NEW HACK | MAKE | BUILD | CREATE

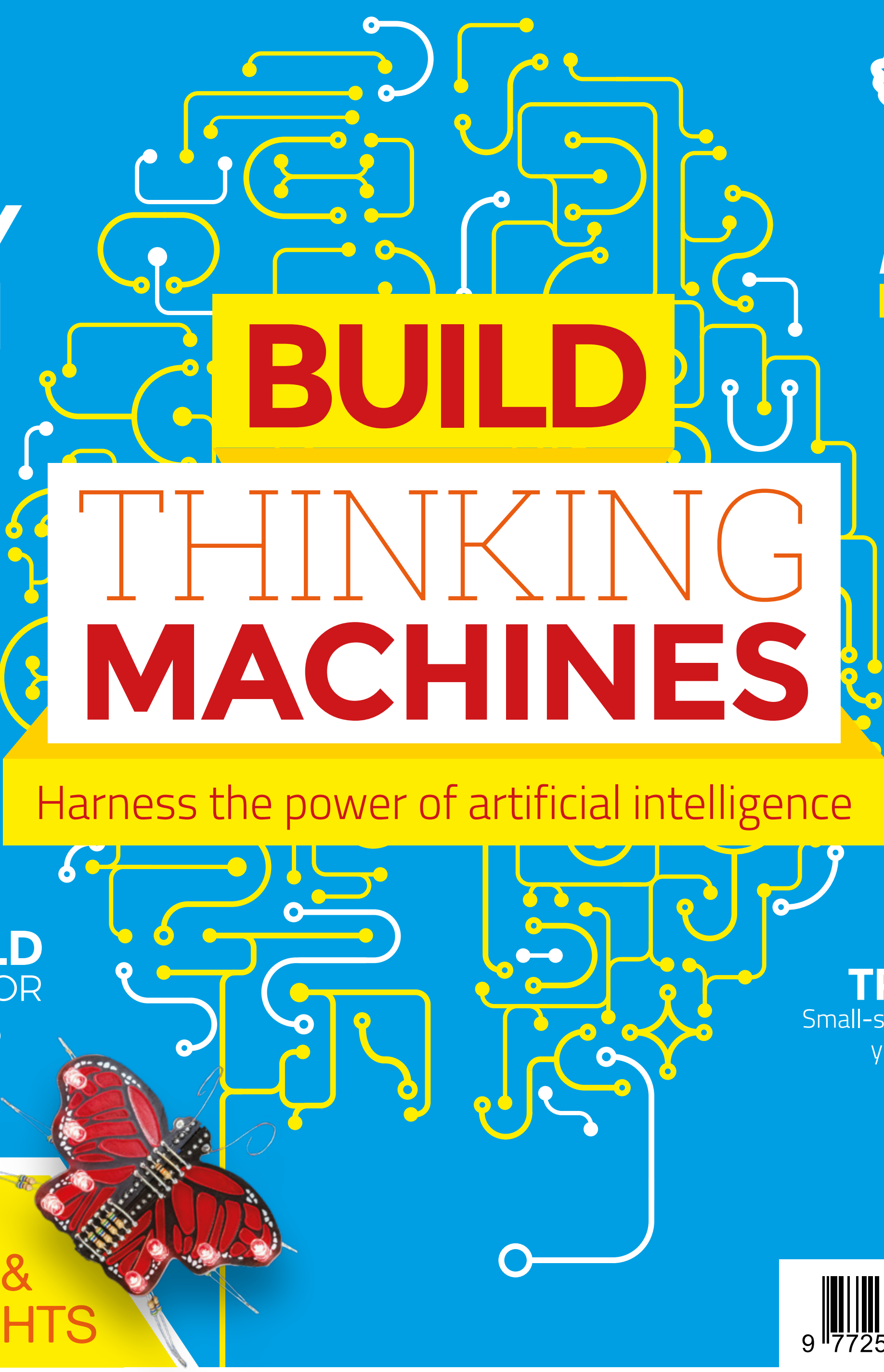
HackSpace

TECHNOLOGY IN YOUR HANDS

hsmag.cc | December 2017 | Issue #01

BECKY STERN

How to get paid for making fun things



BUILD THINKING MACHINES

Harness the power of artificial intelligence

WORLD'S *FASTEST* **RASPBERRY PI**

SMOKE YOUR OWN **BACON***

*or anything else. But mostly bacon

MAKE BEAUTIFUL **MUSIC**

Unleash your inner Robert Moog

HANDHELD CONSOLES FOR **HACKERS**

BUILD A **TREBUCHET!**

Small-scale warfare to satisfy your Napoleon complex

BOLDPORT: BUGS, BITS & BLINKENLIGHTS

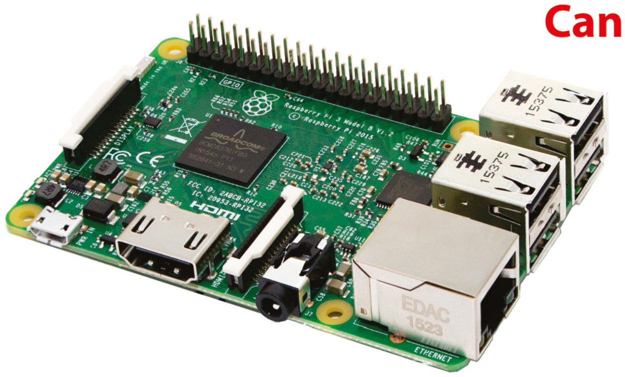
Dec.2017 Issue #01 £6

TRANSISTORS **3D PRINTING** DUCT TAPE **CHANGING THE WORLD**

CanaKit™

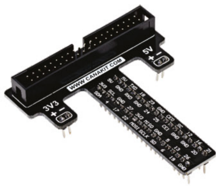
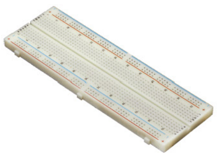
CanaKit Raspberry Pi 3 Ultimate Starter Kit

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 APPROVED RESELLER

Raspberry Pi Zero W

Available at CanaKit!

Wi Fi™ Bluetooth®

Electronic Kits • Electronic Parts • Raspberry Pi • Arduino

Welcome to HackSpace magazine

Hackspaces (often known as hackerspaces outside the UK) are community-run groups that enable people to share access to tools, socialise with like-minded people, and collaborate on projects. They exist all over the world – the chances are that there's one near you that you can get involved with, to develop your own projects and share your experiences with other members.

We want to raise the profile of the hackspace movement. We want to highlight the work that's being done, and we want to get involved ourselves. Most important of all, we want to provide an inclusive forum for the community, where everyone can bring their work together to learn and be inspired by the work of others. That's why hackspaces exist, and that's the entire raison d'être of HackSpace magazine.

We want to raise the profile of the **hackspace movement**

As well as endeavouring to support community-run spaces around the world, any money made by HackSpace magazine goes to the Raspberry Pi Foundation. This UK-based charity aims to put computing in the hands of people around the world through developing low-cost hardware and working with teachers and students.

To ensure that everyone who wants to build things can get access, we're making every issue of HackSpace magazine available as a free download from our website, **hsmag.cc**, on the same day they go on sale.

We're looking forward to working with everyone in the hacking and making community to bring skills, knowledge, and inspiration to as many people as possible. Get in touch with us at **hackspace@raspberrypi.org** if you want to help make this happen.

BEN EVERARD
Editor ben.everard@raspberrypi.org

GET IN TOUCH
hackspace@raspberrypi.org
hackspacemag
hackspacemag

ONLINE
hsmag.cc

Available on the App Store
GET IT ON Google Play

EDITORIAL
Editor
Ben Everard
ben.everard@raspberrypi.org

Features Editor
Andrew Gregory
andrew.gregory@raspberrypi.org

Sub Editors
Phil King, Jem Roberts

DESIGN
Critical Media
criticalmedia.co.uk

Head of Design
Dougal Matthews

Designers
Lee Allen, Mike Kay

Photography
Brian O'Halloran

Illustrations
Sam Alder

CONTRIBUTORS
Lucy Rogers, Andrew Huang, Mayank Sharma, Jenny List, Goli Mohammed, Jenny Smith, Cameron Coward, John Wargo, Eric Coates, Martin O'Hanlon, Andy Clarke, Will Kalif, Gareth Halfacree, Les Pounder, David Crookes, Graham Morrison

PUBLISHING
Publishing Director:
Russell Barnes
russell@raspberrypi.org

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

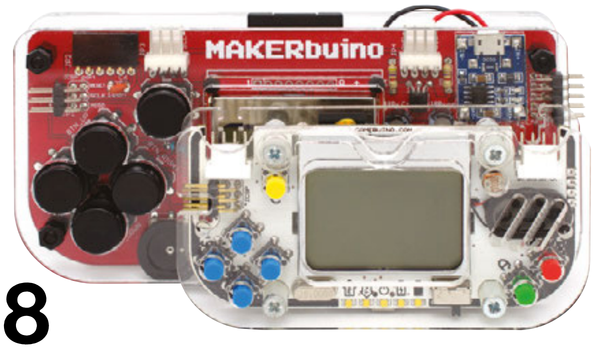
SUBSCRIPTIONS
Select Publisher Services Ltd, PO Box 6337, BH1 9EH
+44 (0)1202 586 848

Mann Enterprises Ltd, Unit E, Brocks Business Centre, CB9 8QP
hsmag.cc/subscribe

Contents

58

54


118

64

22

20

116

Utterly hipster bicycle speed and cadence gauges

By Grzegorz Hołdys   InsightMachinesLab

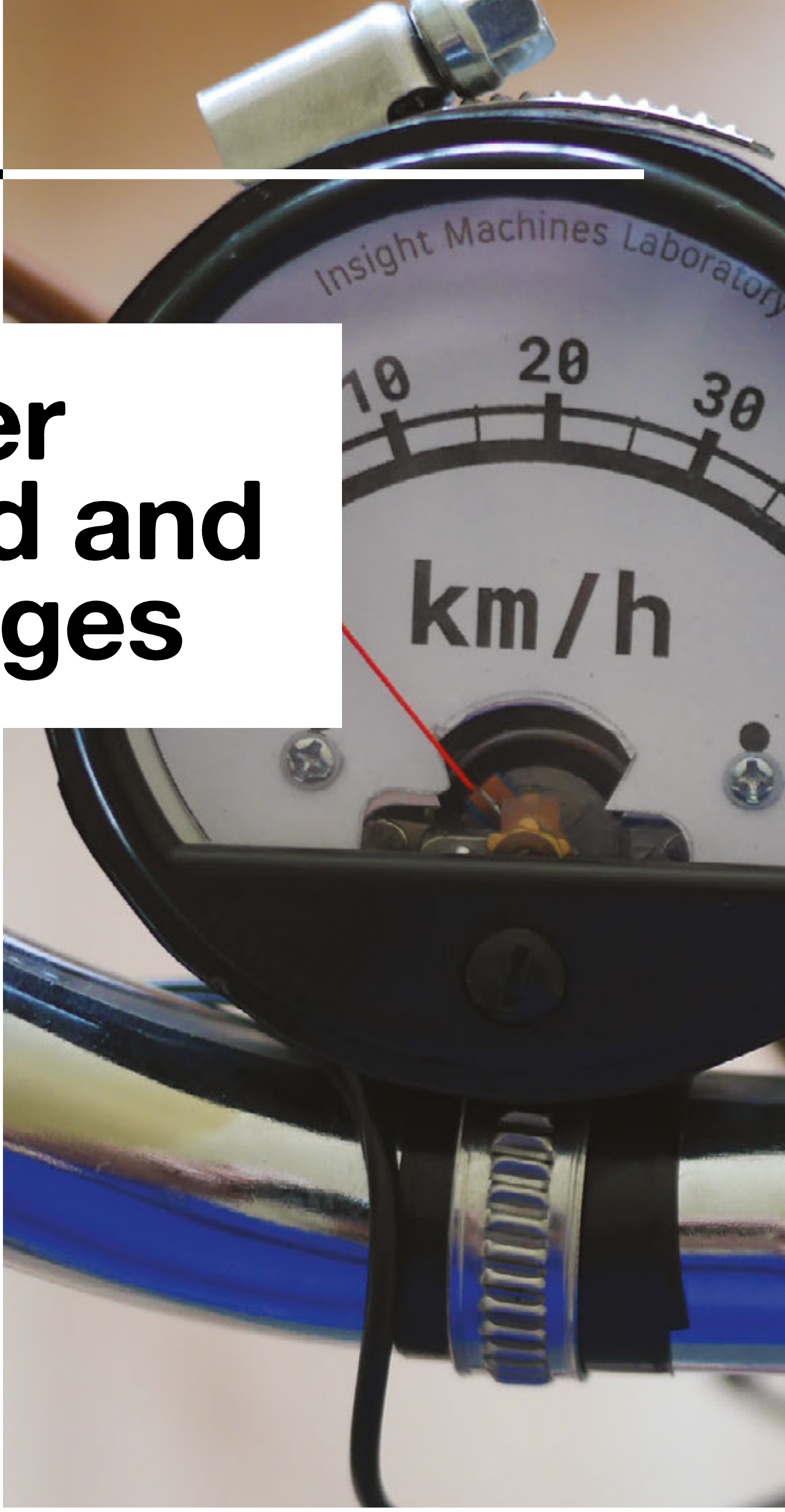
The idea to create analogue gauges for a bicycle appeared in my head when I came across an article on a Polish hackers' site that described how to build a desk clock from analogue voltage meters (hsmag.cc/JBGDAz). I always liked analogue gauges and when I saw how easy it is to convert voltage meters into pretty-much-anything meters, I decided to give it a try.

