Left □
Visible stratification between layers is usually a result of printing at too high of a speed

what's causing your particular issue. The most common of these problems is first-layer adhesion on the build plate. If the nozzle is too high above the build plate, the part won't stick well and will come loose while printing. If it's too low, the first layer will be inconsistent. If your bed isn't level relative to the X and Y axes, you'll get a combination of these two problems. Follow your manufacturer's instructions for setting the height and levelling the bed.

GOOGLE SEARCHES ARE YOUR FRIEND

Another common issue that will dramatically reduce the quality of your print is stringing. This will result in thin strands of plastic that create a kind of web on your part. Most of the time, this can be fixed by turning on retraction and wiping to ensure the nozzle is clean before it moves to a different section of the print. If you still encounter this problem after that, your hot end temperature may be too high for the material you're using. Try turning it down by 5–10°.

Most other print quality problems can be attributed to overly aggressive printing. This will often manifest as shadowing of features on the part, very visible layering, or poor dimensional tolerances. This can be solved by simply printing

at a slower speed, or by reducing the acceleration rate. A printer with a very rigid frame can handle faster speeds and higher acceleration, but most lower-end printers aren't quite as solid. It will take longer to print your parts, but the quality of the result will make it worth it. □

Get printing

Designing objects for 3D printing can be daunting, but don't worry: the 3D printing community has built up a fantastic library of designs that you can print out on websites such as Thingiverse (thingiverse.com). A few of our favourite 3D printable objects from Thingiverse and beyond are:

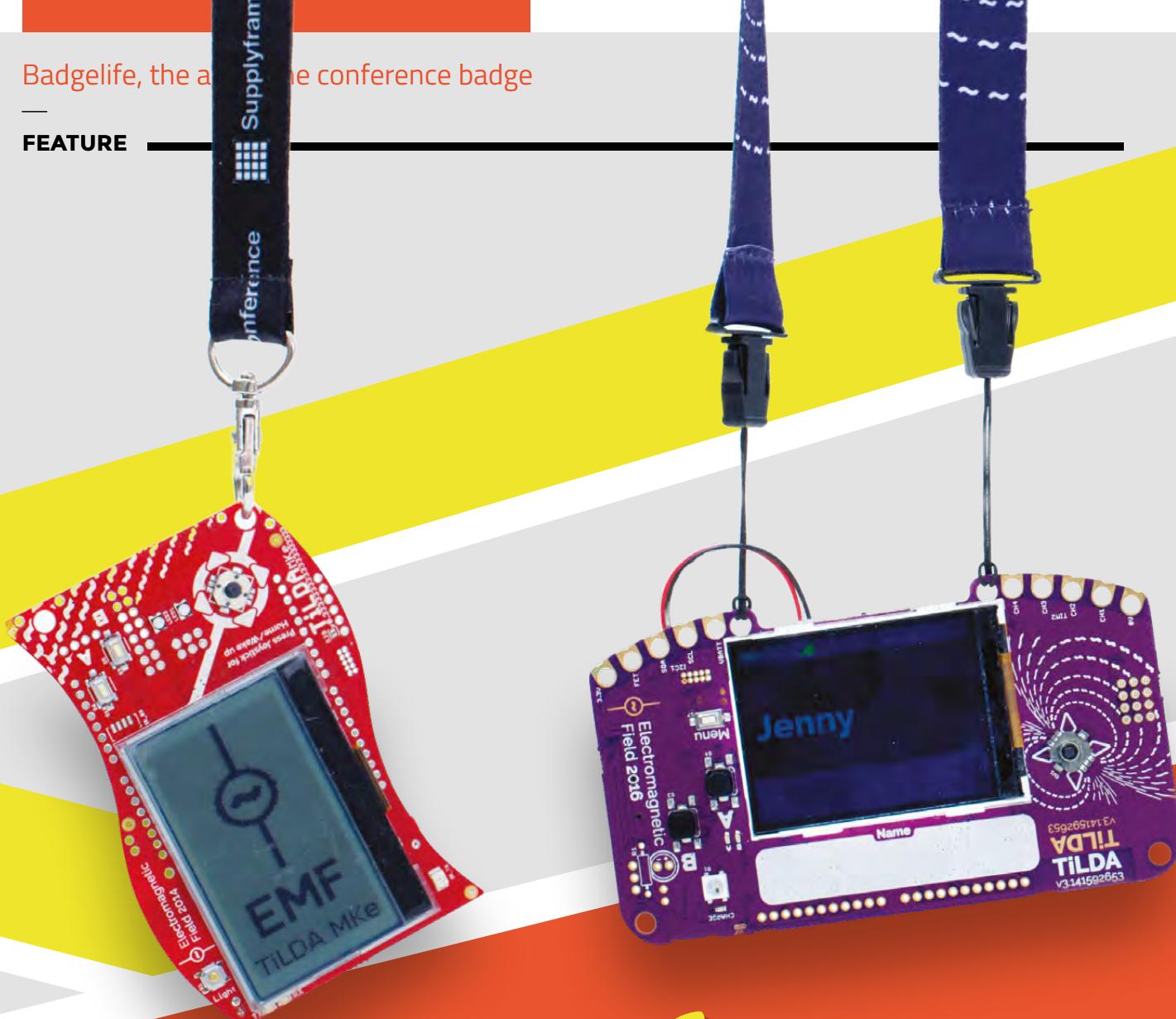
- ◆ **Digital sundial:** The shadow changes into the numerals of the current time; hsmag.cc/PnNPrz
- ◆ **Mechanical vice:** Perfect for holding projects together while working on them; hsmag.cc/mCTtqH
- ◆ **InMoov Humanoid:** A life-size humanoid that can be created on a printer with a 12 cm × 12 cm × 12 cm print volume; inmoov.fr
- ◆ **3D scanner:** Pop your smartphone in this holder and use its camera to create 3D models; hsmag.cc/SNGEjq
- ◆ **Telescope:** The frame for a parabolic mirror telescope; hsmag.cc/lYnaTj

QUICK TIP

Always start with the filament manufacturer's recommended settings, but feel free to experiment from there.

Badgelifie, the art of the conference badge

FEATURE



Badgelifie

the art of the conference badge

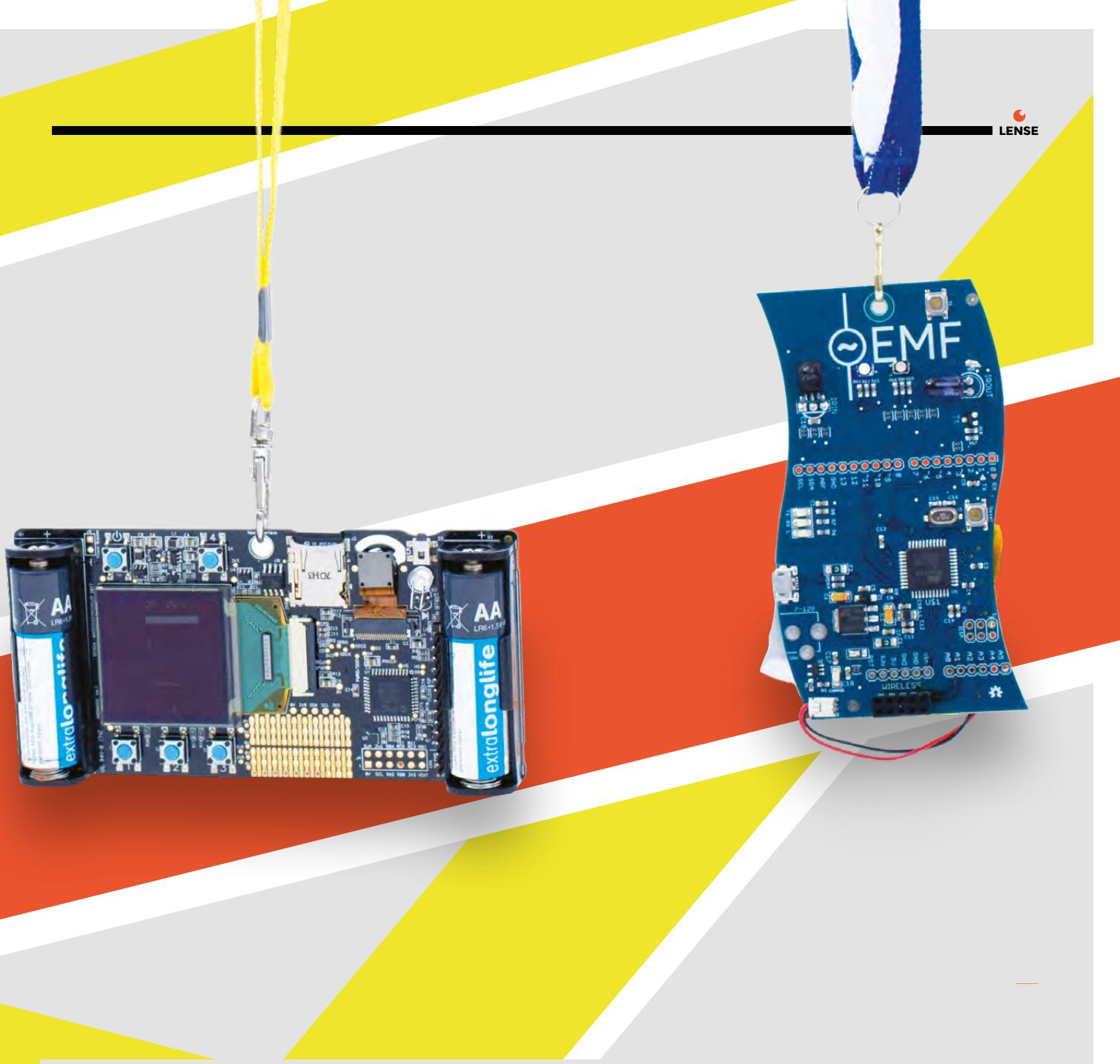
Our community has an art form all of its own, and it's a vital part of our culture



Jenny List

@Jenny_Alto

Jenny is the creator of the @LanguageSpy electronics kits for Raspberry Pi and ham radio. She's also a key member of Oxford Hackspace.



There are times when the art world can appear inaccessible and remote to outsiders. As billionaires pay record prices for lost Leonards, and achingly trendy East London galleries serve up the latest conceptual works to gushing art journalists, the person in the street can be forgiven for feeling a little at a loss. Dismissed even, as though our simple tastes are not worthy of consideration in the face of such sophistication. Sometimes though, artistic movements and new artistic media evolve on their own, outside of the art establishment. They retain their own raw creative edge, until eventually they are 'discovered' and exclaimed over, with examples such as graffiti art gracing the pages of highbrow catalogues and being featured inside those trendy galleries, rather than painted on their exterior

walls in the dead of night. Our community has an art form all of its own, one that came from the grass roots, that boasts huge technical sophistication, and which can deliver exceptionally beautiful results.

We are speaking, of course, about electronic conference badges, the wearable confluence of electronic hardware and PCB art that has taken on a life of its own over the last decade. A modern conference badge will typically be an all-in-one computer in its own right, on a board replete with PCB artwork, complete with colour TFT screen, input devices, USB, and WiFi networking. They will come with a fully hackable operating system, typically with an app store for which code can easily be written by the conference attendees. The announcement of the badge is a key part of the event build-up and, during proceedings, the store will fill up with a huge →

Above All three EMF Camp badges so far; from 2012, 2014, and 2016, plus the 2017 SuperConference badge to boot



Above The front view of a DEF CON 14 badge, showing the LED eyes on the smiley face logo

variety of apps to show off the community's creativity. To give some idea of the complexity involved, the 2015 German Chaos Communication Camp (CCC) rad10 badge, featured a software-defined radio transceiver, and the 2017 Hackaday Superconference badge sported a fully functioning digital camera.

Electronic conference badges first appeared during the last decade, and were fairly simple

affairs. Joe Grand's 200 badge for the DEF CON 14 hacker conference, for example, was a circular PCB featuring the event logo, with the eyes of the smiley face being formed by a pair of LEDs, on the back of which was a coin cell and a PIC microcontroller to vary the LED flash pattern. A hardware hacking contest was run for the badge, something which has become a tradition at events.

Meanwhile, equivalent events were taking the same path. The CCC Sputnik badge had an RF proximity detector system as a demonstration of population tracking for a real-life version of Harry Potter's Marauder's Map.

Over the decade since those early beginnings, there have been many badges produced, all with a spin appropriate to their particular event. It is normal now to expect an LCD screen and some form of wireless module, and badges will normally expose their processor's spare GPIOs and other interfaces to a set of headers, inviting attendees to try their hand at producing add-on boards. Meanwhile, the badge will typically boot into a menu system with pre-installed apps from the event organisers, and a store containing apps contributed by the event attendees.

The official conference badges are, however, only part of the story. Impressive as they are, we would

MR ROBOT BADGE

The 2017 Mr Robot badge, from Brian Benchoff, took a piece of iconography from the US TV show of the same name, and added an LED matrix with a driver chip fed by an ESP82 processor. Its artwork took a custom fabrication process, involving not one but two silkscreen colours, and a custom coloured solder resist layer to provide the skin tone. It carried a game involving assembling a hidden zip file with clues delivered in a series of tweets, then disassembling its contents to find an image hidden at an address in the badge's ROM.



Credit
Brian Benchoff

probably not be bringing you this article were it not for the unofficial badge culture, known informally as Badgelife, that has developed alongside them. Subgroups attending conferences produce their own badges for their members, and individual hackers and groups produce badges merely as an expression of the art form. Into these unofficial badges goes the best that can be found, both in electronics and computing, and in aesthetic design through PCB artwork and shaping. Away from the confines of event requirements, the unofficial badge designers have free rein to exercise all of their prodigious talents.

BADGERS

This year's DEF CON 25 sported the most impressive Badgelife offerings yet, with a common wireless standard shared between the various groups producing badges, allowing them to communicate. The array of badges on offer included plenty of blinkenlights, and more than one that took its artistry beyond the PCB into textile and even woodwork, but since this was DEF CON there was much more to them than simple visual appeal. There were badges with cryptographic puzzles, ones with backdoors waiting to be found through the wireless network, and even a badge that doubled as a miniature quadcopter.

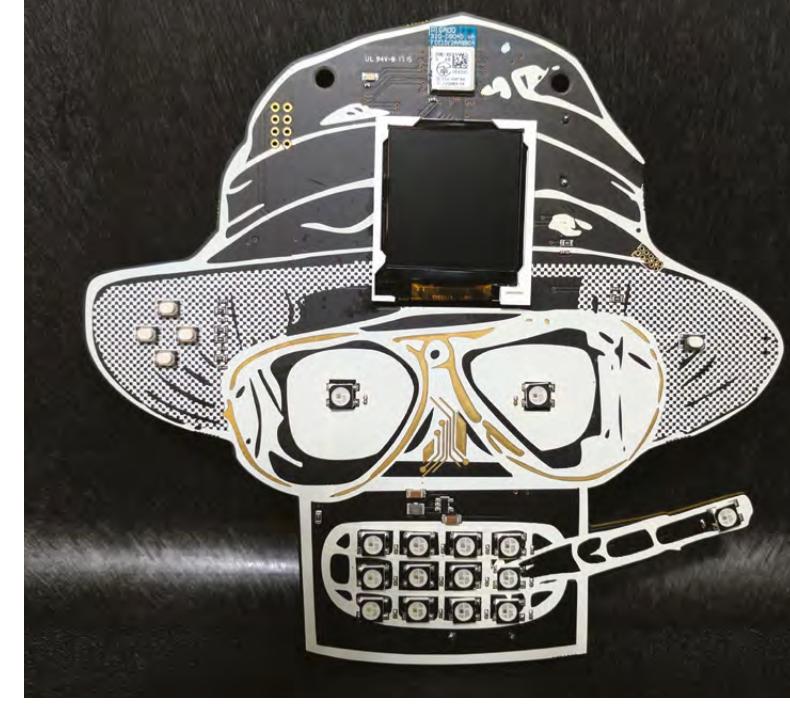
If your attention has been captured by the badge scene, there's nothing to stop you having a go at designing one yourself. Sure, some of the badges mentioned have super-powerful computing and novelty capabilities, but even the simplest circuitry can still make a sought-after badge. The promotional dog badge from the hardware market Tindie, for example, has only a couple of flashing LEDs but, coupled with some well-executed PCB art, it made for a much-admired piece of swag. But, this is your canvas for your artwork, and there are no rules to say that you need to even use a PCB. Impressive badges have been made using stripboard, and even with dead-bug-style soldering.

BUILD A BADGE

When you set about designing your electronic badge, though, there are some requirements that you will have to consider before starting. For example, will the components be robust enough to withstand several days of bouncing around on a lanyard dangling from someone's neck? And are they well-enough protected that they resist damage from snagging on a wearer's clothing? Then there is the tricky question of power; whichever circuitry →

FEAR AND LOATHING ON A BADGE

The 2017 Bender on a Bender badge from ANDIXOR built on the team's offering from a previous year for a badge that had everything. Impressive custom PCB artwork, more blinking LEDs than you could ever need, multimode wireless connectivity, integration with Android phones via an app, and emulation of the CHIP-8 programming language for emulation of classic games. It also had a multiplayer computer security game between all the Bender badges, in which they formed a peer-to-peer botnet, and players became sysadmins of their own badge, with the aim of both protecting their own badge and launching attacks on others nearby.

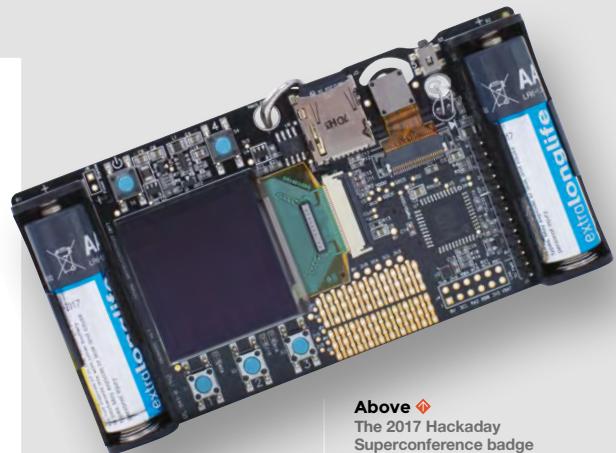


“The components must be robust enough to withstand several days of bouncing around”

Above ◁
The ANDIXOR Bender on a Bender badge

EMF CAMP

The British Electromagnetic Field (EMF) hacker camp has run every two years since 2012, and seeing its official badges side-by-side provides a fascinating window into the evolution of badge technology over that time period. The 2012 badge was an Arduino-compatible microcontroller board based upon an ATMega32U4, but whose only visual interface was a pair of LEDs. By comparison, the 2014 badge was an ARM-based Arduino Due clone, with a joystick and LCD display, and had a series of apps on board, as well as basic wireless connectivity. The most recent badge from 2016 had a much more powerful ARM processor, a colour LCD screen, WiFi, and a full software suite including an app store for which attendees could write their own apps in MicroPython.



Above ◊
The 2017 Hackaday Superconference badge powered by AA cells

you choose must not drain your choice of battery too quickly. You will see badges with everything from CR2032 coin cells to 18650 lithium-ion cells but, if your circuitry requires the latter, then whoever wears the badge may end up with a sore neck.

If you're making a PCB for your badge, and particularly if you're making one incorporating PCB art, you will need to familiarise yourself with the offerings from the various PCB fabrication houses. Boards made in the usual way with a solder mask and silkscreen will offer you four colours to play with, but the exact palette at your disposal will depend upon the work of your chosen board

house. Solder masks can be had in a huge variety of colours, aside from the familiar green. Some of the more expensive Badgelife boards will even use custom combinations of solder masks; however, that level of sophistication requires a relationship with the board house not usually available to mere mortals.

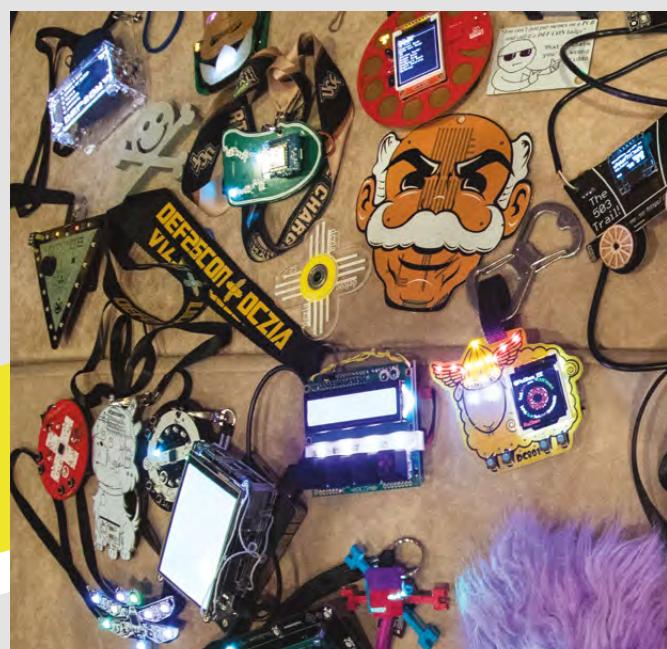
We've seen some of the directions that badges have taken over the past decade, but how about the future? It is certain that the available processor power will increase hugely over the forthcoming years, and that ever more exciting peripherals will be included. An easy prediction is that some official badges will tend towards standardisation around a

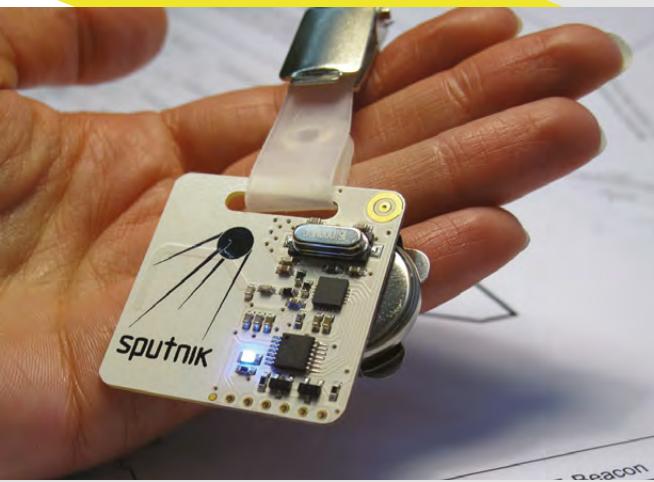
“

Perhaps future badges will break away from the PCB design entirely, and feature custom cases

”

Thanks to Matt Westcott for the loan of some of the badges used in this piece.





Above The Sputnik badge from CCC 25C3

particular processor and software stack, to deliver a known quantity in terms of being ready to run on event day one. Meanwhile, it is probable that we will eventually see badges running a fully fledged operating system, such as a Linux distribution of some kind. In manufacturing terms, it's also an easy prediction that people will find ways to produce more colours on a piece of PCB art, and we should also expect more custom parts appearing. Perhaps future badges will break away from the PCB design entirely, and feature custom cases more of the type you would expect to see on a commercial product. Whatever happens, the Badgelife scene will be an exciting one to watch over the coming years. □

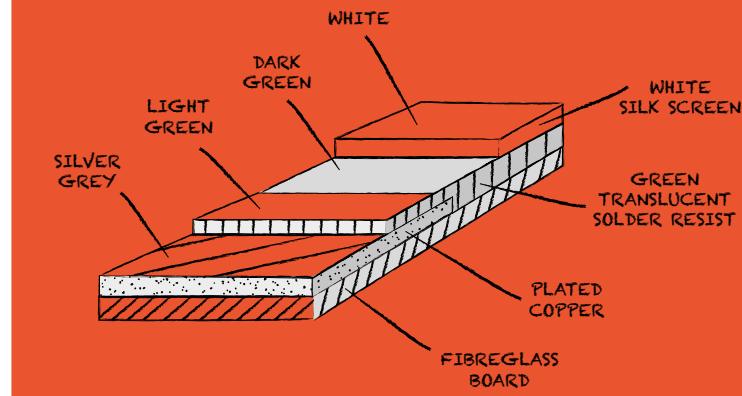


Left The impressive collection of unofficial badges at DEF CON 25

Credit
Cat Murdock
(@catmurd0ck)

PCB: THE ARTIST'S CANVAS

A PCB takes the form of several layers: the fibreglass substrate; the copper conductors, which are usually plated with solder; the solder mask layer that can be almost any colour but is often a transparent green; and finally the silkscreen that would normally carry white text. By combining these layers a simple four-colour palette can be created, with the plated copper being a silver-grey, the solder mask over copper being a light green, the mask over bare board being a dark green, and the silkscreen being white. A PCB artist takes their image and carefully maps their desired image to these available colours, before creating it in a vector graphics package such as Inkscape, from which it can be exported with each colour as the various layers to a PCB CAD package.



We Learn CROCHET

HackSpace magazine travels to the land of dragons to learn a new skill

By Ben Everard



The British late autumn weather is at its most typical as my train pulls into Swansea: it's dark, drizzling, and windy. The Christmas illuminations are up, and the tumbling castle is lit in blue, but my train was late so I've no time to enjoy the view. I put my head down and walk briskly down the high street. I spy a Hackspace logo through metal railings and press the intercom to go in. Buzzed through the doors and, after climbing the stairs, I'm welcomed into Swansea Hackspace by Tim Clark, one of the Hackspace directors.

It's Crafternoon (second and fourth Wednesday of the month), and there's already a group of people

working on their projects. Swansea Hackspace member Sophia has kindly agreed to teach me a new skill, crochet – the art of knotting yarn into fabric using a hook.

I can sew just about well enough to repair clothes, but that's about as far as my textile skills progress. Crochet seems the perfect place to start, as it's the fabric-making craft with the smallest number of instruments – you need only a single crochet hook to get started – so I assumed it's the easiest to learn, though this thought isn't backed up by any evidence.

She begins with a demonstration. She makes a loop with the yarn then, with a hook, beings to loop

Below The Swansea Hackspace Crafternoon





Above With a little experience, you can crochet complex shapes

and pull the yarn, then some things happen that I don't quite catch, and there's a little square of fabric.

At its most basic, crochet is similar to finger-knitting, where you pull a loop of thread through another loop, and tighten the first loop around the second. However, with crochet, these loops are intertwined and sometimes you're pulling a loop through two other

Crochet seems the perfect place to start, as it's the fabric-making craft with the smallest number of instruments

loops. The skill, as far as I can tell, is in remembering where you are in the pattern and pulling the right bits through the right loops.

Sophie sets me to work trying to make a granny square – essentially a chess-board pattern of square knots of crochet and holes.

Before I can start to get into patterns and loops and in-and-out bits, there's a much more basic skill to learn – I'm struggling to master keeping the yarn on the hook. The difficulty is that there are three things to control (the hook, the yarn, and the thing that I'm crocheting) yet I only have two hands. Sophie encourages me to solve this by wrapping the yarn around my finger, but it's hard to keep it tight and, without the yarn under tension, the crochet becomes loose, and the whole thing becomes more of a tangled mess than it's supposed to be. When some loops are tight and some slack, the distinction between loop and not-loop starts to fade and the result is anarchy.

LEARNING TO CROCHET

The basic equipment for crocheting is just a crochet hook and some yarn. If you start with just the basics, you can easily get everything you need to start for under £5 (and that's if there's not a hackspace or makerspace near you with equipment you can use). You do need to make sure that you have the right-sized hook for your yarn (check with the seller for details). It's easiest to learn from an experienced crocheter, if possible – this may be a crocheter you know – or there are courses run frequently around the UK. That said, it's certainly possible to learn from either a book or a set of videos, but you will need a little more patience and problem-solving ability to work out what you need to do. As with so many skills, practice is the key to learning to crochet.

Crochet has one saving grace when compared to knitting (the other main way of making fabric from yarn): it's straightforward to undo mistakes. You can take the hook out of the material, pull the loose end, and watch the mistakes gradually pop out, much like going back to a saved point in a computer game when things go wrong. Without this, I'm pretty sure I'd still be in South Wales tangled in a ball of yarn.

First starting to crochet is a deeply unsociable thing. I spend most of the first hour in Swansea Hackspace staring fixedly at a tangle of yarn trying to work out if what I've done is correct, and if it is, what I'm meant to do next. However, building up the granny square is fundamentally a pattern: three double crochets, one link, three double crochets, →

one link, three double crochets, three links and then repeat. After a few iterations of this loop, my muscle memory starts to develop, and I can do each of these steps without staring fixedly at the yarn. It starts to become an automatic process, a rhythmical movement that relaxes my mind as the fabric slowly (and slightly irregularly) appears in my hands.

Swansea Hackspace Crafternoon is a social event and, as the rhythmical toing and froing of the crochet

THINGS TO MAKE

Once you've mastered the basics of crochet, you can move on to more advanced projects. There's loads of ideas to inspire you at ravelry.com. A few popular options are:

Blanket

Using basic granny squares, you can build up a large (or small) blanket to keep you warm through the winter months

Beanie hat

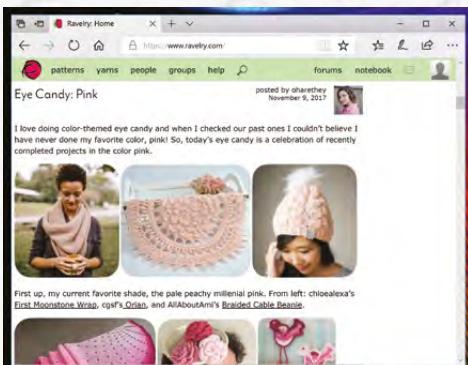
The technique we used to create a ball can be used on a larger scale to create a beanie hat

Cuddly toys

As you master more complex patterns, you can knot your way to complex cuddly toys

Bags

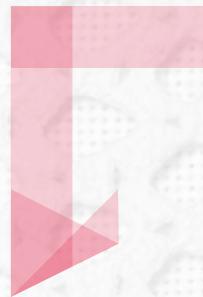
The patterned fabric created from crochet can create striking-looking bags



hooks helps people unwind, the conversation flows. I'd love to be able to report the gossip from my night at Swansea Hackspace, but unfortunately my muscle memory hadn't quite built up enough yet to join a conversation for long enough to work out what was going on before having to return my attention the tangle in my hands.

I'm not going to claim my granny square was a triumph (if you look closely you might notice that there's a square missing), but it is roughly square and it is crocheted.

Sophia's keen to put me to the test, and ups the challenge from two to three dimensions. Unlike most methods of converting a thread into a fabric (like knitting or weaving), with crochet you can vary the amount of knots in each line (or loop in the case



Below Sophia Komninou, our crochet tutor for the evening, demonstrating the basic technique





of circular crocheting). This means that you can make the fabric curve into a third dimension. The simplest way of doing this is by creating something that's broadly spherical, by starting with a circle and varying the number of knots in each ring.

The inner ring has six knots. The second twelve, then eighteen – well, approximately anyway. I never quite mastered the art of remembering where I was in this pattern. With this pattern, the fabric gently curves upwards, and it can be reversed to bring the curve back together again to form a complete sphere.

You can judge my success for yourself, but I must confess that Sophia finished the ball off for me. Not because I couldn't do it, but because I was

pressed for time with the last train home leaving shortly – at least, that's what she told me.

So, can you learn to crochet in an evening? With some guidance, you can learn to make something in an evening, and end up knowing enough to learn more by yourself. Applying the same basic skills, but in different patterns, you can build up complex 2D and 3D designs. □

Above ↗
My not-so-square
granny square

Above Left ↙
What a granny square
should look like, if
done correctly

**With some guidance, you can learn to make
something in an evening, and end up knowing
enough to learn more by yourself**



Left ↙
The slight stretch in
the yarn helps cover
up the worst of the
imperfections

SUBSCRIBE AND SAVE!



SAVE UP TO
35%
on the cover price



FROM JUST
£4
per month

Visit: hsmag.cc/subscribe

SUBSCRIBER BENEFITS ↓

SAVE UP TO 35% ON THE PRICE

FREE DELIVERY TO YOUR DOOR

EXCLUSIVE OFFERS AND GIFTS

GET YOUR COPY BEFORE STORES

Rolling subscription from £4 a month:

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

12-month subscription from £55:

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

PLUS

Digital subscriptions from £2.29 per month



Available on the
App Store



GET IT ON
Google Play

Visit: hsmag.cc/subscribe

THE CANUTE



By Liam Smyth

How Bristol Hackspace is bringing Braille into the digital age



or thousands of blind, deaf-blind, and partially sighted people around the world, Braille is literacy, employment, and independence.

Technological advances for blind and visually impaired people have, however, often failed to keep pace with those developed for sighted people. Working from the Bristol Hackspace, Bristol Braille Technology is committed to developing radically affordable digital Braille technologies.

While e-reader devices have dramatically increased the accessibility of the written word (a £200 e-reader can now store as many books as a grand library), access to technological solutions for Braille readers has been more limited. Braille display units currently on the market can only show one line of text, of between 12 and 40 characters. A useful display, with 40 or more characters, can easily cost upwards of £1500. To read just this article on the cheapest 12-character display, the user would have to refresh the unit 395 times.

Below ■
"It was the best of times, it was the worst of times"

Bristol Braille Technology has developed the world's first multi-line Braille e-reader, the Canute, in order to provide low-cost access to digital Braille. Through our unique electromechanical mechanism, we can display nine lines of 40 characters each, for a total of 360 characters of Braille per page. The Canute can store thousands of Braille files in a single standalone unit. We have achieved all of this in five years, working from the Bristol Hackspace.

The potential impact of this technology is huge. Consider the education sector: only 8% of STEM (Science, Technology, Engineering, and Maths) textbooks are currently published in Braille format. This represents a significant barrier to access for blind and partially sighted children and young adults. Audio technologies cannot adequately convey tabular information, and existing single-line displays cannot render this information with the context necessary for complete understanding. The Canute can render tabular and algebraic information in Braille in context, allowing – for the first time – information only currently available in print to be accessed in Braille.

To give another example, imagine a blind law student required to study and understand a complicated legal test case. To listen to arcane legal arguments narrated by Microsoft Sam via a screen-reader programme is not an attractive proposition, but unfortunately for many blind students there is currently no practical alternative to text-to-speech. Coupled with





Above The Christmas celebrations with the Canute at Bristol Hackspace

automatic transcription software, any written information can be converted by the user, and displayed by Canute. Research tells us that blind people who are Braille-literate are twice as likely to be employed as those who are not, and technologies that can promote and encourage Braille literacy have a key role to play in ensuring that today's blind children have the same life chances after they finish education as their sighted peers.

STOPPING THE TIDE

Bristol Braille Technology is a social enterprise. Our primary business objective is to make multi-line digital Braille affordable to users. Working from the Bristol Hackspace, we have been able to reduce our development costs such that we expect to bring the Canute to market in 2018 at a drastically more affordable price than other, less capable devices currently on the market.

Working from the Bristol Hackspace has offered us numerous advantages as a business. The first five prototype e-readers were built almost entirely using Hackspace tools and resources. As we moved through the proof-of-concept stage, this allowed us to work closely with local Braille readers, testing, developing, and refining our designs in-house. Access to high-end equipment, like the laser-etcher, allowed us to produce complicated electromechanical designs, while keeping our overheads low.

The development of early prototypes at the Hackspace enabled us to test mechanisms, interfaces and functionalities with local Braille readers. This data, in turn, enabled us to demonstrate the potential social benefits of our technology to backers, which allowed us to secure grants and support from organisations like Comic Relief, the Nominet Trust, and the Raspberry Pi Foundation. By targeting funding requests and capitalising on the resources available to us through the Hackspace, we have been able to show our



Above Russell Couper designing the circuit board at Bristol Hackspace

investors that we are able to produce low-cost technological solutions to address complicated social issues. This rapid, low-cost prototyping and development process would have been impossible without the resources available to us at the Hackspace. Ultimately, we have been able to pass these savings on to the end user, meaning that we will be able to retail the Canute e-reader for around £800, a price point comparable to a new iPhone or mid-range laptop.

We can display nine lines of forty characters each, for a total of three hundred and sixty characters of Braille

Ultimately, the Canute project aims to give blind people a choice in how they access their information. Just as sighted people may choose to listen to an audio book on the train, it would be wrong to suggest that a Braille e-reader will replace audio technologies, or be adopted by all blind or partially sighted people. That being said, Bristol Braille Technology aims to offer individuals who choose to access information through Braille affordable access to a low-cost multi-line Braille e-reader. By working with the blind community from the Bristol Hackspace, we have been able to harness technology to address significant social needs. It is at once impossible to imagine the Canute project succeeding without a hackspace, and heartening to envisage other individuals, groups, and projects utilising hackspaces to develop technologies that enhance the quality of life for individuals who have been historically overlooked by mainstream technology companies. →



A NEW APPROACH TO DISPLAYING BRAILLE



By Ed Rodgers

T

he Canute is a radical departure from any existing, refreshable Braille technology. It is more mechanical, more Heath Robinson and, consequently, far more affordable. It is also the first time digital Braille can cover a whole page. And it runs on the Raspberry Pi Zero.

Braille was invented in 1824 by a Frenchman called Louis Braille as a tactile writing and reading system for blind people. It involves embossing a series of dots into paper, six per letter, or cell, spaced one tenth of an inch from the next.

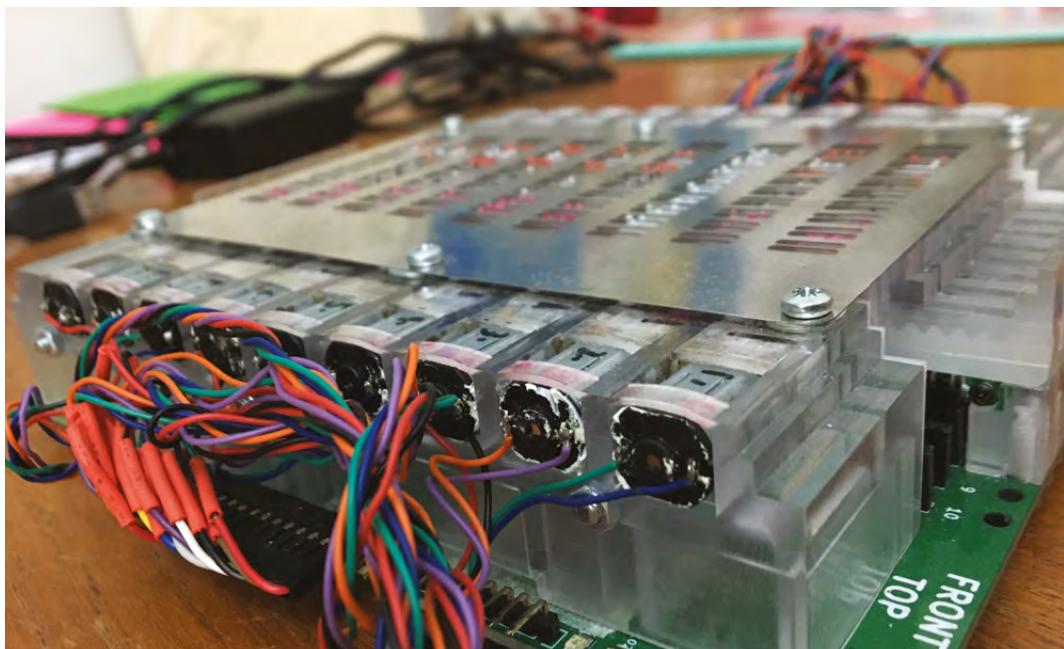
Refreshable Braille was developed over the latter half of the 20th century. It simulates paper Braille through a series of pins that raise or lower to represent digital text. They are typically either connected to a computer to act as a terminal ('Braille display'), or have their own built-in battery, computer, and keyboard ('Braille note-taker').

Right Using off-the-shelf parts has dramatically reduced the cost of developing the Canute

Existing refreshable Braille displays all rely on one technology to raise and lower pins: piezoelectrics. This involves the use of long strips of expensive ceramic under every pin. It is reliable, but hugely expensive and limits the displays to a single line of text at a time. The technology has remained fundamentally unchanged for decades.

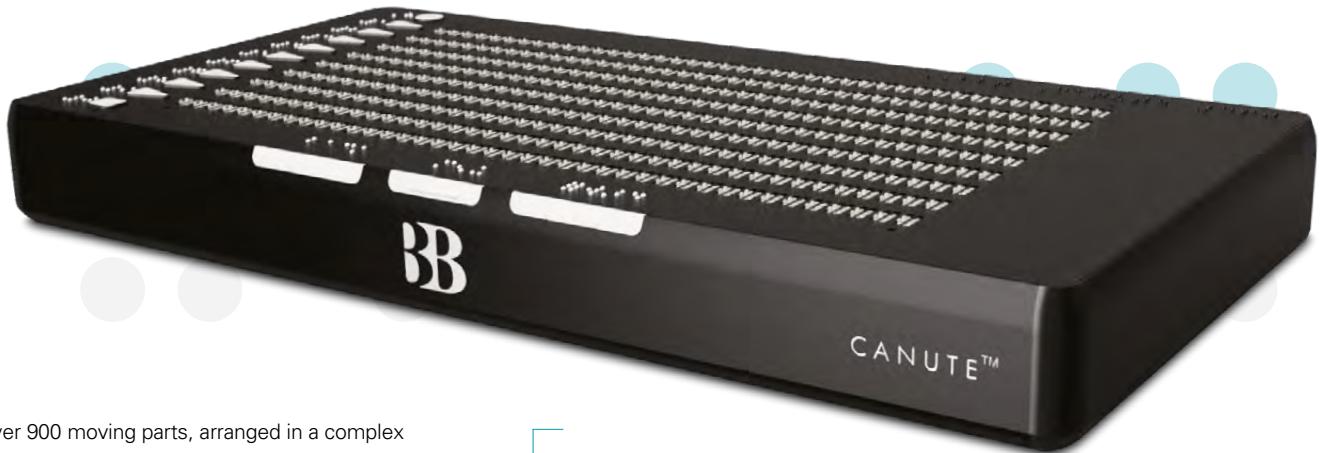
By contrast, the Canute uses off-the-shelf Raspberry Pi Zeros and motors from existing products, coupled with a custom mechanism of cogs, racks, and steel bars. Whereas the piezoelectric approach costs upwards of £40 per cell, this approach is closer to £2 per cell, and has no limit on the number of lines.

The core principle of the Canute is apparent from the first moment it is turned on. 720 tiny plastic octagons, representing 360 cells, spin under your hands in neat lines, from the top of the page down to the bottom. The machine whirs and clicks, and you can feel the slight vibrations of a page turning. This is not an iPad. It has



H





over 900 moving parts, arranged in a complex instrument more akin to a piano than an LCD screen.

On every side of those 720 octagons is embossed a unique combination of up to three dots to represent half a Braille cell: a three-bit pattern with eight combinations. As they are spun between one state and the next, a line of text changes and presents itself to the reader.

RIPPLE EFFECT

When the lines change, one after the other in a ripple down the page, you feel the dots rapidly retracting with a faint chunk, the whir of movement, then a new line is embossed. The Braille is smooth and hard, more like signage Braille you might find on a door than either paper Braille or the soft nylon of Braille displays. You run your hand over it, just as a sighted reader would run their eyes over print; left to right, top to bottom, in paragraphs, bullet points, and headings.

Canute is a stand-alone device that reads Braille e-books from the memory of the Pi Zero, one page at a time, like a Kindle. It has a user interface written in Python and designed with several hundred Braille enthusiasts from a group called the Braillists Foundation. This drives the mechanism built in the Bristol Hackspace over five years and 13 prototypes.

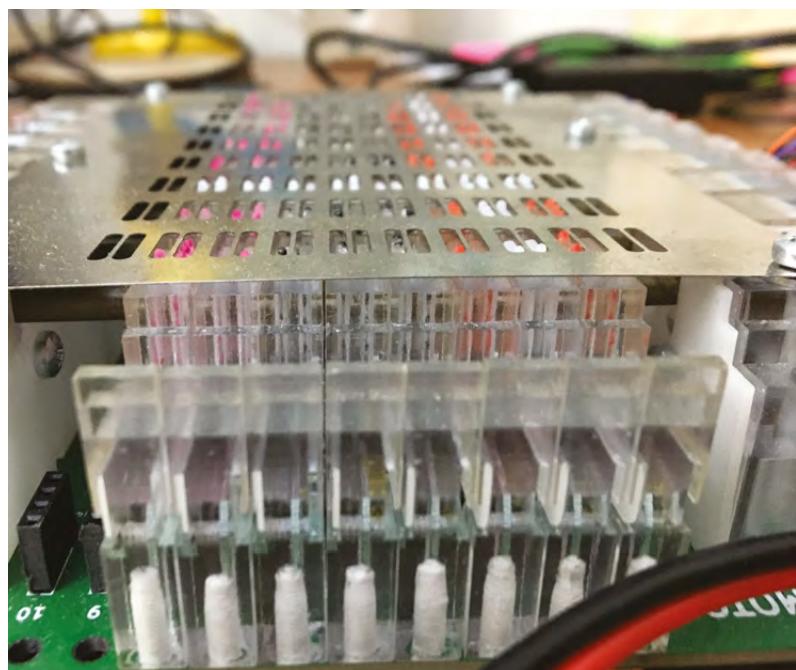
The team behind the Canute, Bristol Braille Technology, created it to help reverse the decline in Braille use. It will achieve this by being dramatically cheaper, by eschewing piezoelectrics, by having a page rather than a single line (thus being far more useful for music, maths, and other subjects), and by being designed by, rather than for, the Braille reading community.

Above all else, the Canute is proof that, in this age of nanometre-scale electronics and multibillion-dollar tech behemoths, you can still achieve remarkable advances with a small team, working from a small workshop, with a limited budget, and active imaginations. □

Above ■
The production prototype getting ready for manufacturing

720 tiny plastic octagons,
representing 360 cells, spin
under your hands in neat lines
from the top to the bottom

Below ■
The mechanical Braille mechanism



HackSpace magazine meets...

HACKER HOUSE

How two crafters turned a hobby into a business

It's not just physical devices that come out of hackspace: by bringing like-minded people together, hackspace can also act as perfect incubators for businesses. One of these that has come across our radar is Hacker House.

From humble beginnings messing about with drones, to advising banks on cybersecurity, Hacker House has come a long way in a short time. And it's the brainchild of just two people: Jennifer Arcuri and Matthew Hickey.

As hardware hackers, they make things to solve problems and, as security experts, they get to advise businesses on how to keep their computer networks (including devices) safe online. We braved the snow and storms of northern England to speak to them about the promise of the Internet of Things, how to keep your smart devices safe from North Korean hackers, and the joys of smashing drones up. →

Below ■

Jennifer Arcuri: "We thought we'd give it six months, see what works in this sort of maker community thing called Hacker House..."



Right, tell us about Hacker House: what is it and how long have you been doing it?

JA About a year as a business, but it started out as a series of events, hackathons, out of a makerspace in East London that we ran, a 1200 square foot apartment I had in Shoreditch. I was just fascinated by building and making things, I was part of the maker movement in New York, and then there was lots of cool stuff in San Francisco around 2014 when I was there.

Everywhere I went, there seemed to be this wave of things being created. I got particularly interested in hackerspaces after visiting 57 spaces in the US along the East Coast, DC, New York, middle of America and the West Coast, just seeing what they were like. People would not only build a thing, but were then making it smart, connecting it to a network. The community that collected around that; some people had development skills, some people were good at building stuff, and I loved the mish-mash of collective thinking that happened.

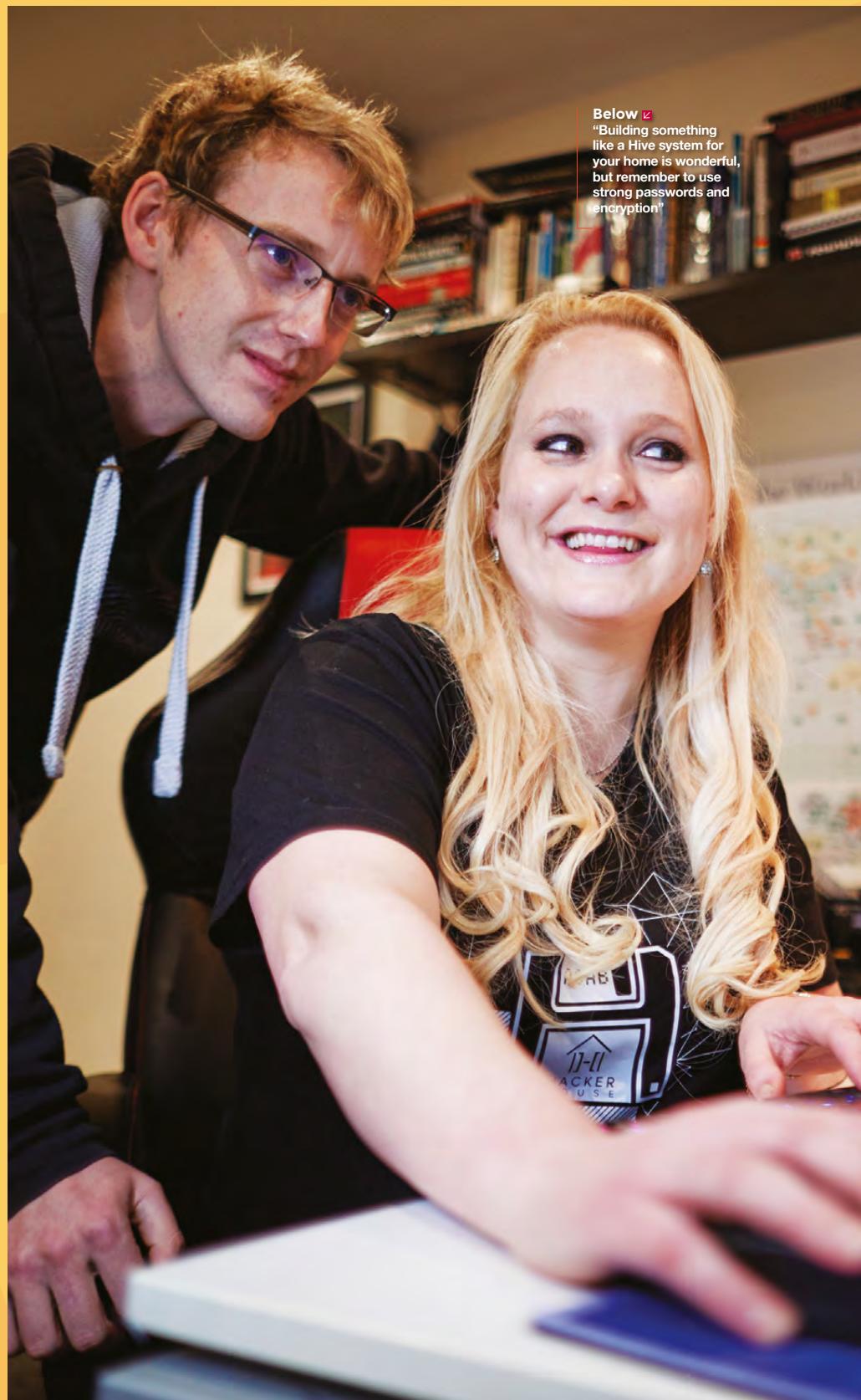
I thought, 'How am I going to get a bunch of people to do that in London?' There were various pockets, but when I first started doing it there was really nothing... overwhelmingly awesome.

MH We started out with 15 hackers, some pizza and beer, and we got to work on various cybersecurity projects. It was like the hackerspace movement applied to cybersecurity. And through Jennifer's work doing that, we met and hit it off, and now we've formalised it into a business where we provide services and training to organisations who are trying to understand cybersecurity matters.

JA We came here [to Cheshire] because we realised there was a lot of stuff we could do with drones, that we couldn't do in London. We came up here to be 'off-grid'. We did a corporate retreat up here, there were three big banks and all these different corporate types, and the first thing they said was "where the hell are we?" It was in a big country house just up the road.

We thought we'd give it six months, see what works in this sort of maker

Below ↴
"Building something like a Hive system for your home is wonderful, but remember to use strong passwords and encryption"



community thing called Hacker House. And then we decided through those six months, what was a viable business, and what was a social movement. We still have a community side to what we do; we publish regular challenges on our blog to try and encourage people to pick up cybersecurity and understand how to do it. But, like I said, we've had a baby in the last year, so give us a few more years and we'll get back to our community stuff.

There's also more space here for flying drones. We were trying to commercialise drone opportunities, but cybersecurity just took over the business. We used to 3D-print little flying quadcopters and teach people about the science behind how drones worked, take them out, and then maybe fly a full-sized one. We would 3D-print all the components so, when we'd fly them into walls and trees, we'd just have to go and salvage the electronics and build a new one.

We'd go outside, light a fire and play some poker, and it worked great.

MH That's what 3D printing is for: build your own stuff. There's no need to buy a bunch of drones when you can just make a bunch of drones yourself, right? You can rip them apart, figure out how they work, then make them yourself. Why buy it when you can build it? That's absolutely where we came from. We were trying to understand what can we do with all this cool stuff! We've got great talent, all these people who understand how the technology works, and we've got businesses that are crying out for cybersecurity skills, and it just made a perfect fit to keep the rebellious spirit, the hacking vibe, and the lifestyle in the business and turn it into what it is now.

We do professional services and we run these workshops and events where we train people to do the careers that we've been doing. Cybersecurity stuff, ethical hacking, vulnerability analysis, stuff like that. We build training modules, we build things to teach businesses, and we kind of embody that creative spirit in the work that we do. We work with a lot of companies

that might have been victims of cyber-attacks, or maybe want to understand how cybersecurity impacts them. That's the niche that we've been able to corner.

Do you get a spike in business every time something like Talk Talk happens?

MH Sometimes. It works two ways for us. When things happen in certain industries, there's usually a sudden rush of 'oh my God, this could happen to us!' So they suddenly want to get in touch with companies like ours to work on better security for their infrastructure. And that's usually a result of something else happening. We actually prefer to work with clients that are adopting cyber into the workplace as part of their culture, because we're able to work better with teams who are more proactive in cyberspace, who aren't just reacting to a competitor getting hacked and panicking.

That's what 3D printing is for:
build your own stuff

It's better to adopt secure thinking into the workplace and empower the users. Telling someone that they've got a bad password and punishing them isn't the right way of doing it; it's much better to empower the people in these organisations to understand how these attacks happen.

It's like your house. If you look at your house, you know to lock the windows and doors when you're going out, because that's what a burglar is looking at. But, we don't do the same thing with computer networks. We don't analyse those networks, we don't probe those networks for ourselves, so we don't know how they're broken, and we don't know how to defend against them.

JA Inherently all computers are broken. We spend so much time trying to explain this to people. Computers are analytical machines – they were designed for problem solving; they were never designed for

secure banking, they were never designed for secure communications. These are just things that we ended up using computers for. So, inherently all computers ship with this flaw that they were never designed to be something that you could use to replace gold or whatever.

Technology has transformed much of the world around us, from social interactions to financing, to the way in which we pay our utility bills and socialise with one another. This technology has been something that we inherited with that problem. They are broken, and they will continue to be broken, and people will continue to find ways to outsmart and beat the computer. Our motto is that all the computers are broken. If you think like that, we can try to fix them: try to understand the ways in which they are broken and engineer solutions around the problems they have.

So, the Internet of Things. It has the potential to transform so many areas of our lives for the better, but the downside of that is that there's an unprecedented amount of data flying about that could be intercepted. Is it all it's cracked up to be?

MH The Internet of Things is an empowering thing. We're going to have more interconnected technologies, more convenience, we're going to get more visibility all kinds of ways. People will innovate in the space, improving the way they prepare food, the way in which we do self-medical diagnosis, so the opportunities for home treatments will be astronomical... it'll be really cool.

JA We chose to have it.

We chose to have it? Did we really?

JA We, as consumers, demanded more immediate 'now now now' products. We loved the fact that we could have our devices connect to each other, connect to a network, communicate, do cool things, all remotely controlled. All of that seemed to →

fascinate us, and continues to do so. It just then asks the question of where security comes into the picture because, up until now, it just hasn't.

Right. You don't need security for your kettle. But then you put it on the internet, suddenly it's a computer, and you need to protect it from all sorts of stuff.

JA Most people wouldn't think about security for their kettle.

MH Security is an afterthought for most of these vendors, because it's not something that's being demanded of them by the consumers.

We've been looking at embedded devices and Internet of Things technologies in the security space since around 2000. Now there are more broadband routers, there's more interconnectivity in the home. A lot of these products were never designed with consumer security in mind, because consumers didn't really demand that.

What we were finding was that a lot of these technologies were shipping with insecure credentials programmed by default into the box, insecure programming protocols, HTTP, Telnet, exposed engineering capabilities; that meant that someone who was able to get physical or remote access was able to reprogram these devices. As consumers, we never really demand: "has this product been security tested? Has it undergone any evaluation? What kind of standards has it followed?"

Do you need your internet-connected fridge to connect to your business PayPal account? You probably don't. You need to work on building better security into devices, but that won't happen unless consumers are demanding them and innovators, and people who are building them, are aware of the need for such matters.

If you look at ransomware attacks and outbreaks like Mirai; Mirai just scanned the internet for a common set of weak passwords, and it got into so many embedded systems because more and more of us are buying internet-connected baby monitors, internet-connected toasters,

fridges and, as we increasingly connect devices into that ecosystem, we're exposing them to all manner of attacks. We're just not there yet where we, as consumers, are demanding the security features in our embedded platforms.

Do you think the public has even started to become aware of the potential threats that could come from poorly implemented devices or is there still total ignorance?

MH I think the public are aware, but when you go into a high street or a retail environment or you're shopping online, how many of us look for security certification tags? How many check whether a device uses SSL, if it's using encryption, how it's managed, how it will work with firewalls? We're demanding a lot of convenience and ability to automatically configure things, but that comes with a greater risk to the security of the devices.

Consumers are more aware their data is valuable, but they're not fully aware of how to secure it, because it's a complex topic.

One thing most security people agree on is that there are some things that just shouldn't be attached to the internet, no matter how convenient it might be. Voting machines, for example. Is there anything else you think shouldn't be on the internet?

MH Many things. Insulin pumps. Pacemakers. Things that have control of your car; do we really need those to be connected to the internet to tell you how high your tyre pressure is? There are certain elements of connectivity that will add additional attack surfaces, that will make our technologies more vulnerable. Things like pacemakers, things that have a critical effect on human life; when they are built, it should be with the utmost due diligence for security guidance and security handling.

At the moment, that's something that isn't enforced. It's not something that

Below ↓

Why buy a drone when you can make your own, and put it back together when it crashes?



manufacturers have to do, it's something that will only be done if consumers demand it from people who are building things.

But we can all build things – we can set the example for the manufacturers. If you've got a Raspberry Pi and an Arduino and you're building your own IoT home heating system/kettle/brewery, for example, what should you bear in mind if you want to keep your device safe online?

MH When you're looking at IoT technologies, you want to think about the network topology that you're deploying, so you're mostly going to be deploying a star network topology with a centralised hub that will be gathering these inputs from other sources around the environment, so you should look at how the information is being gathered and sent, is it making use of HTTP, is it making use of SSL, are you securing credentials, making sure that you're not putting in common user names and passwords such as 'admin' and 'admin'?

If you're using wireless technology, ensure that you're making use of encryption, and not just blasting stuff out to the ether.

Broadcasting toilet habits may seem far-fetched, but we did see a startup looking for funding for its toilet monitor. It's bizarre. We have no idea why you'd want to put that on the internet?

MH Monitoring a person's stool output, and everything else, could be a good way to monitor medical conditions in an early onset. If people are going to smart-connect things like their toilets where it will be gathering information for medical use, you might not necessarily want your medical history accessible to just anyone with a web browser. If you're gathering personal information, only take what you need to build your system, don't gather what you don't need. And don't store it if you don't need it! Storing it will lead to extra complications, extra regulations and controls, things like the GDPR.



Left The fingerprint logo on Matthew's fetching hoodie is comprised of the text of *The Conscience of a Hacker*, composed in 1986

Building technology is a great thing to do. If you think about security from the early onset – you're more likely to build a secure product. Many products that are pushed to market in an insecure way, just didn't do this. They didn't do security by design.

Building things that are opt-out privacy by default, making sure users are aware of what they're turning on, what the information is that they're sending to a device, making choices where they're sending to a device, and making choices

JA A lot of those parents should write letters to the manufacturers demanding better security measures. We're going to see it just like anything, it's just that we're all in that honeymoon stage of 'wow, look at our devices talking to each other'. As we continue to evolve in this space we will demand smarter, but more secure, devices. There will be enough of a response eventually, but it's just, how loud do we cry? That teddy bear incident was a perfect example of just making parents aware that they can go to the manufacturer, to make sure that this bug never happens again. There's a baby monitor upstairs; what can we do to demand that the company gives us better security measures? Eventually they will hear, just like in any market coming into its own.

There's definitely more of an awareness around this. Parents are now understanding that there are security vulnerabilities in these great devices, that vulnerabilities exist. It will take time and enough people crying out for vendors to respond, but it will happen. That's me being optimistic. Because with the Internet of Things, the awesome wow, the gain is far greater than any one parent who wants to hold back their child from having these things. I guess it comes with the territory of building a technology-enabled world, you're going to have to have security as more of a responsibility. □

If you think about security from the early onset, you're more likely to **build a secure product**

where you can secure any kind of endpoint that's got encryption or strong passwords, not leaving engineering facilities in there, not leaving debug code – all crucial.

There was a WiFi-connected teddy bear not long ago that was found to be hackable by anyone wanting to spy on children. Do you think any parent would actually buy those if the vendor revealed what it could do, or would the prospect be so scary that the product just wouldn't sell?

RUBBER BANDS



Get anything under control with this wraparound fix



Mayank Sharma

@geekybodhi

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

They have a knack for sneaking into our homes via newspapers, bundles of mail, and cut flowers, and then multiply like rabbits in the drawers. The flimsy-looking rubber bands don't have the same operational value as some of the other household stationary like duct tape. Yet the physics-defying knick-knacks are an easy fix to some exasperating troubles and the simplest means to bear-hug anything you want held together as a group. The bands are made of rubber, which is one of those ingenious hacks that was in use even several thousand years ago in everything from sandals to jewellery. It wasn't called rubber until several centuries later when famed British chemist Joseph Priestley (who also discovered oxygen) accidentally realised that this material could erase pencil marks on paper.

Thomas Hancock then began mass-producing rubber and turned it into a commercial product by 1819. The following year, Thomas Hancock patented rubber springs for various types of clothing. He commercialised rubber by inventing the rubber masticator, which was a machine that had revolving teeth to tear up rubber scraps. These shredded bits adhered into a solid mass that could then be pressed into blocks or rolled into sheets.

However, things made from rubber back then weren't durable and had a tendency to melt in warmer weather. It wasn't until 1839 that Charles Goodyear discovered the vulcanisation process by heating a

rubber and sulphur mixture to end up with a material that was elastic and strong. He patented the process in the US in 1844. But when he applied for a patent here in the UK, it was challenged by Hancock who had patented a similar method a year earlier in 1843. After a tedious court battle, Hancock came out on top.

We have Stephen Perry to thank for the modern-day rubber band. In 1845, Perry began slicing the vulcanised rubber tube into elastic bands. However they weren't mass-produced until 1923 when William Spencer cut bands from discarded rubber products to wrap newspapers to prevent them from blowing across his lawns. Soon after, standards for rubber bands were defined in the United States in 1925, which governed things like the tensile strength and elongation of the bands.

Throughout history, two types of rubber have been used to manufacture rubber bands: natural rubber or latex from rubber trees, and synthetic rubber. A majority of the rubber these days comes from crude oil. In general, to make synthetic rubber, by-products of the petroleum refining process called butadiene and styrene are combined in a reactor containing soap-suds. This creates a milky-looking liquid latex, which is coagulated into lumps that can be melted down and used. The modern-day rubber bands are created by extruding this rubber into long tubes of varying thickness and diameter. These elastic tubes are then sliced into the circular bands that we're all familiar with.

RUBBER BAND HEAT ENGINE



Project Maker
ADAM MICOLOICH

Project Link
youtu.be/dBXL93984cQ

Left ↳
No, you can't
replace the bicycle
wheel with that of a
Bugatti Veyron and
generate enough
torque to outrun
your hatchback



Who better to demonstrate the thermodynamic properties of rubber bands than a physics professor?

In a superb video, Adam Micolich, an Associate Professor in the School of Physics at the University of New South Wales, Sydney, builds a rubber band-powered heat engine to convert heat energy into mechanical motion.

You'll need a bike rim and bolt cutters to cut through a small section of the aluminium spokes. Bend both of the ends of the remaining spoke into hooks and stretch a rubber band between them. Do this for all the spokes. You'll then have to rebalance the wheel so that the centre of mass is at the axle. Adam found this to be the most frustrating part of the project: "The worst part is, because the bands are a bit flexible, the centre of mass wanders a bit, so when you think you have the wheel true, a small shift means it might not be." Once the wheel is balanced, he places a lamp close to the rubber bands and after a while the wheel starts to turn.

When heated by a hot lamp, rubber bands, unlike metals, will contract and become smaller. As they contract, the centre of mass will shift towards the outer rim of the wheel, which makes it unbalanced. This causes the weight to shift and the wheel begins to spin. And this is how the heat energy from the lamp is converted into mechanical motion. □



CURIOS USES FOR RUBBER BANDS



REMOVE STRIPPED SCREWS

Place a rubber band flat over the screw head to give the screwdriver extra grip.



DEPTH GUIDE

Wrap a small rubber band round the drill bit to always drill to the correct depth.



FINGER EXERCISER

Place a rubber band around all or some fingers and thumb and stretch them as far as possible.



SECURE WALLETS

Wrap some rubber bands around the wallet to prevent it from accidentally slipping out of your pocket.



CAR PHONE HOLDER

Pass a long rubber band through the air vents and place your phone between the two loops.



PREPVENT DOORS FROM LOCKING

Wrap a large rubber band around the handles and cross it to cover the lock.



AN EXTRA GRIP

Wrap a rubber band around the lid of stuck bottles and jars to get a good grip.



SLIP-PROOF UTENSILS

Make glass tumblers less perilous by wrapping several rubber bands around them for tighter grip.



PORTION CONTROL

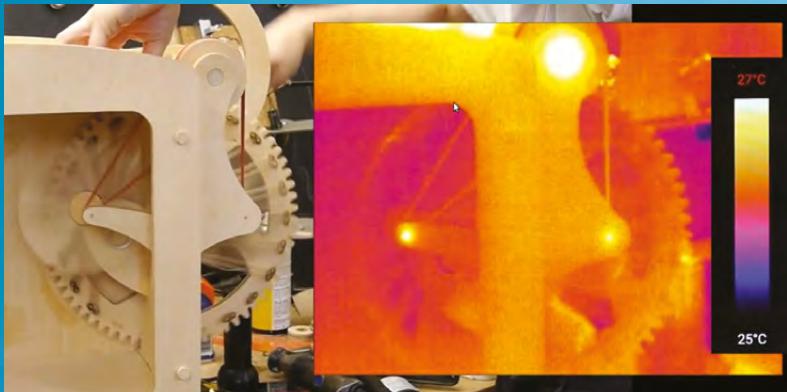
Wrap a rubber band around the neck of a pump dispenser to reduce the amount dispensed.



SLIP-PROOF HANGERS

Wrap a couple of small rubber bands around both ends of the hanger to stop clothes from slipping off.

RUBBER BAND COOLED REFRIGERATOR



There's another thermodynamic property of rubber bands that Ben Krasnow harnessed to build a refrigerator. Ben works at Google's Verily Life Sciences research labs and hosts the Applied Science channel over at YouTube. Ben noticed that rubber bands get hot when stretched and cool when released. This seems to defy the second law of thermodynamics that states that things, like gases, should get hotter when compressed. As Ben explains in the video, it's the rubber band's molecular structure that's responsible for this strange behaviour.

He put this property to use by fabricating a refrigerator with a wheel that stretches rubber bands outside the fridge and releases them inside. He uses a couple of fans to transfer the thermal energy from the bands and circulate inside the fridge. Ben models his parts in SolidWorks and then cuts them with a hand-held CNC router. He cranks the wheel for five minutes and uses a thermal camera to show that the air inside the fridge does cool by a couple of degrees. Again, like Adam's heat engine, the rubber bands aren't efficient enough to replace the traditional refrigerant, but Ben's fridge is a nice demonstration of the rubber band's thermodynamic properties. □

Above The rubber bands contracting are the same as gas being sprayed (or expanded) out of a can

Project Maker
BEN KRASNOW

Project Link
hsmag.cc/GjPdtx



RUBBER BAND GATLING GUN

Tired of shooting your pals with rubber bands from your fingers?

With a weekend and some very detailed instructions from Mark Stevenson, you can build and assemble a hand-held Gatling gun which will help you dominate any crucial rubber band duel. Instead of using CNC machines or laser cutters, Mark cut out the gun using a drill and a jigsaw to make the project more accessible. He has also published all the templates, in PDF format, for the four main components, along with videos to help you fashion them with relative ease. Once you have crafted your gun, you can load over a hundred rubber bands in its barrels and unload them as fast or as slow as you want. Be merciless. □



Project Maker

MARK STEVENSON

Project Link

hsmag.cc/Szeflc

Left ♦

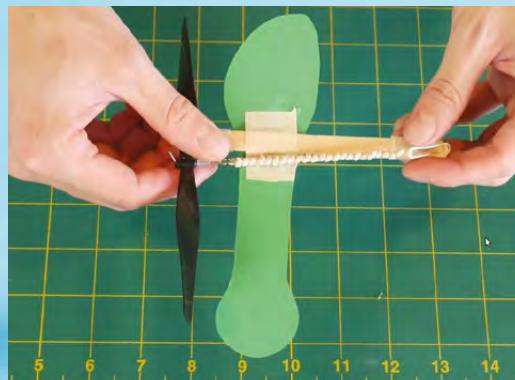
The rubber bands are stretched on the barrels over a thread, which winds when cranked and pulls the bands off the barrel and fires them.

RUBBER BAND HELICOPTERS

Lance Akiyama loves to demonstrate science concepts to kids with hands-on engineering projects. One of the easiest to replicate is the rubber band helicopter.

All you need is a plastic propeller that you attach to a craft stick. Bend and attach a paper clip on the other end and stretch a couple of rubber bands between the two ends. The most crucial bit is the piece of paper cut-out that helps create lateral drag. When you wind the rubber bands and release the helicopter, instead of all the energy travelling down and spinning the craft stick, a majority is diverted towards the propeller and the helicopter takes flight. Every time Lance teaches the project to kids, someone's helicopter would inevitably land on the roof and

they'd feel both distress and pride: "I like to believe that experiences like that are what they think back on when they're thinking of a time they needed to show some grit" says Lance. □



Project Maker

LANCE AKIYAMA

Project Link

hsmag.cc/IHAedl

Left ♦

You can create over a dozen other similar contraptions following Lance's book, aptly titled *Rubber Band Engineer*.

pi-top

Inspiring inventors and creators to seek the skills of tomorrow and create their future, today.



OCR
Oxford Cambridge and RSA



The modular laptop with sliding keyboard



8HR BATTERY
LIFE



14" FULL HD
1080P SCREEN



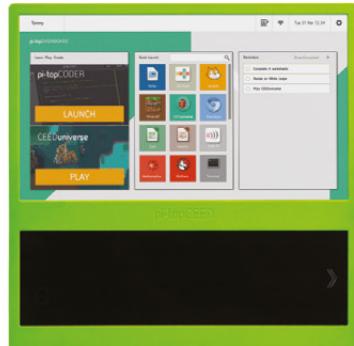
180°
HINGE



CUSTOM
PASSIVE COOLING
BRIDGE



MODULAR
RAIL



The modular desktop



14" FULL HD
1080P SCREEN



MODULAR
RAIL



ADJUSTABLE
VIEWING ANGLES



pi-topPROTO



pi-topSPEAKER



pi-topPULSE

pi-top

Colors 
Raspberry Pi 3 optional

AWESOME INVENTOR'S KIT INCLUDED

20+ projects
to explore

Explore beyond the screen and keyboard by creating with the all-new **pi-top** modular laptop.

Get started with 20+ inventions in the inventor's guide booklet. There are 3 inventor's journeys - Smart Robot, Music Maker and Space Race.

pi-topCEED

Colors 
Raspberry Pi 3 optional

pi-topCEED is the plug & play modular desktop. It's the easiest way to use your Raspberry Pi. We've put what you love about our flagship laptop in a slimmer form factor. Join hundreds of code clubs and classrooms using **pi-topCEED** as their solution to Computer Science and STEAM-based learning.

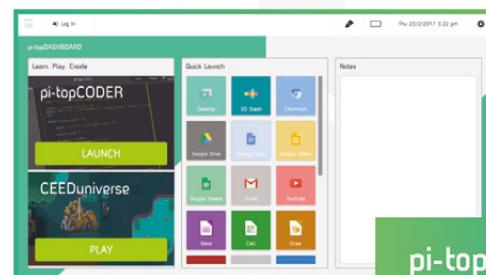
Modular Accessories

Stay up to date with our latest news by following our social media

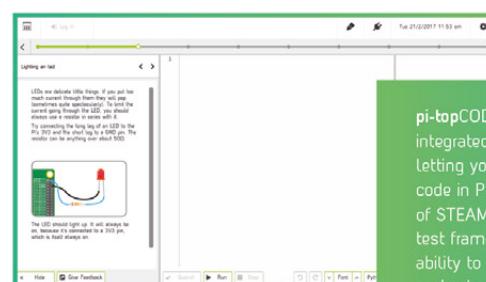


pi-top is an award-winning ecosystem designed to make experimenting, coding and building electronics, simple, affordable and fun. **pi-topOS** is here to guide you through the world of making!

The OCR* endorsed **pi-topOS** (Operating system) platform comes pre-installed on the SD card shipped with every unit. **pi-topOS** software suite lets you - browse the web, - check emails, - create and edit Microsoft Office compatible files. Gain access to dozens of hands-on learning lesson plans with **pi-topCODER** and have fun learning to code with **CEEDuniverse**!



pi-topOS



pi-topCODER has a fully integrated coding environment letting you program hardware, code in Python and learn lots of STEAM skills! Our integrated test framework gives you the ability to assess your own understanding as you learn.



CEEDuniverse
Learn programming concepts through our minigames, for example, learn problem decomposition by solving visual programming puzzles.