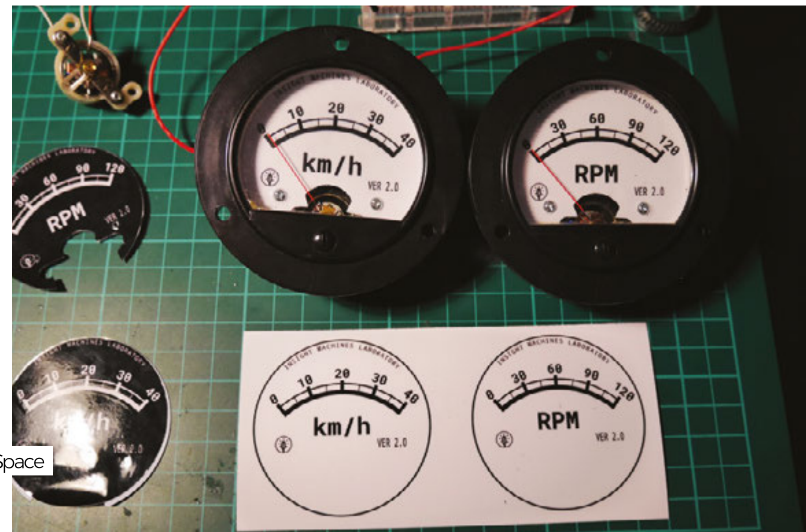
I have an old Peugeot city bike that looks vintage (to some extent), so I figured that it might even look quite cool with analogue speed and cadence meters. I wanted to have two gauges, just like on a motor bike. I had some experience with ATtiny85 and Arduino, so it seemed like a very simple project. The breadboard prototype was trivial. The difficult part turned out to be fitting everything together in the voltage meter's enclosure and mounting the devices on the bicycle's handlebars.

Initially, to mount the gauges, I decided to use worm gear clamps. They held the gauges quite well but were difficult to use and often scratched the handlebars. The newest version (not yet published or even photographed) will use plastic bands taken from CatEye's bicycle headlamps – they are much easier to work with and look better.

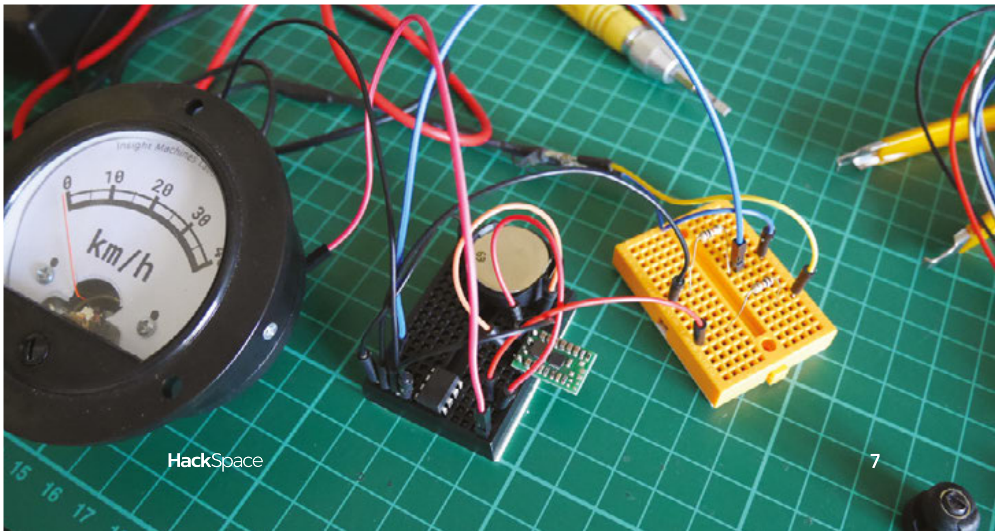
Another challenge was power consumption. The CR2032 battery is small, cheap and light, but it also has very limited capacity (~200 mAh). It quickly turned out that the ATtiny85 along with a switching voltage regulator consumed quite a lot of power. I did not want the meter to have an on/off switch so I needed a way to reduce power consumption, especially when the bicycle was stationary. Reducing power consumption of the ATtiny85 was easy – it runs at 1MHz and is put to sleep when the speed or cadence drops to zero. The voltage regulator that I decided to use has a shutdown pin so it can be deactivated. However, even after shutting down it still consumes power and, in fact, it consumes more than the ATtiny85. In the end, the gauges should work on a single CR2032 battery for about three to four months. That's not bad but not too good either.

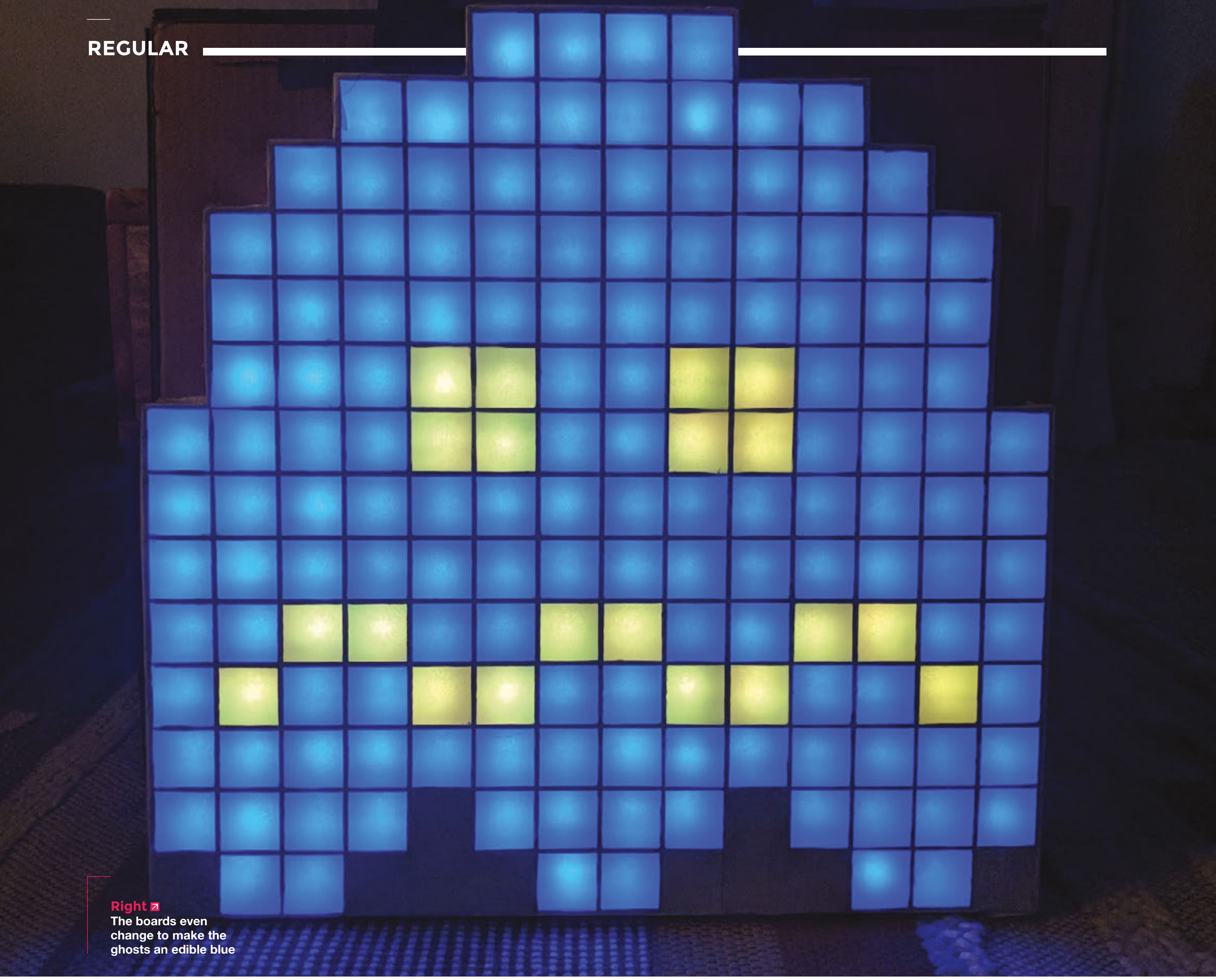
The gauges are currently being tested on a new bicycle – a large and heavy Schwinn cruiser that not only needed some analogue speed and cadence gauges but could also use some automatic lights and maybe a security system… More is definitely to come. □





Above
Now find the coolest bike you can find to give these gadgets the wheels they deserve





Right The boards even change to make the ghosts an edible blue

Pac-Man Halloween

By Ben Muller @pix3lot

I'm an architect living in Philadelphia with my family. I have always been a tinkerer, curious how things work and how to make them on my own. I mostly work on making things that both fill that curiosity and apply to my work as an architect, which is now focused on VR, AR, and writing custom tools for our 3D software.

Halloween is a good chance to try making something or using something that I haven't tried before. Last year we added LED strips as accents to silver clothes to look like some glowing retro silver space alien people. The costumes looked great and I liked the addition of lights to the costumes.

This year we knew we wanted to do something with LEDs again but we weren't sure what. While trying to come up with an idea, my daughter suggested Pac-Man characters. She had just been playing the Namco classic at a birthday party at a bowling alley. Once she suggested that, we ran with it.

Since we wanted it to be family costumes, it was perfect since there are enough characters and we could all be different. I would be Pac-Man, my wife Ms. Pac-Man, my daughter Pinky, and my son Blinky.

I wanted the costumes to look like they came right out of the arcade game. I wanted them to be animated and for each pixel to be legible.

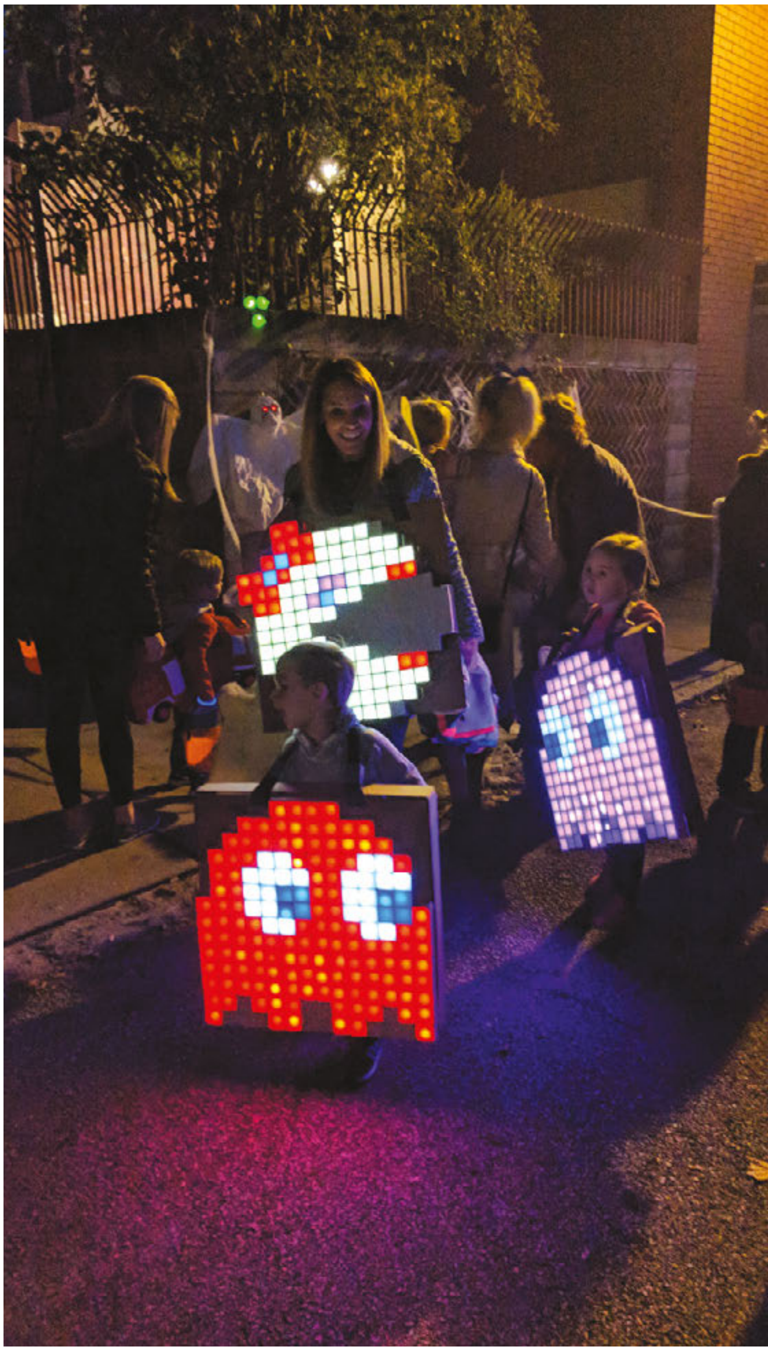
I designed the framework of the panel in the 3D modelling program Rhinoceros by McNeel. We have a laser cutter at my office and I used it to cut out the framework in ⅛" (3 mm) cardboard. The whole panel comprises the base, the slats, and the cover. The base and slats are made of the cardboard and designed so they all slide together, which minimises the need for gluing. The cover is made of drafting vellum, a type of paper.

The lights are strings of individually addressable 12 mm DC5V WS2811 LEDs that are more typically used in outdoor signage. They are inserted into 12 mm holes cut into the base. The slats create the pixels and the vellum diffuses the light to create each pixel.

The LEDs are driven by an Arduino UNO R3. The code is written, compiled, and uploaded to the board with the Arduino IDE. I used the FastLED library (**fastled.io**) to control the LEDs. It's a simple and easy-to-learn library for Arduino that is specifically made for programming individually addressable LEDs.

The LEDs and Arduino are powered by a DC 5 V battery pack (a portable phone charger) and a USB to DC adapter. I did tests on two chargers I had lying around. I got about 12h 40m out of 7800 mAh and 3h 40m out of the 2200 mAh capacity. For most nights out you really only need the small chargers, which is good news.