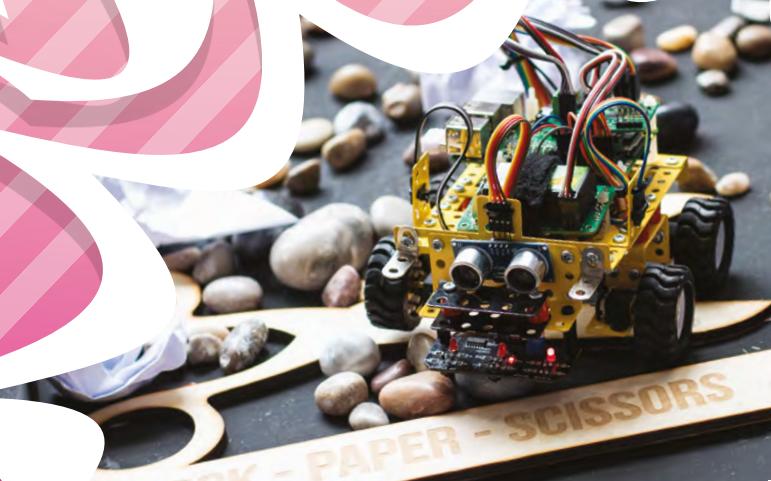
Pi WARS

THE RASPBERRY PI ROBOTICS CHALLENGE COMPETITION



21ST-22ND APRIL 2018
CAMBRIDGE COMPUTER LABORATORY

Piwars.org



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
82

SINE WAVE STYLOPHONE

Create smooth beats from smooth waves

PG
86

LED GOGGLES

Illuminate the long winter nights with some glowing NeoPixel-powered eye wear

PG
92

CUSTOM ARDUINO

Design your own microcontroller based on the famous Uno

PG
98

MUSIC BOX

Drive some clockwork tunes using only a 555 timer and a modified servo

PG
70

SCHOOL OF MAKING

Start your journey to craftsmanship with these essential skills

70 Make a knife

74 Laser-cut an LED clock

78 Get digital data into an Arduino

PG
102

BUILD A CHEESE PRESS

Transform milk into deliciousness with the application of pressure

PG
106

OPENHAB 2

Design your own smart home with an open source hub

How to make a knife without a forge

This knife, inspired by the Japanese Tanto, is made with just some basic hand tools



Will Kalif

@willkalif

Will Kalif is an amateur siege engineer who has built all types of siege engines, ranging from 1 ft (30 cm) miniatures to 10 ft (3 m) behemoths. He is the webmaster and owner of the website: StormTheCastle.com

T

o make this Tanto-inspired knife you won't need ten years of tutelage under a Japanese swordsmith.

You won't even need a forge. You will just need a few basic hand tools to make it, and an inexpensive propane torch to heat treat it. You don't even have to worry about designing the knife; we give you the template right here.

There are two things that are of the utmost importance when making a knife. Firstly, you have to use a really good knife steel and secondly, you have to heat treat it properly.

If you cover these two things, you will end up with a knife that is strong and tough, and that will hold a nice sharp edge.

In this article we strip away all the complicated stuff and pare this simple and beautiful knife-design down to the basics. Everything about this project has been carefully designed to make it as easy as possible for you to make a great knife.

This method of knife-making is called the 'Stock Removal' method. This is because we take a piece of stock steel and remove all the unwanted steel, leaving behind the shape of the knife we want. It is a tried and true method of knife-making, and considered the easiest way to get started in the craft.

BUT THERE IS A TRADE-OFF

Because you aren't using power tools, you will have to use some elbow grease. From start to finish, this knife project will take you about ten hours of work. But once it is done, you will be proud to show off the knife you made with your own two hands.

Right ↗
The completed knife

LET'S GET STARTED

We start with a piece of O1 steel (pronounced oh-one), because it's a steel that's pretty easy to work with hand tools, and it's easy to harden and temper – in short, it's an excellent beginner knife-making steel. There are many other types of steel you can use to make a knife, but with O1 you are guaranteed success as a beginner.

START BY PROFILING THE KNIFE

Create a paper template based on the design in **Figure 1**. Cut out one knife design, and use a spray adhesive to attach it to your piece of steel. Line it up so the straight edge (spine) of the knife is along one edge of the steel. This will save you a lot of hacksaw time.

Hacksaw the profile of the knife. Go all the way around, in effect cutting the shape of the knife out of the steel. Stay as close to the lines as possible, because it will save you work when filing. Take your time with the hacksaw. Take frequent breaks – not just to prevent fatigue, but to prevent the steel from overheating. Oil it frequently. If the paper template is ruined by the oil, remove it with acetone and glue a new one on. If the steel starts to change colour (blue or gold), you are going too fast and changing the temper of the steel. This changing of temper can make it more difficult to cut, or more difficult to harden and temper later.

Once you finish with the hacksaw work, use a mill file to do the detail work – finishing the shape of the knife to its final profile. A mill file is one that is only used in a forward direction. You file forward, lift up the file, return to the beginning, then file forward again. Do not file in a back and forth motion – only forward motion.



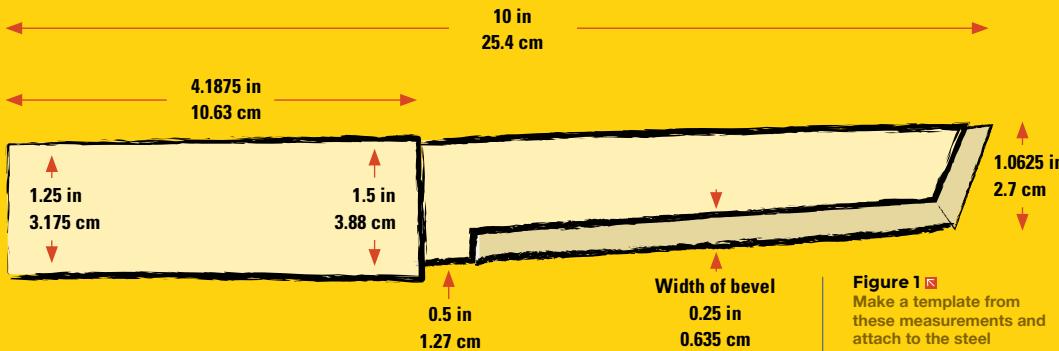


Figure 1 Make a template from these measurements and attach to the steel

BEVEL THE BLADE

Remove the paper template and clean the blade with acetone. Cut out a new paper template, and cut away from it the blade section. This will now act as a guide for filing the bevel of the blade. Affix this new template to the knife and file the bevel.

Use a mill file on the bevel section of the blade to slowly and evenly file down the bevel. Keep your hand at an angle, and avoid rocking your hand. File each side of the knife so the cutting edge of the knife is just about to the centre line of the $\frac{1}{8}$ inch (.3175 cm) thick steel. Leave a small, thin line along the centre line, for sharpening. Take your time with this bevelling. It will take you two to three hours to complete it. As you do it, you will get a feel for how the file acts, and how to keep the bevel line straight.

STEP 3 HEAT TREATING

Heat treatment is done in two steps: hardening and tempering.

HARDEN THE BEVEL OF THE BLADE

Without a forge it would be difficult to harden the whole knife but, for functional use, we can use a

propane torch to just harden the bevelled section of the knife, which will be enough to give a very serviceable blade.

Use the torch to slowly heat the whole bevelled section. Go slowly, and start out with the torch about 15 cm (6 in) from the steel. Use constant back and forth motions along the length of the bevel.

Everything about this project
has been carefully designed to
make it as easy as possible for
you to make a great knife

For the first two minutes it will seem like nothing is happening. But, the steel is slowly heating up. Continue heating slowly, and continue with the motion. You will see the steel start to change colour, first to a gold then to a blue.

You want the whole bevelled section to change colour at the same time. Direct the heat with more focus on areas that are slower to change colour. →



SIMPLICITY OF DESIGN

This knife is specifically designed for the do-it-yourself fan to make with just a few simple tools. The spine is straight, to avoid a lot of the hacksaw work, and it makes bevelling the cutting edge of the blade easier. The steel we use (O1) is easily worked, and thick enough to give us significant strength. The template is included so you don't have to design your own knife. You can make copies of the template, cut out the knife shape, and adhere it to your piece of steel. Everything has been carefully designed to make your knife-making experience a fun and easy one. You will end up with a durable, and functional Tanto-styled knife.

YOU'LL NEED

- ❖ **1 piece of O1 steel**
 $\frac{1}{8}$ inch thick
(.3175 cm), 12" long
(30.5 cm) and 2" wide (5 cm)
- ❖ **1 piece of hardwood**
For the handle: 5" long (12 cm), 2" wide (5 cm), 2" thick (5 cm)
- ❖ **A hacksaw**
Plus several bi-metal blades
- ❖ **Two-part epoxy glue**
Gorilla glue works just great
- ❖ **Two clamps**
- ❖ **Mill file**
- ❖ **Wood rasp**
- ❖ **Propane torch**
For heating the steel
- ❖ **Oil**
For quenching
- ❖ **Small steel vegetable can**
- ❖ **A polishing oil**
To finish the handle, we used Tung oil
- ❖ **Various grades of sandpaper and emery paper**
- ❖ **Acetone**
For cleaning (optional)
- ❖ **A vice**
Optional, but nice to have
- ❖ **Can of spray-on adhesive**

Left ❖
Affix the template in a way that will minimise hacksaw work

How to make a knife without a forge

SCHOOL OF MAKING



Above ↗
Finish shaping the
knife with a mill file

SAFETY

Wear safety gloves and safety goggles when heat treating your knife. Remember, just because the steel isn't glowing red, doesn't mean it isn't dangerously hot. At no point during the process should you touch the knife with a bare hand. Use a tool, vice grips, or pliers until you are absolutely sure that it has returned to room temperature.

Right ↗
Go slow when
heating the knife

Get the whole bevel section to red, and then quench the knife in motor oil.

Let it sit in the motor oil until it's cool. Your knife is now hardened, but it is also very brittle, and can snap or crack very easily.

We balance this out with a second heating. This second heating is called 'tempering'. We heat the steel again to a temperature that makes it softer and more resilient to use.

Clean the blade of the knife with some emery paper, removing the black so you can see the steel colour cleanly. Your knife is hardened. Emery paper will not affect it much, but will clean all the soot and

carbon off of it. When tempering the blade, we want to heat it to the point where it turns a golden colour.

Repeat the same process as you did with hardening the blade. Heat the bevel of the blade with a steady back and forth motion, taking your time, until it evenly turns a golden yellow colour. Then quench it in motor oil. Your blade is now tempered.

Be cautious when tempering and go very, very slowly! It will seem not to be heating up at all then, all of a sudden, parts of it will turn blue. Those blue sections are over-tempered and too soft! So go carefully and get the bevel of the knife to evenly heat to a golden colour.

If at this stage you get too much blue colour, you should start over again. Cool the knife, then begin again by hardening and then tempering.

Once the blade has cooled after hardening and tempering, you can clean it up with emery paper. You can progress through finer and finer grits, to get a mirror finish on the blade, if you desire.

MAKE AND ATTACH THE HANDLE

Cut two pieces of hardwood to be slightly larger than the handle section of the knife, and cut each piece to be about 2 cm ($\frac{3}{4}$ in) thick. You can vary this thickness depending on how beefy you want your knife handle to be, and how big your hand is. Before adhering the wooden handle halves to the



knife, ensure that you have very flat surfaces on the wood. Lay a piece of 100-grit sandpaper on a table, and rub the handle halves over it until they are very flat and smooth. This ensures a very stable and secure bond between the wood handles and the knife steel.

Mix and apply a liberal amount of the two-part epoxy to the handle halves, then apply them to the knife and clamp in place. Excess epoxy will squeeze out of the edges of the bond. Clean it up with a rag, and do a final cleaning of it all with acetone. Set the clamped assembly aside for 24 hours.

SHAPE THE HANDLE

When ready, use a wood rasp to shape the handles. The knife handle is not perfectly rectangular, but is slightly wider near the bevel than at the butt end. Make a note of this, and rasp the handle halves to mimic this shape. Take your time when rasping, and frequently hold the knife to see how it feels in your hand.

Once the shaping of the handle is done, treat the wood with some kind of oil. Tung oil is our preference, but others can be used. For a deep rich handle look, apply a coat of oil, buff it with a rag, let it dry, then lightly sand it, and repeat with another coat. Repeat this as many times as you prefer.

FINISH BY SHARPENING THE BLADE

Sharpening a knife is a bit of an art, and it takes practice and muscle memory to do easily and efficiently. Use a sharpening stone and a small amount of a thin oil, moving only in backward strokes. Face the blade toward you and push the knife away from you, repeating slowly and evenly on both sides of the knife.

Below ↴
Two-part epoxy makes a very strong bond between steel and wood



Above ↴
You can see some of the gold colour on the bevelled edge of the knife

Left ↴
Raspining the shape of the handle is a simple job

A standard edge would require you to sharpen a double-sided knife like this one to 20 degrees on each side. This gives a total edge angle of 40 degrees. You can make a more durable edge, that will last longer, by creating a more obtuse angle. For this knife, we recommend you sharpen each side to between approximately 22 and 25 degrees. It will be less sharp, but more durable.

GOING FURTHER

You have lots of options if you would like to take this project further. You can polish the blade of the knife by sanding it with emery paper, going through progressive grits all the way up to 1200-grit. This will give the blade a mirror finish. You can also make a wooden sheath for the knife.

Once this knife is made, you probably won't be called upon to perform any ninja duties, but you will have a durable and good-looking knife that you can be proud of, because you made it with your own two hands, and not a single power tool. And that's something any ninja would be proud of. □



QUICK TIP

There are many different oils you can use to finish the handle including Tru-Oil, teak oil, or boiled linseed oil.

Laser-cut a glowing clock

Use a laser cutter and some basic Arduino skills to create an LED-lit clock



Mike Fischthal

@pixelacademy

Mike Fischthal is the CEO and Founder of the Pixel Academy, a technology and STEM-focused provider of after-school and camp programs in New York City. WhatWillYouMake.com

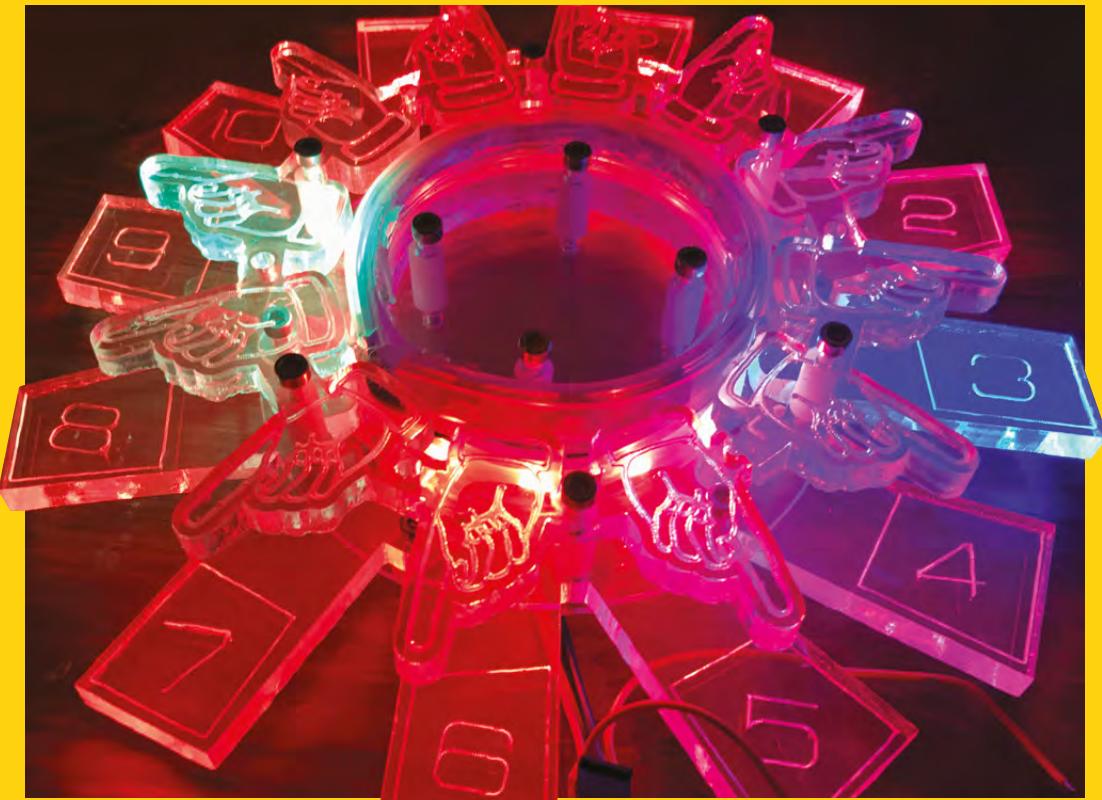
You can use LEDs and a laser cutter to make custom illuminated signs, but can you make them animate? Yes! We'll use a clock as an example and create some sweet glowing displays.

STEP 1: DESIGN

For this project we're going to be using light-guide acrylic which, unlike regular acrylic, is embedded with tiny particles that will reflect light in all directions. Source lights will spread out conically, illuminating everything both directly and through indirect reflection. We want to control the illumination for specific elements within

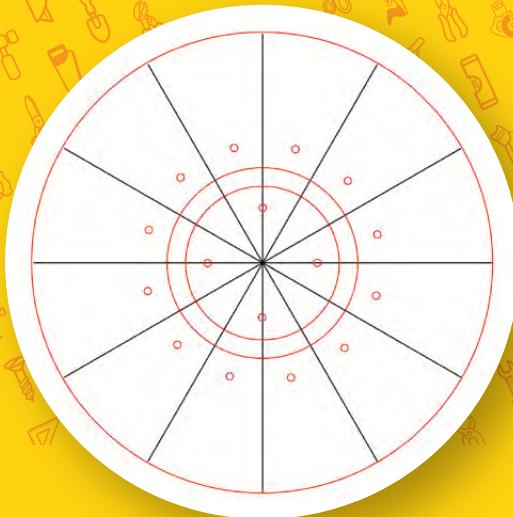
the same sheet of acrylic, so we're going to need to get each element to fall within the cone of a single LED. Furthermore, to minimise the likelihood that other LEDs or internal reflections will illuminate the wrong elements, we're going to place our LEDs in the centre and project light outwards, and we're going to cut out any negative space between elements.

Each layer of the final piece will have a small circle that the LEDs will wrap around, a gap for the LED strip, and then a large surrounding ring that has the designs we're going to illuminate. This tutorial is going to focus on a single-layer design, but you can add as many as you're willing to make. After a certain point (around



Right ↗

As long as you have the mechanism running well, the design is up to you

**Above ↗**

To keep things symmetrical, we decided to use more screws than needed, with one between every hour on the outside ring and just four on the inside circle

three), you won't be able to see the bottom-layer designs through the top ones.

STEP 2: MEASURING

The size of our clock is going to be limited to the size of acrylic we can cut. So all we need to figure out is how big to make the inside circle that the LEDs are going to fit around.

We want exactly twelve LEDs on the inside circle, one for each hour on the clock. Take your LED strip and measure the distance from the centre of the first LED to the centre of the twelfth LED. On the WS2812 addressable LEDs, that works out to about 200mm (you could also measure the spacing of one LED and multiply by twelve, but that increases the margin of error).

$$\text{Circumference} = \pi * \text{diameter}$$

$$\text{Diameter} = 200 \text{ mm} / \pi$$

So, using the WS2812, our diameter for a circle of twelve LEDs is about 64mm.

The WS2812s are surrounded by a 5mm-thick plastic cover. So in deciding how much of a gap we want to leave between the inner circles, we need to add that to each end of our diameter, making the new value 74mm. Give yourself a little extra clearance and make the design's internal radius an even 78mm.

STEP 3: VECTOR ART

Every brand of laser cutter has its own custom software, but every program will work with vector art as an input. We're going to use Adobe Illustrator, but any vector art program will work.

Start by creating an artboard that is the size of your acrylic, 8 x 12 in (20 x 30cm), to give a visual framework for how much material you have to work with. Click on the ellipse tool (L), then left-click on the artboard to bring up the ellipse dialogue (don't click and drag). Enter the diameter of your internal circle (for the LEDs) in the width and height (64mm) and press OK.

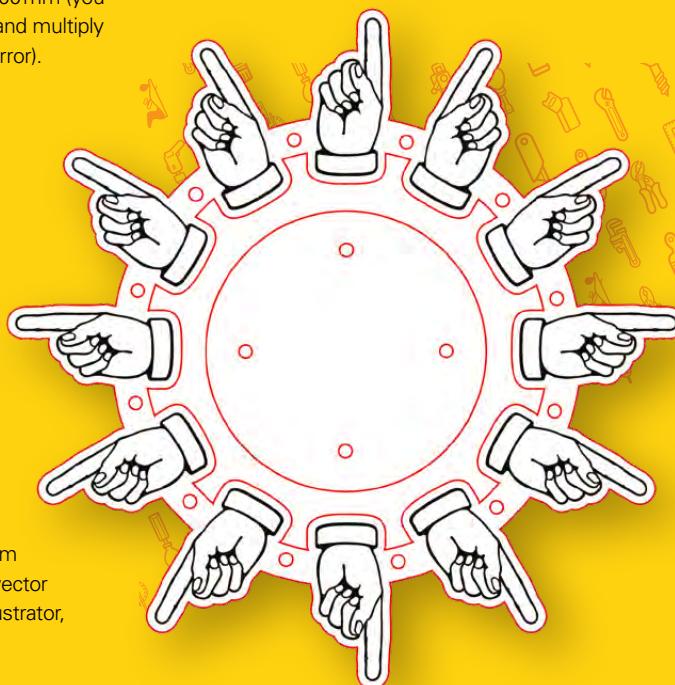
Repeat the process and add two more circles for the inside diameter of your design ring (78mm), and an outer diameter reference ring (20cm / 8in).

Open the Alignment window by going to **Window > Alignment** or press **SHIFT+F7**. Select all three circles and align the objects along the vertical centre and horizontal centre so they are all concentric.

Click the Line tool (\) and draw a vertical line from the top of the artboard to the bottom. Align everything vertically and horizontally again to put the pivot point of our line directly in the centre of our circles.

Select the line and go to **Object > Transform > Rotate** to bring up the rotation dialogue. To get our twelve hours on the clock, we need to rotate this line around the circle by one twelfth of 360 degrees. Enter 30 degrees as the angle and then instead of clicking OK, click the COPY button. Repeat the process by going to **Object > Transform > Transform Again (CMD+D)** until you have your twelve lines.

Now we have our template for the two pieces of a single layer. Before we start to make the actual designs, we want to think about how everything is going to come together. →

**YOU'LL NEED**

- ❖ A laser cutter
- ❖ 1 × sheet of light-guide acrylic
hsmag.cc/VctviU
- ❖ 1 × sheet of 1/16th inch clear acrylic
hsmag.cc/uCHoZZ
- ❖ Addressable RGB LED light strip (e.g. WS2812)
hsmag.cc/ifXIMA
- ❖ An Arduino and computer with Arduino IDE
- ❖ 16 × M3 30 mm screws
hsmag.cc/KfweRU
- ❖ 16 × M3 nuts
hsmag.cc/abXVBW
- ❖ Assorted M3 spacers

OPTIONAL

- ❖ Plastic polishing compound
- ❖ Hot glue gun
- ❖ Additional sheets of light-guide acrylic

Left ↗

The top-layer hands are solid fill and no stroke, so they will be rastered

SCHOOL OF MAKING

DESIGNING FOR LASER CUTTING

- ◆ There are two modes the laser will operate in:
 - Rastering, which takes a greyscale image and rapidly turns the laser on and off while scanning across your material to etch a dithered pattern. This is slow because the laser scans back and forth across your material.
 - Vectoring, which uses vector paths to etch lines of a fixed depth in your material. This process is fast because the laser moves along the path of your line and it can be used to engrave or to cut, with a cut being a strong enough engrave to penetrate the material.
- ◆ The technique you want is determined by how you set up your art: objects in Illustrator have a fill and a stroke. Fills are interpreted as solid objects for rastering, while strokes are interpreted as single paths for vectoring. Strokes can accidentally be interpreted as solid objects for rastering if they have a width. For example, a 2mm stroke will be interpreted as a 2mm-wide box. To ensure your laser sees your stroke as a vector path, you need to give it a near zero width. Strokes of different colours will let you distinguish vector cuts from vector engravings.

QUICK TIP

Rastering looks at the image as a whole, as a printer would, so only the top layer of art is engraved.

Below

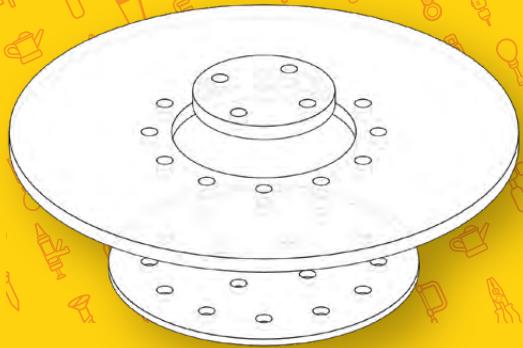
Check to see if the insides of the screw holes have fallen into the honeycomb

We're going to have (at least) two layers: one layer of light-guide acrylic (with two separate pieces) and one backing layer that will hold the insides and outsides together.

Open the Layers window (**F7**) and create a new layer to hold our screw holes. Lock the layer that the circles and lines are on so you don't accidentally move them in the next steps.

Every maker has their go-to screws and fasteners and here we're using M3 bolts – small and sturdy.

Using the ellipse tool, add a circle with a 3.1 mm diameter (3mm will work but since we're using nuts to fasten everything together, it doesn't hurt to leave a little room for error). Copy that circle to anywhere on your design where you want to place your screws.



Above Here's a rendering of how our clock will come together

With all of your layers unlocked, select the Artboard tool (**SHIFT+O**) and, while holding the **ALT** key, drag your artboard down to make a duplicate. We'll use this as our template for the backing piece. All it needs is a hole for our wires to come out the back and a way to mount the clock on the wall.

With the templates for all three pieces done, lock your M3 holes in the layers window and go ahead and create your designs!

STEP 4: LASER CUTTER

Import your art into your laser cutter software. If any rasters are shown as vectors or vice versa, check your laser cutter's instructions to see if they have any special import requirements.

Remove the top film from your first acrylic sheet and line it up in the laser cutter on top of the honeycomb. Focus your laser to the top of the acrylic and close the lid.

The first tab in the Full Spectrum software is for rastering, where you could adjust your greyscale image and create dithers. Because we're working with light-guide acrylic, a dithered image will just get washed out, so we're sticking with solid black. You may see your vector lines showing up in the raster preview as well (they are, after all, part of the image that would print). Adjusting the threshold down slightly usually gets rid of them, or you can reimport the raster job with the 'Ignore Thin Vector' setting turned on. There really isn't anything wrong with rastering the areas you're going to later vector, but it will increase the time your job takes and shorten the lifespan of your laser tube.

For the actual rastering process, there are two settings: speed and power. The higher the power and the slower the speed, the deeper your etch. Keep the power at maximum and slow down the raster to half speed (50) so we can get a deeper etching.

Before running any job, you should check to make sure everything is lined up correctly. Sometimes (mostly

through human error) the dimensions are wrong or your material is slightly off alignment. Lasers have a 'Run Perimeter' option that will trace out the bounding box of your job. Watch the red alignment beam as the laser runs the perimeter of your job and take note if it wanders off your material.

If it's all contained, you're good to go. Run the raster.

When the raster job completes, anywhere from a few minutes to a few hours later, we can switch over to our Vector tab. Notice how each coloured stroke shows up as its own task for the laser, and that we can give each task its own custom settings.

Vectoring settings are a bit of a trial and error process. The power of your laser, the type of focus lens, and the material type and thickness all have to be factored in. Unlike the rapid scanning process of rastering, vectoring focuses the beam on a specific spot to rapidly burn it away. Because of the added heat, it's easy to set cardboard on fire or blacken a sheet of wood with char. So, for flammable materials, you want to vector fast and with low power to prevent the material around your cut from burning. Acrylic, though, is a much more forgiving material and is really hard to set on fire. If you overpower your vector engrave, the worst you're likely to do is go too deep and accidentally make a cut. Normally, best practice is to do a test cut, but light-guide acrylic isn't cheap so we're just going to take it slow. And by slow, we mean fast...100% speed and 100% power.

A shallow cut for our vector engrave will be deep enough to produce a bright reflection, so a single, quick pass is all we need.

The vector cut is a bit trickier. A thick material like the light-guide acrylic won't be cut all the way through even at 100% power. So we need to make multiple passes, each one cutting a little deeper. Again, because acrylic doesn't burn, we can have the laser move pretty slowly on each pass, giving the laser more time to vaporise each spot.

So our initial settings for the vector cut are going to be 100% power, 20% speed, and four passes. Run the job perimeter again to make sure your vector art doesn't go out of bounds and then hit GO.

When your job is done, confirm you were able to cut all the way through the acrylic, but don't move the material. If you end up having to run more passes you'll have messed up your alignment. One way to tell if you cut all the way through is to check if the insides of the screw holes have fallen down into the honeycomb.

Attach your outer rings to the backing using the M3 screws. If you made multiple layers, separate them using spacers to give your LED strip some extra room.



STEP 5: ARDUINO

Cut your LED strip to a length of twelve LEDs, peel off the backing, and wrap it around the inner mounting circle. There are only three wires: +5V, ground, and Data. Data is unidirectional, so one LED's data-out (DO) needs to connect to the next LED's data-in (DI). Connect all three wires to your Arduino (+5 to +5, ground to ground, and DI to pin 4).

These addressable LEDs work with Adafruit's NeoPixel library which you can add to your Arduino IDE by going to Manage Libraries and searching for NeoPixel.

In the example file that comes with the NeoPixel library, all you need to do is change the number of LEDs to match the number of LEDs you used in your project:

```
#define PIN 4
#define LED_COUNT 24
```

Upload the example to your Arduino and the LEDs should cycle through a rainbow of colours. Align your centre circle so the LEDs best match up with the hours on your clock, and then screw the centre into place. All that's left to do is add some code. You can set a specific LED to light at each hour or try something a bit more exciting like having the LEDs pulse with each second and display different colours to distinguish between the hour hand and the minute hand. □

Above 9:15 with the top layer (green) showing minutes and the bottom layer (blue) showing hours

SAFETY TIPS

Make sure that you are aware of the safety requirements for your laser cutter.

Reading digital data on the Arduino platform

Learn how to read external data in an Arduino project



John Wargo

@johnwargo

John is a professional software developer, writer, presenter, father, husband, and geek. He is currently a Program Manager at Microsoft, working on Visual Studio Mobile Center. You can find him at johnwargo.com

```

SerialCallResponse | Arduino 1.8.5
File Edit Sketch Tools Help
SerialCallResponse $ 
// BTNPIN defines the Arduino input pin to which the
// button is connected
const int BTNPIN = 2;

// Specifies the amount of time the button must stay pushed for it
// to trigger the LED on or off. Increase this value if your LED
// flickers
const unsigned long DEBOUNCE_DELTA = 50; // milliseconds

// btnState stores the current button state (HIGH or LOW)
// initialize it to LOW so the LED stays off until the sketch
// reads a HIGH state for the button input
int btnState = LOW;
// A place to store the previous loop's button state
int prevBtnState = LOW;
< >

Done compiling.

Sketch uses 222165 bytes (21%) of program storage space. Maximum is 1044464 bytes.
Global variables use 31572 bytes (38%) of dynamic memory, leaving 50348 bytes free.
59 NodeMCU 1.0 (ESP-12E Module), 80 MHz, 115200, 4M (3M SPIFFS) on COM3

```

Left The Arduino development environment includes code highlighting, to help you spot typos in your code

YOU'LL NEED

- ❖ An Arduino or Arduino-compatible device

We recommend the Arduino Uno for first-time users

- ❖ Momentary push button

- ❖ A 10 kΩ resistor

- ❖ Breadboard

- ❖ Breadboard jumper wires

In the previous article in this series, we showed you how to blink the built-in LED on an Arduino device. Here, we'll show you how to use a push button to toggle the LED on and off. This article illustrates one way to read digital data using an Arduino.

Arduino boards offer several ways to interact with external hardware components, in all cases this means sending a signal to, or reading data from, an external device. Those inputs and outputs, coupled with the logic you've coded in your project's sketch, are the meat of any Arduino project. Arduino inputs come in two formats: analogue and digital, in this article, we'll cover one way to use digital inputs.

Each digital input on the Arduino can read two values: LOW and HIGH. LOW is a constant defined within the Arduino IDE that essentially means zero (or very little) voltage. A value of HIGH references

the highest voltage value the Arduino can support (typically 3V (volts) on an Arduino operating at 3V, and 5V on an Arduino operating at 5V).

Note: Any Arduino device you use for your projects will have one or more digital inputs; these usually double as digital outputs as well. You learned how to use a digital output in the series' previous article.

You might be saying to yourself: "How useful is a digital input if it's only either on or off? That's only one bit, right?" On the Arduino, digital inputs are used in two different ways: to read point-in-time input values, such as the status of a button, or to read a stream of binary digits (bits) values which an application converts into more useful data such as bytes, or numbers. In this article, you'll find out how to use a digital input to read the status of a push button.

DIGITAL INPUTS CAN READ SINGLE VALUES OR STREAMS OF DATA

In the previous article in this series, we showed you how to use the default Arduino Blink sketch to turn an Arduino's on-board LED on and off on a specific interval. In this project, we'll extend that project, and use a button to turn the LED toggle the status of the LED. When the push button is depressed (pushed), the LED turns on. When the push button is up (open), the LED turns off.

Before we wire up the circuit, let's take a look at the code (you can find the complete code for the example at hsmag.cc/KTioNX).

The sketch defines the **BTNPIN** constant used to identify the Arduino digital input pin to which the button is connected. Following a common convention, we created the constant name in all capital letters, making it easy to distinguish constants from variables in a sketch. You'll populate this constant value with the pin number for your particular hardware implementation.

Next, the sketch defines the **btnState** variable, which is used to store the current state of the button; this value is used to determine whether to turn the LED on or off. Notice how we initialised the variable to **LOW**; this isn't required, but gives the sketch a fallback in case it can't read the button, setting the LED to off by default the first time through the loop.

```
// BTNPIN defines the Arduino input pin to which the
// button is connected
const int BTNPIN = 2;

// btnState stores the current button state (HIGH
// or LOW)
// initialize it to LOW so the LED stays off until
the sketch
// reads a HIGH state for the button input
int btnState = LOW;
```

In the sketch's **setup** function, the code sets the mode for the Arduino I/O (input/output) pins used by the sketch. The sketch calls **pinMode** to set the default LED pin (defined in the Arduino IDE's constant **LED_BUILTIN**) to output mode, then calling **pinMode** again to set the push button pin to input mode. Finally, the function turns the LED off, through a call to **digitalWrite**, just to make sure we start with the LED in a known state before the first loop begins.

```
// The setup function runs once every time the
Arduino
// powers up or resets (after a sketch update, for
example)
```

STAYING CONSTANT

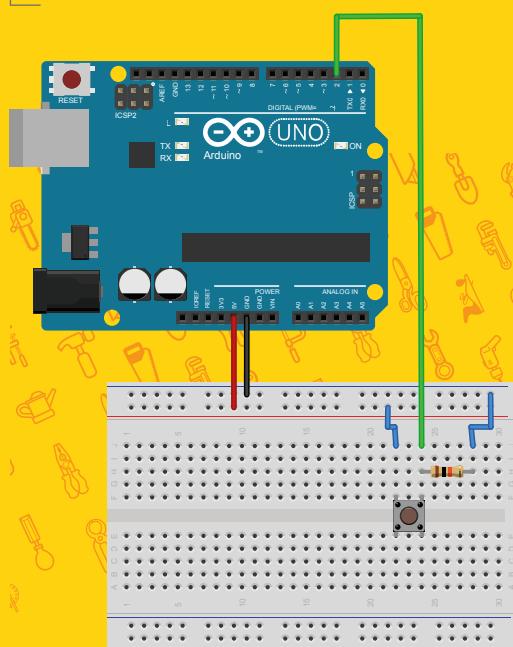
A good practice for developers is to use constants to store values used in multiple places in a sketch. The **BTNPIN** constant is a good example for this; by pulling the value into a constant defined at the beginning of the sketch, you make it easy to change this value if the hardware configuration changes (if you connect the button to another digital input pin, for example). You could skip this step, but if you later changed the input pin for your project, you'd have to locate every place in the sketch where it's used, then change each instance. For this small sketch it's not that big of an issue, but for larger sketches it's much easier to do it this way and make one change that affects the whole sketch instead of many little edits, and potentially missing one.

```
void setup() {
    // initialize digital pin LED_BUILTIN as an
output.
    pinMode(LED_BUILTIN, OUTPUT);
    // initialize the push button pin as an input:
    pinMode(BTNPIN, INPUT);
    // set the initial state of the LED (off)
    digitalWrite(LED_BUILTIN, btnState);
}
```

In the sketch's **loop** function, the code reads the button status through a call to **digitalRead** and stores the result in the **btnState** variable. Next, the code →

Figure 1 ↴

The Fritzing tool (fritzing.org) can be a great way of designing your circuits before starting on your breadboard



QUICK TIP

The resistor is used in this circuit to help force consistency of digital input values. Without the shunted circuit to ground, there's no clear definition of LOW vs. HIGH, and the input could 'float' at an indeterminate value without an input value applied. With the resistor in place, there's a clear definition of LOW when the button is open through the connection to ground. With the button pushed, the 'slower' path (through the resistor) is ignored because it's a more expensive route than the direct route to the digital input.