The board and chargers are attached to the panel with Velcro and a strap is added to make it all portable. □



Stranger Things lights

By Liz

@BlitzCityDIY

I made the Stranger Things lights project to celebrate the *Stranger Things* season 2 premiere, for which I hosted a viewing party. The project is in two parts and is coded using Adafruit's Circuit Python.

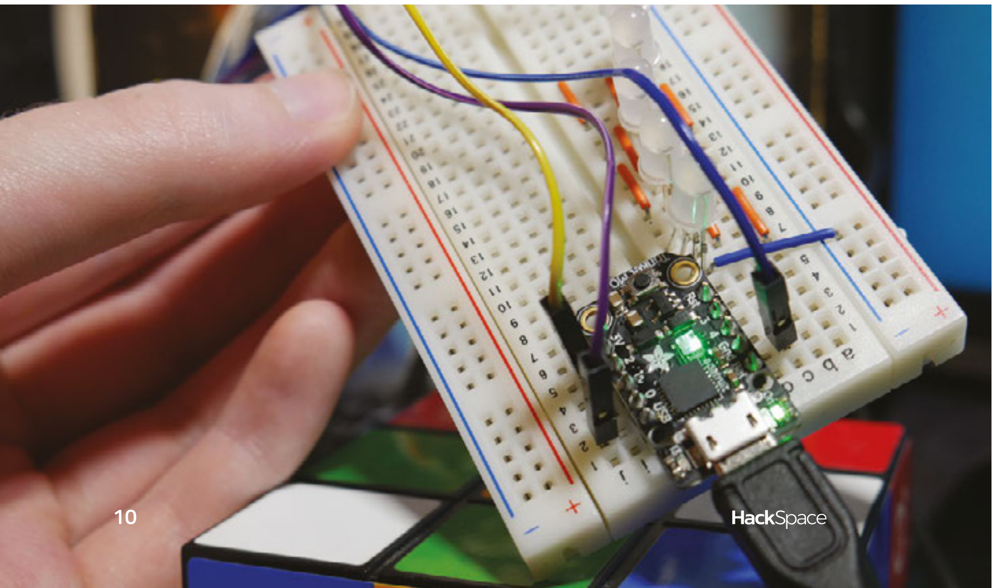
The first part is a recreation of the alphabet wall that the character Joyce Byers uses to communicate with her son Will while he's trapped in the Upside Down. I have a cycle of light routines that loop continuously running on an Adafruit Trinket m0 board to recreate messages from the show along with some fun light effects.

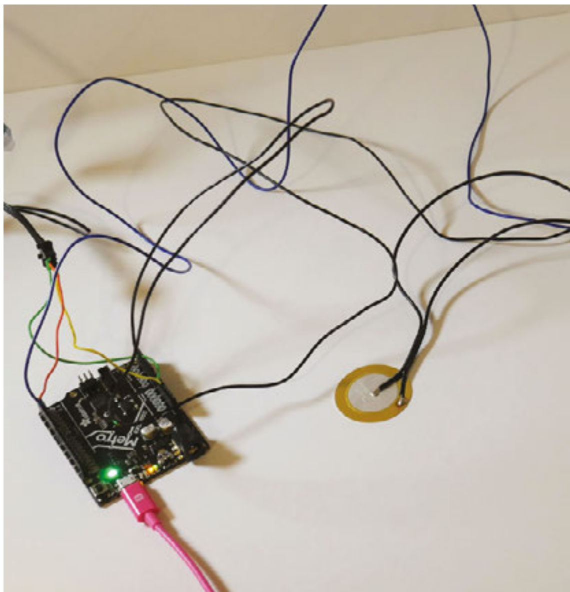
The second part is a recreation of the lights that Joyce strings throughout her home to also communicate with the Upside Down. I have 200 lights strung throughout my apartment that are connected to an Adafruit Metro m0 board along with a cluster of piezo sensors. I soldered the piezos together to basically make one giant sensor that is hidden under a rug. When the rug is stepped on, it triggers the lights to light up one by one followed by some startling (and strange) light effects that last for about two minutes.

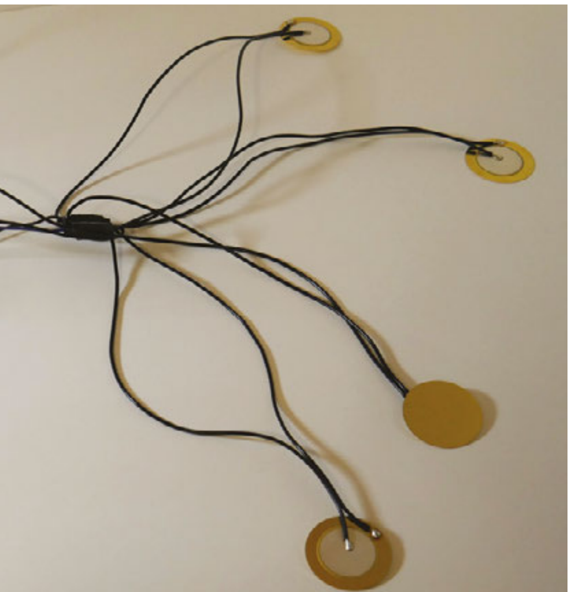
It was a really fun project to work on and if you're interested in making it yourself, I have a write-up on **Hackster.io** that goes into a bit more detail and includes the code files.

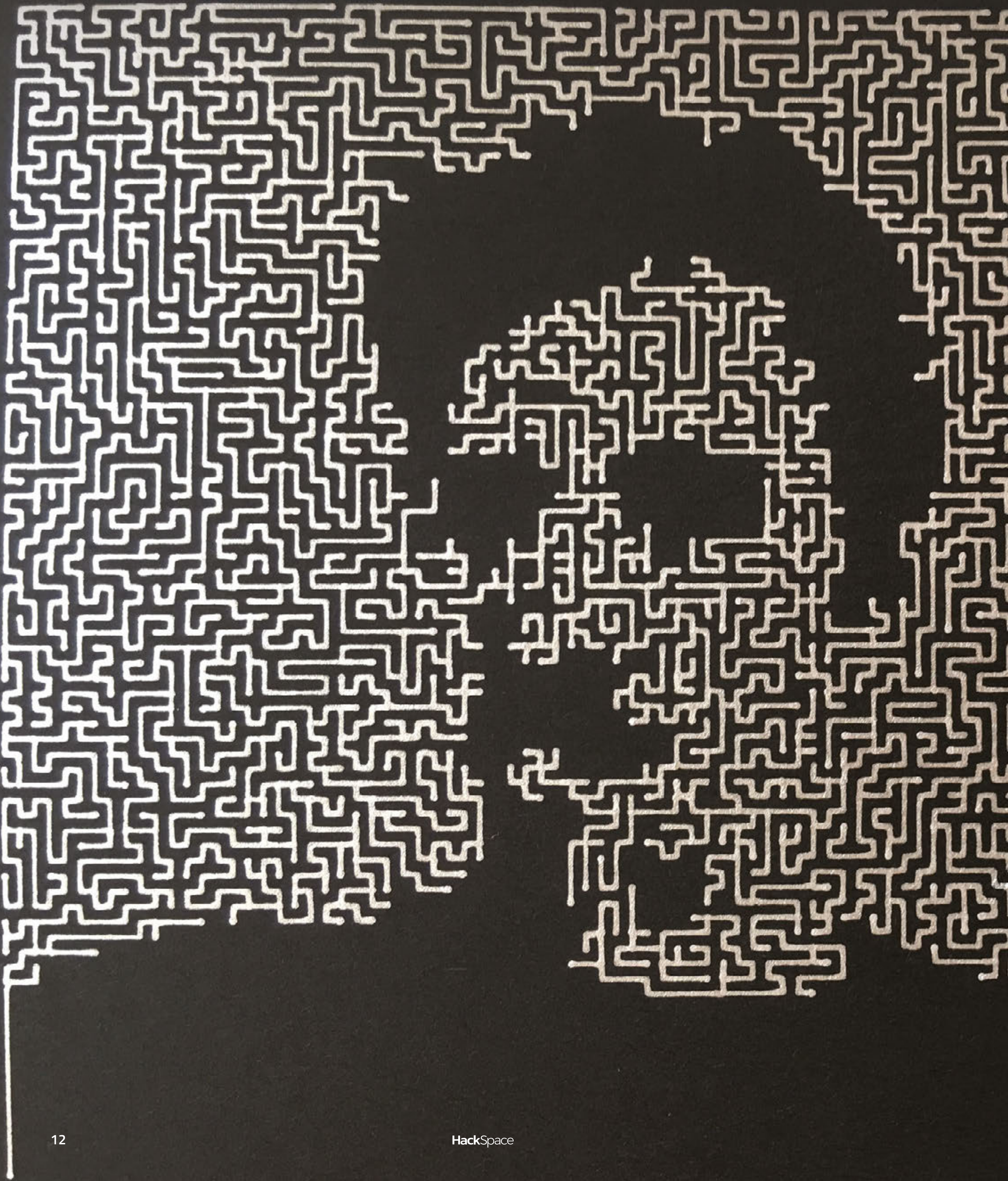
I'm a female DIY-er on a quest to gather and share knowledge. My handle, Blitz City DIY, is a reference to two of my favourite bands: the Ramones and Yeah Yeah Yeahs. I love the open-source community and how it empowers people, from beginner to expert, to learn and create. When I'm not working on projects, I can be found hanging out with my two adopted cats named Winnie and Harriet. □

Right
Prepare your home for the coming of the Demogorgon









Plotter art

By Seph Gentle @josephgentle

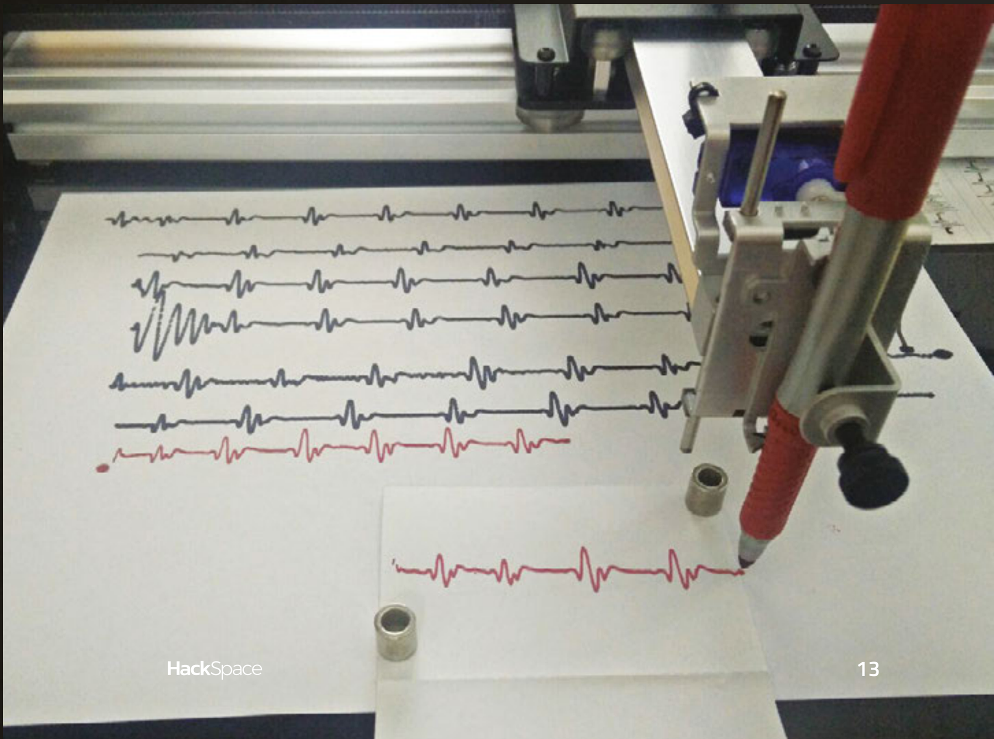
I got in to plotter art when a friend in the US got an AxiDraw plotter and I wanted one. I was worried it would gather dust so I made a deal with myself that I was only allowed an AxiDraw if I made at least ten pieces of artwork for it first. I made ten and didn't stop.

It turns out making art is great fun! I can't draw, but I can do maths and write code, so this is the perfect medium for me. Watching the plotter draw some cool maths I wrote is hypnotic.

The artworks are made using custom JavaScript. For each piece, I just write a simple function that returns a list of paths (lists of points). It turns out you can make lots of cool shapes with tiny functions – [my] Joy Division-inspired piece is less than 30 lines of code. I hand the paths to some Python code the AxiDraw team wrote to generate a list of plotter instructions. It's all open source, though – I'm tempted to rewrite everything and talk to the plotter directly.

The most enjoyable part of the process is usually showing off what I made. And I love the back and forth on Twitter – lots of my favourite work comes from mushing together other people's ideas and playing around.

Its also great coaching my friends through making things. Apparently we're all just a little mathematics away from art: no drawing required!