SCHOOL OF MAKING

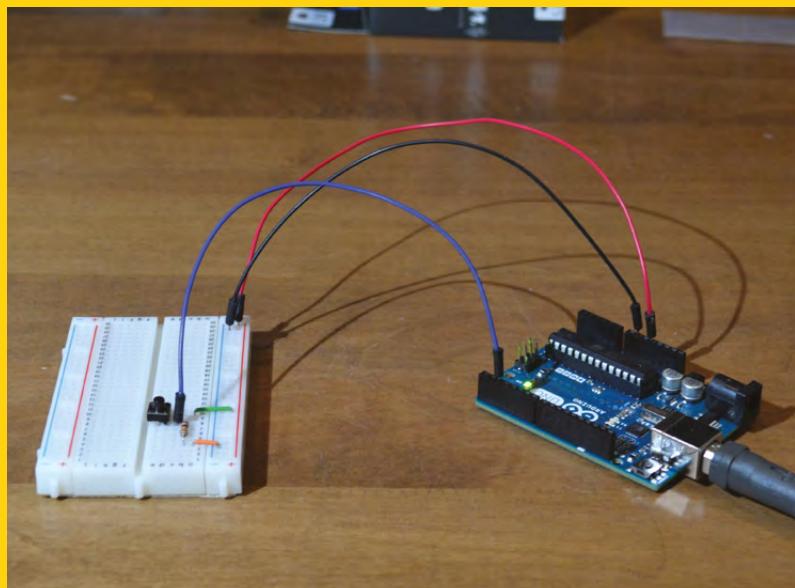
BOUNCING ALONG

Bouncing and debouncing are terms used when describing interactions with electrical connections like the one we have in the push button used in this project. As a button or switch begins a connection or disconnection, there's an uncertainty in the connection as the contacts move. A button potentially makes multiple intermittent connections until the button contacts connect solidly; this is called bouncing. To mitigate bouncing, Arduino developers implement debouncing, a mechanism used to force a single signal from the button through some extra code. In this example, the code debounces the button connection by forcing the application to wait a minimum amount of time with a connection before considering it to be accurate.

uses the value in `btnState` to set the LED status using a call to `digitalWrite`. When `btnState` is `LOW`, the code turns the LED off; when `HIGH`, it turns it on.

```
// The loop function runs repeatedly as long as a
// sketch is
// loaded and the Arduino has power.
void loop() {
    // Read the state of the button; it's a digital
    // input,
    // so possible returned values are HIGH or LOW.
    btnState = digitalRead(BTNPIN);
    // Use the measured value to set the LED state
    digitalWrite(LED_BUILTIN, btnState);
    // This whole function can be simplified to the
    // following
    // single line of code:
    // digitalWrite(LED_BUILTIN, digitalRead(BTNPIN));
}
```

Below The complete circuit assembled and running with an Arduino Uno



The code, as shown, breaks that action into two steps: reading the value from the input pin into a variable, then using that variable value to set the output on the default LED pin. That's a great way to do it when you're illustrating how to do something, but you'll use less memory and get better performance in your sketch if you consolidate the two steps into one, as shown in the commented line in the code (shown here uncommented):

```
digitalWrite(LED_BUILTIN, digitalRead(BTNPIN));
```

Here, the result from the call to `digitalRead` is passed as an input to `digitalWrite`. You won't get tremendous performance benefit doing this here but, for larger sketches, especially when you're bumping up against memory limits on the Arduino device, it's a useful approach.

PUSH TO START

Push buttons are mechanical devices, and as you're pushing or releasing the button, there's no guarantee that the Arduino can get a solid reading every time the button is pushed or released. To accommodate this, you can adjust your sketch so it debounces the button connection, ensuring that the button has been pressed for a minimum number of times before triggering a change in LED status.

In the following example, we've enhanced the previous example to include debouncing; you can find the complete code for the following example at hsmag.cc/pEzXyu.

At the beginning of the code, the sketch defines the same `BTNPIN` constant and `btnState` variable used in the previous example. We've also added the `prevBtnState` variable to keep track of the previous state of the button, and the `ledState` to track the current state of the LED. The `lastToggle` variable keeps track of the time the button state changed. Finally, the `DEBOUNCE_DELTA` constant defines the number of milliseconds the sketch waits before it believes in a reading from the button. You'll see all of these in action later in the sketch.

```
// BTNPIN defines the Arduino input pin to which
// the
// button is connected
const int BTNPIN = 2;
// btnState stores the current button state
//(HIGH or LOW)
// initialize it to LOW so the LED stays off
until the sketch
// reads a HIGH state for the button input
```

```

int btnState = LOW;
// A place to store the previous loop's button
state
int prevBtnState = LOW;
// Used to track the current state of the LED
int ledState = LOW;
// Stores the last time the status of the button
changed
unsigned long lastToggle = 0;
// Specifies the amount of time the button must
stay pushed for it
// to trigger the LED on or off. Increase this
value if your LED
// flickers
const unsigned long DEBOUNCE_DELTA = 100; // milliseconds

```

The `setup` function is precisely the same as the previous example.

```

void setup() {
    // initialize digital pin LED_BUILTIN as an
output.
    pinMode(LED_BUILTIN, OUTPUT);
    // initialize the push button pin as an input:
    pinMode(BTNPIN, INPUT);
    // set the initial state of the LED
    digitalWrite(LED_BUILTIN, ledState);
}

```

In the `loop` function, the code reads the button using `digitalRead`, just like the previous example. Next, the code checks to see if the current state of the button is the same as it was the previous time the loop executed. If it isn't, the code stores the current time in the `lastToggle` variable.

AROUND AGAIN

The next time through the loop, if the button state hasn't changed, the sketch checks to see how long it's been since the last toggle (by subtracting the value in `lastToggle` from the current time). If the button state hasn't changed in more than `DEBOUNCE_DELTA` milliseconds (`if ((millis() - lastToggle) > DEBOUNCE_DELTA)`), then the sketch knows it has an accurate button reading, and it toggles the LED.

```

void loop() {
    // Read the current state of the button
    btnState = digitalRead(BTNPIN);

    // Is the button in the same state as the last
    time
    // we came through the loop? No? Then we need
}

```

```

to record
// the current time (in milliseconds)
if (btnState != prevBtnState) {
    // store the current time in milliseconds
    // It doesn't matter what the actual time is,
all we need
    // to know is how long did the button stay in
this state
    lastToggle = millis();
    //Reset our previous state, so this check
skips next time
    prevBtnState = btnState;
} else {
    // OK, the button states (current and
previous) are the same
    // Lets see if they've been the same for
DEBOUNCE_DELTA
    // milliseconds
    if ((millis() - lastToggle) > DEBOUNCE_DELTA)
    {
        // the button's been pushed (or not pushed)
for at
        // least debounceDelta milliseconds, so its
time to
        // toggle the LED if needed
        //Is the LED at the same state as the button?
        if (ledState != btnState) {
            // No? Then toggle it
            digitalWrite(LED_BUILTIN, btnState);
            //Then reset the LED status
            ledState = btnState;
        }
    }
}

```

To test either of these sketches, wire a button into an Arduino (see **Figure 1**, page 79). On one side of the button, the connection shunts from the 5V connection through the 10kΩ resistor to ground (GND). The other button connection routes to the digital input pin 2. With the button pushed, a connection is made from the 5V source to the digital input, bypassing the resistor and forcing the circuit to **HIGH**. When the button is released, the connection to the digital input pin disappears, and the voltage runs through the resistor to ground, making it **LOW**.

Using the Arduino IDE, upload the code to the Arduino device and try pushing the button to toggle the LED on and off. Play around with the value in the `DEBOUNCE_DELTA` constant to see how it affects the sketch's reaction to the button.

Don't forget, all of the project source code is available at hsmag.cc/dMDWFx. □

QUICK TIP

The Arduino's `millis()` method retrieves the current time in milliseconds since the Arduino started running the current sketch; it doesn't give the sketch an accurate time, but does let the sketch track how long it has been since a previous measurement.

BIG DELTA

`lastToggle` and `DEBOUNCE_DELTA` are both long integers because the sketch uses them to calculate time deltas, and time values are very large integers. Even though `DEBOUNCE_DELTA` is a small number (comparatively), since the sketch will be doing arithmetic using those values, we made them the same type to avoid any conversion issues.

Make sweet music with the Sineophone

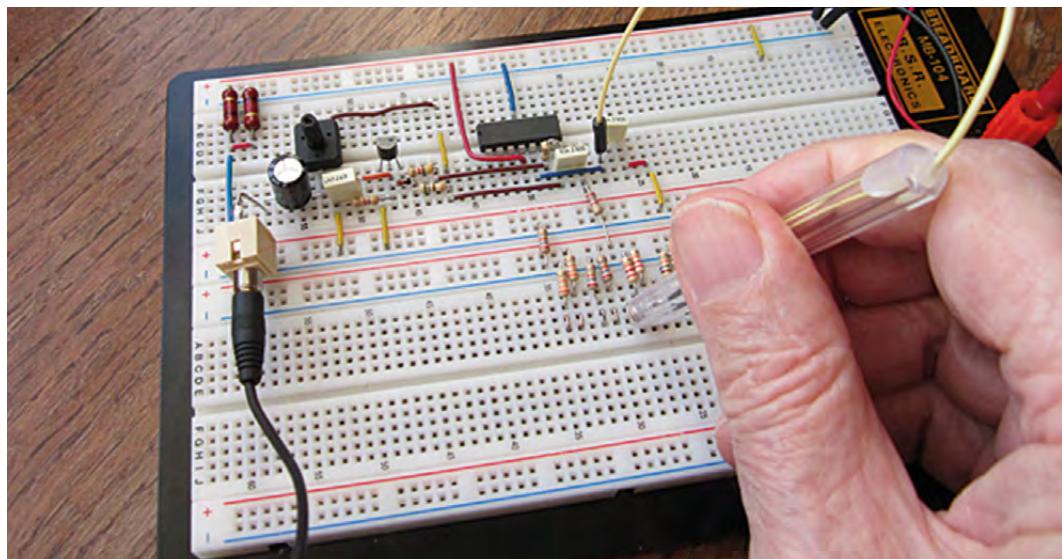
A smooth new take on a favourite electronic music machine



Eric Coates

↗ hsmag.cc/cabTmd

Eric Coates, BSc (Hons) MA has lectured on electronics in technical colleges and acted as examiner and moderator for several UK technical educational boards. He is the founder and CEO of learnabout-electronics.org



Electronic music covers an amazing array of instruments and techniques. Music created by entirely electronic means began back in the 1930s with instruments such as the Theremin, and continued through the 1960s and 1970s with the much-loved and much-copied Stylophone.

Most of these copies, like the original, used the humble 555 timer to produce square waves of varying frequencies for a musical note that had a harsh sound due to the many harmonics present in square waves.

THE SINEOPHONE IS BORN

This made me think, "How would it sound if we used sine waves instead of square waves? Perhaps a similar fun instrument, but with a sweeter sound." A Sineophone maybe!

Therefore to begin with, we need a sine wave oscillator that operates at audio frequencies and can be easily tuned to produce the different frequencies required for different notes.

CHOOSE YOUR OSCILLATOR

Looking at sine wave oscillators, there are a number of standard designs that produce excellent sine waves using capacitors and inductors, but at radio frequencies higher than the required audio range. Audio oscillators mostly use capacitors and resistors.

The basic design of an audio oscillator is illustrated in **Figure 1**, and consists of an amplifier with a feedback circuit that provides positive feedback by connecting the output signal from the amplifier back to its input, so the feedback signal is in the correct phase to add to the signal produced by the amplifier.

The components in the feedback circuit also determine the frequency of the signal produced. The two most popular oscillators for producing sine waves at audio frequencies are the phase shift oscillator and the Wien bridge oscillator. You can learn more at hsmag.cc/nKZzqy.

As the phase shift oscillator does not produce particularly well-shaped sine waves, and to alter the frequency of its output requires the values of three

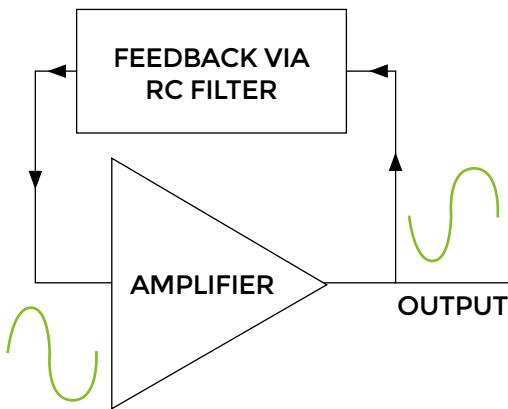


Figure 1 An oscillator is basically an amplifier with positive feedback

components to be changed simultaneously, which is tricky, the Wien bridge oscillator, shown in its most basic form below, is our circuit of choice this time.

Controlling the gain of the Wien bridge oscillator is important – the amplifier gain must exactly cancel out the losses in the feedback and frequency control sections of the circuit. There are several ways of ensuring this: the basic Hewlett Packard filament lamp method, or a pair of diodes in the feedback path are both popular methods or, most successfully, using a JFET in place of the filament lamp in **Figure 2**. As the JFET method not only controls the gain as efficiently, but also allows for an output with amplitude, it is basically the circuit shown in **Figure 3** that we will be using for the Sineophone circuit.

Now let's look at the complete Sineophone circuit in **Figure 4**, where it's not as easy to recognise the Wien bridge network. The series capacitor resistor arm of the network is now made up of C3 and R5 (now 27 kΩ) plus the right-hand resistor of one of the resistor pairs on the 'keyboard' at the bottom.

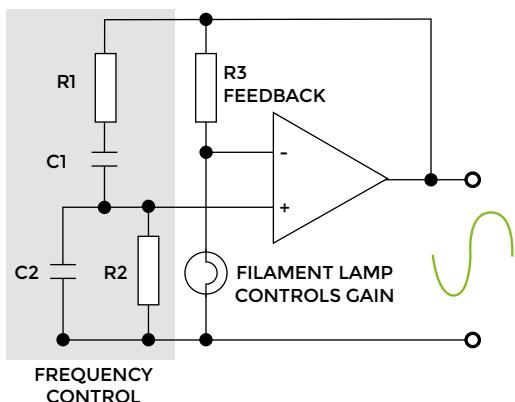


Figure 2 The Wien bridge oscillator with an RC network to control frequency and a filament lamp to control gain

COMPARISON

Take a moment to compare the circuits in **Figures 2** and **3**, and notice that R1 and C1 in **Figure 2** have been swapped around to become C4 and R6 in **Figure 3**. The reason for this is that we will need to change the values of the two resistors in the Wien bridge network for each different note played, and this will be much easier if the resistors R6 and R7 share a common connection as shown in **Figure 3**.

TAKE A NOTE

When it's powered up nothing is heard, as the Wien bridge network isn't complete. Each note is produced by connecting one pair of equal value resistors on the keyboard to pin 5 of the op amp IC1 via the 27 kΩ resistors R5 and R9 using a contact on the tip of the stylus. The parallel arm of the Wien bridge network now comprises the left-hand resistor of the selected pair in series with R9 (also 27 kΩ), both connected in parallel with C2. The series section of the bridge is now the right-hand resistor of the keyboard pair in series with R5 and C3.

The tip of the stylus bridges the two contacts at the open ends of both resistors of the pair for that note. Now the midpoint of the Wien bridge is momentarily connected to pin 5 (the non-inverting input) of IC1. This completes the oscillator circuit and so a sine wave note is produced. Different notes are produced as different frequencies by using different value resistor pairs for each note. The values of the resistors in each pair are added to the 27 kΩ values of the main Wien bridge resistors R5 and R9. The values of the two 10 nF Wien bridge capacitors C2 and C3 are unchanged over the range of available notes.

KEYBOARD DESIGN

The prototype Sineophone can produce a scale of eight notes and, for each note, it's first necessary →

HEWLETT-PACKARD

Although the Wien bridge had been around since 1891, it was not until 1939 that William Hewlett, a student at Stanford University in California, incorporated it into a practical audio frequency sine wave oscillator that has low distortion and a variable frequency and also a stable amplitude. This was achieved by the inclusion of an ordinary filament lamp as the gain control component. In collaboration with fellow student David Packard, the firm of Hewlett Packard was created to manufacture (initially in Dave Packard's garage) a useful, inexpensive, and highly successful audio signal generator.

YOU'LL NEED

- ◆ **Breadboard** with solid and flexible wire links
- ◆ **Semiconductors**
 - 1 × LM324 IC
 - 1 × 2N3819 JFET
 - 1 × 1N4148 diode
- ◆ **Capacitors**
 - 1 × 0.47 µF polyester
 - 2 × 10 nF polyester
 - 1 × 220 µF electrolytic
- ◆ **0.5 W resistors**
 - 2 × 1 kΩ
- ◆ **0.25 W resistors**
 - 1 × 560 Ω
 - 2 × 3.3 kΩ
 - 2 × 4.7 kΩ
 - 2 × 5.1 kΩ (2 extra 4.7 kΩ used in prototype)
 - 2 × 9.1 kΩ (2 extra 10 kΩ used in prototype)
 - 1 × 10 kΩ
 - 2 × 13 kΩ (15 kΩ used in prototype)
 - 2 × 18 kΩ
 - 2 × 22 kΩ
 - 3 × 27 kΩ
 - 2 × 33 kΩ
- ◆ **Miniature variable resistor**
 - 1 × 4.7 kΩ
- ◆ **1 × headphone socket**
- ◆ **Headphones** (approx. 30 Ω)
- ◆ **For the stylus**
 - An old Bic pen
 - A small dome-headed bolt
 - Hot glue
- ◆ **Power supply options**
 - 1 × 9 V battery (with connector leads)
 - 2 × 9 V batteries (with connector leads, connected in series to give 18 V)
 - 12 V DC supply (and matching connection for breadboard screw terminals)
- ◆ **Current requirement** approximately 16–20 mA

TUTORIAL

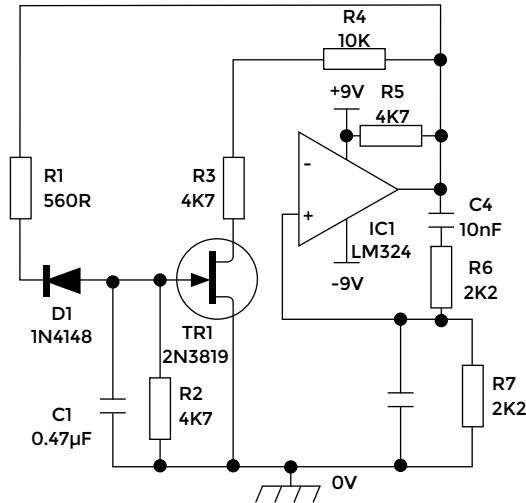


Figure 3
In the improved Wien bridge oscillator, a JFET acts as a variable resistor to automatically control gain

MUSIC LESSON

Beginning at the piano note of middle C (C4), the frequencies of the notes in our scale are therefore:

C4 = 261.63 Hz
D4 = 293.66 Hz
E4 = 329.63 Hz
F4 = 349.23 Hz
G4 = 391.99 Hz
A4 = 440.00 Hz
B4 = 493.88 Hz
C5 = 523.25 Hz

to know the frequency of the notes to be used. An eight-note scale has been chosen, assuming that the 'A' note used (A4) is 440Hz.

No flat or sharp notes have been included but these could also be created, given enough space on your breadboard. Lists of notes and frequencies are available from a number of websites. The frequencies listed above are rounded to two decimal places.

HITTING THE RIGHT NOTE

To calculate the appropriate resistor values for the Sineophone keyboard we need to know the appropriate formula for producing any particular frequency using the Wien bridge oscillator. The basic formula for the frequency of oscillation is:

RESISTANCE VALUES

Using the above formula the total resistance values required for each of the eight notes used will be:

C4 at 261.63 Hz will require $1/(2\pi \times 261.63 \text{ Hz} \times 10^{-9}) = 60.832\Omega$ or $60.8\text{k}\Omega$
D4 at 293.66 Hz will require $1/(2\pi \times 293.66 \text{ Hz} \times 10^{-9}) = 54.197\Omega$ or $54.2\text{k}\Omega$
E4 at 329.63 Hz will require $1/(2\pi \times 329.63 \text{ Hz} \times 10^{-9}) = 48.283\Omega$ or $48.2\text{k}\Omega$
F4 at 349.23 Hz will require $1/(2\pi \times 349.23 \text{ Hz} \times 10^{-9}) = 44.573\Omega$ or $44.6\text{k}\Omega$
G4 at 391.99 Hz will require $1/(2\pi \times 391.99 \text{ Hz} \times 10^{-9}) = 40.602\Omega$ or $40.6\text{k}\Omega$
A4 at 440.00 Hz will require $1/(2\pi \times 440.00 \text{ Hz} \times 10^{-9}) = 36.172\Omega$ or $36.2\text{k}\Omega$
B4 at 493.88 Hz will require $1/(2\pi \times 493.88 \text{ Hz} \times 10^{-9}) = 32.225\Omega$ or $32.2\text{k}\Omega$
C5 at 523.25 Hz will require $1/(2\pi \times 523.25 \text{ Hz} \times 10^{-9}) = 30.417\Omega$ or $30.4\text{k}\Omega$

As 27 kΩ of the above values are made up by R5 or R9, so 27 kΩ must be subtracted from these values to find the ideal values for the keyboard resistors:

60.8 kΩ – 27 kΩ = 33.8 kΩ for C4
 54.2 kΩ – 27 kΩ = 27.2 kΩ for D4
 48.2 kΩ – 27 kΩ = 21.2 kΩ for E4
 44.6 kΩ – 27 kΩ = 17.6 kΩ for F4
 40.6 kΩ – 27 kΩ = 13.6 kΩ for G4
 36.2 kΩ – 27 kΩ = 9.2 kΩ for A4
 32.2 kΩ – 27 kΩ = 5.2 kΩ for B4
 30.4 kΩ – 27 kΩ = 3.4 kΩ for A5

$$f_{\text{osc}} = \frac{1}{2\pi RC}$$

Therefore the value of R can be found by:

$$R = \frac{1}{2\pi f_{\text{osc}} C}$$

...which means a suitable resistance to generate A4 for example from our circuit would be:

$$R = 1/(6.283 \times 440 \times 10^{-9}) = 36172\Omega$$

As we will be using standard E24 series resistors, the choice of values is somewhat limited, and because of these limits it is not going to be possible to simply choose a single E24 resistor for each of the chosen notes that gives an absolutely accurate frequency. Any error can be reduced however by using large value fixed resistors for R5 and R9, to take up the bulk of the required resistance value, therefore leaving only the relatively small remaining resistance to be made up by each of the two resistors on the keyboard. So any error in the calculated frequency will only be a certain fraction of this smaller value. Also, bear in mind that each of the E24 series resistors chosen may not be exactly the value indicated by its markings as the actual resistance could vary within its own tolerance limits of +/-5%.

However, the commonly used E24 resistor range only has twelve preferred values to cover each decade of resistances, i.e. 1 to 10Ω , or 10 to 100Ω , or 100 to 1000Ω etc. and each initial value may deviate by +/-5%. Therefore, by using this range of resistors to cover the eight-note sequence, we can choose from multiples or sub-multiples of the following values:

10, 11, 12, 13, 15, 16, 18, 20, 22, 24, 27, 30,
 33, 36, 39, 43, 47, 51, 56, 62, 68, 75, 82, 91

C4 needs $33.8\text{k}\Omega$ so choose $33\text{k}\Omega$ giving a frequency of 265.26Hz (an error of 3.63Hz)

D4 needs $27.2\text{k}\Omega$ so choose $27\text{k}\Omega$ giving a frequency of 294.73Hz (an error of 1.07Hz)

E4 needs $21.2\text{k}\Omega$ so choose $22\text{k}\Omega$ giving a frequency of 324.81Hz (an error of 4.82Hz)

F4 needs $17.6\text{k}\Omega$ so choose $18\text{k}\Omega$ giving a frequency of 353.68Hz (an error of 4.45Hz)

G4 needs $13.6\text{k}\Omega$ so choose $15\text{k}\Omega$ giving a frequency of 378.94Hz (an error of 13.05Hz)

A4 needs $9.2\text{k}\Omega$ so choose $10\text{k}\Omega$ giving a frequency of 430.15Hz (an error of 9.8Hz)

B4 needs $5.2\text{k}\Omega$ so choose $4.7\text{k}\Omega$ giving a frequency of 502.07Hz (an error of 8.19Hz)

A5 needs $3.4\text{k}\Omega$ so choose $3.3\text{k}\Omega$ giving a frequency of 525.26Hz (an error of 2.01Hz)

These calculated errors do not include any differences due to the +/-5% tolerance figure for each resistor so can only be considered to be approximate. Using these values on our prototype does create some audible errors in the sequence of notes; most noticeable on G4, A4 and B4 and, for greater accuracy, it could be an advantage to use some resistors from the closer tolerance E48 range of resistors.

As a general rule, you should try to choose resistors that (combined with the 27K of R5 or R9) give a resulting resistance within 1K or less of the calculated value. Sometimes this may be made possible by adding an extra resistor having a low value to each resistor in the pair, to bring the total resistances for that pair as close as possible to the required result, depending on what your aim is and how pitch perfect you intend to be.

STYLING THE STYLUS

The stylus is basically just a wire that temporarily connects the two keyboard contacts to the non-inverting input of the op amp. However, to make a good contact between the stylus and both resistors, a small dome headed bolt soldered to the stylus wire should do the job well. However, if you don't want to solder, you could easily fix the wire to the bolt using a matching nut. The main criterion is to use a bolt with a domed or round head, preferably one that is nice and shiny, so as to make a good electrical contact between the two small wire loops on the breadboard. The wire for the stylus is a couple of flexible breadboard links, but any (not too thick) flexible wire would do just as well, so long as it will easily plug into the breadboard.

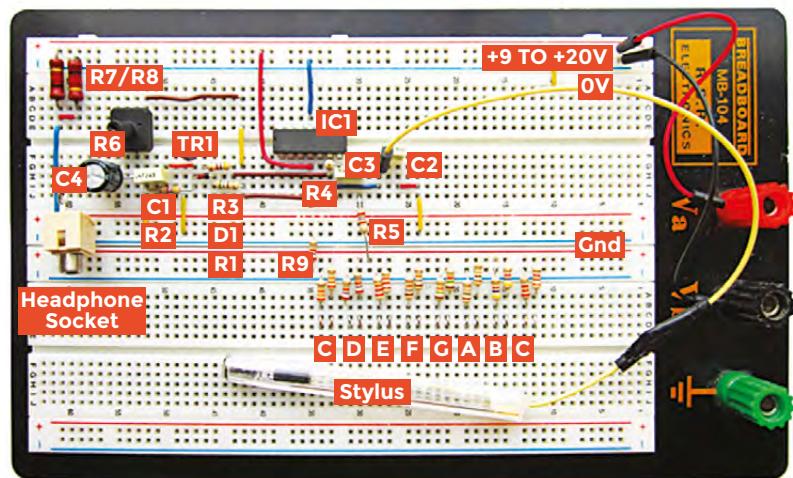
To make the stylus easy to handle, the case of an old ballpoint pen (the cheap kind) was shortened by cutting a small amount from the pointed end, so that it nicely fitted the diameter of the bolt, then cutting off about half of the other end to make it not too unwieldy. Finally, the two ends of the stylus were sealed with hot glue, making sure not to get any glue on the contact surface of the bolt.

THE POWER SUPPLY

The Sineophone can be powered from any DC supply between 9V and 20V, and requires less than 20mA of current. So, 1 x 9V battery or 2 x 9V to make 18V would be fine, or a handy DC 'Wall Wart' type of supply can be used, 12V seems to be about ideal.

WANT MORE NOTES?

You may also wish to extend the range of notes on your keyboard, which can easily be done by using the



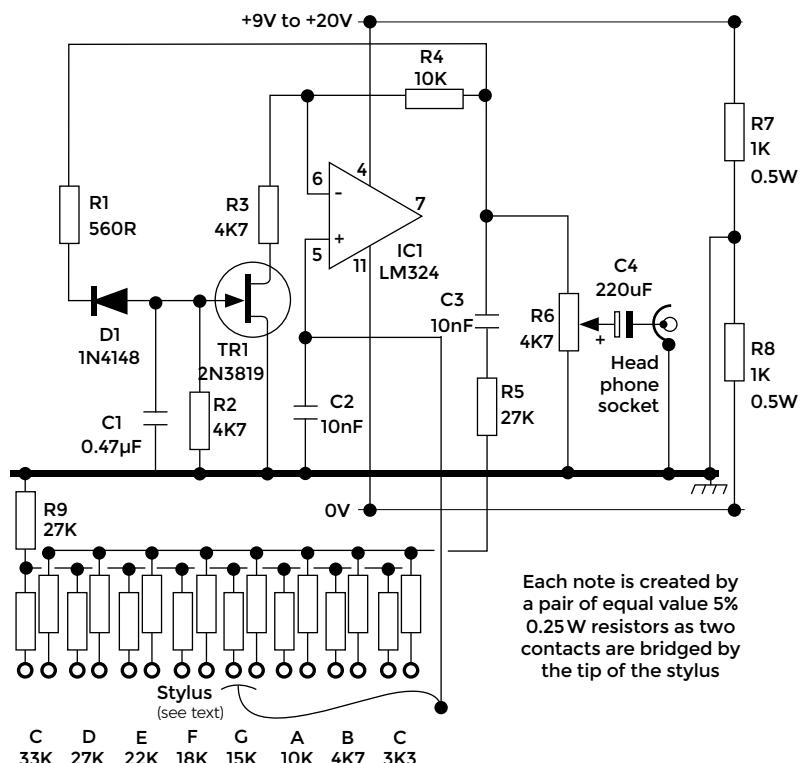
method of calculation described above, for each of the extra frequencies needed.

WANT MORE POWER?

The Sineophone project is intended to introduce a different approach to building a musical instrument, so additions such as extra power amplifiers haven't been included. The output from the JFET-controlled Wien bridge oscillator should be sufficient to drive a pair of headphones (in mono), and a simple volume control (R6) is included to keep the sound level under control. However, it is possible to use the output to drive a power amplifier, for some open air busking. □

Above The Sineophone layout on breadboard showing the 'keyboard' resistors and stylus

Figure 4 The complete Sineophone circuit, with keyboard and stylus



Build custom LED snowboarding goggles

Shine bright this winter with LED snowboarding goggles



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, chronicling her adventures in making.

Winter is here, and that means snow-blanketed days and long, dark winter nights. In other words, it's the perfect season for flashy wearable projects! In this tutorial, we'll put a strip of programmable LEDs into a pair of snowboarding goggles for a futuristic look. Sport them in the snow or at the lodge, and bring a blinking rainbow of joy to your winter festivities.

If you're a generalist, rejoice! We'll do some soldering, sewing, hot gluing, and even a little programming. This is a great project for beginners with a little bit of soldering experience. For advanced folks, this is a fun, one-day build.

Wearable projects are a great way to combine different types of making and design, and there are plenty of ways to customise your build. Your first design choice is: what goggles to use? Snowboarding goggles are a good base to start with, as most will have ample room for the LEDs, and foam sides that are easy to pass wires through. If you've already invested in an expensive pair of snowboarding goggles, you may want to purchase a less expensive pair for this. Your author purchased the pair used here for about \$40 (£30).

A strip of Adafruit NeoPixels works perfectly in this project, and can be cut to size for a custom fit. If this is your first project using NeoPixel strips, prepare to be hooked! These are flexible metal strips that are packed with individually addressable LEDs. They come in several varieties and, in this project, we're using the most densely packed option: a whopping 144 LEDs per metre. We'll remove the protective sleeve the NeoPixels come in but, for projects where the strip is exposed to the elements, you'll want to leave it on. You can choose between a black or white base strip, the black disappears nicely behind darkly tinted goggles.

An Adafruit Gemma microcontroller is another good choice for this build: it's small, sewable, →



TUTORIAL



YOU'LL NEED

- ❖ **Snowboarding goggles**
- ❖ **Adafruit NeoPixel digital RGB LED strip**
- ❖ **Adafruit Gemma microcontroller**
- ❖ **30 AWG silicone-coated wire**
- ❖ **3.7 V LiPo battery**
1600 mAh
- ❖ **4-way stretch Spandex**
to match your goggles, about
20 cm square
- ❖ **Black gaffer tape**

and has a built-in on/off switch. Instead of sewing the circuit, we'll solder our electrical connections and use the remaining holes to secure the Gemma in place. Programming the Gemma is easy with the Arduino IDE and, even if you're new to coding, we'll get up-and-running quickly by modifying an example sketch included with the NeoPixel library. For experienced coders, this is a great place to make this project your own with a unique animation.

A word about safety: you should be able to wear these modified goggles comfortably over your eyes. We'll be backing the LED strip with tape, but a small amount of light may still come through. Remove the

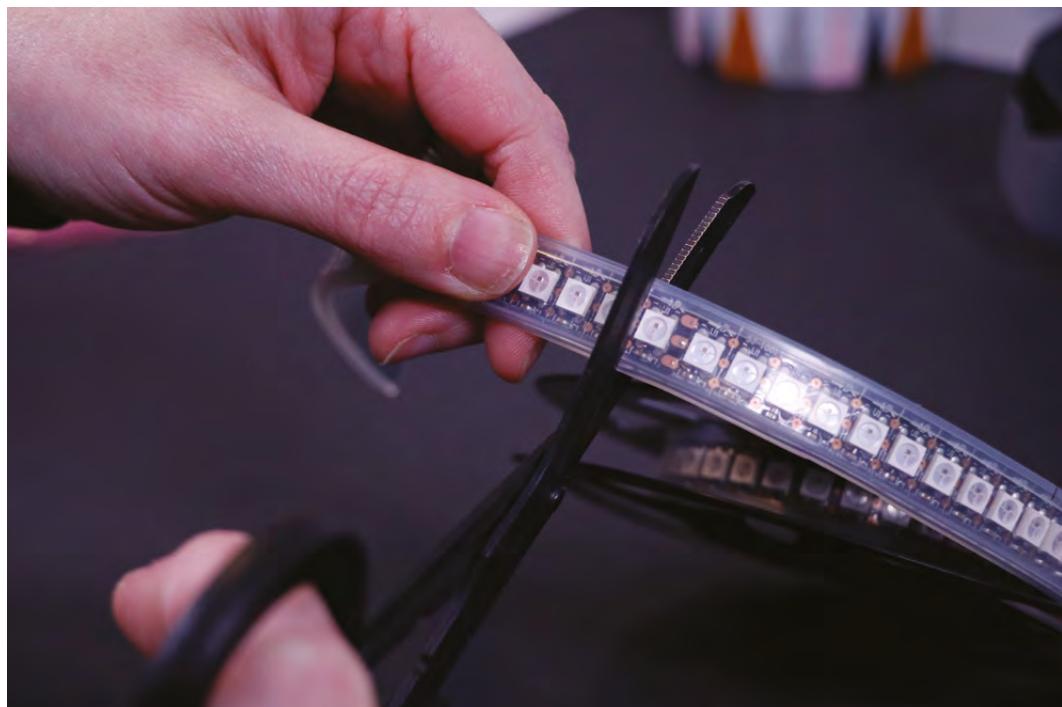
goggles if you experience discomfort. You should have good visibility out of the goggles, however some peripheral vision may be blocked, and you should not wear them while actually snowboarding, skiing, piloting a spaceship, or doing any other activity where impeded vision would be dangerous.

The circuit in this project is simple and versatile, it can be used in many other wearable projects where a strand of programmable LEDs would work. You could use a NeoPixel ring instead of a strand for an arc-reactor-inspired glove. Or, swap the snowboarding goggles for a dog collar, and make your furry companion festive and visible during winter walks.

PREPARE THE NEOPIXEL STRIP

Start by measuring the inside of your goggles. Remove any protective film on the inside of the visor, and measure the length of the visor on the inside, at the very top of your goggles. Our goggles measured about 18cm.

Next, cut your neopixel strip to size using your goggle measurement. Note the cutting lines on the strip. Cut through the middle of the contacts, so that each piece has usable contacts to solder to. The contacts are very tiny, so aim carefully before you cut. Wire cutters are fine for this, but trauma scissors are useful. Cut through the silicone sleeve and the NeoPixel strip at the same time, then remove the strip



Above ❖
Fabric tape measures
are easier to bend
around the goggles

Right ❖
A single snip and
our LED strip is the
correct length



Above The NeoPixels should leave enough space to see properly

from the sleeve. The cut corners will be sharp, it's a good idea to round them, but be careful not to cut off the contacts or any circuitry.

Test the fit by placing the strip into your goggles. It should fit comfortably at the very top of the visor, with a little bit of space at each end. If it buckles or flexes, it's too long. Cut off one LED and check again.

INSTALL THE NEOPIXEL STRIP

Cut three lengths of 30 AWG silicone-coated wire, about 15 cm long each. Use different colours of wire to keep things organised.

Strip and tin one end of each wire. Note the directional arrows that show which way data flows through the NeoPixel strip. Tin the contacts on the input end of the NeoPixel strip, and solder the wires to each contact. We used red for power, black for ground, and white for data. Add strain relief to the wires by applying a small amount of hot glue on each connection.



Above Measure twice, cut once and your strip should fit perfectly

Place the NeoPixel strip into your goggles with the LEDs face down against the visor and the wires pointing to the right. Thread a few centimetres of the wires through the eye of the yarn needle. Use the needle to pass the wires through the foam barrier at the side of your goggles to the inside of the goggle strap. Remove the needle and keep it handy.

Position the NeoPixel strip at the very top of the visor and apply a generous bead of hot glue to the top edge. Do one short section at a time, and be careful while holding the strip in place, as the hot glue will heat up the metal. Our strip felt secure with glue just along the top edge, but feel free to add some glue to the bottom edge if you think you need it. For strain relief, also apply a bit of hot glue to the wires on the right, just before they exit through the foam barrier.

Cover the back of the NeoPixel strip with black gaffer tape. Overlap the bottom edge slightly to secure the strip to the visor and block most of the light from the LEDs. We found it easiest to apply the tape in pieces, rather than using one long piece.

Thread the wires through the yarn needle again, and this time pass them through to the outside of the elastic strap. Give the wires a little extra slack

at this point so they're not strained when putting on and removing the goggles.