Right
This is Seph's heartbeat, on a name badge

Objet 3'd art

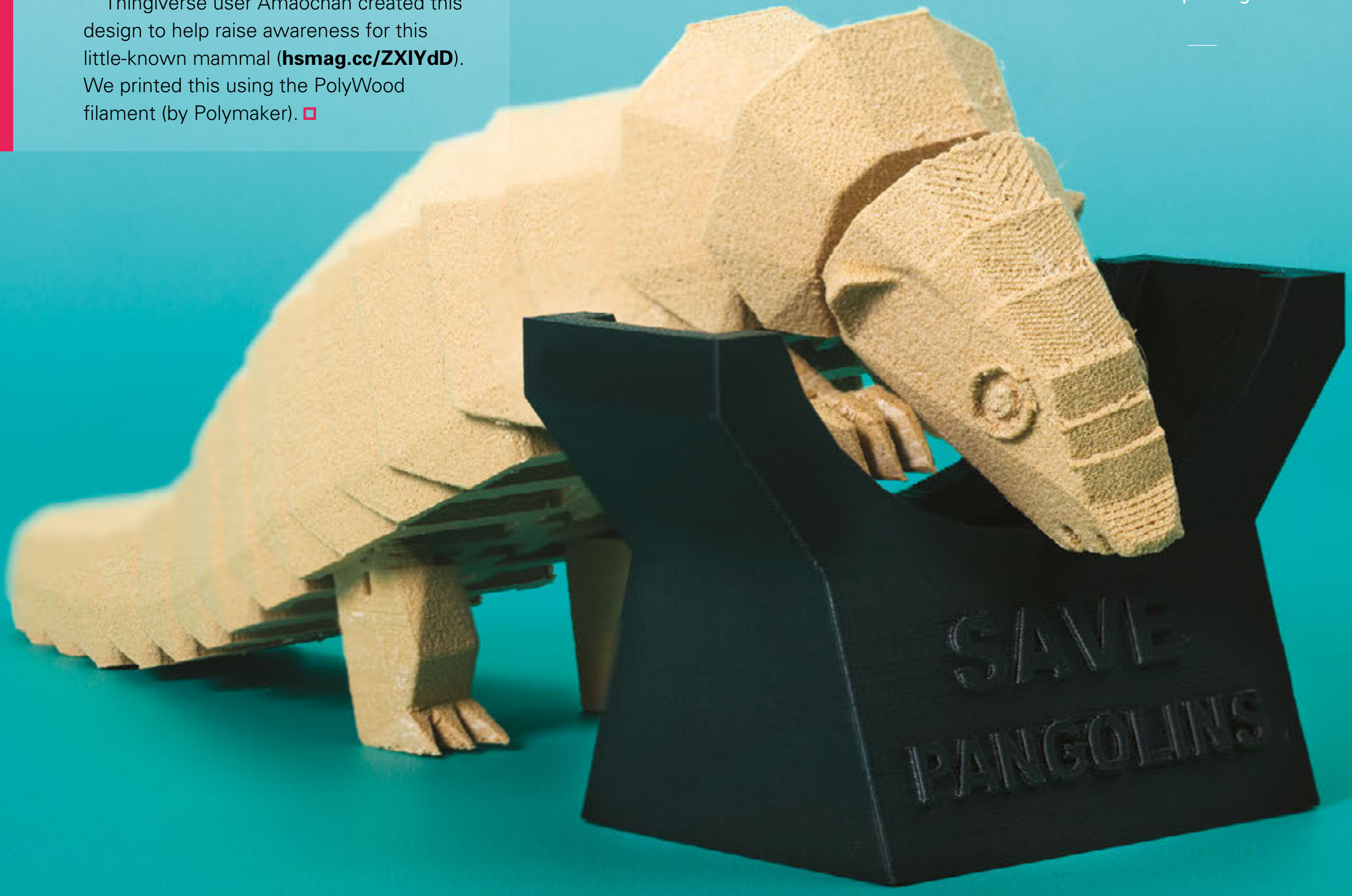
3D printed artwork to bring more beauty into your life

Pangolins are the only mammals with keratin scales to protect them from attackers. Of the eight species found across Asia and Africa, all are classified as Threatened With Extinction by the International Union for the Conservation of Nature, and two are classed as Critically Endangered.

Thingiverse user Amaochan created this design to help raise awareness for this little-known mammal (**hsmag.cc/ZXIYdD**). We printed this using the PolyWood filament (by Polymaker).

3D PRINTING
Supplied by 3D HUBS

Head to 3dhubs.com
for local 3D printing services



The interplay of transparent filament and light can lead to some great effects. Here we've printed a biologically inspired lamp holder using Green Transparent ColorFab PLA/PHA. You can create your own lamp using the files shared by Thingiverse user Nervoussystem at **hsmag.cc/wKANqO**.

For a similar, but personalised, lamp, you can tweak this design using the CellCycle web-based tool at **n-e-r-v-o-u-s.com/cellCycle**. This lets you alter the parameters that are used to generate the cell structure in the ring. □

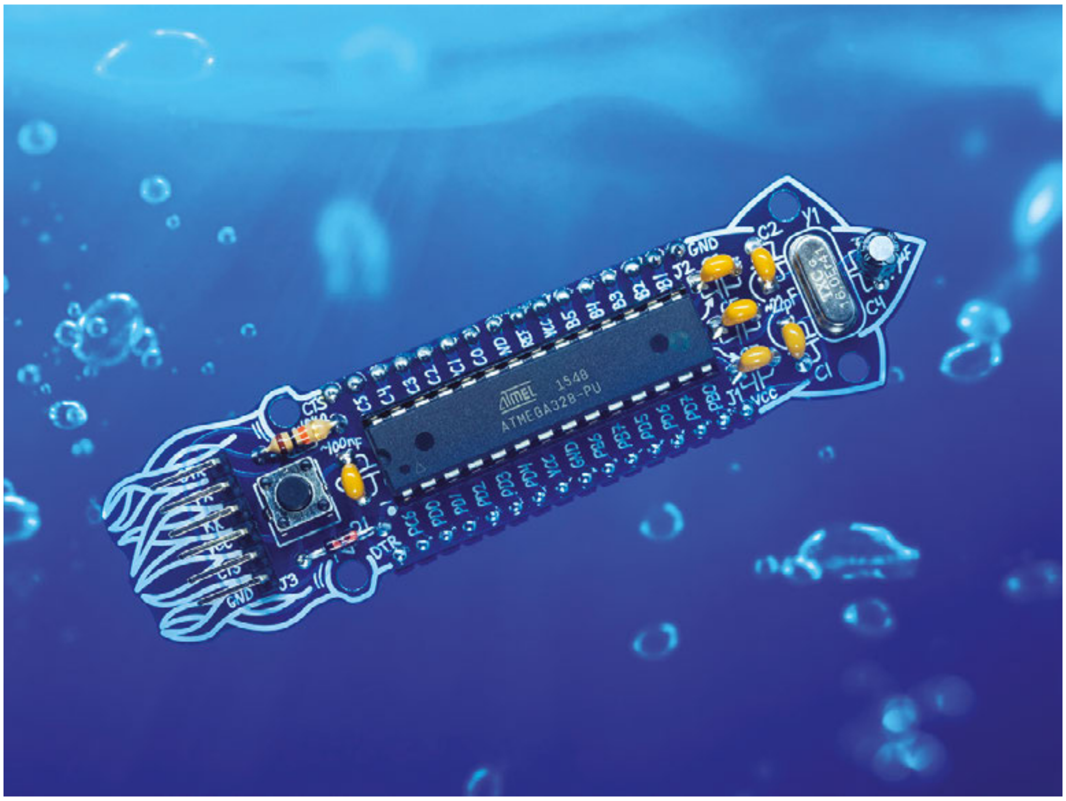
Meet The Maker: Boldport

The company that melds art and PCB fabrication

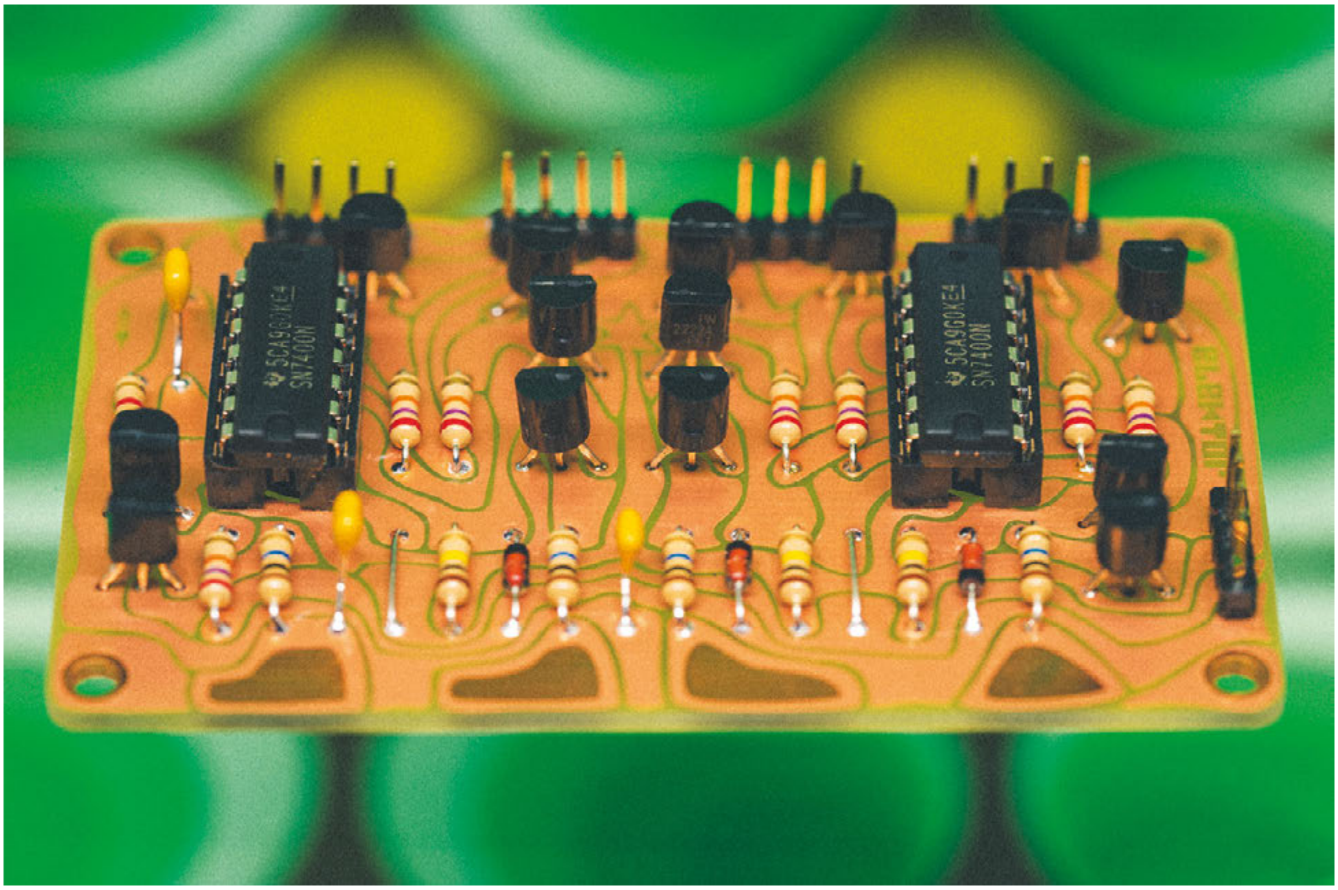
Printed circuit boards are mostly green rectangles. The copper traces carrying current to the various components go in straight lines – and the various resistors, capacitors, and chips line up in nice neat rows. Nothing about this is based on aesthetics. Electrons don't care what colour your solder mask is, nor do the components worry about being lined up. Boldport founder Saar Drimer frees circuits from this visual monotony.

PCB design software is part of the problem: it helps users build functional boards, not beautiful ones. Unsatisfied with the available options, Saar built his own: PCBModE, which converts between SVGs (which can be created using most vector drawing programs) and Gerbers, which are sent to PCB manufacturers. Freed from the constraints of CAD software, he's been able to unleash his creativity on the boards he makes.

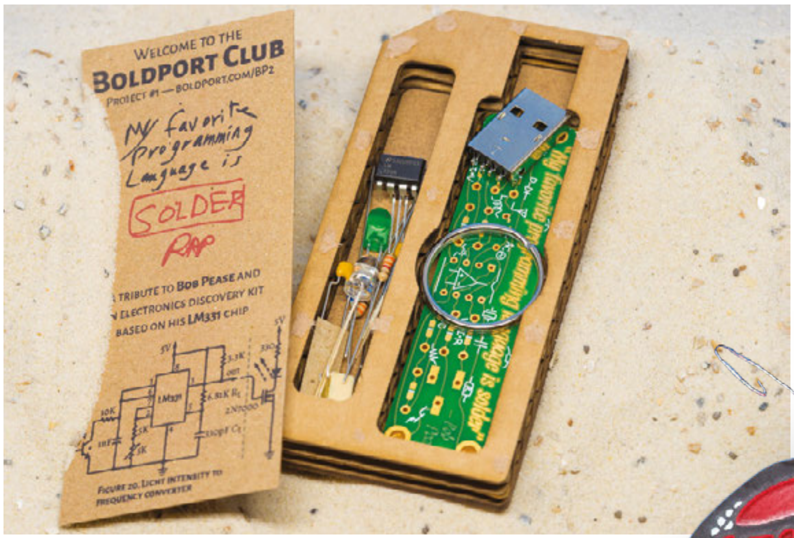
Saar isn't the only person who appreciates good-looking boards. He's started a subscription club to share his designs with the world. Each month, he packs up a board and components into a soldering kit that he sends out to subscribers. Launching in 2016, the club has already grown beyond Saar's expectations and Boldport has had to take on a new member of staff (Ben Barwise) to help keep up with demand. You can see all their work (and sign up for membership) at **boldport.club**. □



Right
An Arduino-compatible board that fits in a breadboard … or a coral reef



Images courtesy of Erbsland Art, erbsland-art.com


WELCOME TO THE
BOLDPORT CLUB
PROJECT #1 — BOLDPORT.COM/BP2
My favorite programming Language is
SOLDER
RAP
A TRIBUTE TO BOB PEASE AND AN ELECTRONICS DISCOVERY KIT BASED ON HIS LM331 CHIP
FIGURE 20. LIGHT INTENSITY TO FREQUENCY CONVERTER
My favorite programming language is solder

Above
The MOSTAP project uses a 40-year-old circuit to provide a touch-sensitive input

Below
The Monarch uses a shift register and logic gates to make the LEDs flutter

Above
The first Boldport Club project was a tribute to circuit designer Bob Pease

The joy of making

Making stuff is a form of play
– so let's break the rules and have some fun

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**

Back when I started school, aged five, everyone had a hand-knitted jumper. Often it was made especially and tailored to fit. By the time I was ten, you were teased if you had one – could you not afford to buy one ready-made? Handmade items were often seen as second best. People who made things for a living seemed to be a dying breed. Making was not a career path.