This is a great project for beginners with a little bit of soldering experience. For advanced folks, this is a fun, one-day build

CONNECT THE GEMMA

The Gemma microcontroller will be mounted where the wires come through the strap. Hold the Gemma in place, with the JST battery connector to the left. Find the ideal length for your wires by holding them up to their respective pins. Your power wire goes to Vout, ground to GND, and data to D0. Cut the wires accordingly, then strip and tin the ends.

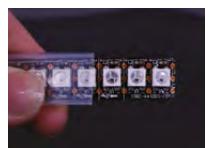
Feed the wires to their pins on the Gemma from the back to the front, and solder the connections on the front of the Gemma. Dab a bit of hot glue onto each connection on the back of the Gemma and let cool. Gently pull the slack of the wires to the inside of the strap, so the Gemma sits flat on the outside of the strap. Then, glue the Gemma in place and stitch to the elastic band through the unused GPIO holes (D1, A1, and 3Vo) with a needle and thread.

Download the Arduino IDE at hsmag.cc/fKwgoe.

Learn more about using Gemma and NeoPixels at hsmag.cc/mcjLa.

TOOLS

- ❖ Measuring tape (the soft type)
- ❖ Soldering iron and solder
- ❖ Wire strippers
- ❖ Helping hand tool with clips
- ❖ Wire cutters or trauma scissors
- ❖ Hot glue gun and glue sticks
- ❖ Sewing machine (optional)
- ❖ Needle and thread
- ❖ Computer and micro USB cable



Above It takes just three solder joints to connect the LED strip

Build custom LED snowboarding goggles

TUTORIAL

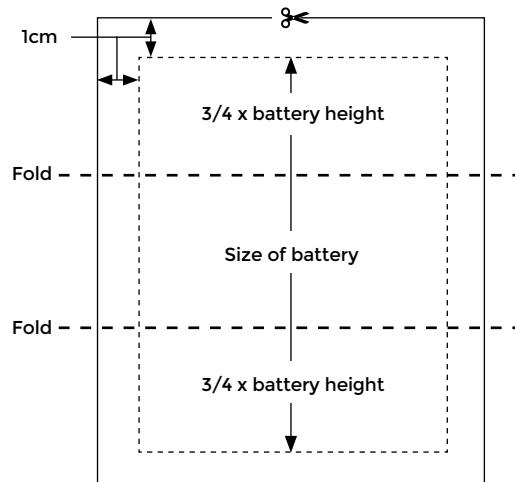


Figure 1
The pattern for the
battery holder

PROGRAM THE GEMMA

Use a micro USB cable to connect the Gemma to your computer, and start the Arduino IDE. If this is your first time using NeoPixels, you'll have to download and install the Adafruit NeoPixel library. Find the Library Manager in the Sketch menu: Sketch > Include Library > Manage Libraries. Search for 'Adafruit NeoPixel by Adafruit' to find the library, and click install.

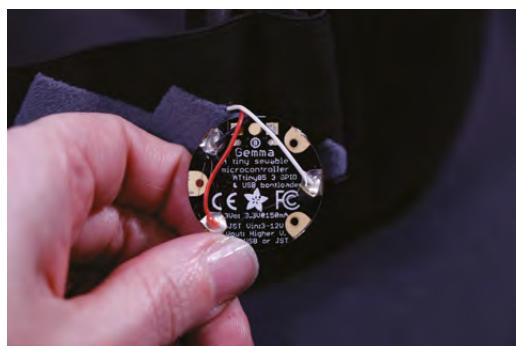
The library comes with some example code sketches that we can modify for use in this project. Go to File > Examples > Adafruit NeoPixel and choose Strandtest. This will pull up a sketch that cycles through several animations for testing your NeoPixel strip. First we'll test our strip, and then we'll modify the code to only show one of the animations the whole time.

Before uploading the code to the Gemma, we need to update two values in it to match our project: the pin we've connected our strip to, and the number of pixels in our strip. This sketch makes it easy to change these values by putting them at the top. Find this line near the beginning of the code:

QUICK TIP

If your project needs more I/O pins than the Gemma has, Adafruit also makes the Flora which is similar but bigger. The Lilypad is an official Arduino board similar in size to the Flora, and makes a great option for wearables projects.

Right ♦
The large pads on the
Gemma are easy to
solder to



#define PIN 6

We've connected our strip to pin 0, so change '6' to '0'.

A few lines down from that, find this line

```
Adafruit_NeoPixel strip = Adafruit_NeoPixel(60,  
PIN, NEO_GRB + NEO_KHZ800);
```

In this line, the first parameter in the parenthesis is the number of pixels in the strip. Change '60' to the number of pixels in your goggles (23 for us).

To upload, we need to tell the Arduino IDE what board we are using: in Tools > Board, select Adafruit Gemma 8MHz. In Tools > Programmer, select USBtinyISP. You're ready to upload!

Press the tiny black button on the Gemma to enter bootloader mode. Bootloader mode will last for about two minutes, a red light will flash on the Gemma during this mode. While the bootloader is active, click the upload arrow at the top of the strandtest sketch, and you should see the code compile and upload to the Gemma. In

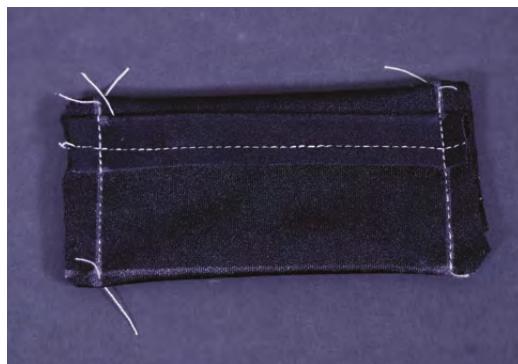
a few seconds, your LEDs should turn on! Watch the pretty lights cycle through the test animations, and decide which one is your favourite.

Now let's go back into the code, and choose one animation for the Gemma to play. Scroll down to this line in the code:

void loop() {

This is the beginning of the section that cycles through the example animations. We'll simply tell the program to ignore the animations we don't want by adding two slashes at the beginning of their lines. Look through the list and find the animation you want to keep (we chose rainbowCycle). Type // at the beginning of the other lines in the list. Here's how ours looked:

```
void loop() {  
    // Some example procedures showing how to  
    // display to the pixels:  
    // colorWipe(strip.Color(255, 0, 0), 50); // Red  
    // colorWipe(strip.Color(0, 255, 0), 50); //  
    // Green  
    // colorWipe(strip.Color(0, 0, 255), 50); //  
    // Blue  
    // Send a theater pixel chase in...  
    // theaterChase(strip.Color(127, 127, 127), 50);
```



```
// White
// theaterChase(strip.Color(127, 0, 0), 50); //
Red
// theaterChase(strip.Color(0, 0, 127), 50); //
Blue

// rainbow(20);
rainbowCycle(20);
// theaterChaseRainbow(50);
}
```

Leave the rest of the code as it is. Press the black button on the Gemma to start the bootloader again, and upload your new code. If your animation plays as expected, hooray! Unplug the USB cable, and let's move on to the battery.

MAKE THE BATTERY HOLDER

We suggest you cover the LiPo batteries with gaffer tape for extra protection, and mark their mAh rating on the tape. Draw up a pattern for the battery holder. Start by marking the size of your battery on a piece of four-way stretch Spandex. Then use the diagram (**Figure 1**) to mark out the rest of your pattern. Hang on to this diagram for future projects, just scale it to the size of your battery.

Cut out the holder, then fold and sew the top and bottom edges as shown. Stretch the fabric slightly as you sew – this will build some stretch into the seams so they don't break when in use. Fold along the fold lines, right sides facing in, and sew the side seams. Trim threads and corners, and flip the holder right side out.

Note where the wires come out of the battery: at the centre of one side. Snip a small hole into one side of the battery holder for the wires to feed through. Do not cut through the seam.

Now, insert the battery into its holder: feed the JST connector wire into the opening of the holder and carefully pull through the hole you made in the side, then simply slide the battery into the holder. It should be a snug fit, but you shouldn't have to wrestle too desperately with the battery to get it in the right place. And you should always handle LiPo batteries with care.



INSTALL THE BATTERY

Hold the battery and holder in place on the elastic band next to the Gemma, and mark the placement of the corners on the band. The wires of the battery should reach the JST battery connector on the Gemma comfortably. If your wires are a bit too long, like ours were, move the battery back or gently form them into a loop.

Remove the battery from its holder, and sew the holder into place on the band, stretching it to reach the corner points you marked. Hand-sew around all four sides with a whip stitch or back stitch. Insert your battery back into the holder, and plug the battery into the Gemma. If you formed your battery's wires into a loop, sew the loop down so they won't catch on anything.

LIGHT IT UP!

Flip the tiny switch on the Gemma to ON, and your LEDs should fire up! Take your new flashy goggles for a spin as they are, or turn them into a sci-fi costume piece by adding painted foam pieces or scavenged greebles. Show us how you customised this project and made it your own! □

Above ♦
LiPo batteries are easy to connect and can power a wide range of projects

Left ✎
A modified strap hides the circuit

Below ♦
The coloured LEDs add a futuristic feel to the goggles



How to develop your own custom Arduino

Discover how to turn your Arduino project into a custom PCB



John Teel

@JohnTeelEE

John Teel is president of Predictable Designs, which specialises in helping entrepreneurs, startups, makers, inventors, and small companies develop and launch new electronic products. John was previously a microchip design engineer for Texas Instruments.

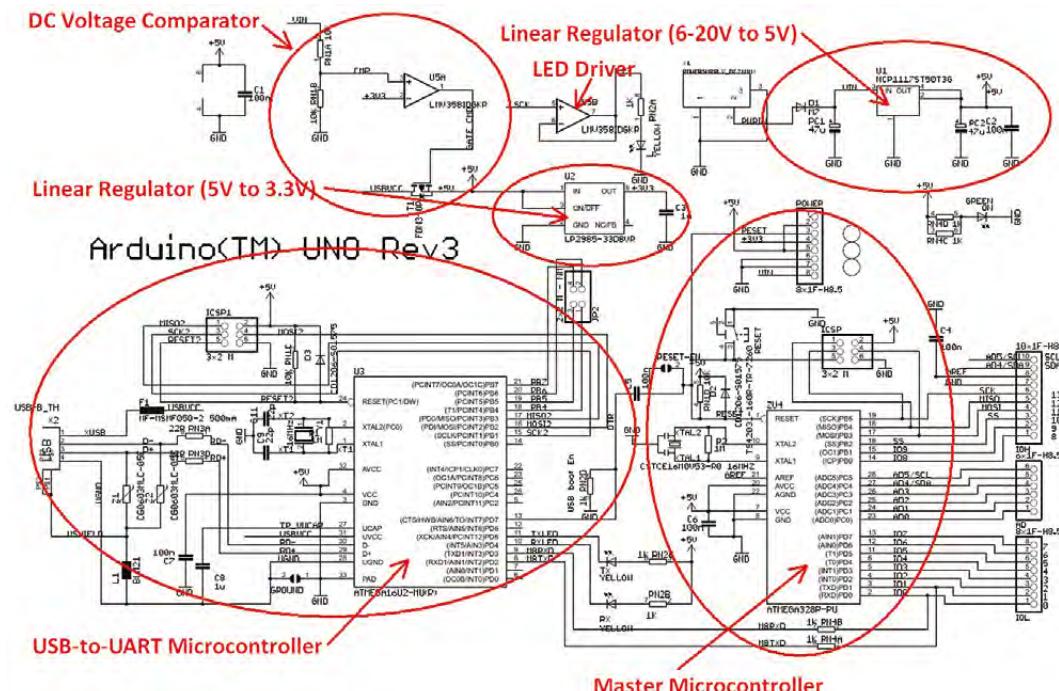


Figure 1 Open-source schematic circuit diagram for the Arduino Uno

In this article you'll discover, step-by-step, how to transition from an Arduino-based project to a custom PCB design.

The Arduino microcontroller development platform is an easy way to create your own electronic DIY project. But, if your ultimate goal is to bring your product idea to market, then you're going to eventually need a custom PCB design.

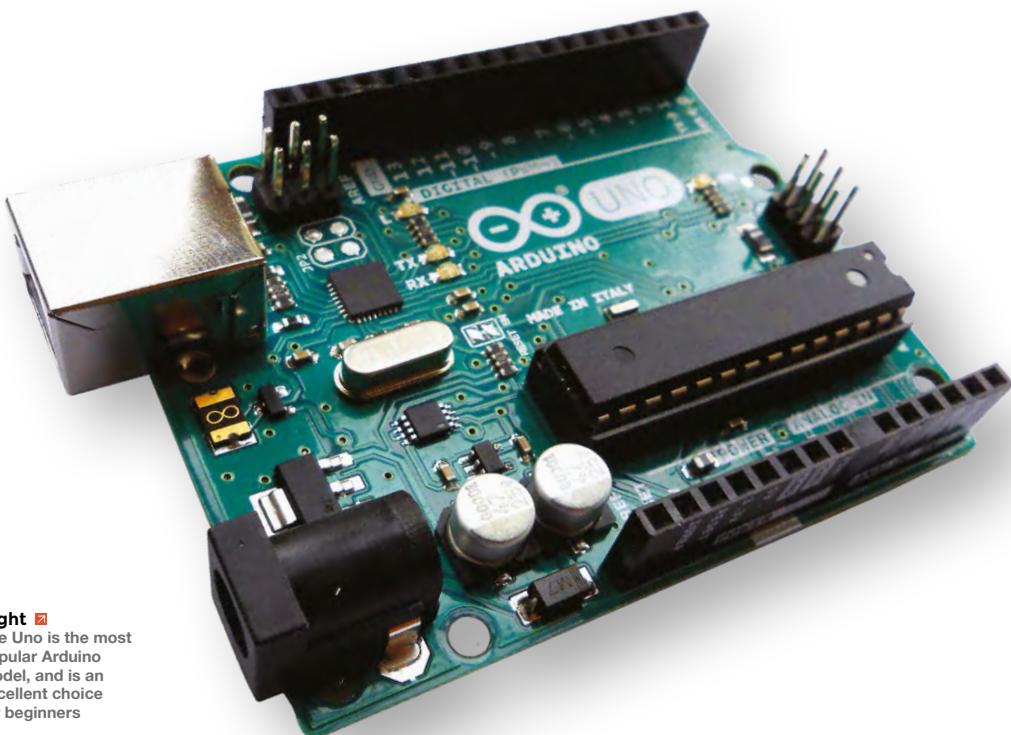
In this article you'll discover exactly how you can develop your very own custom microcontroller PCB. The considerable time and cost required to develop a custom PCB only makes sense if you hope to eventually sell your product. It can also be a fun learning experience to develop a custom PCB for your personal projects.

CHOOSING THE MICROCONTROLLER

Most models of the Arduino use an 8-bit Atmel AVR microcontroller. There are a few that are based on a more powerful 32-bit microcontroller, including the Arduino Zero, Due, MKRZero, and MKR1000. We'll be primarily focusing on the Arduino Uno in this article, since it is the most popular model available.

Note that the Uno microcontroller is packaged in a Dual In-line Package (DIP), which is then mounted in a through-hole socket. A socket facilitates the easy replacement of the microcontroller if it were to become damaged. Use of a socket may be handy in a dev kit, but a socket is almost never a good idea for a custom design you plan to sell.

Arduinos are open-source, which of course means that you can simply download the schematic circuit



Right ↗
The Uno is the most popular Arduino model, and is an excellent choice for beginners

diagram and the PCB layout design files for reference, or to use as a really handy starting point for your own custom board.

In the Arduino Uno schematic diagram shown in **Figure 1**, you may notice there are actually two separate microcontrollers (labelled U3 and U4).

U4 is an Atmel ATmega328P microcontroller, whereas U3 is an Atmel ATmega16U2 microcontroller. Both of these microcontrollers are part of Atmel's 8-bit AVR line.

The ATmega328P is the core microcontroller that is actually running your sketch code. The ATmega16U2 microcontroller is programmed to only act as a USB to UART converter. This is necessary because the ATmega328P does not provide any embedded USB functionality.

As you may already be aware, the main purpose of USB on an Arduino is for programming purposes. It's the simplest way to connect your Arduino to a computer without the need to lug out any extra fiddly hardware.

As this article will discuss in further detail, this is not the case when designing your own custom board. Instead, you will need a special piece of external hardware to handle this USB-to-UART translation. Unless you require USB for some other reason, the ATmega16U2 portion of the circuit can be removed.

Shown in **Figure 2** is the schematic diagram for the custom version of the Arduino Uno discussed in this article. The main differences are that the

“
The considerable time and cost required to develop a custom PCB only makes sense if you hope to eventually sell your product
”

ATmega16U2 has been removed, the power circuitry has been redesigned to use a battery, and an LCD display has been added.

DISPLAY

The Nokia 5110 LCD display can interface to any microcontroller using only three to five digital output pins. The display is powered by 3.3V, so you'll either need to run it at 3.3V, or use level shifters between the display and the microcontroller. →

NEED SOMETHING MORE POWERFUL? UPGRADE TO 32 BITS

There are countless microcontrollers available with much higher performance than the relatively simple 8-bit Atmel microcontrollers used in most Arduinos.

For example, much more powerful ARM Cortex-Mx microcontrollers are available. Cortex-Mx is a very popular 32-bit processor architecture implemented by many microcontroller manufacturers. Microcontrollers are available that are not only much faster than the ATmega328, but which also include significantly more memory and peripherals, to boot!

YOU'LL NEED

- ❖ Arduino Uno
- ❖ Atmel ATmega328P microcontroller
- ❖ Nokia 5110 LCD module
- ❖ Texas Instruments TLV70233 linear regulator
- ❖ ST Microelectronics STB08PMR USB battery charger
- ❖ ST Microelectronics STC3100IQT battery level monitor
- ❖ AVR programmer

QUICK TIP

You will minimise the cost to assemble your PCB if you strictly use surface-mount technology (SMT) components, and no through-hole packages.

TUTORIAL

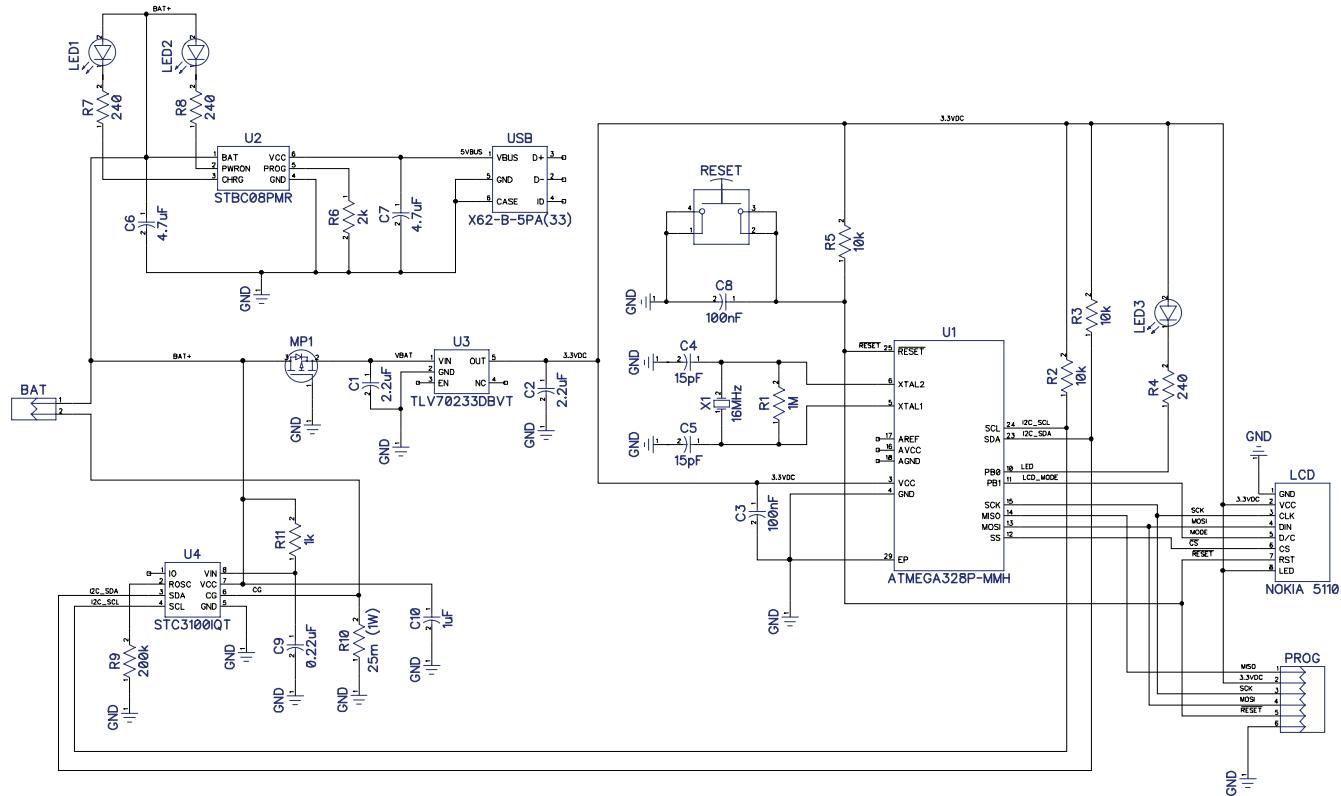


Figure 2
The schematic diagram for the custom battery-powered Arduino design discussed in this article

The Arduino Uno runs the microcontroller at 5 V, so level shifters would be required. However, we've chosen to power the microcontroller from 3.3 V, which eliminates the need for any level shifters.

Referring to **Figure 2**, the display should be connected as follows:

First, connect the VCC and LED pins to a 3.3 V supply, and the GND pin to ground. Now, verify that the backlight LEDs illuminate.

Next, you need to connect the digital data lines. Connect the RST (reset), CS (chip select), D/C (data/command), DIN (data in), and CLK (clock) lines to GPIO pins on the microcontroller.

The RST pin could also be connected to the reset signal, so the display resets automatically any time the microcontroller is reset. The chip-select line can also simply be tied to ground so the display is always selected. Wiring the reset and chip-select pins in this fashion reduces the number of GPIO pins required from five to only three.

QUICK TIP

The key to being able to understand complex circuit schematics is to break them down by each function. First, understand each function, then focus on how the pieces all fit together.

POWER CIRCUIT

Arduinos are designed to be powered by an external DC power supply. A 6-20V supply can be connected to the power adapter plug and from that a 5V supply is generated (U1 in **Figure 1**). Alternatively, 5V can be supplied directly via the USB connector.

For the custom Arduino however, we're going to make it function with a lithium-ion battery which can

be recharged via the USB port. A lithium-ion battery provides a voltage supply of about 3.7V. In our design, we'll use a linear regulator to step down this voltage to 3.3V.

The Arduino Uno uses a Texas Instruments LP2985 linear regulator rated for load currents up to 150 mA. That is sufficient for our application, but in order to give more room to grow we are instead using a TLV70233 rated at 300 mA.

The big downside to linear regulators, compared to switching regulators, is that they can often be extremely inefficient and waste power by dissipating it as heat. This can be especially critical for battery-operated designs, of course. The amount of power wasted by linear regulators is equal to the load current times the voltage difference between

REGULATIONS ■ SWITCHING EFFICIENCY

Switching regulators rapidly turn a series switch on/off at a controllable duty cycle. The duty cycle determines how much charge is sent to the output. Inductors and capacitors are energy storage components that supply energy during the off period. Since the series switch is either on or off, and never operates in the linear region, it wastes very little power. Switching regulators may have an efficiency of 90-95% whereas linear regulators are less than 50% efficient in some applications.

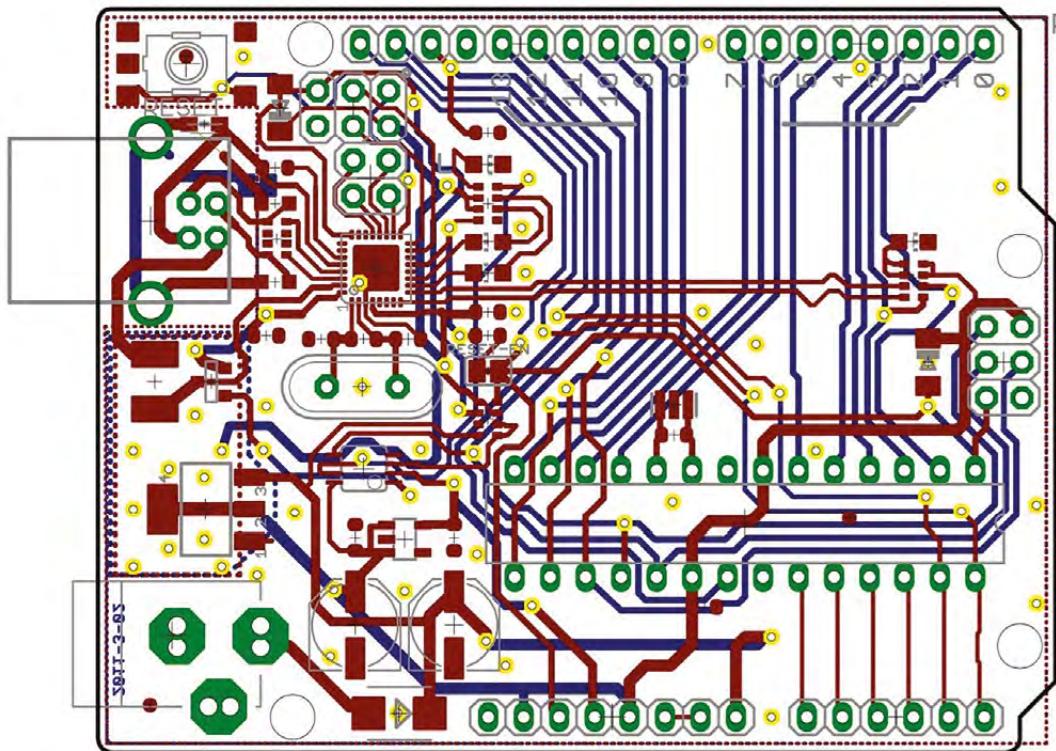


Figure 3 The PCB layout for an Arduino Uno is available to use as a reference for your own PCB design.

the input and the output pins: $P = VI = (Vin - Vout) * LoadCurrent$.

If an application requires a large input-output voltage differential, and/or a high load current capacity, then linear regulators are not likely the best solution.

A linear regulator works well in this application because the difference between the input and output voltage is low ($3.7V - 3.3V = 0.4V$), and the current draw isn't that high.

For recharging the lithium-ion battery we're using the STBC08 battery charger from ST Microelectronics. Always use a battery charger IC specifically designed for the type of battery being used, no cheap substitutes.

DISPLAY ■ RETRO COOL

The LCD display used in this project was originally used in the Nokia 5110 mobile phone that was introduced way back in 1998. By today's smartphone standards, these are quite primitive monochrome displays, but they still have lots of useful applications. They measure about 1.5" diagonally with a resolution of 84×48 pixels, and can be used to display graphics or text. They also feature a white backlight for improved readability in low-light conditions.

For the custom Arduino, we are going to make it function with a lithium-ion battery which can be recharged via the USB port

Charging a lithium-ion battery is comprised of three stages: pre-charge, constant current (CC) fast-charge, and constant voltage (CV) termination charge.

During the pre-charge stage, the battery will be charged using a small trickle current that is about 10% of the fast-charge current.

Once the battery voltage reaches about 3.0V, the charger will enter the fast-charge constant-current (CC) phase.

The battery continues to be charged in constant current mode (fast charge) until the battery reaches about 4.2V. At that point, the charger switches to constant-voltage (CV) mode.

On the STBC08, the fast-charge current is set with a resistor up to a maximum of 800mA. When charging via a USB port you must be careful to not exceed the maximum current capacity of the USB port. The easy solution is to limit the charge current to a maximum of 500mA since all USB ports can handle at least 500mA.

Normally, the fast-charge current should be limited to a maximum 1C charge rate to prevent →

TUTORIAL

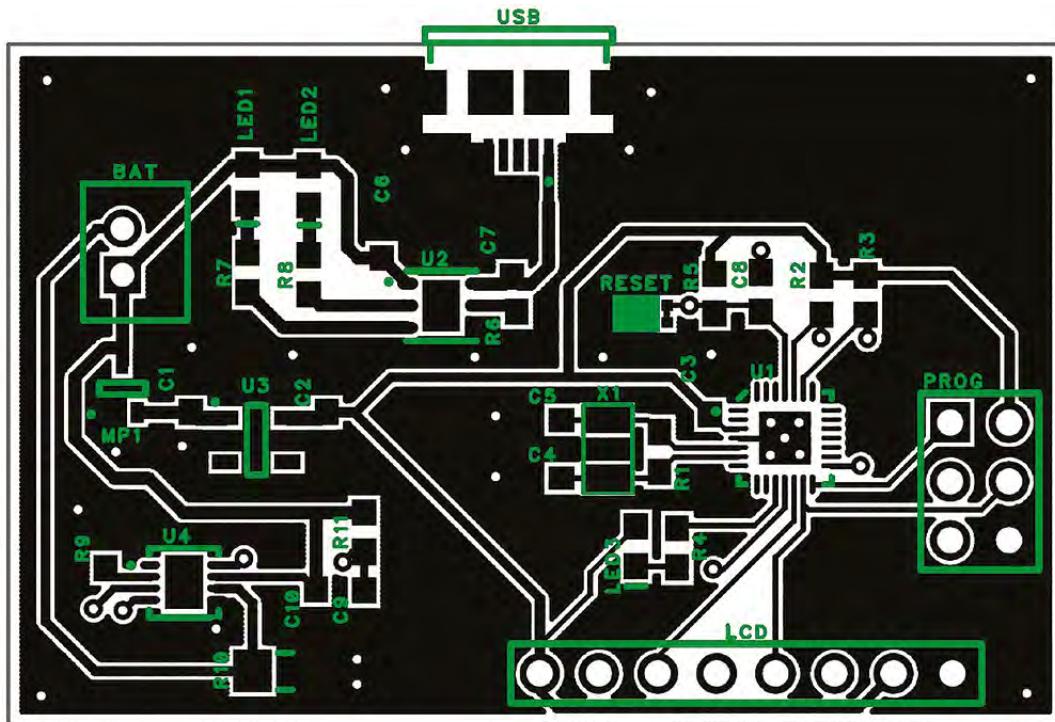


Figure 4 Shown here is the PCB layout for the custom Arduino board

any overheating and early battery degradation. But, every battery is different, so be sure to refer to the datasheet for the specific battery being used.

The other function that is typically required for a battery charger design is a way to indicate the battery charge level. A lithium-ion battery has a very non-linear discharge curve, so it's impossible to measure the state of charge by simply measuring its voltage.

To accurately monitor the charge level for a lithium-ion battery, the best solution is to use a chip designed for this purpose, such as the STC3100 from ST Microelectronics.

The STC3100 interfaces to the microcontroller via a simple two-wire serial interface called I₂C. The microcontroller can now precisely monitor and report the battery charge level.

PRINTED CIRCUIT BOARD (PCB)

The schematic circuit is an abstract engineering diagram. Design of the PCB layout is necessary to turn the abstract schematic diagram into a real-world circuit board.

All Arduinos are open source, so you can easily access the PCB layout design files (see **Figure 3**). However, your PCB design will most likely need to be developed from scratch to meet the size and shape requirements for your specific product.

A microcontroller running at only a few tens of megahertz (MHz), without any wireless functionality, is a moderately easy PCB to design. Designing a PCB requires significant technical skills, but is well within the capability of most makers.

Always be aware that PCB layout design becomes significantly more complex as clock speeds reach into hundreds of MHz, and inevitably, even more so for GHz speeds. You'll find that the PCB layout for a high-speed microprocessor board, such as a Raspberry Pi, is going to be considerably more complicated.

There are two general precautions for laying out a microcontroller PCB.

First, the crystal (which provides a highly precise clock for timing purposes), and its associated load capacitors, must be carefully laid out. This includes placing these capacitors as near as you possibly can to their corresponding pins on the microcontroller chip.

Secondly, be sure to place decoupling capacitors as near as possible to the pins being decoupled.

Once you have the PCB layout design completed (**Figure 4**), and have run the necessary verification to ensure it matches the schematic, it's now time to order the boards.

You will need to send your PCB shop the PCB layout files (in a format called Gerber), along with your bill of materials (BOM).

QUICK TIP

Older linear regulators required the input voltage to be a couple of volts higher than the voltage on the output. Now, most linear regulators are classified as Low-DropOut (LDO) regulators, meaning they can operate with an input voltage only a few hundred millivolts above the output voltage.

All Arduinos are open source, which means you can easily access the PCB layout design files



FIRMWARE DEVELOPMENT

As previously stated, an Arduino is programmed by connecting it to your computer via USB. Using USB allows the Arduino to be programmed from any computer without the requirement for any special hardware.

On the other hand, a custom microcontroller circuit is usually programmed using a simpler serial protocol such as UART, SPI, SWD, or JTAG.

Since these protocols aren't supported by most computers, a specialised hardware device called an in-system programmer (ISP) is required.

They are called in-system programmers because they allow programming while the microcontroller is embedded in the full system. Programming the microcontroller in this fashion allows for much easier testing and debugging.

An ISP is generally used with a specific line of microcontrollers. For example, the Atmel ISP is called the AVRISP mkII (**Figure 5**) and is used to program their 8-bit AVR line of microcontrollers such as the ATmega328P, discussed in this article.

The next step is to begin porting your Arduino sketch over to the native code required for the specific microcontroller chosen.

An Arduino, and custom microcontrollers, are both programmed using C++. But Arduino has greatly simplified the programming process by including an extensive library of numerous functions.

For example, on an Arduino, in order to define a GPIO pin as an output you would call the **pinMode()** function as follows:

```
pinMode(PinNumber, OUTPUT);
```

Then, to set this output low you would call the **digitalWrite()** function as follows:

```
digitalWrite(PinNumber, LOW);
```

Whenever you call either of these functions, the critical code is executed within these two already defined functions.

These functions won't be automatically available when transitioning to a custom microcontroller. Instead, you will just have to develop this fundamental code yourself.

FINAL THOUGHTS

The Arduino is celebrated as a great place to get started with a proof-of-concept prototype, or for just learning all you can about electronics. However, if you hope to sell your product then you'll eventually need a custom PCB in order to

reduce your product's cost, and to fit your desired form factor.

Although by no means trivial, it is within the capabilities of most makers to design their own custom PCB. This is especially true for relatively low-frequency circuits, such as for the microcontrollers used in Arduino.

For higher frequency circuits, such as high-performance microprocessors or wireless functions, it would be advisable to use modules in most cases. The use of a module will lower the design complexity and reduce your FCC certification costs.

Finally, it is recommended that you always be sure to get your design reviewed by an electronics engineer before you proceed with ordering prototype boards. □

QUICK TIP

The 1C charge rate means, for example, if you have a battery with a capacity of 500 mAh, then the maximum fast-charge current should be 500 mA. A 2C charge rate would indicate you can charge this battery with up to 1000 mA of current.

Figure 5 ◊
A custom microcontroller without a USB communication port requires a special hardware device called an in-system programmer (ISP). Shown is the Atmel AVRISP mkII used for programming the firm's AVR series of microcontrollers



Marvellous mechanical music box

Build a touch-activated music box with no coding



Andy Clark

↗ @workshopshed

After an aerospace apprenticeship and electronics degree at Imperial College, Andy took a job as a software engineer. For the last ten years he's been making and repairing in a shed at the bottom of the garden. You can see more of his exploits at workshopshed.com

EXPERT TIPS

I like to use a tool called a Surform, which looks like a small cheese grater. This is good at removing wood a bit faster than sanding alone and can be used to quickly get the wood to the correct size. Take care with the ends as you might cause the wood to splinter or 'break out'.

Coding is great, because it gives your projects flexibility to change. However, for simple well-defined systems, dedicated hardware can do the job just as well.

This project uses a dual version of the classic 555 timer to control the system. A modified micro servo drives a mechanical music mechanism and the wooden box acts as a resonator to amplify the sound. A touch sensor module triggers the timer.

To make the box we need six pieces of wood. I'm using 10mm prepared pine from a local DIY store. It comes smooth all over, but you need to watch for warping when you buy. To save time, I bought pieces that were the size of the box I wanted to make.

Start with the top and bottom, as those are the largest pieces. The length needs to be the length of the servo plus music box together, plus 2 x wood thickness, plus 2 x 3mm for the lining. Best leave a little extra so the servo is not pressing against the

wall. Also allow a few mm for sanding. The width is the width of the servo plus space for the servo arm to rotate, plus 2 x wood thickness, plus 2 x 3mm lining.