When young, we are encouraged to make things. Give a group of six-year-olds a pile of junk (empty washing-up liquid bottles, cardboard tubes, etc.); add some tape, shiny paper, and a few pipe cleaners and the creations will be fantastic. Rockets, castles, robots, dragons, trains – children will have fun and let their imaginations run wild.

When it was discovered I had the genetic make-up and attitude for academia – which meant I did well at exams – the opportunities to make creative things at school diminished. You were either practical or academic. Not both.

Fortunately, I gained a lot of practical skills outside of school. At home I was surrounded by people who made things. I was also a member of the Girl Guides and later Scouts – where using the available resources to solve a problem was encouraged. I found that here I could combine my academic knowledge with my practical skills. This is when I became a maker.

Our day jobs can stifle creativity, and making is limited to only a few professions. A manager's role is often to make sure things are done to specification – without variation or improvement.

But making can be a hobby as well as a profession. Homemade, bespoke, and artisan products are becoming cool. Making is on TV. There are local craft markets. Websites such as Etsy are thriving. People are not only making things but others are buying these things.

Making is something we can all enjoy. We don't have to make a junk model to unleash our inner child – to be open to ideas, use our imaginations, and have fun

From knitters and potters to computer and electronics experts, people now share skills and ideas on the internet in videos, blogs, and even social media. Physical meeting places such as hackspaces and other community groups allow people to share tools, skills, and ideas face to face.

Making is something we can all enjoy. We don't have to make a junk model to unleash our inner child – to be open to ideas, use our imaginations and have fun – but we could…

How do you celebrate the joy of making? □

Raspberry Pi: the duct tape of computing

Versatility is a feature: why the Raspberry Pi is still useful

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.

Duct tape is one of those things you'll find in virtually every toolbox. It's a jack of all trades but master of none, so despite being in every engineer's workshop, you'd be surprised to open a consumer product and find that it's held together on the inside with duct tape.

The Raspberry Pi is in many ways the duct tape of computing. I've come to find it indispensable in the lab – its a GPIO-to-internet box that's also powerful enough to host and compile complex Git repositories. Furthermore, its native toolchain can directly target most ARM-based embedded projects. As a result, I've retired most of my JTAG dongles: why carry around a USB adapter when I can get a fully fledged development environment and JTAG-over-GPIO (via openOCD) that I can SSH into?

If the Pi is the duct tape of computing, Arduinos are like Scotch tape – great for light applications around the home; and the industrial SOMs are like speciality adhesives

The Raspberry Pi is also cheap enough that I can afford the convenience of a new module for every project, rather than attempting to extract the board from the unruly tangle of wires that inevitably sprouts from its GPIO headers. And it's available enough that I can count on getting a new one almost anywhere in the world. This last point is crucial: the friction-free supply chain for Raspberry Pis mean I can do design in Singapore, demos in the USA, and development in China on the spur of the moment, without spending an arm and a leg on courier fees.

Like duct tape, the Pi isn't perfect for everything – its strength comes from its versatility and availability. The turnover rate of new Pi models can be frustrating; they're almost but not quite perfectly cross-compatible between models. The form factor and connector layouts are also a bit clumsy, and there are situations where I've wished for more I/O capability. They also have a tendency to fail at the worst times, which is why, whether I'm walking into a big demo, or venturing out to Burning Man, I'm sure to pack a spare Pi plus backup copies of the SD card image.

If the Pi is the duct tape of computing, Arduinos are like Scotch tape – great for light applications around the home; and the industrial SOMs are like specialty adhesives – perfect for their intended application, but too specific for the toolbox. And so, despite being designed originally for the education market, the Raspberry Pi's versatility and ubiquity has earned it a place in this engineer's toolbox, right next to the duct tape. □

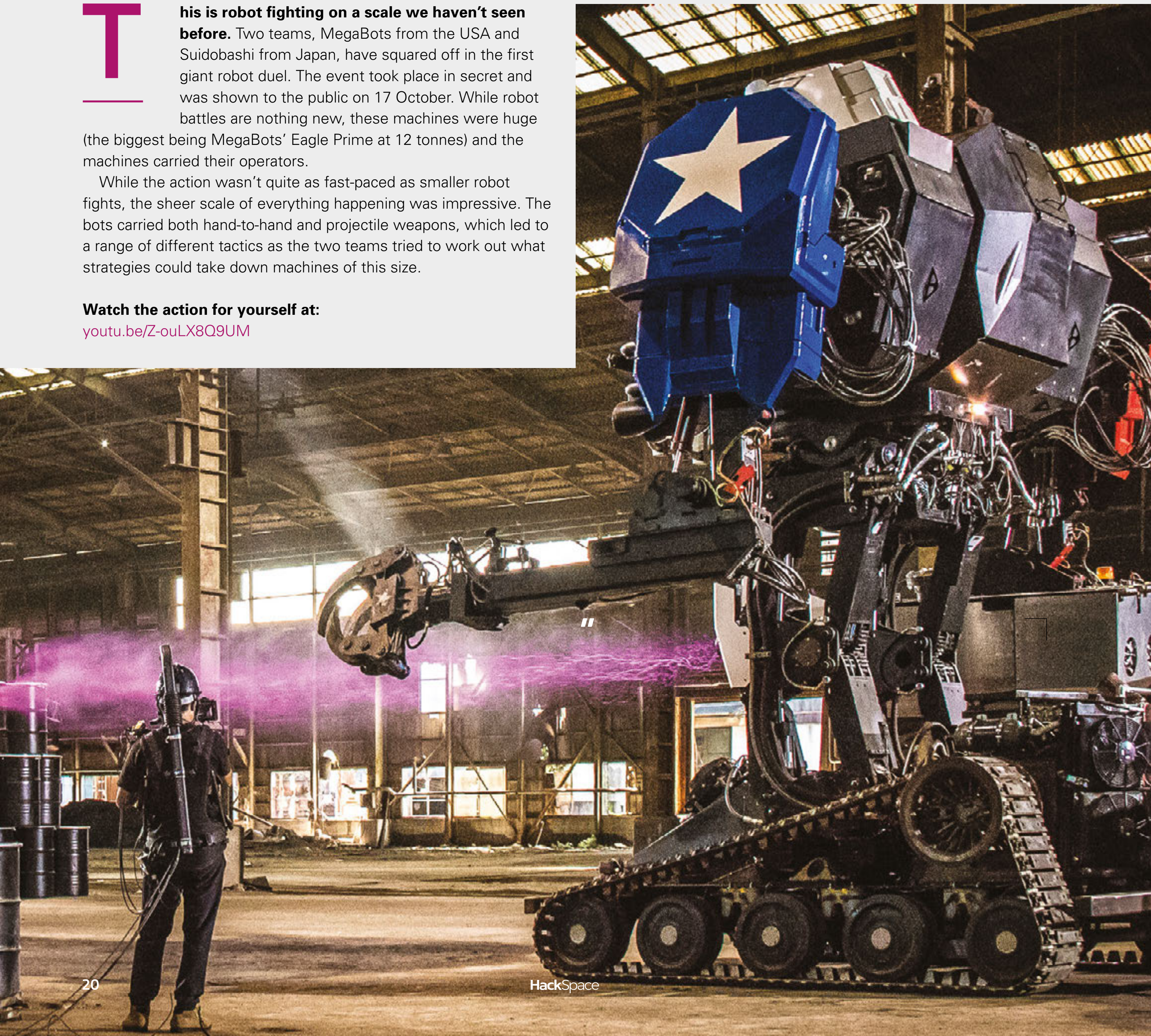
Mecha Death

Robot battles go super heavyweight

This is robot fighting on a scale we haven't seen before. Two teams, MegaBots from the USA and Suidobashi from Japan, have squared off in the first giant robot duel. The event took place in secret and was shown to the public on 17 October. While robot battles are nothing new, these machines were huge (the biggest being MegaBots' Eagle Prime at 12 tonnes) and the machines carried their operators.

While the action wasn't quite as fast-paced as smaller robot fights, the sheer scale of everything happening was impressive. The bots carried both hand-to-hand and projectile weapons, which led to a range of different tactics as the two teams tried to work out what strategies could take down machines of this size.

Watch the action for yourself at:
youtu.be/Z-ouLX8Q9UM

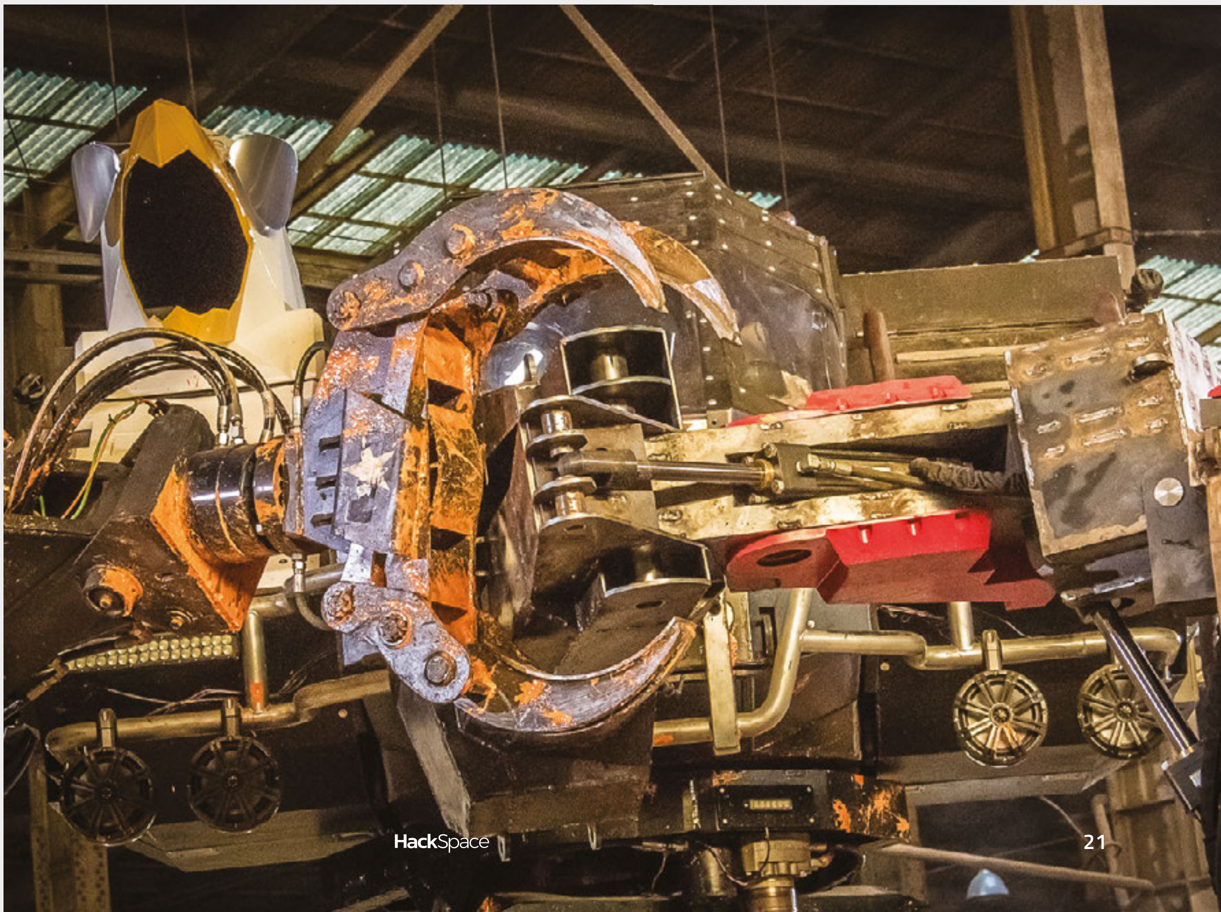


While this was a one-off battle, MegaBots are looking to start a league of live giant robot combat events. As we go to press, there aren't yet details of what this will entail, but they have aired a test fight between Iron Glory and Eagle Prime in which they tried out different combat styles that could be used. Stay up-to-date with the latest developments at **megabots.com**.

Robot fighting comes in all shapes and sizes. The most famous are Robot Wars (in the UK) and Battlebots (in the USA), but there are other mechanical combat disciplines and fights around the world. Generally, robot fighting is split into weight classes, with the smaller classes being more accessible to hobbyists as they don't require as much metalwork. Smaller robots such as the antweight class (known as fairyweight in the US), which have to be under 150 grams, are often 3D printed. ◻

For more details on local rules and competitions take a look at fightingrobots.co.uk for the UK **or** robotbattles.com in the US.

Images Michael Maulin



Hackspace of the month: Your Hackspace!

Everyone has within them the instinct to create, as well as the human need to be around people with similar interests to us. But not everyone has a garage, shed or spare room (or the space and money for a CNC router, laser cutter, welding equipment etc). And so the ancient ones created the makerspace: a kingdom where all could come and unleash their creativity on the world.

Except, it isn't like that. If you have a place you can go to solder components together and build whatever you next project might be, it's not because someone else came along and made the space for you. You and your group have most likely had to beg, borrow and scrounge tools, made do with cast-offs, and hunted high and low for a suitable building to work in. And then there are the arguments about who empties the bins, who tidies up, who orders more printer filament when it's running low… it's a miracle that maker/hackspaces exists at all.