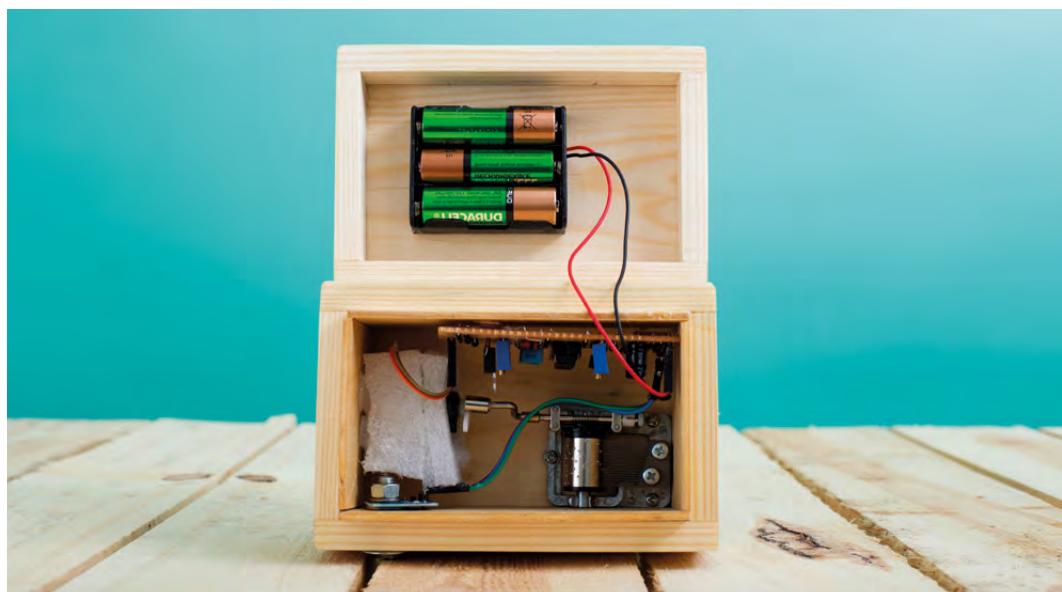
The sides need to be the length minus 2 x thickness. The height of the sides is simply the internal depth of the box. Again, the servo needs space to rotate.

Finally, the ends are sized to fit but should be length minus 2 x thickness and the width minus 2 x thickness. You need to ensure that the ends are at right angles to the sides.

TURN BLOCK INTO BOX

Once the glue is dry, sand your block all over, ensuring that the ends are flat and smooth. Add a small bevel to the edges of the box to make it more pleasant to handle.

With a square and a ruler draw a straight line all the way around the box about two thirds of the way



**Right ♫**

A good box like this can be filled with anything, but we prefer to fill it with MUSIC

QUICK TIP

If you want to drive a bigger motor, then a power Darlington transistor can be used.

A protection diode ensures that the transistor is not damaged by reverse current from the motor. Ensure that your motor is wired so that it rotates in the opposite direction to the music box mechanism.

The last component to mention is the 100 µF capacitor. That is connected across the supply to ensure that the voltage does not suddenly drop each time the motor turns on. Without this you may find that the sensor and timer behave erratically.

up the side. Then, carefully saw along that line to separate the lid from the box. Sand away the saw marks, but try not to sand too much. Keep checking that your two halves still match nicely.

The last assembly step for the box is to add the lining. This is made from 3 mm ply. Cut this to size so that about 10 mm protrudes above the base of the box. Ensure that all of your pieces are the same size and don't forget to allow for the thickness when measuring. Finally, you can finish the box with varnish or Danish oil to give your box protection from the elements and to show off the grain. Two or three coats is recommended.

Once the varnish is dry, drill a hole in the front on one side for the coach bolt. Thread the bolt into the hole and add a washer. Tighten with a nut so that the

square part of the coach pulls tightly into the wood. Now remove the nut.

Drill a hole in the middle of the sensor pad. Thread this onto the bolt, followed by the second washer. Finally, add a nut to secure. You can test the sensor works using the battery pack and a multimeter to see the output voltage.

SERVO SURGERY – WARRANTY VOIDED

A servo consists of a motor, gearbox, position sensor, and a control circuit. For the music box, we just need the motor and gears. The control circuit can be removed and the gears modified to allow continuous rotation.

A micro servo case is held together with small but long screws; remove these and any stickers →

YOU'LL NEED

- ◆ Wood plank
- ◆ 3 mm ply
- ◆ Coach bolt or similar
- ◆ 2 × washer
- ◆ Hand-cranked music box mechanism
- ◆ Micro servo
- ◆ Foam packing material
- ◆ TTP223B capacitive touch sensor module with the sensor pad separate from the electronics
- ◆ 556 dual timer
- ◆ 14-pin DIL socket
- ◆ 1 × signal diode
- ◆ 1 × rectifier diode
- ◆ 2 × 100 kΩ trim pot
- ◆ 1 × BC549 general purpose transistors
- ◆ 1 × TIP31 power transistor
- ◆ 1 × 0.1 µF capacitor
- ◆ 3 × 10 nF capacitor
- ◆ 2 × 100 µF polarised capacitor
- ◆ 1 × 100 kΩ resistor
- ◆ 4 × 1 kΩ resistor
- ◆ Battery box
- ◆ 3 × AA batteries
- ◆ Connectors
- ◆ Stripboard
- ◆ Solder
- ◆ Wire
- ◆ Danish oil or varnish

TUTORIAL

TOOLS

- ◆ Saw
- ◆ Fine cross-head screwdriver
- ◆ Wood glue
- ◆ Drill and drill bits
- ◆ Multimeter

QUICK TIP

Do not try to drive the motor from the output gear, it could strip the teeth off the gears.

Below

The output gear has wider teeth so it can take more torque



A ROUGH GUIDE

Sandpaper comes in different grades or grit sizes, represented by the numbers on the back. For shaping wood, a low number such as 80 grit is good. For smoothing and finishing wood, a higher grade such as 180–240 is better.

that stop you opening the case. Open the case slowly as the parts may fall out.

Open the back of the servo and pull out the small circuit board. Cut or desolder the three wires connected to the potentiometer. Cut or desolder the two wires that are connected to the motor.

Remove the circuit board. Solder a $0.1\ \mu F$ capacitor across the motor terminals and attach two wires that lead out of the case.

Close up this side of the motor and open the opposite side. The main output gear has two features that stop it from rotating continuously. The first is a peg on the underside. This will clash with the other gears if it rotates too far. The second is a

A modified micro servo drives a mechanical music mechanism and the wooden box acts as a resonator to amplify the sound

flat area on the potentiometer shaft that aligns with a D-shaped hole in the gear. Carefully round off the hole in the output gear: you can do this with a small flat-bladed screwdriver. Cut off the peg. Replace the gear on the gear train and rotate the second gear in the train to check that it can complete a full revolution.

THE ONLY THING BETTER THAN A 555 TIMER IS TWO 555 TIMERS

The control circuit for the music box uses a 556 integrated circuit, which is two 555 timers in a single package. The first 555 is running in Monostable mode and needs a ‘low’ signal to trigger it, but the sensor module provides a ‘high’ signal when it is triggered. To invert this signal we can use a simple transistor switch to connect the input to low and a resistor to pull it high when the transistor is turned off.

The first timer is configured in

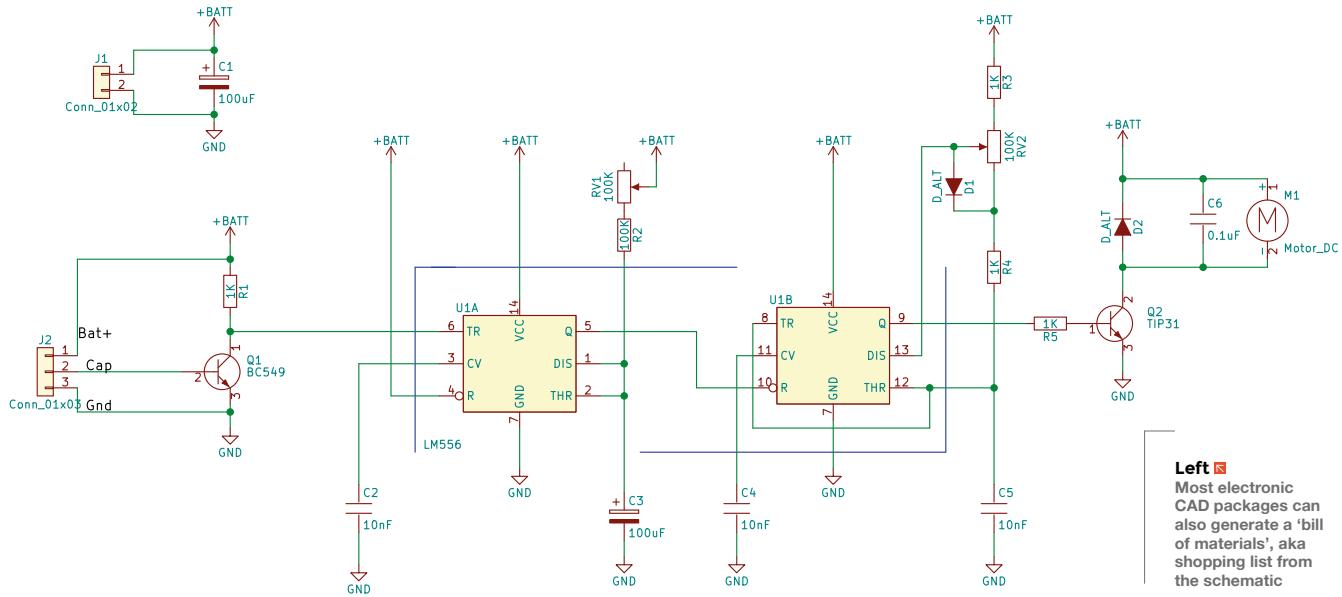
‘Monostable’ mode. This means that, once triggered, the output stays high for a period of time before resetting. The period is defined by the resistors and capacitor connected to the threshold and discharge pins. The recommended values will set the period to around 15 seconds, which should be enough for a couple of plays of your music. The trim potentiometer will allow this to be fine-tuned.

When the output of the first timer is high, it will enable the second timer via its reset pin. This is configured in ‘Astable’ mode. In this mode the timer will send out a series of pulses. The frequency of these pulses is defined by the resistors and capacitor connected to the threshold and discharge pins. The trim potentiometer and diode allow the pulse width of these pulses to be adjusted. This is known as pulse-width modulation or PWM and can be used to control the power to the motor and hence the speed of rotation.

As the motor takes more current than the 555 can provide, a power transistor is used to turn the current on and off to the motor.

PUTTING IT ALL TOGETHER

When building the stripboard, you can design using squared or lined paper and pencil. It is also possible to use an electronics CAD package such as KiCAD to design your board.



Left Most electronic CAD packages can also generate a 'bill of materials', aka shopping list from the schematic

QUICK TIP

Some modern microcontrollers come with built in capacitive sensors, so you could design an equivalent circuit with one of those.

Cut the tracks as per the diagram, remembering that their positions will be flipped when you turn the board over. Solder the wires and small components first, along with the DIL socket. Then add the taller capacitors followed by the transistors. It is usually worth using connectors to attach any external components such as the battery pack, sensor, and motor.

Position the music mechanism and servo in the base of the box. Ensure there is space for the servo arm, and the music mechanism handle can rotate without hitting the sides of the box. Secure the music mechanism with screws.

The motor needs to be at the right height and position so the output shaft aligns with the axis of the music mechanism handle. Cut a rectangle in the foam to fit the servo. Secure the foam into the box using hot glue.

Test that the sensor activates the motor. Adjust the duration and speed trim pots so that the music will play at a suitable pace. You can now enjoy your music box. ☐

QUICK TIP

The sensor module will take a few seconds between each touch to recalibrate.

Right Include packing under the clamps to stop them marking the wood



Build a cheese press

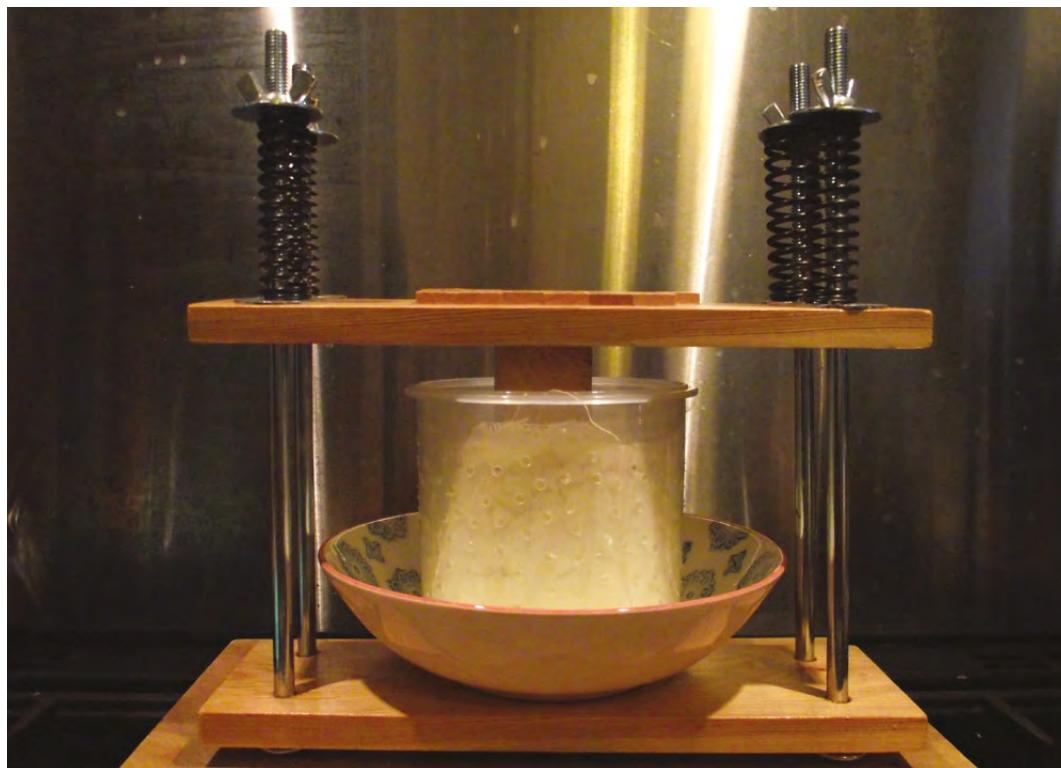
TUTORIAL

Build a cheese press

Because squeezed curds are more delicious than they sound

S, Q₁₀, U, E, E, Z₁₀, E,
F₄, O, R,
C₃ H₄, E, E, S, E,





Cheese is basically curdled milk that's had much of the whey removed. A cheese press is a device for removing the whey by squeezing the curds (typically wrapped in a cheesecloth) until they reach the desired consistency. The resulting cheese can either be eaten fresh or aged to make a mould-ripened cheese such as Brie or Stilton.

There are loads of designs for cheese presses, varying from the simple (like the one in this tutorial) to the complex. In general, the more complex presses can apply more pressure, which allows you to both make a bigger range of cheeses, and larger cheeses.

Our design is kept as simple as possible because the simpler it is, the less there is to go wrong. There are two planks – one at the top, one at the bottom – that are squeezed together using four bolts (tightened with wingnuts). Compression springs between the wingnuts and the plank even out the pressure.

The compression springs need to be able to take enough weight without collapsing completely. Look for ones that can take more than 50 newtons of force. Fortunately, it's trivial to switch them around, so you can experiment to see what works best with the sorts of cheese you're making.

With all the parts assembled, building the press is simple: you just need to drill holes for the bolts in the corners of the planks. These need to be far enough in from the edge that the planks don't split when

Above ◊

You may wish to place your mould in a bowl to catch the whey as it drains out of the curds

pressure is applied. We put them 25 mm in from both the end and the side, but it doesn't have to be exact. We drilled holes 2 mm larger than the bolts we used to give us a little wiggle room if they're not exactly vertical or in quite the same place on the upper and lower planks. Once it's in use, the force being applied will keep everything stable, so this slight wobbliness isn't a problem.

The only other thing to do is cut the chocks. These are just 100 mm × 100 mm × 100 mm cubes of wood that can be inserted between the upper plank and →

FURTHER HACKING

Do you have an insatiable desire for more cheeses and more hacking? If you want to set yourself up to create an even wider variety of cheeses, you'll need a temperature and humidity controlled environment. If you want to mould-ripen cheese, you're going to need to create the right environment for your chosen mould to grow. This depends on the particular cheese, and typically is the cool, damp air found in many caves – which is exactly where cheeses have been traditionally aged. Unless you happen to have access to a hole in the ground, the best DIY approach is usually to modify a fridge.

Ben Everard

@ben_everard

Ben Everard is the editor of HackSpace magazine and a maker whose projects always seem to lead to food, including an irrigation system that waters his vegetables and a 1947 radio (converted to a Bluetooth speaker) that keeps the music flowing in his kitchen.

YOU'LL NEED

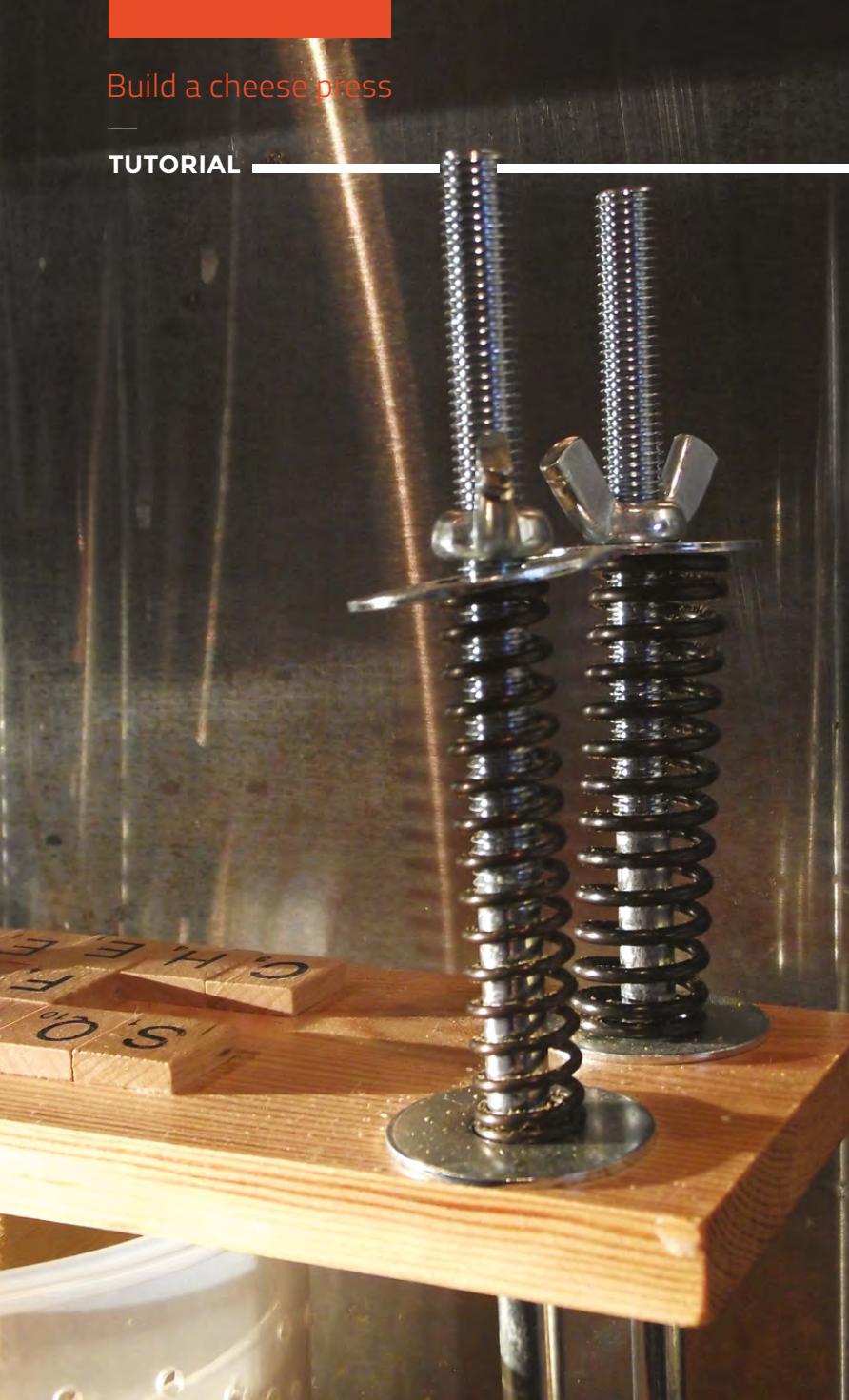
- ◊ 4 × 30 cm M10 bolts
- ◊ 4 × wingnuts
- ◊ 8 × washers
- ◊ 4 × compression spring
- ◊ 2 × wooden plank 18 mm × 169 mm × 400 mm (approx)
- ◊ Wood 100 mm × 100 mm × 200 mm (for chocks)
- ◊ 12 mm drill bit
- ◊ Handsaw
- ◊ Cheese mould

FOR THE CHEESE

- ◊ 4.5 litres whole milk
- ◊ 130 ml white vinegar or lemon juice

Build a cheese press

TUTORIAL



Left ↗

The washers, wingnuts, and springs are used to apply pressure to the curds

the plunger (in the mould) to apply the pressure in the correct place. We opted for chocks, rather than a solid bar, so they can be inserted and removed depending on how full the mould is.

In addition to the press, you'll need a mould. By this, we don't mean the blue fluffy stuff that makes Stilton delicious (though you can get some of that as well), but something to shape your cheese. You can buy these or make them. They should be shaped to allow the pressure from the press to push a plunger into the mould, pressing the cheese curds together, and with space for the whey to escape as the pressure increases. Ours is made from a cylindrical Tupperware pot with holes drilled in the side and the top trimmed down to make it fit inside. If making your own, make sure that it's made from food-grade material, as some plastics can leach dangerous chemicals into the cheese – this is a particular risk here as the cheese is acidic.

Ideally, you'll want to know how much pressure your cheese press is exerting. You may remember from school that:

$$\text{pressure} = \text{force} / \text{area}$$

You need to know the force your press can exert and divide this by the area on the top of your cheese mould. Technically, force should be measured in newtons but, in practice, it's often measured using weight – 1 kg of force is the force of gravity on a 1 kg weight.

If you have scales that can measure at the appropriate degree of accuracy, then you can use these to calibrate your press (by measuring the length of the springs at different amounts of force you can then reapply this amount of force without using the scales). You can get by without doing this, at least with basic cheeses, but you'll have more success if you can consistently apply the correct pressure to your cheeses.

MAKING CHEESE

That's the press made. Let's get on with making some cheese. Paneer is a great first cheese. It's fairly straightforward to make, but still demonstrates much of what you need to make cheese. As it's a fresh cheese without any ageing, you can enjoy the fruits of your labour quickly.

The first part of making paneer (as with most cheeses) is curdling the milk. With paneer, this is

OTHER PRESSES

You don't actually need a cheese press to make cheese. Anything that can apply pressure to a cheesecloth-wrapped ball of curds will do. One common option is to place a pan full of water on top of the cheese. This works to an extent, but it can be difficult to get the balance correct to supply even pressure over a long period of time.

If you find yourself wanting to make larger cheeses or applying more pressure than our little press can handle, the next

step up is a 'Dutch' press, which uses levers and weights to apply pressure to the cheese. While these are more complex than ours, they're still within the reach of someone with modest woodworking skills. If you're really looking to increase the quantity you're making, you might want to go hydraulic. A car-jack and a suitable frame should be able to apply vast amounts of pressure, but make sure the frame's strong enough as any structural failure could be catastrophic.

done by adding something acidic, but with other cheese, this is often done by adding rennet.

Heat your milk in a large metallic pan until it's 90°C. Milk has a tendency to stick to the bottom of the pan and burn, so heat it slowly and stir it often. It's not a complete disaster if a little of it does stick to the bottom of the pan, but the less that does, the better the cheese will be. Once you've hit 90°C, turn the heat off, add the vinegar, give the mixture a good stir, then put a lid on the pan. You should see the milk start to split and curdle almost instantly, but it's best to give it at least ten minutes to properly curdle. If it doesn't curdle, you could try adding a little more vinegar and waiting longer.

While this is happening, it's a good idea to scald the rest of the equipment. This will help keep everything as sterile as possible. Pour boiling water over the cheesecloth, mould, plunger, and anything else that will come into contact with the curds. As paneer is eaten fresh and not aged, it's not quite as essential to keep things sterile, but it's good cheese-making practice to keep things clean.

Now that the milk has split, you need to extract the curds into your mould. First, place a cheesecloth in the mould, then – using a slotted spoon – transfer the curds into the mould, leaving as much of the whey behind as possible. It's important to drain the curds in the spoon for a few seconds before putting them in the mould, as this means there's less whey for the press to remove.

Once you've got all the curds in the mould, fold the cheesecloth over the top, place the plunger on there, and put your cheese in the press. Start by adding a little pressure, then slowly build it up by giving the wingnuts a twist every half hour or so. You should

PICKING MILK

If you struggle to get curds, the problem could be the milk. Our paneer recipe has worked well with all the supermarket milk we've tried it with, but other recipes can be more fickle. Most milk is treated to stop it separating (this is called homogenising), and this can cause problems. If you have access to farm-fresh milk, this can be more successful.

In general, the more fat in the milk, the better. Whole milk is the minimum you should use for cheese making. In the UK, gold-top milk from Jersey cows generally has a higher fat content. You can also artificially increase the fat content by adding cream to the milk before starting with the cheese making.

It's best to experiment with the milk you have available to find what works best for you.

MOULDY CHEESE

Cheese gets really comes to life when bacteria and fungi get to work to develop complex flavours. Traditionally these are introduced by storing cheese in the same place as other cheeses and letting the spores transfer across. This is unreliable and risky for the amateur, as it's hard to guarantee what moulds will transfer and it is possible to end up with a dangerous strain of bacteria growing in your cheese. Fortunately, you can buy cultures that you can add to your cheeses. There are three common types of cheese culture:

* **Mesophilic:** The most common cheese culture, this can be used in both fresh and aged cheeses.

* **Thermophilic:** Used in cheeses heated to a hotter temperature.

* **Penicillium:** The fungus that gives blue cheeses their distinctive colour and texture.

Recipes will often call for more than one of these and they may need to be added at different points in the cheese-making process. There's a great range of recipes to try at cheesemaking.com.

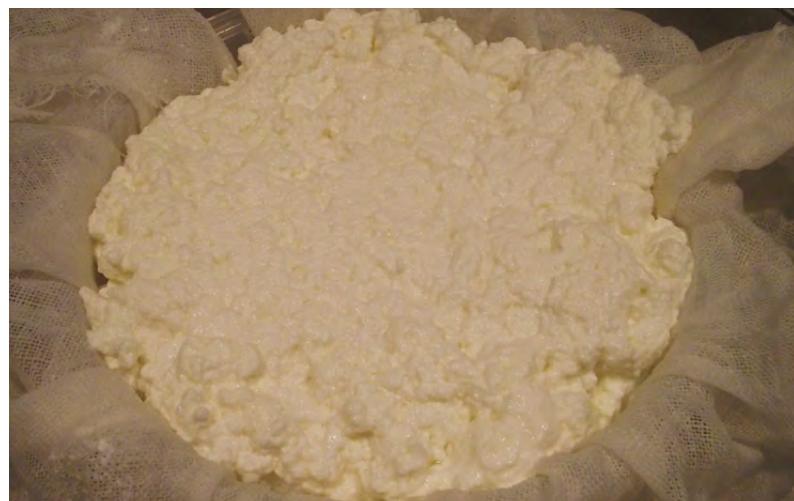


build up to around 150 – 200 pascals pressure (which, in our 20cm diameter mould, translates to a little over 10kg weight). You should find that the plunger moves down the mould as the pressure is increased. You might find out that you have to add an extra chock to help it push the plunger deeper into the mould.

The cheese will need at least a few hours of pressure, and overnight is best. Once that's done, take the pressure off the cheese, take the cheese out of the mould, and you have your own paneer, which is almost ready to eat. Paneer is typically eaten cooked – either fried or grilled. It's delicious on its own, but even better when added to curries. □

Above Move beyond paneer and recreate your favourite cheeses with recipes from cheesemaking.com

Below Curds in the mould waiting before applying pressure. The more whey you drain before placing the curds in the mould, the better your cheese



Code your own smart home with OpenHAB 2

A DIY home automation solution for everyone

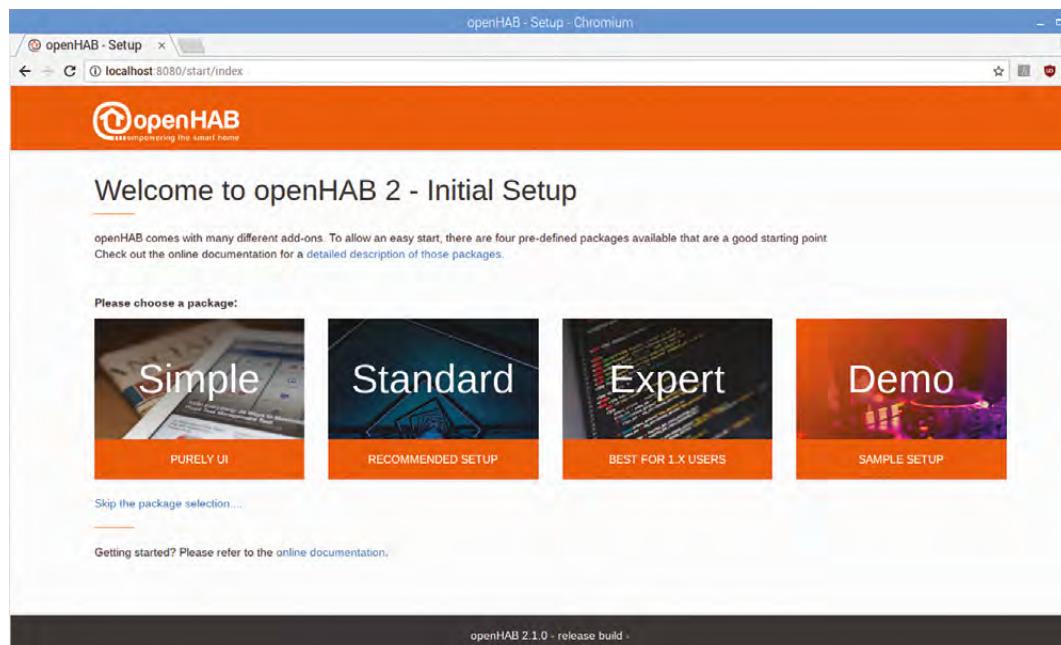


Inderpreet Singh

@ip_v1

An electronics engineer by profession, Inderpreet Singh is also a freelance maker and author. Based in Amritsar, India, he is a fan of the Raspberry Pi and has a great interest in home automation

Figure 1
Setting up things is really easy with OpenHAB 2 and can be done graphically using the web UI



YOU'LL NEED

- ❖ A Raspberry Pi 2 or 3 with power adapter, keyboard, mouse, and display
- ❖ 16GB memory card
- ❖ Latest version of Raspbian OS pre-installed
- ❖ An internet connection

Home automation can be a thing of beauty or the stuff of nightmares, depending upon your project size and the kind of software you use. OpenHAB, by the Eclipse Foundation, is the leading open-source home automation project out there, and boasts of features such as portability and in-built support for most connectivity standards. In this tutorial, we take a look at setting up your own Raspberry Pi OpenHAB 2 device, and how to connect it to the internet. Additionally, we take a brief look at the powerful rules engine of OpenHAB 2 and show you how to automate a simple task.

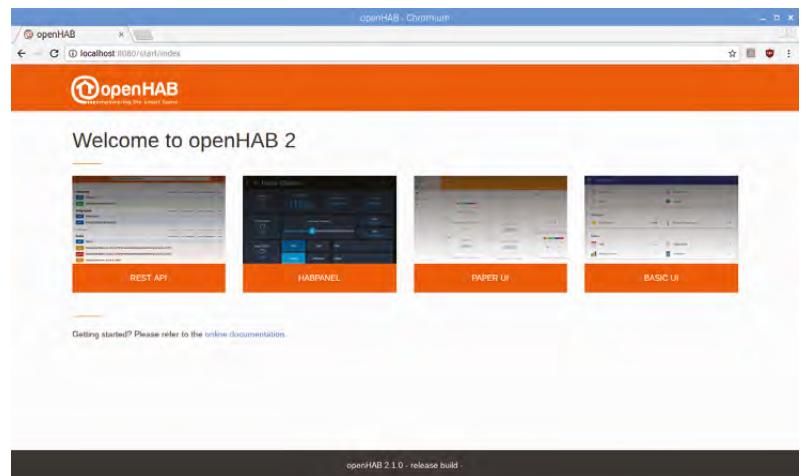
OpenHAB version 2 is the latest release of the offering and can be run on just about anything that has the Java Runtime. There are four parts to this simple guide:

1. Setting up OpenHAB 2
2. Getting data into OpenHAB 2
3. Adding rules to automate a task
4. Access OpenHAB over the internet with myOpenHAB

PART 1 SETTING UP OPENHAB 2

There are a number of ways to set up OpenHAB 2 (hereafter referred to as OH2) on the Raspberry Pi, including the dedicated OpenHABian distribution that works out of the box – hsmag.cc/UPeutC.

Unfortunately, this option does not include a standard desktop and is meant to be used without any kind of monitor or keyboard attached directly to it. So instead, on this occasion, we will use the OH2 package repositories and the apt package manager to set up the service.



STEP 1: Adding the repositories

To allow apt to download and install OH2, we need to add the repository link, as well as add HTTPS support. Open a Terminal and enter the following command:

```
wget -qO - 'https://bintray.com/user/downloadSubjectPublicKey?username=openhab' | sudo apt-key add -
sudo apt-get install apt-transport-https
echo 'deb https://dl.bintray.com/openhab/apt-repo2
stable main' | sudo tee /etc/apt/sources.list.d/
openhab2.list
```

This method has the advantage of being able to update OH2 using apt when newer releases become available.

STEP 2: Install OH2

To install OH2, we need to first synchronise the packages and then install using the following command:

```
sudo apt-get update && sudo apt-get install openhab2
```

This will install the OH2 base system. However, to talk to the various devices and protocols, add-ons are necessary. You have two choices at this point; you can either select to add all the extensions right now or add them as per requirement later. You may, like us, prefer the latter approach since the idea is to have it connect to the internet anyway, and we can always download the necessary stuff.

STEP 3: Running OH2

The next step is to run OH2 for the first time, and for that you have to run the following command:

```
sudo systemctl start openhab2.service
```

This will start OH2 in the background, and if you optionally want to start automatically every time the Raspberry Pi boots up, execute the following command:

```
sudo systemctl enable openhab2.service
```

The first boot should take around 15 minutes and you will see the CPU usage ramp up during that time. Be assured that once it has loaded, the CPU usage will drop considerably.

STEP 4: First configuration

In order to access the UI, start the browser on the Raspberry Pi 2 and point it to the address **http://localhost:8080**. You should be presented with the interface shown in **Figure 1**.

We want to be up and running without a lot of messing around, so we will let OH2 create everything

for us this time. This is done by clicking on the 'Demo – Sample Setup' button which will create all the necessary files for us.

Congratulations! You now have a working system with a template all ready and running.

STEP 5: Let's walk around

Once the configuration is complete, the UI will present four options to the user, as shown in

Figure 2. Here is a brief tour of what we have ready.

OpenHAB by the Eclipse Foundation is the leading open-source home automation project out there and boasts of portability and in-built support

If you followed the provided directions, you should have the following options in front of you.

BASIC UI: This is the traditional face of the OpenHAB system. With the demo setup, it is pre-populated with a number of buttons, sliders and icons and we will configure it according to our requirements. Everything you see is courtesy of the file **/etc/openhab2/sitemaps/demo.sitemap**. This file is where the UI configuration data is, while another file at **/etc/openhab2/items/demo.items** is what powers the configuration of the various properties of the elements shown. →

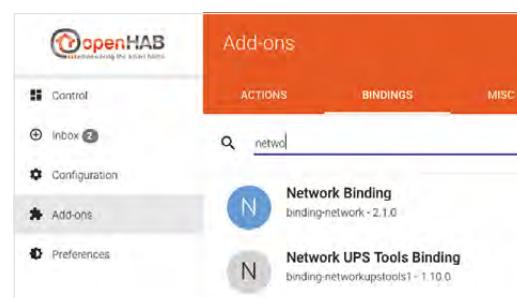


Figure 2 The OpenHAB2 welcome screen presents a number of options for configuration and customisation with ease

Figure 3 Adding functionality such as bindings is just a matter of few clicks

TUTORIAL

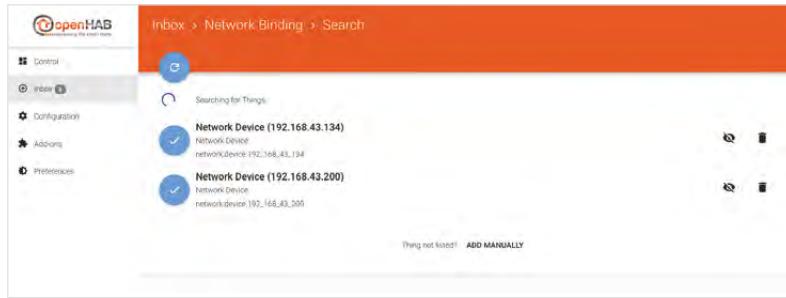


Figure 4 OpenHAB automatically detects new devices and helps configure them

HABPANEL: The newer UI for OH2 is the HABPanel, which features a modern look and feel to the system. It can be configured independently of the Basic UI with a plethora of features. The customisation is done using the web interface and no coding is required for the most part.

PAPER UI: In contrast to its predecessor, OH2 features a GUI for configuring and connecting things for automation. We can install add-ons, as well as detect and add existing devices on the network without writing a single piece of code.

REST API: This menu allows the configuration of services that can be accessed using URLs and HTTP requests. Simply put, this will be of interest if you want to access OpenHAB from a different application, or want to create your own user interface. An in-depth discussion of this topic is beyond the scope of this article, although you are welcome to refer to the original documentation for more information.

PART 2 GETTING DATA INTO OPENHAB 2

Next, we will modify the demo files to allow data to be fed to OH2. We will use Leafpad as our main editor, which is already installed and can be accessed by typing the command in the Terminal window:

```
sudo leafpad &
```

STEP 1: Add weather data

Getting the latest weather update is easy, and by default, the Yahoo Weather binding should be pre-installed in the demo setup. Selecting **Configuration > Things > Weather Information** will allow the selection of Temperature, Humidity and Pressure to be viewed. It will default to Berlin as the location.