Which is why we're making it our mission to highlight a makerspace every issue, to inspire others, to give pointers and to show off. We want to hear from you, about what you're doing and why you're doing it, but just as importantly we want to know how you curate the space itself. Here's what we want to know about your hackspace…

WHEN DID YOU START?

An obvious one this, but it's good to kick off with the basics. What we've found is that a lot of makerspaces with an impressive output, a lot of tools, and some great-quality builds have only been around for a couple of years. If you're new to anything, there can be a natural reluctance to put yourself forward, but we want to hear from everyone, no matter whether you're just starting or have been around for years.

WHO IS IT FOR?

Some makerspaces spring up in university towns, as a way to channel the creative direction of ex-students. Some grow out of a local employer that might have tools or space to hire cheaply, and some are the results of random chance throwing the right people together at the right time.

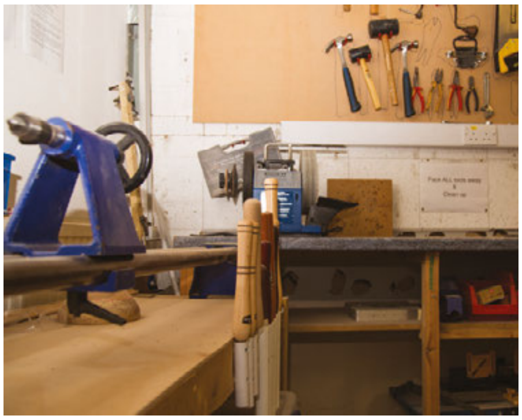
What all makerspaces have in common is a lot of hard work. How did your hackspace come to be? Are you sponsored by any local organisations? It's easy to say you're for everyone, but how do you make sure you get a mix of people? →

Left
Not every hackspace is lucky enough to have such a large selection of tools at its disposal (many thanks to Cambridge Makespace for letting us take the photos)

Above
What software do you use? Do you share your source files?

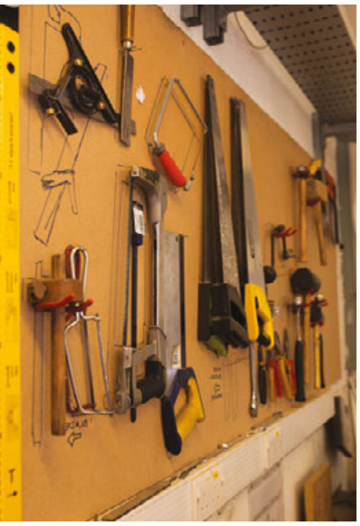
Right
'Obsolete' hardware suddenly becomes more useful if you can acquire it for free

Left
We love the smell of laser cutters in the morning. Stay safe!



Right
Sharing a space gives everyone a responsibility to keep it tidy

> **Some makerspaces spring up in university towns,** as a way to channel the creative direction of ex-students



WHAT EQUIPMENT DO YOU HAVE?

Anyone thinking of joining a makerspace probably fits into one of two categories: those with a project in mind, and those who want to make something but aren't sure what yet. In the latter case, a large determinant of what gets made will be what kinds of tools you have available. It's a good idea to have an up-to-date list of the equipment you have, especially the rarer or more unusual stuff.

HOW DO I BECOME A MEMBER?

Hackspaces usually have open days for non-members, but how do you go from that to becoming a full member?

Most hackspaces will charge a token fee, but one person's 'token' can be someone else's substantial investment. Do you have reduced fees for students or the unwaged?

HOW DO YOU SHARE YOUR IDEAS?

Do you have a mailing list, IRC group, Facebook group, Twitter, or something else? It's natural to want to show off (that's how we get suggestions to make our work better), so it's nothing to be shy about. Some brave souls even venture out to meet people in real life. Does your group attend maker meet-ups?

WHAT HAVE YOU MADE RECENTLY?

If you ask anyone from a hackspace what's going on there, their first instinct will probably to say "all sorts!". Well, be specific, and give us photos! What is it, who made it, what does it do, how did you make it? Most importantly of all, how does it make the world better?

WHAT TRAINING DO YOU DO?

This won't apply to smaller groups, but if you have potentially dangerous kit, such as welding equipment or laser cutters, you'll need at the very least to have safety training. After that, it comes down to how much time you can invest in showing new members how to make the ideas inside their heads. □

Left
People! The most important ingredient of any hackspace

Below
What are you working on right now? Let us know!

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.

HackSpace
TECHNOLOGY IN YOUR HANDS

LENS

HACK | MAKE | BUILD | CREATE

Uncover the technology that's powering the future

FEATURE

The INTELLIGENCE Makers

An increasing number of DIYers are giving their projects a mind of their own

When the Raspberry Pi was first launched in 2012, it was designed as a way for kids to learn to code without spending oodles of money. It wasn't long before tech-minded DIYers realised that the puny little device has enough processing power and functionality for an array of different fun and practical home projects. Arguably, the explosive growth of the Raspberry Pi is because of the hobbyists, who have been hooking up all kinds of actuators and sensors to build cool new things.

But the Raspberry Pi isn't the only single-board computer (SBC) fuelling this exponential growth of the maker community. One of the greatest advances in computing hardware has been the proliferation of microcontrollers. A microcontroller consists of a processor with a small instruction set, some memory, and programmable input/output circuitry contained on a single chip. Microcontrollers are usually packaged with supporting circuitry and connections on a small printed circuit board.

One of the most successful and most popular microcontrollers is the Arduino platform, which made its debut in 2005. It was originally aimed at hobbyists who needed an affordable microcontroller to make their projects more interesting. Thanks to its ease of use and versatility, the Arduino quickly gained the attention of a larger audience and a wider variety of projects. In fact, together these microcontrollers and single-board computers like the Raspberry Pi, Banana Pro, BeagleBone Black, CubieBoard and others have brought traditional resource-intensive fields like artificial intelligence and robotics into the realm of the scrubby DIYer.

In this feature we'll look at some projects that haven't just used these embeddable computing platforms intelligently, but have used them to build real intelligence into their creations.

Intelligent IoT

Why settle for a smart home when you can have an intelligent one?

The Internet of Things (IoT) is pitched as the ultimate upgrade that'll turn your house into a smart home. The everyday household devices are all connected to the internet and to each other and together will automate mundane tasks such as replenishing the detergent and restocking the kitchen. While such a level of home automation isn't quite yet available to the masses, you can build upon the work by the bunch of trailblazers that are building intelligence into various aspects of their daily chores.

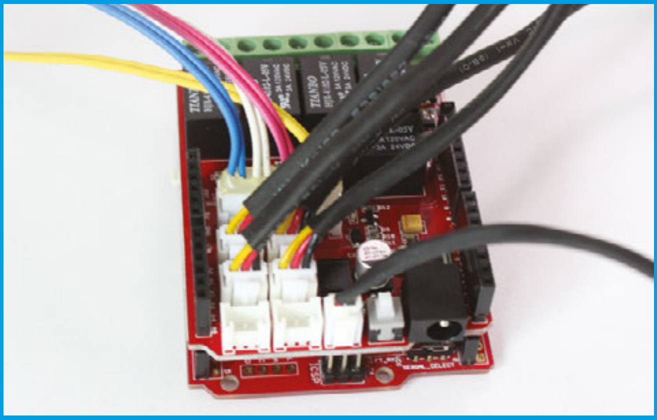
NURTURING NATURE

When Shenzhen-based hardware facilitation company Elecrow had to move out of its old office, it couldn't trust its neighbours to take care of its potted plants. So Elecrow designed a plant watering system to keep an eye on the moisture levels in the pots and automatically water them when they're too dry. It has refined its original design and the new one (**hsmag.cc/HhpgXb**) is easier to cobble together and can water up to four plants with one node.

The system uses inexpensive soil moisture sensors in each pot, a water pump, and a couple of servos to control the distance and angle of the water spray. The components are controlled by the company's home-brewed Crowduino Uno, which is an Arduino-compatible board based on the ATmega328 microcontroller.

Elecrow has shared the code that you can push to your Arduino Uno, along with detailed instructions on how to connect the soil moisture sensors and the smart pump shield to the Arduino. Once you have everything set up, the Arduino microcontroller will keep an eye on the readings from the soil sensors. When it detects a drop in moisture levels in any of the plants, it'll move the watering assembly to the pot that's dry and activate the pump to spray the plant. You can tweak the code to modify the parameters as per the physical setup of your plant pots.

Saddam Khan has built upon this design and added the SIM800 GSM module (along with a voltage regulator) to the configuration (**hsmag.cc/emevVp**) to send regular alert messages with the reading from the soil sensor along with the status of the water pump. In true DIYer style, he has also described the procedure to create your own simple soil moisture sensor. In addition, he's explained the workings and the code that drives the system in great detail.



Right
There's plenty of room on the Arduino to add all sorts of sensors. You can check the level of water in a dish and refill it to make sure your quadruped friends are never thirsty

Credit
Instructables.com

TRICK THE TREAT

We all love our pets and it kills us to leave them home alone. John Saunders, who owns the NYC CNC machining and prototyping shop, cobbled together a machine (**youtu.be/PAHrBA0jYAo**) for his two-year old Vizsla dog named Judd that dispenses treats whenever Judd receives an email.

Inspired by John's design, Walter Miraglia created a version of his own (**thingiverse.com/thing:2187877**) that is much simpler to put together. In Walter's version you send an email to your pet's email address. The system checks for email every ten minutes or so and if it finds any new unread emails, it dispenses a treat. It then waits a few seconds before snapping a picture of your pet munching the treats. The system then replies to the yet unread emails and attaches the recently snapped picture. Finally, it marks the email as read, to prevent it from falsely triggering the treat-dispensing system.

Both John's and Walter's systems are powered by the Raspberry Pi. Walter combines it with an Arduino Mini to control the servo motor that drives the auger to dispense the treats, as well as some LEDs to signal incoming treats.

Auditory Intelligence

Converse with your projects

Nothing speaks intelligence like speech itself. Grant Gibson's toddler enjoyed *Toy Story 3*, but never took a fancy to the Talking Chatter Telephone that Fisher-Price reintroduced to coincide with the release of the movie. So Grant, like any affectionate maker parent, prised open the toy and made it come alive by replacing its internals with a Raspberry Pi B+, a WiFi dongle, and some Python code.

Grant's version of the telephone (**hsmag.cc/VLTZAA**) retains the original rotary dial, but he's added a sensor to detect when the receiver is off the hook and a servo to animate the toy's eyes. The chatter smartphone uses JSON format APIs to pull data from various online services such as Rotten Tomatoes and Forecast.io. He's also configured a push notification system that takes input from a Twitter account via IFTTT. Thanks to this, Grant's phone automatically announces when the International Space Station is about to pass overhead and when he's left the office.

While his son has outgrown the toy in the three years since the build, the project still gets used, shares Grant: "Just last month I was invited to give a talk and demonstration to his school classmates about this and other Raspberry Pi projects we've worked on together."

Noting the importance of cheap SBCs and microcontrollers to the DIY community, Grant says: "Arduino was revolutionary for me, giving me the tools to connect an internet-connected PC to the physical world. What started as a hobby – my first project was connecting my door-bell to an SMS gateway – turned into a career, building physical games and machines rather than conventional software and websites. Raspberry Pi took that to the next level, effectively replacing both the PC and Arduino with a single SBC."

SMART COOKING

If we had a penny every time we put the tea on to brew and forgot about it, we'd have bought our own tea estate. James Pavur probably had the same problem, which is why he designed the TeaPi to automate the process of brewing tea (**hackaday.io/project/156-tea-pi**). You just tell TeaPi for how long and at what temperature you want your tea brewed, and the Raspberry Pi will activate the connected kettle, measure the temperature, and lower the tea in with a servo motor. Once the tea leaves have been in for the desired time, it'll lift them out again.

Coffee drinker Bastian Slee took another approach. He ripped open his Philips Senseo coffee machine and hooked it up to a Raspberry Pi 3 (**hsmag.cc/HqFkXx**). He then used Alexa together with the AWS Lambda compute service and the AWS IoT platform to send voice commands to the Raspberry Pi, which interprets the commands and works the machine to prepare a perfect cup of coffee.

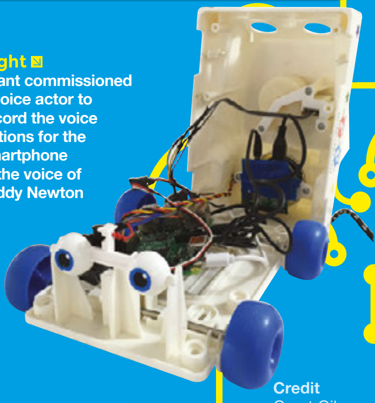
Converse with the Raspberry Pi

There's no dearth of voice recognition systems available for the Raspberry Pi. One of the newest and most popular is the open source artificial intelligence system Mycroft AI. You can run Mycroft on a Linux desktop, your Android smartphone, and of course on the Raspberry Pi.

The Raspberry Pi flavour of Mycroft is called Picroft, and you can grab the .img file and transfer it onto an 8GB microSD card, power up the Pi, and then follow brief instructions (**hsmag.cc/glUdky**) to pair it with an online account.

You'll also need to hook up a pair of speakers and compatible microphones (**hsmag.cc/fTwdmS**). This shouldn't be much of a problem since, according to the project, Picraft works with microphones based on the CM108 chip, which you can find inside a large number of cheap off-the-shelf mics.

Perhaps the best thing about Mycroft is that it's easily extensible and you can teach it a new skill writing some code in Python. You can find a detailed tutorial on coding a new skill (**docs.mycroft.ai/skill.creation**) and can also look at the existing skills for inspiration (**github.com/MycroftAI/mycroft-skills**).



Right Grant commissioned a voice actor to record the voice options for the smartphone in the voice of Teddy Newton

Credit Grant Gibson

Visual Intelligence

They've got an eye on you

One of the most popular applications of artificial intelligence is the detection and recognition of faces. SBCs like the Raspberry Pi have enough horsepower to run the Open Source Computer Vision library (OpenCV) which is a collection of programming functions that allow computers to see.

Tony DiCola combined the Pi and OpenCV to build a treasure box that unlocks only when it detects a known face (**hsmag.cc/TZOYiU**). The treasure box uses the official Raspberry Pi Camera Module to capture your image when you press a button to unlock the box. If it recognises the face, the box will ask the small servo to rotate the latch and unlock the box.

Instead of a camera and OpenCV, Sam Brown uses the Walabot sensor that can look through walls and track moving objects to prank trick-or-treaters (**hsmag.cc/duWxRG**) as they approach your house. Using the Walabot Pro, he designed the tracker to peer right through the front door and look out for the Halloween visitors. As they make their way to the front door, it'll pounce on them with scary sounds and customised messages depending on their number and how close they've come.

EYE SPIES

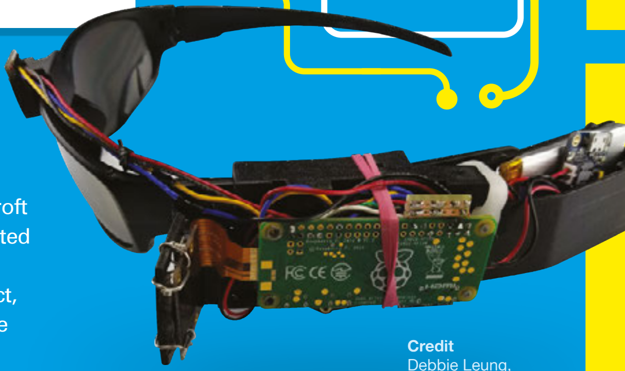
You can take the image recognition skills up a notch with the Clarifai API, which helps computers recognise images. Greg Voronin paired the Clarifai API with Mycroft to build a Smart Eye atop a Raspberry Pi 3 that can recognise the objects placed in front of it. He built the project in two stages. Initially he used Picroft, Mycroft's version for the Raspberry Pi, to invoke the camera and snap a picture of the object. It was then sent to Clarifai for processing and Picroft subsequently spurted out any associated tags and concepts.

Buoyed by the success of the project, Sam then read about natural language generation (NLG) and used Python's Pattern module to turn those tags into a simple sentence. So the second version of the Smart Eye (**hsmag.cc/quEyNT**) can describe the objects placed in front of it in simple sentences, like "I see a watch". You can also ask it whether it sees a particular object, like "Do you see a cat?" For this task, it first separates the nouns and adjectives from your query and then compares them with the ones it received from Clarifai.

While there are other image processing libraries, Greg chose Clarifai because of its ease of use. "With Clarifai you can concentrate on the application, and not have to work out the AI details unless you want to, [which makes it] ideal for makers," he explains.

Capitalising on the computer's ability to replicate the behaviour of the human eye, a group of four engineers and makers are working on a project to help visually challenged folks navigate the world independently. While they aren't the first to use technology to assist disabled people, the team have been able to miniaturise the entire system to fit on a pair of sunglasses thanks to off-the-shelf SBCs. Their prototype, called Visioneer, uses a couple of cameras, and sensors such as accelerometers, to enable blind people to sense their environment via a bone conductor that doesn't interfere with their ability to hear. The captured raw data is first processed through a local neural network and OpenCV before being turned into vibrations, and the entire operation is managed by the minuscule Raspberry Pi Zero.

The group has published Visioneer's entire build process (**hsmag.cc/xSYeUy**) and is competing for this year's Hackaday Prize.


Credit
Debbie Leung, Visioneer

Above
If the Visioneer team win the Hackaday Prize, they plan to use the award money to improve the prototype and even market it as a visual aid

Plug and play AI

Makers have utilised the power of vision on a variety of projects built with SBCs like the Raspberry Pi. Intel recently unveiled the Movidius Neural Compute Stick (NCS) (**developer.movidius.com**) that reduces the wizardry to a plug-and-play USB stick. Just plug the NCS stick into the USB port of the Raspberry Pi to give it the ability to identify the objects it sees through the camera. The image-recognition task is offloaded to the stick, which frees up the processing power of the Raspberry Pi for other tasks. The £69.99 stick consumes very little power and has 12 SHAVE processor cores in the Myriad 2 Vision Processing Unit (VPU) at its disposal for lightning-quick object recognition.

While initially the stick's SDK only supported a 64-bit Ubuntu 16.04 desktop installation, in August 2017, Neal Smith, Senior Software Engineer at Intel, announced support for the Raspberry Pi and also uploaded a video and a guide for users to sample the NCS's image recognition capabilities (**hsmag.cc/jTIQIF**).

Spatial Awareness

Who knows when, where and what

Plugging sensors into the projects and interpreting their results is a more rudimentary type of intelligence when compared to interpreting sights and sounds, which represent a more complex source of sensory input.

At the Maker Faire 2010, Steve Hoefer programmed a gumball machine (**hsmag.cc/KgNclP**) to dispense treats only when someone knocked a particular pattern, which in this case was the popular 'Shave and a Haircut' pattern. The machine uses a piezo sensor to pick up the sounds from a knock-panel, that are then interpreted by the Arduino. When it detects the correct rhythm (ignoring tempo, so the speed of the pattern makes no difference), the Arduino asks a servo motor to release the treat.

PATTERN RECOGNITION

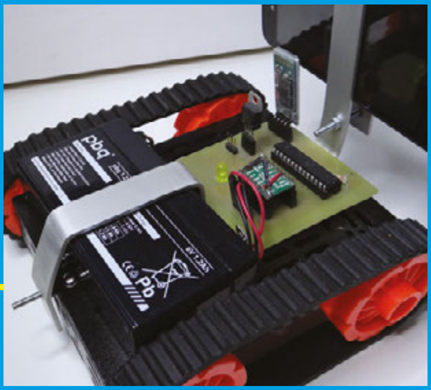
In the same vein as the secret knock gumball machine, Zack Schollz trained his Raspberry Pi to recognise patterns as well. But it did so while he played the piano and after a while Zack's PianoAI (**rpiai.com/piano**), true to its name, automatically started filling in style-appropriate tunes as Zack's duet partner! In the video he's posted, Zack jams on the piano for about 20 seconds before the PianoAI takes over the melody in between Zack's pauses.

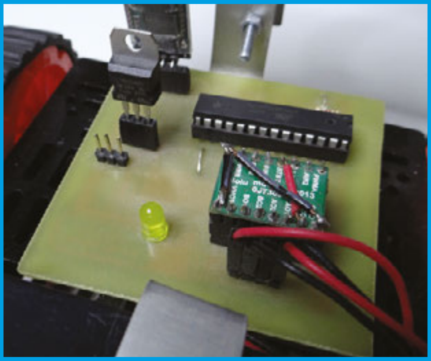
Zack has detailed the process of writing the AI in great detail. He initially used Python but then moved on to Go because of its speed. He also tried using neural nets and then experimented with a few different pattern-recognition algorithms before settling on a modified Markov chain algorithm that he tweaked as per his own style of playing the piano. He has also posted several videos to give us a sense of the evolving skills of the PianoAI.

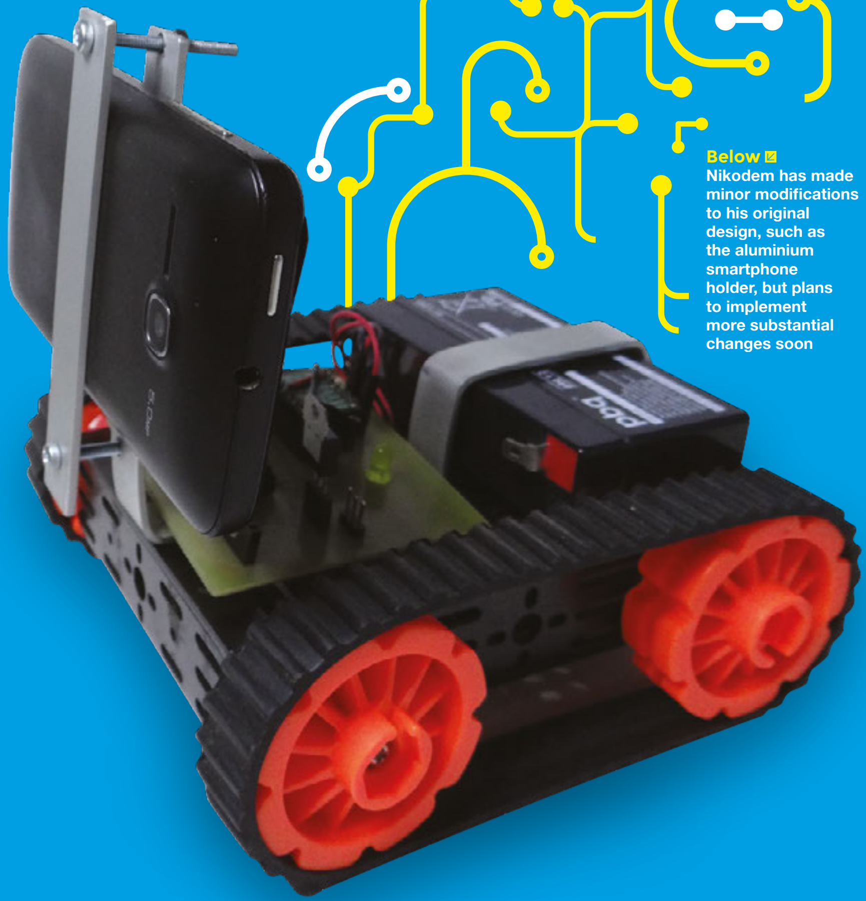
Staying on the theme of picking up patterns, when Nikodem Bartnik became interested in robotics, he built an object-tracking robot (**hsmag.cc/LwBgwf**) that picks its target based on a colour (red in this case). He placed an Android smartphone on the robot that passes everything its built-in camera sees via a custom app to the OpenCV library for processing. The app calculates the arithmetic average to home in on the colour it's tracking. This information is then sent to the on-board Arduino that guides the robot to the object.

Below Debashish believes that SBCs and microcontrollers are helping hobbyists and DIYers become active participants in the evolution of technology

Credit Debashish Buragohain







Below Nikodem has made minor modifications to his original design, such as the aluminium smartphone holder, but plans to implement more substantial changes soon

> Their autonomous robot is capable of exploring the area while avoiding any obstacles. It can also detect bottles and bring them to a designated storage area

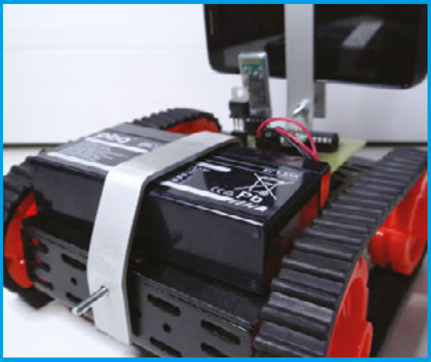
Other people have adapted this into object-tracking robots of their own. One created by Rohan Juneja (**hsmag.cc/UqYvvn**) uses the official Raspberry Pi Camera Module instead of the Android phone and processes the image with a Raspberry Pi.

Karl Kangur, along with fellow students Marcel Starein and Chun Xie, used OpenCV's image recognition capabilities to build a litter-collecting bot (**hsmag.cc/IpPEDk**) as part of their Master's degree semester-end project.

Their autonomous robot is capable of exploring the area while avoiding any obstacles. It can also detect bottles and then use the simple bottle storage system designed by the trio to retrieve and bring the bottles to a designated storage area. Their recycling bot uses four infrared sensors to detect obstacles and the PiCamera Python module to detect discarded bottles. Karl and his team won the competition – between five teams of students – and have also published a detailed paper on their project (**hsmag.cc/taJITy**).

Below Using OpenCV image recognition, Nikodem's robot is able to track and follow red objects

Credit
Nikodem Bartnik



FIST BUMP

In his quest to get to grips with Arduino, Debashish Buragohain built a robot car (**hsmag.cc/wOdSIA**) that understands voice commands, can answer simple questions, and is equipped with the HC-SR04 sensor to avoid obstacles.

The HC-SR04 emits ultrasonic sound waves and records the time it takes for them to bounce back from an object like a wall. The Arduino then calculates the distance to the obstruction and asks the robot to stop moving if the distance is less than 30 cm. It then changes direction towards the left or the right depending on which side has the greater obstruction-free space. →

Teenager Tamas Imets has been building things for over a decade and has created several ground as well as aerial robots

Fuelling the growth of open source robotics

Inexpensive yet powerful single-board computers have reshaped the robotics landscape not just for the hobbyists but for researchers and educators as well. "When I was in graduate school (late 1990s and early 2000s)," says Brian Gerkey, CEO of Open Robotics, "getting a decent computer on a mobile robot meant using expensive industrial hardware like PC-104 stacks, building a custom machine around a smallish desktop system board and then figuring out how to power it, or using a laptop that would inevitably be borrowed for another purpose. Now you can build a robot like the TurtleBot 3 Burger around a low-cost but very capable computer like the Raspberry Pi 3 and power it with a USB cable. This trend will accelerate as robotics continues to benefit from the descendants of technology originally designed for the mobile device market."

Open Robotics produces the Robot Operating System (ROS) which is one of the most popular open source middlewares. While you can write some Python code to automate certain simple tasks, complex robotics needs a middleware software 'glue' that binds the hardware and makes it easier for robot builders to program their creations. ROS came to life in 2007 at Stanford University and then matured in the Willow Garage incubator before the team handed over the BSD-licensed code to Open Robotics.

Juan Miguel Jimeno heads the Linorobot project (**linorobot.org**), which publishes specifications for a collection of open source ROS-compatible robots to students, developers, and researchers. Juan believes that the affordable computing platforms "allow DIY roboticists to run their own robot application on a homebrew platform and leverage ROS's powerful robotics development framework. This breaks a lot of barriers, especially for students, in accessing sophisticated and complex software applications that run some of today's advanced robots like Atlas and NASA's Robonaut."