Fire up Leafpad and open the file **/etc/openhab2/things/demo.things** and change the WOEI (Where On Earth ID) and the name of the city. The WOEI can be obtained from **hsmag.cc/cdVXmj** by entering the city name, state, and country.

Confirm that the data is correct by visiting the Yahoo Weather page for the relevant city, and the data should be identical. It is that simple!

HELP ONLINE

The OpenHAB project is well documented on the website, hsmag.cc/ihngAc. If you want to find out more about the rules syntax, take a look at hsmag.cc/IUICwe. If you're having difficulty, there's a community forum of OpenHAB users. Take a look to see if anyone's experienced your issue before (it's particularly useful as a source of information on integrating products and services with OpenHAB), or ask a question to see if someone can help.

STEP 2: Detecting devices on the network

In addition to getting data from the internet, we can also add devices that are already on our network, such as other computers and cellphones.

In the Paper UI, click on **Add-ons > Bindings** and type in 'Network' in the search bar, as shown in **Figure 3** (page 107). Install it using the relevant button.

Once installed, OH2 can now ping devices on your network such as cellphones.

In the Paper UI, click on **Inbox > Search For Things > Network Binding**.

You should be able to see a number of things pop up with their IP address, as shown in **Figure 4**, and clicking on the blue tick mark will add it to the Control menu.

Add a phone or laptop that is connected to the network and then visit the Control menu. When the device is disconnected from the network, the switch will show OFF and vice versa. The item will refresh after 60 seconds to reflect the connection status, and this delay can be adjusted in the Configuration menu.

PART 3

AUTOMATE THINGS WITH OPENHAB 2

In order to make the best use of OH2, we will configure it to switch OFF a light in the absence of a device. It requires some modification of the configuration files, as explained next.

STEP 1: Adding an item

In order to use the data from the 'Things', we need to create an 'Item' for each. An 'Item' is like a data object that we can create to house the properties of the 'Things'. These can be of the type Switch, Text, Number, String, and even complex ones like colour, date etc.

First, we need to obtain a 'Thing ID' from the **Paper UI > Configuration > Things**.

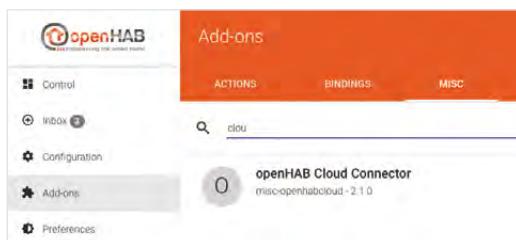
Next, use Leafpad to open the file **/etc/openhab2/items/demo.items** and at the end of the file, add the following entry:

```
Switch my_phone "My Phone's
Wifi is [%s]" <network>
{channel="network:device:8dff330:online"}
```

Where the format is as follows:

```
ItemType ItemName LabelInQuotes Icon
BindingDirectiveInCurlyBraces
```

The '**network:device:8dff330:online**' will be somewhat different and unique in all cases.

**Figure 5**

OpenHAB 2 comes with a plethora of add-ons to ease the configuration process and connectivity with devices, as well as cloud services

STEP 8: Adding a device and switch to the Basic UI

Open `/etc/openhab2/sitemaps/demo.sitemap` in Leafpad and add two entries under the first `Frame {` instance as:

```
Text item=my_phone
Switch item=Light_C_Workshop
```

Here, the first line displays the Switch Item as a Text and the second line adds a pre-existing item from the `demo.items` file. Once you save this, the Basic UI should be updated to reflect two new items.

The workshop light can be turned on and off like a normal switch, but the network icon will only change when your device is connected to the network. Try it out now.

STEP 2: Rules and actions

We can automate the process of turning OFF a light when our device is disconnected from the network. Use Leafpad to open the `/etc/openhab2/rules/demo.rules` file and at the end add the following rule:

```
rule "Office Light"
when
    Item my_phone changed from ON to
OFF
then
    Light_C_Workshop.
sendCommand(OFF)
end
```

The rule is simple and detects a change in the `my_phone` object's state. Once a relevant change is detected, it will send an OFF command to the light, regardless of its current state. That's it! Try turning ON the light and then disconnecting from the network.

PART 4

CONNECT WITH MYOPENHAB.ORG

In order to be able to access the device over the internet, we can employ the free service at **hsmag.cc/dBekSB**. Visiting the link in the browser, it requires four pieces of information, including an

SUGGESTION TIME A FEW OPENHAB 2 IDEAS

Some ideas to try out with your new OpenHAB 2 installation

- Use with Phillips Hue light bulbs (or other compatible smart bulbs) to change colour with weather.
- Make an alarm clock with the rules engine. This could integrate with your lighting, sound system, or anything else connected to your OpenHAB instance.
- Connect with a Kodi media centre to turn off lights when a movie is played. If you're really keen to get hacking, link a popcorn-making machine in as well.
- Monitor power consumption with WeMos Sensor Sockets.

email address, a password of our choosing, and two identifiers from the working instance of OpenHAB 2.

STEP 1 Install the cloud service

In order to communicate online, OH2 uses a service add-on which can be added from the **Paper UI > Add-ons**

> **Misc > openHAB Cloud Connector**, as shown in **Figure 5**. Installation is just a click of the relevant button.

QUICK TIP

OpenHAB 2 can be downloaded for Windows and Mac and run on a laptop for testing on the move.

STEP 2 Getting the keys

Once the add-on is installed, we can use Leafpad to open the files `/var/lib/openhab2/uuid` for the UUID and `/var/lib/openhab2/openhabcloud/secret` to get the remaining pieces of information.

Point the web browser to **hsmag.cc/dBekSB** and then copy and paste the information into the appropriate box. Registering will redirect the browser to the connected instance of the OpenHAB 2 session, as shown in **Figure 6**.

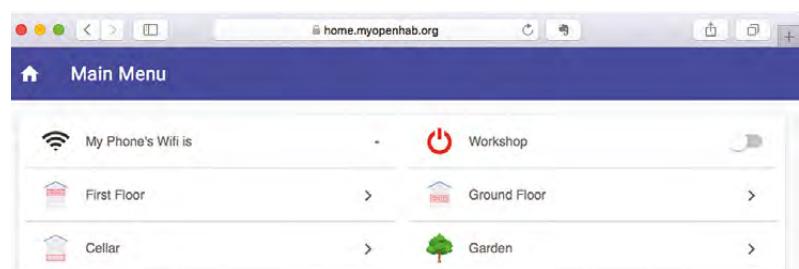
Now the system can be controlled from anywhere in the world, as long as there is an internet connection.

CLOSING REMARKS

In this guide, we have just scraped the surface – OpenHAB 2 is a hugely powerful solution that can be used in a number of applications. We urge you to refer to the official documentation for samples and use cases, and explore more of the possibilities. Happy hacking!

Figure 6

Setting up a system that can be accessed over the internet has never been so easy



HackSpace

TECHNOLOGY IN YOUR HANDS

Download the app

Out now for smartphones & tablets



£2.29

rolling subscription

or

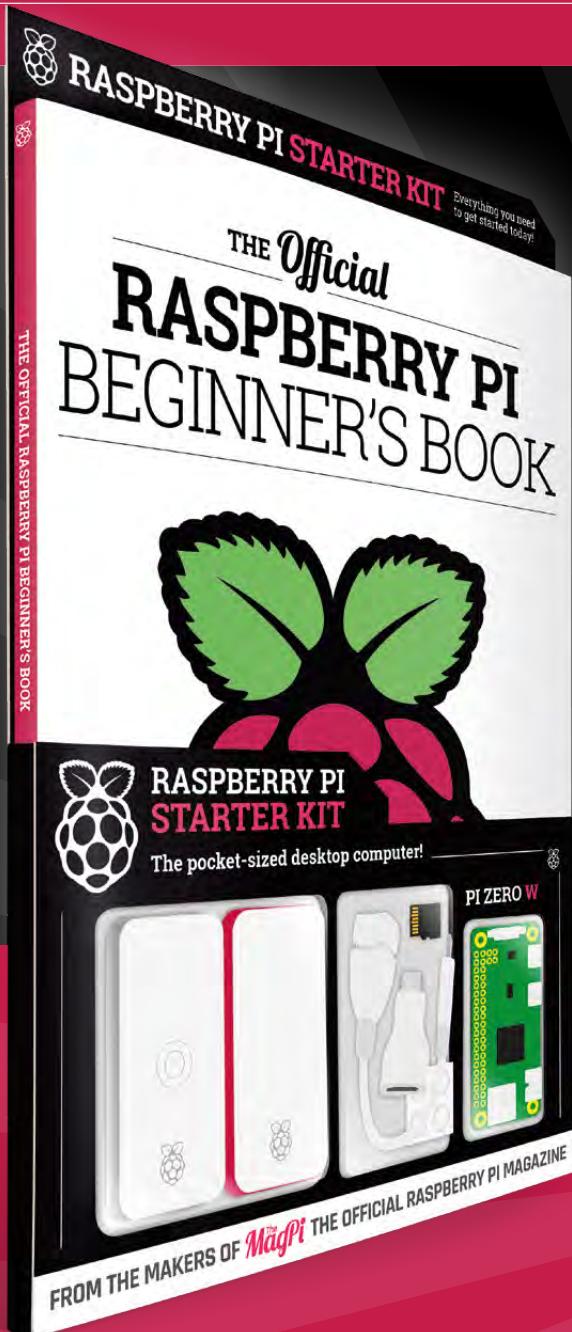
£26.99

subscribe for a year



THE *Official*

RASPBERRY PI BEGINNER'S BOOK



LEARN
COMPUTING
THE EASY WAY!

Includes

- Pi Zero W computer
- Official case with three covers
- USB and HDMI adapters
- 8GB microSD card
- 116-page beginner's book

Available
now

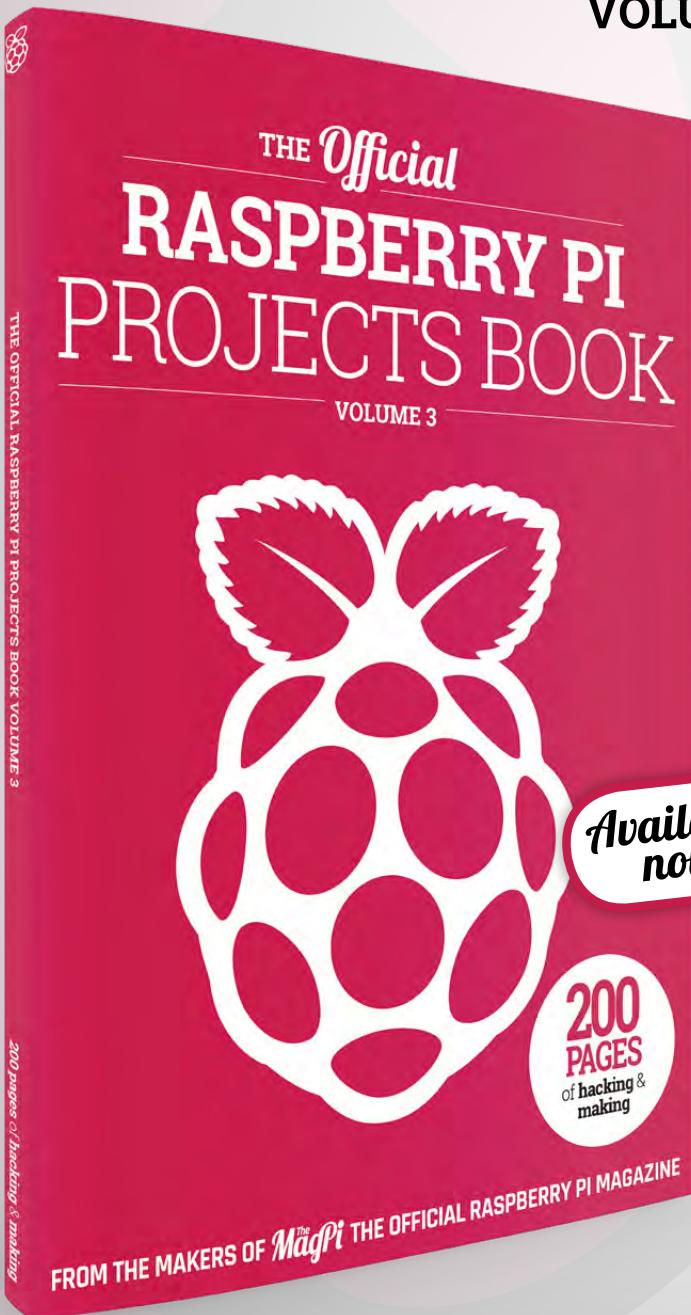


Buy online: store.rpipress.cc

£12.99
200 pages of
Raspberry Pi

THE *Official* RASPBERRY PI PROJECTS BOOK

VOLUME 3



Amazing hacking
& making projects
from the creators of
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

store.rpipress.cc

plus all good newsagents and:

WHSmith **BARNES&NOBLE**



FIELD TEST

HACK | MAKE | BUILD | CREATE

Hacker gear poked, prodded, taken apart, and investigated

PG
114

REVIEW

101 HERO

Is this \$99 3D printer the one to bring cheap home making to the masses?

PG
116

BEST OF BREED

Four Arduino IDEs, clamouring for your love and attention

PG
120

BATTLE TESTED

Essential kit needs thorough testing. This issue: the trusty K40 laser cutter

PG
124

AIY VISION KIT



Google's artificial intelligence tech, packaged in the cheapest possible way for us to mess around with.

REVIEWS

125 Raspberry Pi Programmable Power Switch



128 Bearables

126 Adafruit Grid-EYE

129 Book Review

101Hero: The \$99 printer

\$99 | 101hero.com

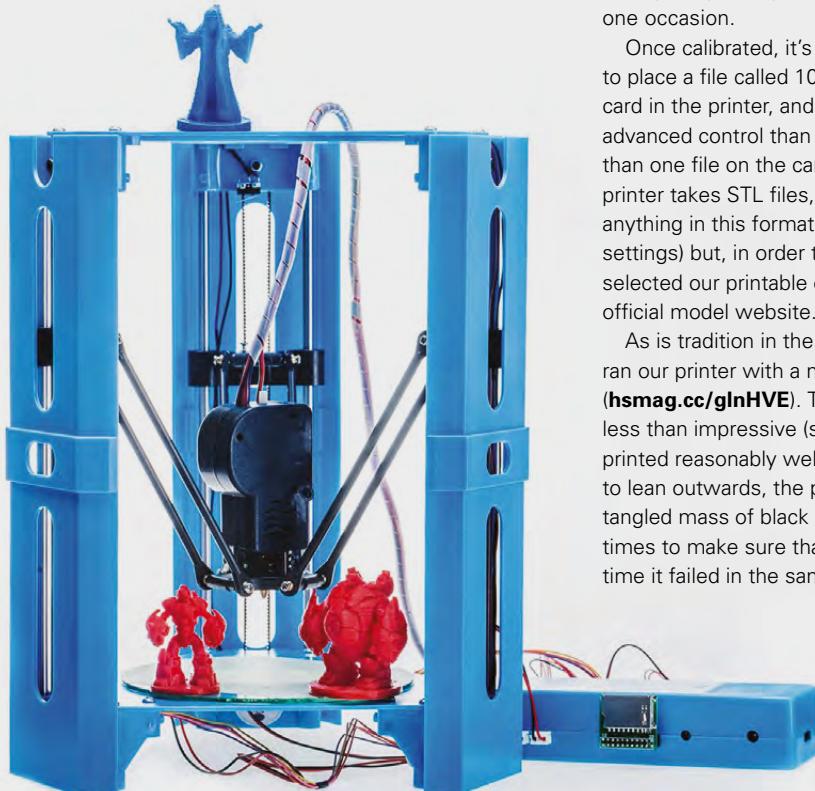
By Ben Everard

 [ben_everard](#)

3

D printers vary hugely in cost, from thousands of pounds to, in the case of the 101Hero, under \$100 (£75).

The 101Hero comes as a kit that you have to assemble, but this is incredibly straightforward – it's just a case of slotting the top and bottom on the three pillars, and joining everything together with a few bolts. While the simplicity of this makes it easy to assemble, it's also concerning. 3D printers aren't fundamentally complex machines. Typically, they're made of three stepper motors that control the position of a print head in three dimensions, and a print head that heats and deposits plastic. The challenge in creating a



3D printer is getting sufficiently accurate control to place the molten plastic in exactly the right position. An error of just a millimetre or two is enough to completely ruin a print (and good-quality prints require even more accuracy). While the 101Hero is easy to put together (requiring the user to add just nine bolts), the resulting structure isn't very stable.

Once assembled and plugged together, the first task is to calibrate the printer. This means adjusting the printer so that the head is exactly where the printer control software thinks it is. With the 101Hero, you need to adjust three hex bolts (one for each motor) to remove any error. It's not a very precise process, and ends up with our print head hitting the printing surface harder than we'd like on one occasion.

Once calibrated, it's time to print. You just need to place a file called 101hero on an SD card, put the card in the printer, and turn it on. There's no more advanced control than this, so you can't keep more than one file on the card, or select what to print. The printer takes STL files, so you should be able to print anything in this format (provided they have the right settings) but, in order to make this a fair test, we selected our printable objects from the 101land.com official model website.

As is tradition in the 3D printing world, we first ran our printer with a model of a small boat (hsmag.cc/gInHVE). The results, were, shall we say, less than impressive (see right!). Although the base printed reasonably well, as soon as the hull started to lean outwards, the print failed, leaving us with a tangled mass of black plastic. We tried the print a few times to make sure that it wasn't a one-off, but each time it failed in the same place.

Left ♦

The simple design makes the 101Hero easy to put together, but it's also part of its downfall

“ The stepper motors in this printer are prone to losing teeth, which renders them useless ”

MOTORING ALONG

Unsure if the error is in the printer or the model, we tried a second example from the official model website, this time of a small rocket: hsmag.cc/LZBYXD. This time, the print completed but the problem became immediately apparent to us – there's a broken motor.

It seems we're not alone in having problems with the 101Hero motors. The stepper motors in this printer are prone to losing teeth, which renders them useless. In principle, switching motors in the printer should be straightforward, but spare parts aren't easily available, and the 1:32 motors are a little hard to find.

Not all the problems in our prints are due to the motor problem. As well as the lean, you can see quite a bit of wobble in the lines. This is all due to the instability in the frame. As the print head moves, you can see the whole printer jiggle, and with each jiggle, the plastic is deposited out of place. There's a community of hackers working on improvements for this printer, from the basic (which includes using elastic bands to tension the arms holding the print head in place) to the more advanced, such as reinforcing the frame and upgrading the motors. These can make matters better, but we're yet to see anything that can improve the situation enough to make the 101Hero a worthwhile investment.

If you're lucky and careful, you can get a 101Hero to print, but even in the best case, the prints are going to be low quality. It might be cheap for a 3D printer, but \$99 (plus shipping) is still quite a bit to gamble on something that might not ever really work properly, and won't work that well if it does. If it at least worked reliably, even at a low quality (or spare parts were readily available), we might be able to recommend it for printing models or other pieces that can accept the inaccuracy.

The simple fact is that making something accurate and strong enough to produce good 3D prints isn't yet this cheap. □



Left ↴
The knotted mess of filament in a failed print will be familiar to most 3D printer owners.

VERDICT

Too delicate and unreliable to for us to recommend as a 3D printer, even at this price.

L **3**/10



Arduino IDEs

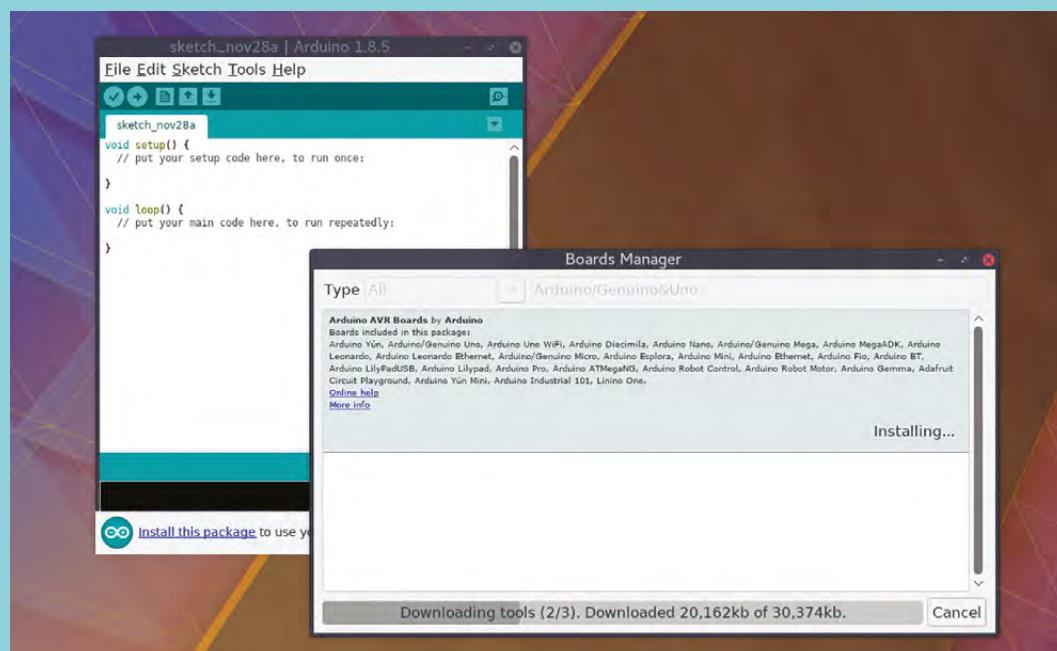
Boost your hacking productivity with a little help

M

any of us enjoy tinkering with Arduinos precisely because they're physical; their flashing lights, bleeping speakers, knobs, dials, and sensors are a world away from screens

and keyboards. And yet, a screen and keyboard is needed to tell your hardware to do anything useful. This is why the development environment for your Arduino code, the IDE, needs to be as innocuous and easy to use as possible, whilst still offering as much power as you need. And there are many Arduino-compatible IDEs to choose between, from platform-specific commercial applications, to open-source plug-ins for your favourite text editor. There are graphical IDEs, command-line IDEs and toolchains that let you manually build and push your code to the device. There's almost too much choice!

Which is why we've pitted a handful of our favourites against one another, keeping to the Arduino ideals of openness and flexibility. This means we've ignored those that may be too specific or platform-oriented. If you are already a keen Visual Studio, Eclipse, Sublime or even Xcode user on macOS, you may want to start with some of the excellent plug-ins Arduino users have built for those, keeping within your familiar environments. Atmel Studio for Windows is also worth considering, as it's a very comprehensive IDE that bundles great hardware support with excellent testing and debugging (there's also the Atmel AVR Toolchain on Linux if you'd rather roll your own). The four we've looked at are excellent all-rounders. They can be run anywhere and on anything, and each is capable of helping you build spectacular Arduino projects. Read on to discover which did best!



Left ♦
The official Arduino IDE is many people's introduction to microcontroller programing

Right ♦
Zerynth's Python support will help some programmers

Arduino IDE 1.8.5 vs Zerynth 2.1

ARDUINO IDE ◆ OPEN SOURCE | arduino.cc

ZERYNTH ◆ FREE UP TO 5 VIRTUAL MACHINES PER DEVICE | zerynth.com

The ‘Arduino IDE’ is the official development environment and, as such, it’s the environment you’re most likely to have used. It has a perfunctory flat design, framed with thick lashings of Arduino blue that’s difficult to miss. Being official is obviously advantageous, and this is still the best way to get started with Arduino and to familiarise yourself with the environment and processes.

You simply need to install and run the IDE, accept the package download request for your specific hardware, paste some code, and click ‘Upload’. Progress is clearly tracked via the output pane situated below the editor, which will also display any problems or errors as they occur, and the editor itself is clear, easy to work with and has excellent syntax highlighting. But that’s it.

The IDE has changed very little over the years; there’s no code completion, no showing where a variable was defined, and no integrated API references (although there is a link).

There are several IDEs that innovate on the official IDE. Windows users could look at the commercial Programino, for example, which brings many of the same concepts and styles found in the original up to date, or there’s the free and cross-platform MariaMole, which adds themes and compiler tuning to a very Arduino-similar IDE. But for a really different approach, there’s Zerynth. This dumps the Processing programming language for Python and offers mass device programming and OTA updates via a system that’s based on a virtual machine. But Python is the main draw, alongside the drag-and-drop support for all kinds of devices, functions, and logic, and it’s an ideal midpoint between Processing and C, the latter of which is more commonly used.

One important caveat is that Zerynth is limited to prototyping boards such as the Arduino Due, and requires an online account, but the editor is configurable and offers more control than the default IDE, as does

the material for getting started, the examples, and the potential for cloud hosting and updates. This flexibility costs complexity, and running your code through a VM to a prototyping board is harder than using the Arduino IDE, and way more CPU-intensive. However, Zerynth is a great choice if you’ve got mass production ambitions, or want to use your Python skills.

The Arduino IDE will neither overwhelm the beginner nor impede the expert, but its lack of a modern UI and more innovative features make it feel like an evolutionary dead end. Zerynth has the opposite problem, bundling too much into a single package that’s initially difficult to navigate. But, the use of Python over Processing (or C!) is a strong advantage, especially if you have Python skills.

Ultimately, we’d like to see something that’s a better combination of both these IDEs, innovating on the original editor without making its on-boarding guides and tutorials redundant. □

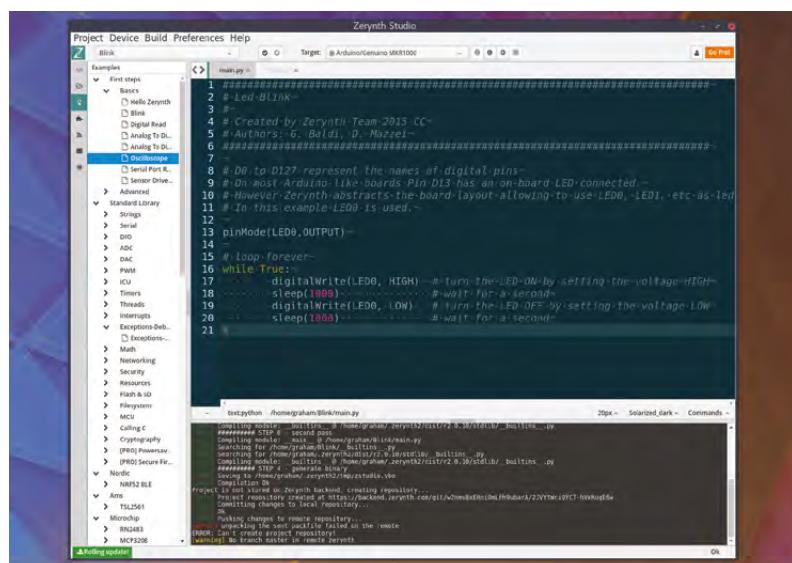
VERDICT

Arduino IDE
Simple and capable but little has changed in years.

8/10

Zerynth 2.1
Use Python with ambitious hardware deployment.

6/10





PlatformIO IDE

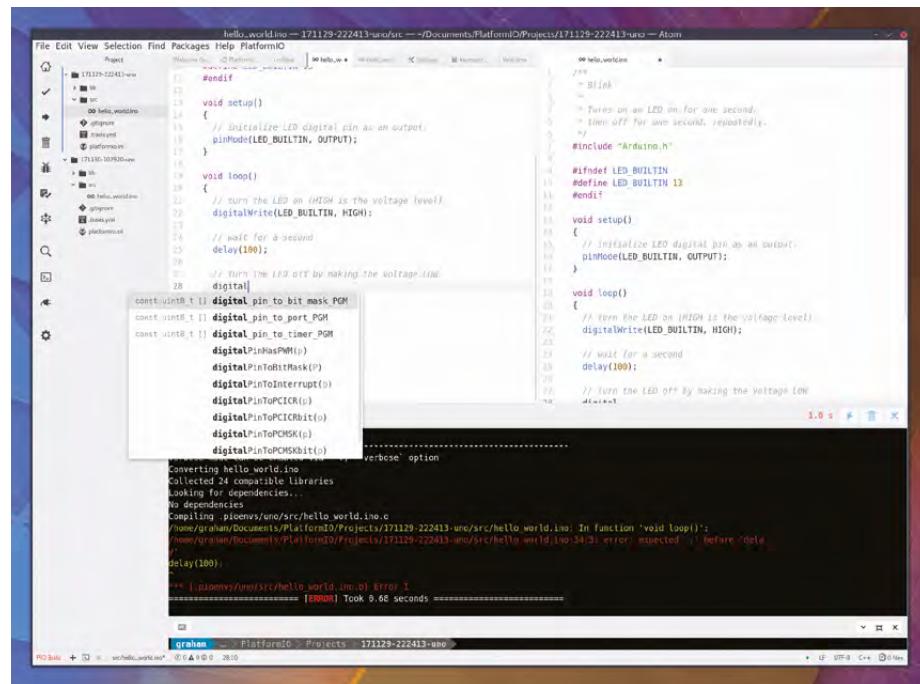
PLATFORMIO IDE ◆ OPEN SOURCE | platformio.org

PlatformIO is a sprawling set of tools, written in Python, that help you build and deploy code on a huge variety of embedded and microcontroller-based platforms. You set up a build chain, write some code, build the code and send it to your device. The difference with PlatformIO is that its device support is phenomenal, and its tool manages the dependencies and requirements of those devices independently.

The Atom editor, at the heart of PlatformIO, is developed by code hosting behemoth GitHub. It supports dozens of languages, as well as many of the processes used by team using GitHub, and as such, has become hugely popular. Its ubiquity has helped Atom become more of a platform than a text editor, enabling users to easily augment their editing functionality with any number of plug-ins; one of which is the PlatformIO IDE.

PlatformIO IDE is the official graphical development of the PlatformIO project. Thanks to Atom's plug-in and packaging integration, installation is as simple as a search and click, transforming your editing environment into a fully fledged Arduino IDE. Vitally, this IDE doesn't feel like a plug-in. The toolbar on the left becomes instantly populated with PlatformIO-specific commands, such as 'Build', 'Upload', alongside opening the embedded serial port monitor, and you can use any of PlatformIO's toolchains, as well as your old Arduino libraries, with very little effort. Just start a new project, select your embedded platform, and start coding. PlatformIO will download all the libraries you need for your chosen board. You can even import your existing 'Arduino IDE' projects and choose to use your local copy of the original Arduino libraries.

Atom's editor is marvellous and utterly configurable, from fonts, colours and styles, right through to code completion, integrated search and keyboard shortcuts. Code completion is one of the best things about this IDE, as you only need to start typing to see the names of matching classes and



methods, and while this works best when you start working with its native C++, it still works with your original Processing projects. If you're a beginner, especially with a poor memory, this can really help you to get started, as you don't need to remember the syntax or even the required data types in the code. The editor will also help with mistakes, such as missing semicolons, and integrates PlatformIO's excellent debugging tools.

The only real disadvantage with the PlatformIO IDE is Atom's dependence on Electron, and subsequently, technologies built for Chrome and Node.js. These make the stack resource-hungry. This isn't important for most desktops and laptops, but it may be important if you want to house your IDE on a Raspberry Pi in the workshop to avoid splats of solder destroying your screen and/or keyboard. □

Above
PlatformIO IDE offers excellent code completion, Atom-based plug-ins, and even the ability to automatically upload your executable code via FTP

VERDICT

Widely used, widely supported, powerful, open, and adaptable.

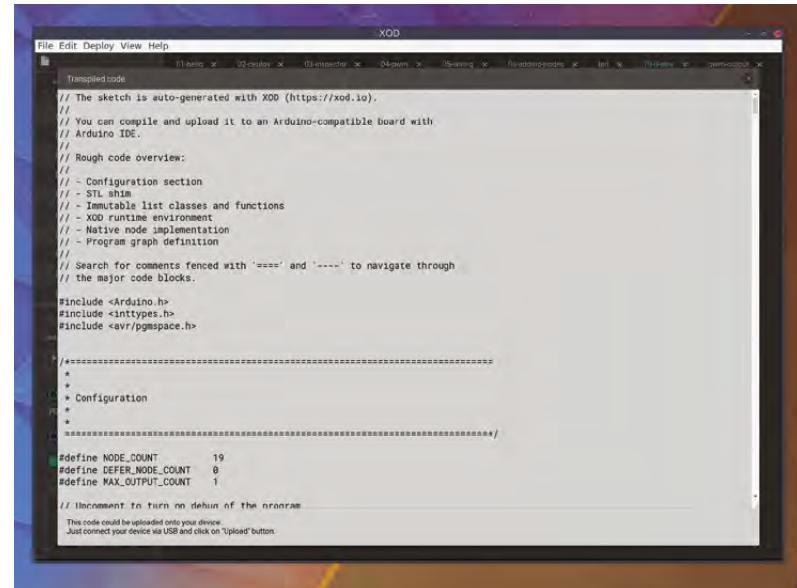
10/10

XOD

XOD ◆ OPEN SOURCE | XOD.IO

There are several development environments that work with Arduino and attempt to implement a visual programming interface. These replace text-based programming languages with a jigsaw of interconnecting elements that implement logic, operators, variables and functions, known as graphical programming blocks. These can be seen in the Arduino-compatible version of mBlock and 'snap4Arduino', both of which integrate the same plug-and-play visual interface created by Google's Blockly, and used in Scratch. Visual blocks can also be seen in Embrio, a real-time development tool that steps back slightly from the graphical blocks to show chunks of code connected via patch cables of inputs and outputs, much like Pure Data or MaxMSP in the realm of real-time audio.

XOD implements its own visual language, and can run either online or 'offline' from your desktop. This flexibility is thanks to the environment being built with the Electron framework, and you can even use the xod.io-hosted version to create your own projects without installing anything further. You will need the desktop version to upload your project to real hardware, however, or you can paste the raw source code generated by XOD into the original

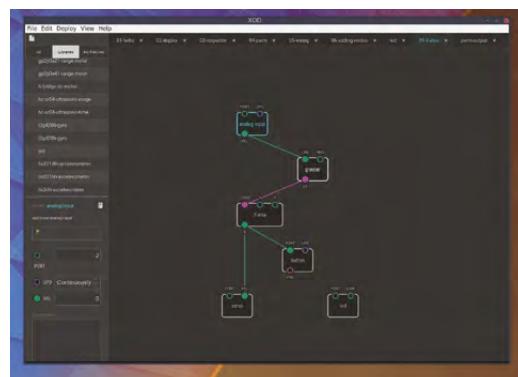


Arduino IDE and run it from there, which could be a useful way of teaching people to code first visually and then from the text source code.

A project is made up of patched-together visual objects. These 'nodes' are dragged from hierarchical menus on the left, and include logic, unit conversion, and even common hardware types, such as buttons, LEDs, gyroscopes, and sensors. To turn an LED on, for example, you send a value representing the Arduino port connected to the LED to the LED node's PORT input and a brightness value to the LUM input. These values can be changed by other nodes, or you can change them using the inspector on the left, just as you might in a visual design tool. In true modular synthesizer style, each of these visual elements is made from even simpler visual elements, and you can see how they're constructed by simply double-clicking on them. This is a brilliant way of learning how chunks of logic work and how you might re-engineer them in your own physical projects.

The real question is, though, whether XOD can stand on its own as a development environment. Graphical programming blocks are obviously easier to understand and use if you have no programming experience, but XOD's implementation of the idea is much closer to real programming, and for small projects, is often quicker at getting results. Even outside of a learning environment, the rapid way you can adjust the configuration of sub-nodes, quickly re-patching the same nodes (or method), is something that can't be accomplished so easily with source code. This makes XOD ideal for prototyping and for quickly testing out ideas, regardless of whether ultimately you export the code and continue development in a text-based IDE. □

Above XOD replaces writing code with a spaghetti of joined-up modules and logic



VERDICT
A completely different approach that helps ideas grow more creatively.

8/10

BATTLE TESTED

K40 Laser Cutter

What's the super-cheap K40 laser cutter like to own, set up, and use?



Martin O'Hanlon

@martinohanlon

Martin is the co-author of *Adventures in Minecraft*, a Raspberry Pi trainer, and blogger at stuffaboutco.de

I have been using the widely available Chinese-made K40 for over a year now; it's been a steep learning curve and in this battle test we'll take an in-depth look at getting it working, common issues, useful upgrades, and tips and tricks.

The K40 is a comparatively small 40W laser cutter and engraver; they're widely available on the internet and while they follow a very similar design and layout, there are a huge number of variations.

It is typical in design: the laser tube mounted is across the back, the cutting head moves along a

2D track, while right-angled mirrors reflect the laser through a lens and focus it on to the cutting bed.

The work area is approximately 300 mm × 200 mm (roughly A4), and it's capable of cutting and engraving on a variety of materials including paper, card, acrylic, plywood, MDF, and natural timber up to 6 mm in thickness (depending on the material).

I have used my K40 for a variety of projects, from creating robot chassis to boxes for holding resistors, to dinosaur-themed keyrings for my son's ninth birthday.

While it requires time, patience, and investment to get and keep it working, it is surprisingly capable.

NOT QUITE PLUG AND PLAY

The first thing that struck me when it was delivered was the sheer size and weight of the box it came in. Out of its box, the main unit measures 81 cm × 50 cm × 26 cm, but it will come to you incredibly well packaged.

The box contains:

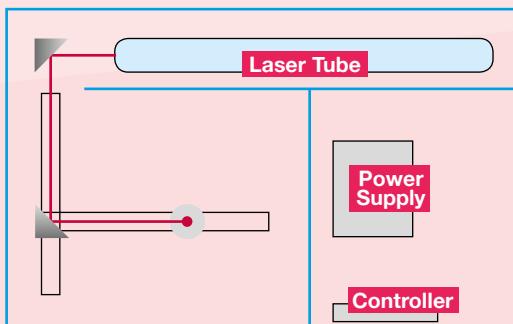
- ◆ Laser cutter
- ◆ Extractor fan and hose
- ◆ Water pump
- ◆ Software on CD
- ◆ USB cable
- ◆ USB security dongle

The CO₂ laser tube is water-cooled and, whilst a water pump is included, a suitable reservoir capable of holding about 20 litres of water is required. I used a plastic packaging box, drilled two holes in the side for the inlet and outlet pipes, filled it with distilled water, and submerged the water pump.

Using a transparent container has the advantage that you can see how much coolant is in the reservoir,

Above Created using the K40 laser cutter, this HackSpace box is ideal for storing all those tiny components



**Above**

The K40 is not 'plug and play' – it requires installation and maintenance



but also more importantly whether it is flowing when the pump is turned on.

If the laser cutter is in an unheated building, or there is a risk that the temperature could fall to near 0°C, it is essential that antifreeze is added to the water. If the water freezes, the CO₂ tube will crack and will need to be replaced – this is a mistake I learned first-hand!

A COOLING BREEZE

Ventilation is required to pull away fumes while in use, especially when cutting acrylic. The supplied fan attaches loosely to the back of the unit and a cable tie is used to hold the flexi tube in place. Using tape to seal up the gaps around the fan improves the ventilation and is well worth doing once everything is set up.

The controls are really simple: there's a main power switch, push switch to enable the laser, a test push button to fire the laser, and a current regulation dial which adjusts the power to the laser.

The mA meter shows the laser power usage and is controlled by the current regulation dial; anticlockwise will turn the power down to its lowest setting. There are no markings on the dial, so I marked up the dial

with the relevant power levels; this is useful when setting up a project.

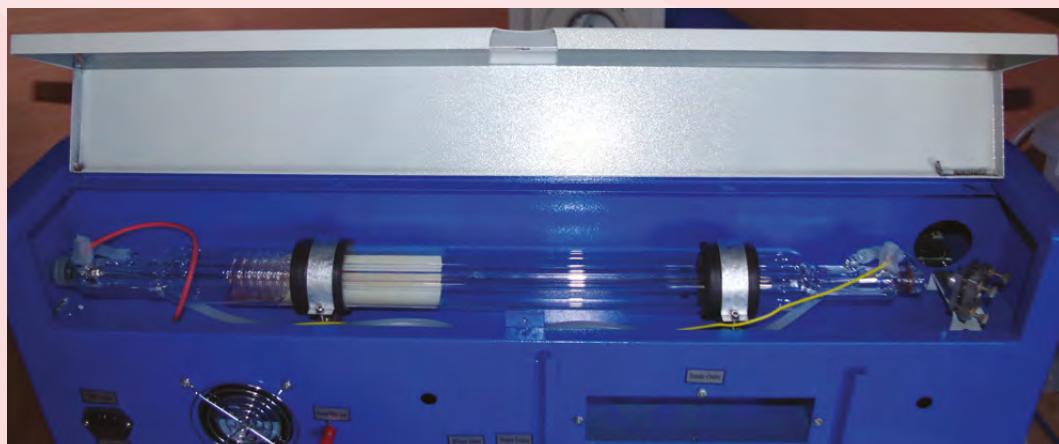
Using as little power as possible to complete the job, and not running at full power, will extend the life of the laser. I found that I rarely had to run the laser above 11 mA to complete most tasks.

When cooling and ventilation was in place, I could test-fire the laser. I placed a sheet of paper under the cutting head, turned the water pump and extractor fan on, powered it up, enabled the laser, and pressed the test button (as quickly as I could) – there was satisfying 'click' and a hole appeared in the paper. Success!

LASER SIGHTED

When delivered, my K40 was aligned almost perfectly and I was able to get started straight away. Comments online suggest this is not the norm and many people find themselves needing to align the mirrors to get good results. I found that over time, I needed to adjust the alignment to keep the machine working well. There is a great guide and how-to at hsmag.cc/CGhNMT.

The mirrors were very dirty when it arrived, no doubt as a result of the transit, but they are easy to →

**Above**

The K40 installed in the shed. You'll need a strong platform and access to ventilation

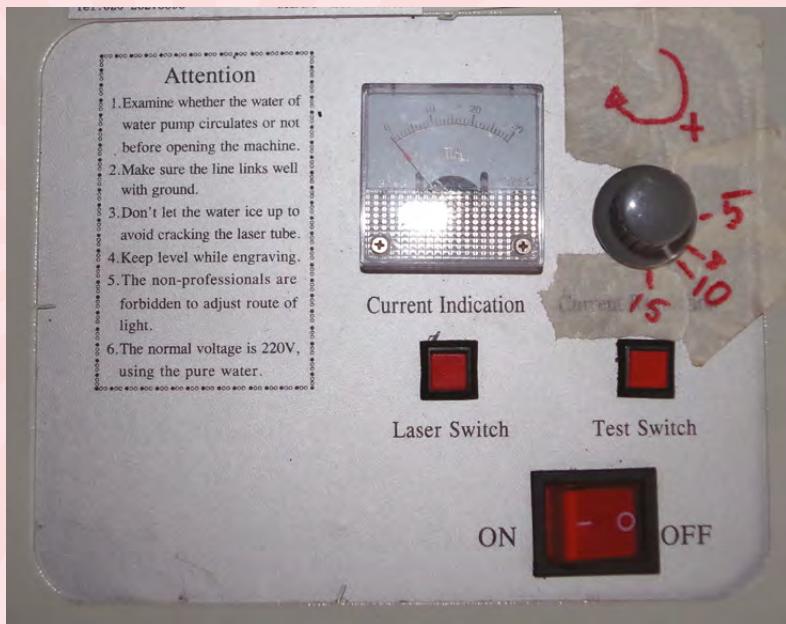
WARNING

The K40, like all laser cutters, should be handled with extreme care: accidents can and do happen. It is, after all, a high-voltage laser capable of cutting through plastic, wood, and humans. If in any doubt – don't! Always use safety equipment, particularly goggles, and have a fire extinguisher nearby. Remember, your safety is your responsibility.

Left

The 40W laser tube is water-cooled

BATTLE TESTED



Above Mark up the current regulation to help set the correct power

clean with a cotton bud and some alcohol. Cleaning the mirrors periodically also really helps to keep the laser working well.

MIRROR SHINE

You will need a PC to drive the K40 via USB and software is provided for Windows; using Linux or macOS is possible, but it's a more complicated install and there is no 'official' software support.

The software includes two packages, LaserDRW and CorelLASER. LaserDRW is a simple drawing package and driver; CorelLASER is a version of CorelDraw 12 (of questionable authenticity in my opinion) integrated with the LaserDRW driver.

You will need the supplied USB security dongle to use LaserDRW; it acts as a security device and it's not possible to operate the K40 without it. Getting a replacement could prove difficult so we recommend you keep it in a safe place.

If you have a very simple project, LaserDRW is really easy to use and any Microsoft Paint user will feel at home. For anything more complicated, you will need to use CorelLASER and, even though it's an older version, it's still a really powerful vector-drawing package and will be able to fulfil all but the most advanced users' needs.

It's also possible to use the popular open-source vector drawing package, Inkscape, with the use of a LaserDRW extension: hsmag.cc/klpnIM. Whatever the drawing package, when cutting or engraving you are ultimately greeted by the LaserDRW engraving manager. This is where the project is set up to be engraved and cut. Getting my project set up has usually involved trial and error before I got the settings just right, but as I gained experience this became easier.

There are five basic factors that need to be considered when setting up your project:

- 1. The layout** – does it need to be cut, engraved or both?
- 2. The material** – what is it made of and how thick is it?
- 3. The power** supplied to the laser
- 4. The speed** the laser will move
- 5. The number of passes** the laser will make

There is a really useful K40 wiki on reddit (hsmag.cc/UVwNDs), which includes a section on materials and gives some suggested settings for different types and thicknesses.

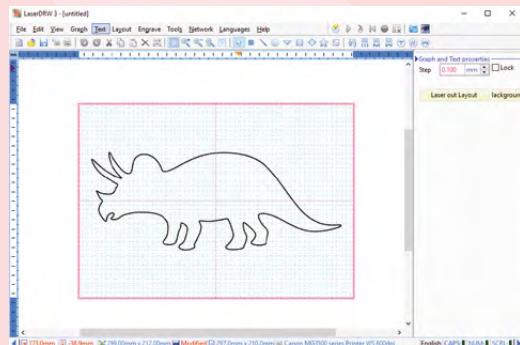
TEST TEST TEST

I had to select my controller from the Model drop-down list as it doesn't autodetect, and frustratingly there was no way of determining which model I had without trying each one in turn. Even more frustratingly, sometimes when I picked the wrong controller, it would 'work' but at the wrong speed, making it more difficult to find the right one.

LaserDRW allows you to engrave and cut but not both at the same time. You can however create multiple tasks which it will perform in sequence. In practice this means either creating separate images for the engrave and cut or using multiple layers, which you 'print' in turn, adding each task separately.

ENGRAVE FIRST, CUT SECOND

There's no doubt about it, the K40 is a 'no frills' device, but that doesn't mean it's restricted, and there are many DIY modifications you can make to improve its performance. I added a honeycomb base to improve heat transfer and stability of the



Above LaserDRW is a simple drawing package and software driver

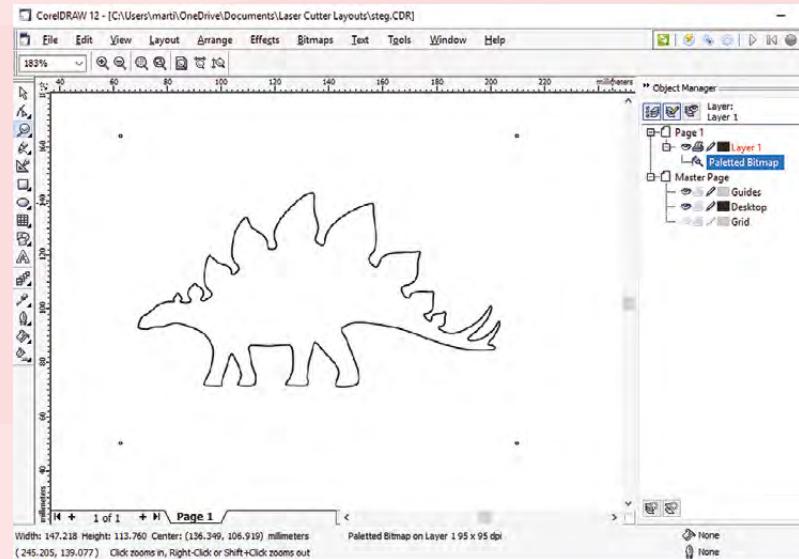
SETTINGS

Determining the settings to use for a specific material can be difficult. Resources online will give hints and tips, but the composition of the material will make a difference.

My starting point when testing a new material is:

- Engraving – 300 mm / second at 6 mA
- Cutting – 8 mm / second at 10 mA

When using a material for the first time, do a series of test cuts and engravings. I cut a small square and engrave the settings I used onto it – that way, when I find the perfect combination, I keep the square for future reference.



cutting base; this is a cheap and easy upgrade. The honeycomb mesh is inexpensive and can be found easily online.

Adding an 'air assist' is next on my list; this improves cutting performance by blowing air from a compressor through a nozzle mounted on the cutting head.

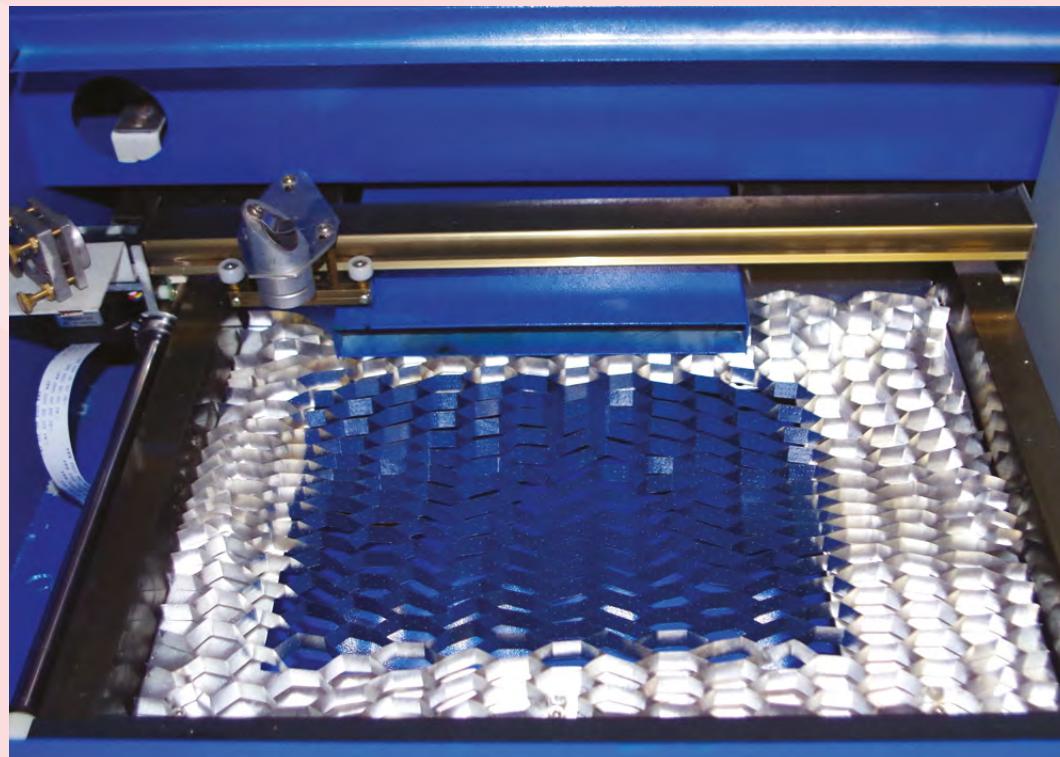
The controller can also be changed to deliver greater precision and allow the K40 to operate as a standalone machine without the need of a PC and the LaserDRW driver software.

You would be hard-pressed to find a cheaper laser cutter than the K40. It's cheap because of what has been left out, not necessarily what's been put in: it really is bare-bones. The versatility of the K40 is that you can modify it, adding in what the manufacturer didn't.

The K40 isn't exactly 'plug and play', but a community of makers has built up around it providing advice and support, which helps fill the gap. Even in its basic form, it's a capable machine and with the right operation you can get some good results. □

Above CoreLASER is the amalgamation of CorelDraw and the K40 driver

Left The honeycomb base provides a better base



VERDICT

Bare-bones and tricky to use but, given some attention, it can produce good work.

6/10

AIY Vision Kit

\$45 | aiyprojects.withgoogle.com

By Lucy Hattersley  @lucyhattersley

After creating one of the most desirable maker projects around (the Google Home-esque Voice Kit), Google is back with this AIY Vision Kit. This time, the premise is to build a Google Clips-like device capable of performing image recognition. Point the Vision Kit at a dog, cat, or computer mouse and it'll be able to tell you what it's looking at. It can also detect human faces and even emotional signifiers like smiling or frowning.

The kit comes with most of the parts you need, including the cardboard frame, lens, privacy LED (so people can tell when it's filming), RGB LED, and piezo buzzer. The stand-out feature is the VisionBonnet board, packing a Movidius MA2450 Vision Processing Unit chip. Unlike the Voice Kit, the Vision Kit does not require a connection to Google's Cloud Service to perform AI. Instead, it's all crunched locally on the VisionBonnet.

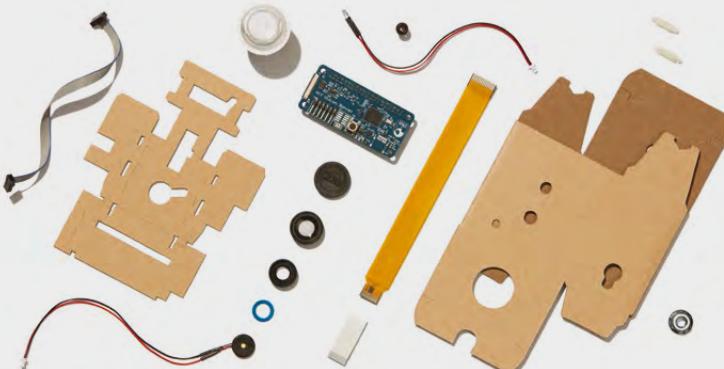
The only other way we know to attach an MA2450 to a Raspberry Pi is via Movidius's Neural Compute Stick USB dongle, costing £69 (developer.movidius.com). Once you've set up the project, you can load object recognition models and use them as part of your own builds. Google is providing three models to begin with: a cat, dog, human detector; a face detector with emotion inference (happy or sad, for example), and an object detector that can identify and label 1001 common items. Down the line, we expect more models for people to



load onto their kits. Google is also planning a companion Android app, so you can view what is being captured without needing an HDMI monitor attached.

For its part, Google seems intent on kick-starting an AI community, whose members can build useful models for one another. In the interim, makers are well prepared for with a GPIO pin breakout on the VisionBonnet. This can be used to hook up the device to electronic components and put it at the heart of your own builds. You'll be able to use these to connect servos and DC motors, plus extra lights and buttons.

This is a fantastic kit that's ideal for learning the intricacies of artificial intelligence and adding machine vision to your projects. How useful it'll be in the long term depends on community support, but we have a lot of faith in the AIY community. □



Left While the cardboard housing makes it easy to get started, for a more permanent project you'll need a more robust case

VERDICT

The easiest way to add object recognition to your makes.

9/10

Raspberry Pi programmable power switch

\$17.99 | nanomesher.com

By Rob Zwetsloot

 @robzwetsloot

Compared to Arduino devices and other microcontrollers, the Raspberry Pi can be a bit power-hungry if you're trying to maximise your power efficiency in a project – especially one that may not have access to a plug socket – so you'll find yourselves going to quite some lengths to save on power usage. Turn off the Raspbian desktop. Disable the Bluetooth and wireless. Use as few USB devices as possible.

All these help, but even when the Raspberry Pi is off it's still being supplied a little power, and either way you'll have to somehow remove and reattach the power supply to get it to turn on again. This is where a good power switch comes in.

The Power Switch from Nanomesher fills this role. Switches for the Pi aren't new and there are various pre-existing solutions you can buy or make yourself. However, this switch takes the concept a step further by including an IR remote, and making the switch itself entirely hackable and reprogrammable.

The switch does this by making the ATtiny85 microcontroller on the board removable. You can then access it via an Arduino Uno to reprogram it. If you have an Uno, you just need to follow the instructions on the Nanomesher website on how to get it all connected up: hsmag.cc/VnFDPm. You can also get a Power Switch kit that includes an Uno and some jumper cables to wire it all up, although you'll have to pay a little more for that privilege.

The switch itself works by connecting to some of the GPIO pins on the Raspberry Pi, as well as acting as a bridge to the micro USB power port on the Pi. After installing a few scripts on your Pi, you can then use the remote to simply turn the Pi on and shut it down. The first one is very simple as it activates the bridge which sends power to the Pi, while shutting down is handled by sending a shutdown command,



and then simply waiting for it to complete before cutting the power.

This is of course all simple enough stuff – the hackable part comes in by letting you add extra functions to the remote, such as tweaking timings, sensing when the Raspberry Pi is on or off, and a hard shutdown where, as the name suggests, it will just cut the power altogether.

It all works very well. Setting it all up is pretty quick and, although the remote controller does feel a little flimsy, it does the job. You could probably replace it with something else if you wished. It can be a bit of a faff to reprogram it off the Pi; however it does work as advertised, and customising the included Sketch is easy enough.

Does it achieve the central goal of saving on power, though? Probably not, at least not as much as using a less power-hungry device, but it does at least allow you to manage power to the Raspberry Pi much more easily, and that's good enough for us. □

Above The basic kit doesn't come with much, but it's just enough to connect it to your Pi

VERDICT

A great idea executed fairly well, although being fully programmable by the Raspberry Pi it connects to would be preferred.

8/10

Adafruit AMG8833 Grid-EYE Breakout

\$39.95 | adafruit.com

By Martin O'Hanlon

 @martinohanlon

The **AMG8833 Grid-EYE Breakout**, from Adafruit, is an infrared thermal camera, packaged on a really convenient breadboard-compatible board. We have to admit, it has an impressive set of features for such a small board:

- ◆ A 8x8 pixel sensor, capable of measuring temperatures ranging 0°C – 80°C (32°F – 176°F)
- ◆ An internal thermistor for measuring the ambient temperature
- ◆ An interrupt that can trigger when a temperature change is detected
- ◆ In-built voltage regulator and level shifter for 3.3V and 5V, making it compatible with Arduino and Raspberry Pi
- ◆ Communication via I²C
- ◆ The Grid-EYE comes unsoldered, giving you the option of soldering the supplied pins or soldering connections directly to the breakout

GET THEM WHILE THEY'RE HOT

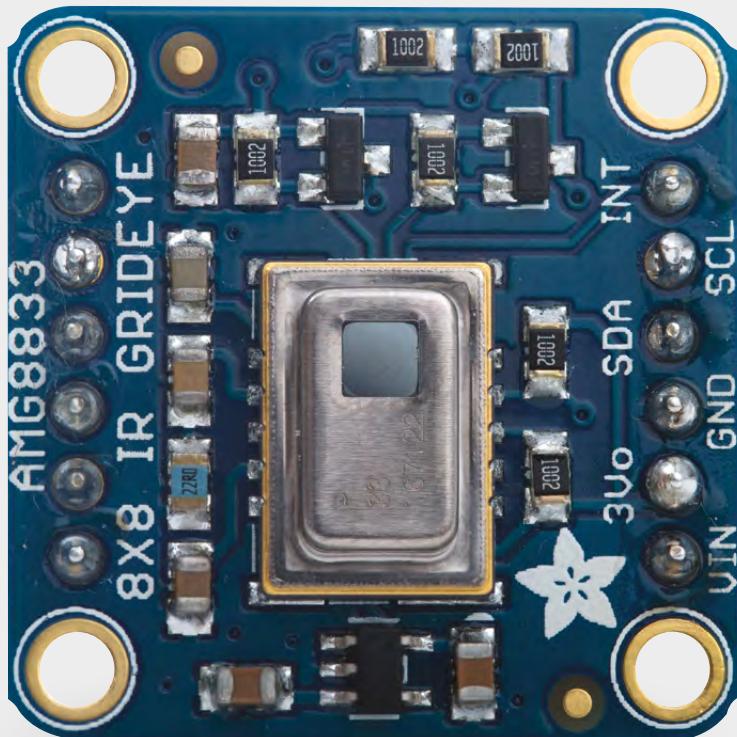
Once soldered, it needs connecting to power, ground, and I²C pins (SDA and SCL); there is also an optional interrupt pin. Power can be either 3.3V or 5V; the breakout board will automatically level-shift the output voltage based on the input.

Mounting the Grid-EYE onto a breadboard and using an Adafruit T-cobbler provided a solid platform for testing the device, and it was connected up in just a few minutes.

Adafruit provides installation instructions, software libraries, and examples for Arduino and Raspberry Pi which are thorough and well written, including a step-by-step walkthrough of the setup, software installation, and code required to create a 'mini thermal camera' using the Grid-EYE and an external LCD screen.

It's worth noting that, when using a Raspberry Pi, you don't need an external LCD screen – the image will also appear on the standard desktop.

Below ♦
The compact form will fit into most projects

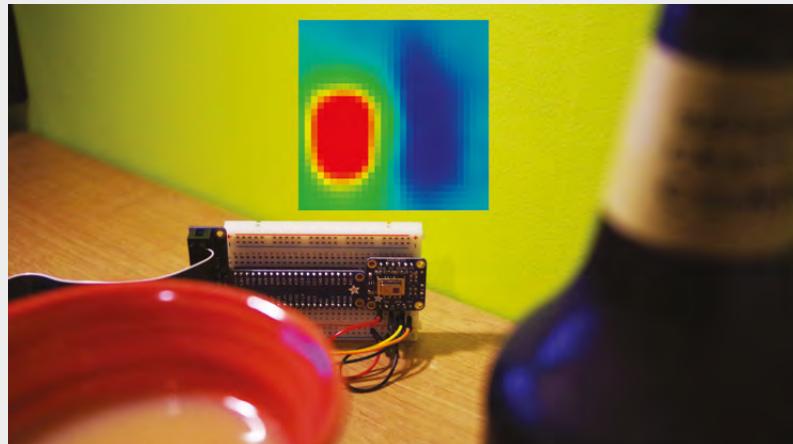


The 8x8 sensor returns a list of 64 temperature values (in °C) from the ‘pixels’. The thermal camera example converts these values into colours ranging from blue to red, and displays them on the screen. The Raspberry Pi example uses ‘bicubic interpolation’ to turn 64 pixels into a much bigger screen, making it appear to be higher resolution than it actually is.

The sensor is able to cope with differences in temperature extremely well. When presented with a steaming mug of tea and a bottle of cold beer, it could differentiate between the two items and show a clear boundary between them.

The supplied software libraries reduce the heavy lifting required to read data from the sensor, but the APIs lack documentation, so you will find yourself looking through the examples and, most likely, the code to understand more advanced features. That said, we were able to adapt the thermal camera example to make use of Raspberry Pi’s Sense HAT as a display with the information available.

Using the interrupt features of the board, you can use it to detect a change in temperature. This is particularly useful when you are trying to determine when a ‘living thing’ is present; such as detecting when someone enters a scene, or creating a wildlife camera.

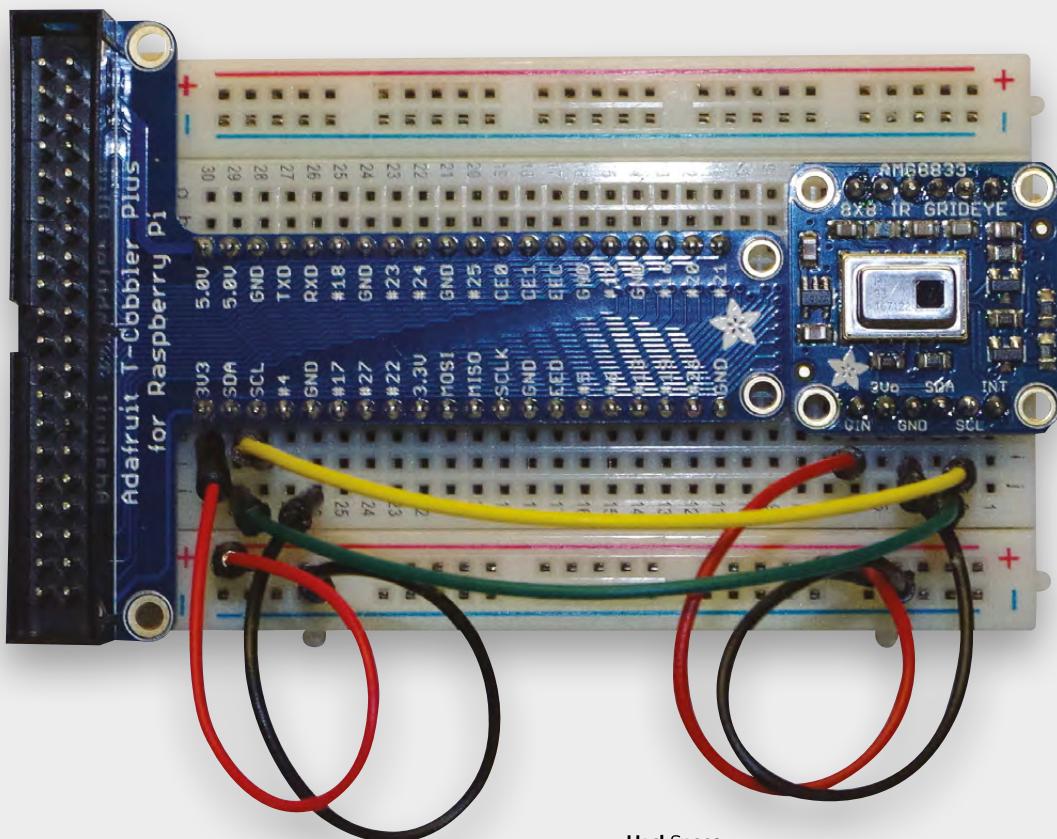


Adafruit claims the Grid-EYE can detect a human from a distance of up to 7 metres (23 feet); this seems accurate, but our experience showed that results were dependent on the environment, with a warmer ambient temperature resulting in less accurate results.

The interrupt (INT) pin needs to be connected to a digital input (GPIO) pin on your Arduino / Raspberry Pi, and enabled using the software libraries, passing a temperature range on which the sensor should trigger, for example when a temperature of 20°C – 30°C is detected. When triggered, the interrupt pin will be set low, whereby you can read which pixels on the sensor triggered the interrupt, enabling you to determine where in the scene the temperature change occurred. □

Above Hot tea and cold beer aren’t often enjoyed together

Below The Grid-EYE is simple to connect to either a Raspberry Pi or Arduino



VERDICT

Adafruit’s AMG8833 Grid-EYE Breakout packs a lot of functionality into a very compact package, and creates many options for hacking and making.

8/10

Bearables Badge Kit

£12.50 | pimoroni.com

By Phil King

 @philking68

At first glance, you might think Pimoroni's new range of woodland-themed, multicoloured LED badges are just a cute wearable novelty item. You'd be wrong. Not only can they be connected to and triggered by analogue sensors, using conductive thread, but they're designed to be hackable.

While the Bearables components are available individually, there are also two full kits, each featuring a badge, sensor, a generous 3 metres of conductive thread, and even some sewing needles. The only difference is that the Bear badge kit includes a tilt-switch motion sensor, while the Fox kit comes with a light sensor. Both badges and sensors have two metal hooks on the rear, around which you repeatedly loop the conductive metal thread used to connect them and sew them onto an item of clothing or a bag – you can't put them in the wash, though, as they don't like water.

Powered by a CR2032 coin cell that provides three to four days of fully lit use, the badges can be controlled manually using a small button on the edge to switch between various light patterns on the twelve coloured LEDs (blue, green, yellow, orange, red, and

pink). Holding the button down for a few seconds puts the badge to sleep or, if a sensor is connected, selects trigger mode – in which case, depending on the sensor type, either motion or a lack of light will then activate the badge LEDs.

PROGRAMMING THE BADGE

With no obvious connection ports, you may be wondering how you go about hacking the badge. The five tiny metal pads on the rear provide the means – via the I²C bus – to program the badge's PIC16F1503 chip using a suitable board, such as a Raspberry Pi or micro:bit. To help you out, Pimoroni has created a Bearables Python library which allows you to switch the LED pattern, control individual LEDs to code your own patterns, use the button to trigger events, and read the voltage across the sewable hooks.

Those badge hooks can be used to read raw ADC values (0–255) from pretty much any analogue sensor, so you could attach different ones such as a sound sensor. You could also use the hooks to read pins that are pulled high or low on attached microprocessors, opening up all sorts of possibilities for using external data sources to trigger badge patterns. □



Above Each badge features twelve LEDs of various single colours that can be lit in twelve preset patterns or your own custom creations

Left The badge and sensor may be sewn onto clothing using the supplied conductive thread



VERDICT

Great value for money, these cute LED badges are very hackable, offering a host of possibilities.

9/10

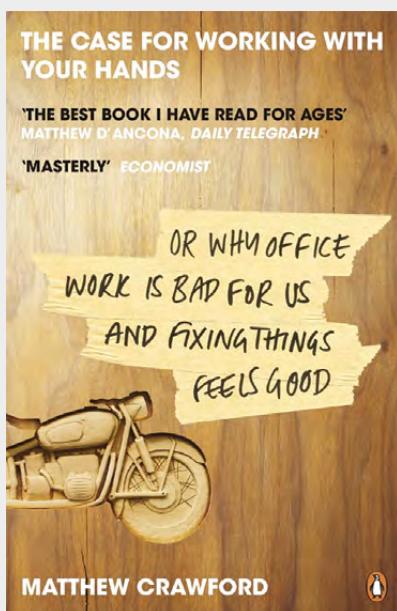
The case for working with your hands

Matthew Crawford ◇ £9.99 | penguin.co.uk

By Richard Smedley

Forget standing desks ("sitting all day is killing us!", they cry), it's the contradiction between the idea of knowledge work and the reality of a dumbed down environment of 'expert systems' which is making us feel bad.

One in which we follow scripts and processes set out by knowledge engineers – mirroring the time and motion studies which took decisions away from skilled craftspeople, and gave us back abstract parts in a process of production.



Matthew Crawford speaks up for a blue collar alternative of real creative work, posited against the anodyne ersatz creativity of consumer choice and individuality. Or, rather, not creative work, but the less glamorous virtue of attentiveness cultivated by the mechanical arts.

What's important is the discipline inherent in working with an external reality. Something which forces us to abandon an egocentric view of the world as revolving around oneself, and really pay focused attention to the myriad possibilities in, say, a motorcycle that won't start. The reality of the immediate and the present, over the abstract.

There are echoes of Robert Pirsig's *Zen and the Art of Motorcycle Maintenance*. But Pirsig was concerned with the value of quality in and of itself. Crawford's work is a manifesto (albeit one encumbered in academic language) for the unengaged young person to bypass college and take up a trade, discovering a role that is not just better-paying than much knowledge work, but rewarding in a fuller sense.

For much of the book, half of the population is missing: 'The case for working with your hands' describes a brotherhood of craftsmen; all the tradespeople mentioned – recalled or putative – are male, and women only enter in the roles of artist or manager. This makes Crawford often read like a Victorian anthropologist (particularly given the sociological bent of the language), and the world of vintage motorcycle repair shops and counterculture figures that the author inhabits seems like an anachronistic throwback.

This, sadly, is also a part of a long tradition of American literature on self-reliance and individual agency. But, despite its description of an incomplete world, this personal account of 'honest trade' offers valid insights into a deeper engagement with work, and can inspire the reader to rethink their whole world view. □

VERDICT

A problematic but essential overturning of our assumptions about the education treadmill, work, and fulfilment.

7 /10

TIP US OFF TO A
INTERESTING PROJECT

BE HACKSPACE
OF THE MONTH

REACH OUT
TO FELLOW
HACKERS

SHARE YOUR
THOUGHTS
ON THE
LETTERS
PAGE

WE NEED YOU



If you love to hack and make stuff, why not get paid to share this with the world? We need passionate people to write features, tutorials, and reviews.

GET IN TOUCH

We want your tips, comments, and questions!

✉ hsmag.cc/hello

🐦 @HackSpaceMag

HackSpace



TECHNOLOGY IN YOUR HANDS

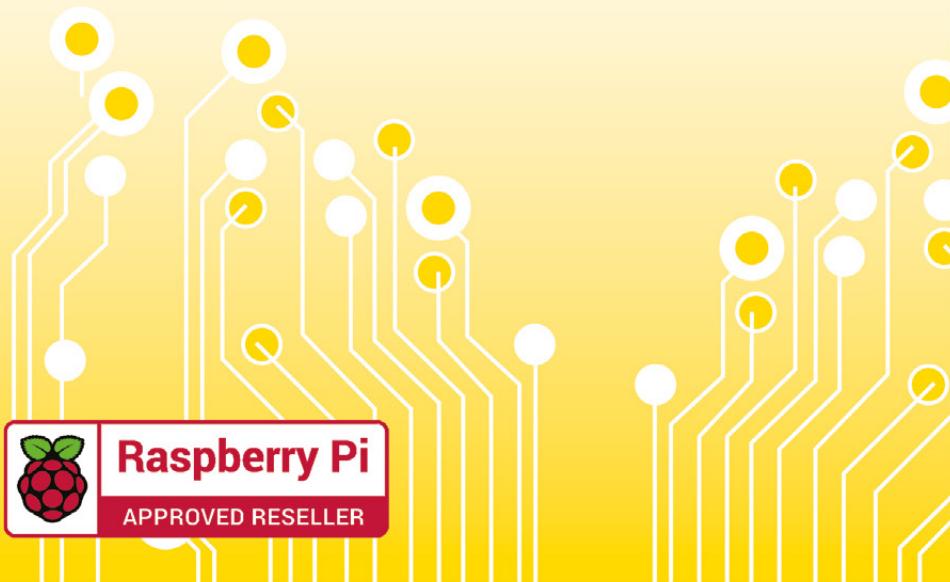
— hsmag.cc —

CanaKit Raspberry Pi 3 Complete Starter Kit



Kit Includes:

- ✓ *Raspberry Pi 3 Model B
Quad-Core 1.2 GHz 1 GB RAM*
- ✓ *On-board WiFi and Bluetooth*
- ✓ *32 GB MicroSD Card (Class 10)*
- ✓ *CanaKit 2.5A Power Supply*
- ✓ *High Quality Case*
- ✓ *HDMI Cable with CEC support*
- ✓ *MicroUSB Reader*
- ✓ *Set of Heat Sinks*
- ✓ *GPIO Quick-Reference Card*
- ✓ *CanaKit Quick-Start Guide*



Available for worldwide shipping at:

WWW.CANAKIT.COM

\$74^{.99} **£59^{.99}** **€64^{.99}**
US DOLLARS EXCLUDING VAT EXCLUDING VAT

Raspberry Pi is a registered trademark of the Raspberry Pi Foundation.
CanaKit is a registered trademark of Cana Kit Corporation.