The robot can also be controlled through an Android smartphone over Bluetooth by using the phone's built-in accelerometer.

The 13-year-old maker wants to install a PCB in order to make the batteries last longer.

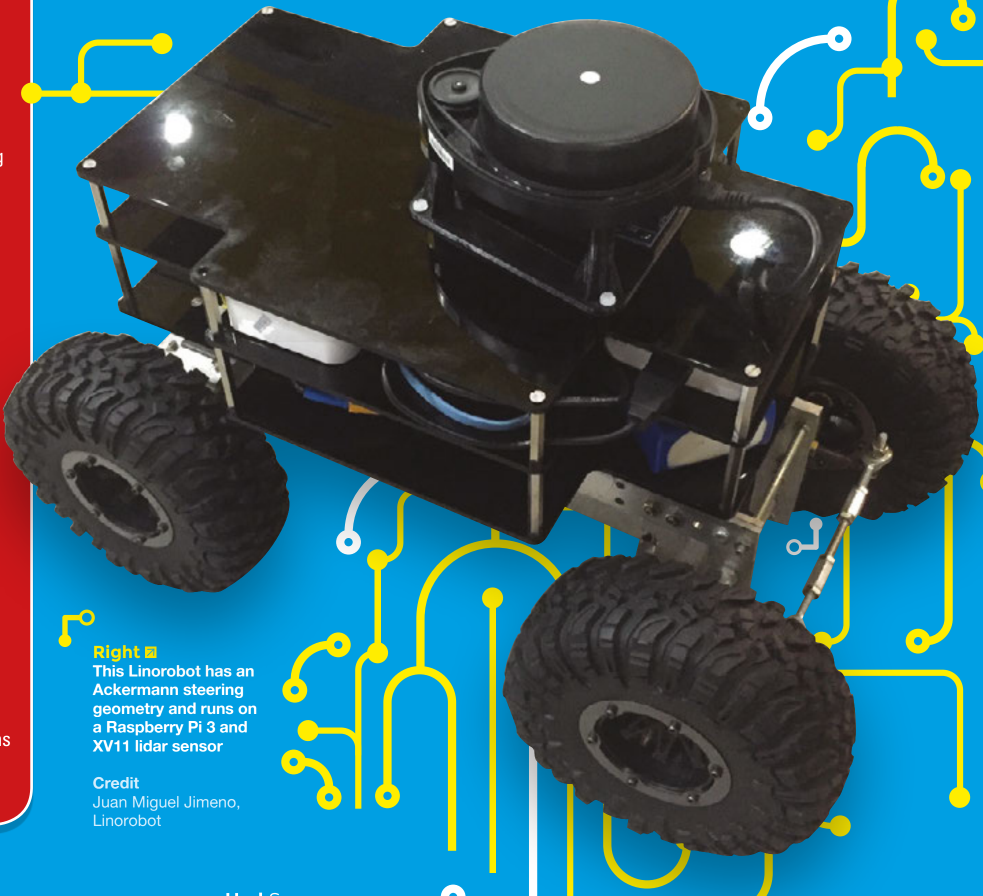
He also wants to give his robot new capabilities such as the ability to find its way around a maze, track faces and more, but is restricted by the lack of GPIO pins on the Arduino Uno. Debashish says he'll first have to switch to "a more capable board like Arduino Mega 2560 or Raspberry Pi" before he can implement other features while still keeping the price of the robot within reasonable limits.

Another teenager, Tamas Imets has been building things for over a decade and has created several ground as well as aerial robots. His latest is the intelligent flying robot drone (**hsmag.cc/tBxoWa**) that can track faces and objects and can even avoid obstacles.

Tamas's inspiration for the quadcopter came from a nasty bike accident in 2016 in which he lost a lot of blood before he was spotted by a couple who called emergency services. His drone uses the Pi Camera Module to follow a red ball or his face. A Python script, running on a Raspberry Pi Zero W on board the drone, captures the different shapes and then controls the flight path depending on what it's been trained to spot. Tamas hopes one day his drone will be able to rescue people stranded in the mountains. He's also shared code for using an Arduino-based MultiWii controller in case you want to control the drone manually.

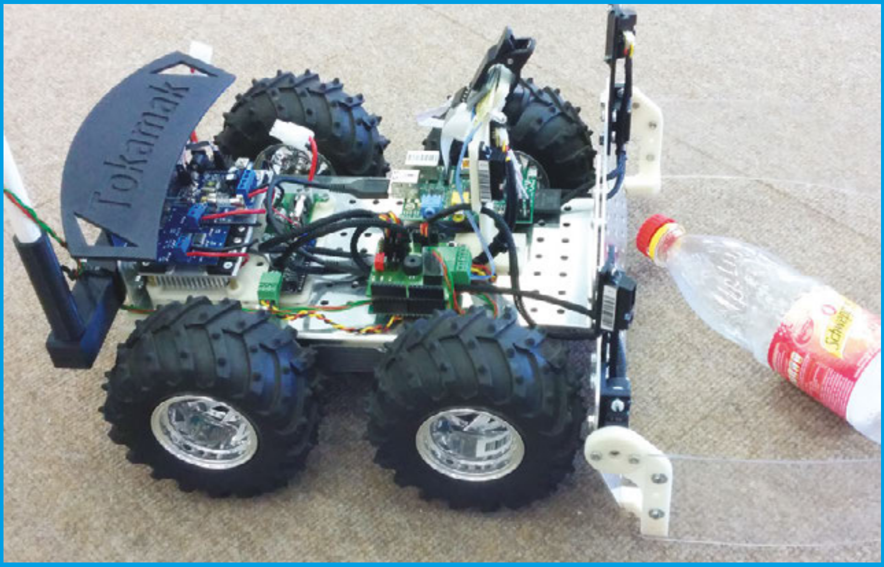
FINDING THEIR FEET

Then there's Renee Glinski. She's a prolific roboticist, but her right of passage to robotics was the self-



Right
This Linorobot has an Ackermann steering geometry and runs on a Raspberry Pi 3 and XV11 lidar sensor

Credit
Juan Miguel Jimeno, Linorobot



Left The bottle-recycling bot's project report makes for an informative read as it discusses various strategies for aspects of the build, such as movement and object detection

Credit
Karl Kangur

balancing robot she named Eddie (**thingiverse.com/thing:694969**). She built it using SparkFun Blocks and it's powered by the Intel Edison Compute Module. Eddie runs Yocto Linux and you can clone its balancing code from GitHub (**github.com/r3n33/EddieBalance**). The self-balancing robot can carry a variety of loads and adjusts accordingly. Renee has also posted impressive videos of Eddie going up and down slopes without losing balance. You can drive your Eddie via remote control over WiFi, or even out of sight with a first-person view camera that streams to a web browser.

A bunch of engineering students have also created a self-balancing robot (**hsmag.cc/yDPchl**) that uses an accelerometer and a gyroscope to feed data to the on-board Arduino, which also drives the motors to keep it upright. To keep their two-wheeled Segway-like robot from toppling over, they need to measure the roll or the angle of inclination of the robot, and then drive the motors in the opposite direction to negate the roll and maintain a vertical position. They've described the advantages and disadvantages of using the accelerometer and the gyroscope to measure the angle and how they've used them both with two algorithms. The guys want to extend this to a self-balancing bicycle.

A hat tip to Google

At the start of 2017, Google announced its plans to bring its artificial intelligence, machine learning, and other developer tools to the Raspberry Pi. In early May the firm released the Voice HAT (Hardware Attached on Top) board that adds AI voice recognition to the Raspberry Pi 3. In essence it allows you to interact with a Raspberry Pi 3 in pretty much the same way you do with the Amazon Echo or Google Home.

Initially launched as a free accessory with the print edition of The MagPi issue #57, the Voice HAT leverages Google Assistant's SDK along with the Cloud Speech API. The issue's kit included the Voice HAT add-on module, a speaker, microphone, cables, button, and a cardboard enclosure to put it in.

The Voice HAT is part of a bigger Google initiative dubbed AIY, or Artificial Intelligence Yourself, and Google has announced that it is working on more artificial intelligence projects to follow its Voice Kit for Raspberry Pi. The Voice HAT allows tinkerers to familiarise themselves with a voice interface. "We'll soon bring makers the 'eyes', 'ears', 'voice' and sense of 'balance' to allow simple, powerful device interfaces," wrote Billy Rutledge, director of AIY Projects at Google in a blog post. In the same post, he shared some of the ways DIYers were extending Voice HAT into their own projects.

GAME ON

Programming an autonomous robot to traverse a maze and then retracing the shortest path is a wonderful example of artificial intelligence. It took Patrick McCabe three attempts to finally design a robot that could find its way around a non-cyclic line maze (**hsmag.cc/aNqOyw**). His robot uses the well-documented 'left hand on the wall' algorithm to traverse the maze. However, it was the second trick of optimising and shortening the path it travelled which was a bit trickier to implement. Patrick's explanation of the algorithm makes for a nice, informative read.

The robot is driven by an Arduino microcontroller that uses a reflectance sensor, which emits an analogue voltage based on the amount of infrared light that is being reflected from the surface. The Arduino interprets the voltage feedback from the sensor and controls the motors to keep the robot on the straight and narrow.

If you want an AI you can go up against, follow Juan Pedro and Jose Julio's guide to put together an air-hockey-playing robot (**hsmag.cc/rXIvDO**). Their robot 'sees' through a custom app that's running on an Android phone mounted over the air hockey table. The app helps it analyse and predict the trajectory of the puck and also makes decisions to either attack or defend in real time. Once the app has made its decision, it communicates with an Arduino Leonardo mounted on the table, which in turn drives the two stepper motors to control the movement of the robotic pusher. The robot is capable of both attacking and defending and you can use the Android app to adjust its difficulty level.

Learning Machines

To boldly go where no program has gone before

Machine learning helps whatever implementation of artificial intelligence you're using to improve its own algorithm by processing a large amount of data. One of the most popular tools for helping computers decipher what they are looking at in a way that's intelligible to machine learning is TensorFlow. It's an open source library released by Google in 2015 to build and train deep learning models. TensorFlow has been trained by Imagenet, which catalogues several million images.

It takes an awful lot of computing power to create a machine learning model to do something like recognise images. To assist makers, however, Google has released several reference models that ship with the TensorFlow library, and makers can then use these in their builds.

John Naulty, whose San Francisco neighbourhood has a two-hour parking limit, used TensorFlow to keep an eye on the meter maids whenever he's working from home.

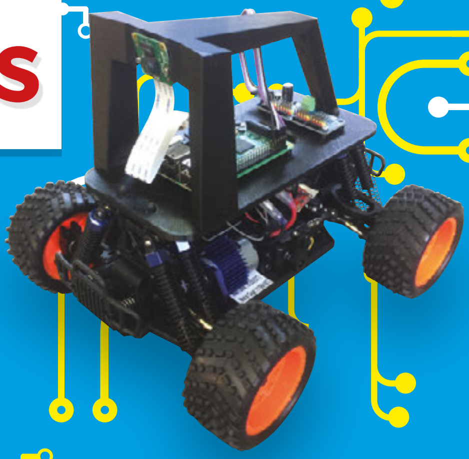
He wrote the Meter Maid monitor (**peoplesparking.space**) during the TechCrunch Disrupt Hackathon, to combine TensorFlow's image classification capability along with a motion detection and speed measuring program running on a Raspberry Pi.

When the Pi's camera detects an image of a moving car in its field of view, it snaps it and passes it to TensorFlow for analysis using its trained data to recognise the meter maid vehicles. If the snapped image is a valid match, it uses Twilio to send a message to John's phone with the image of the captured vehicle for verification.

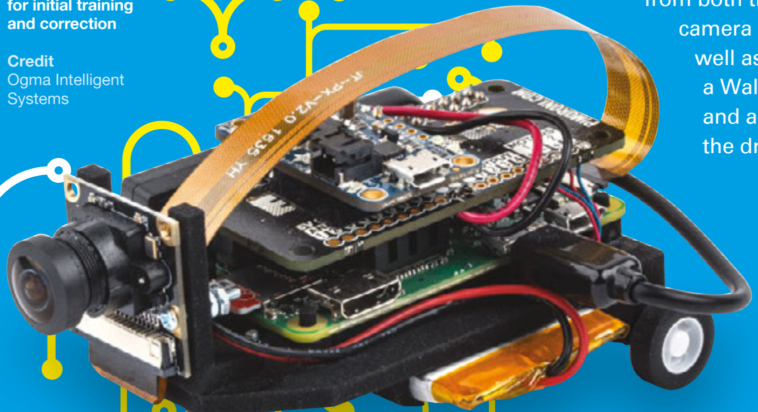
EYES ON THE ROAD

When Asad Zia bought a Honda Civic that lacked the company's proprietary driver assistance system, he decided to create his own with the help of TensorFlow. Asad's implementation (**hsmag.cc/SknPfC**) provides pre-collision alerts and can also detect pedestrians. It relies on images snapped from both the car's camera as well as from a Walabot, and alerts the driver if it encounters an object of interest in the same spatial coordinates. Walabot has several scan profiles or modes for different use cases. This project uses the sensor profile that takes high-resolution images at a slower capture rate. This mode provides image data in the spherical coordinates that his script first converts to Cartesian coordinates. The entire system is powered by a Raspberry Pi while the alert mechanism runs on a Raspberry Pi Zero. Asad has explained the functioning of the hardware components and the system's implementation in great detail.

You can also use TensorFlow to build a robot that's capable of recognising objects. Lukas Biewald has detailed the process in an easy-to-follow guide (**oreil.ly/2d2FBZL**) along with the code (**github.com/lukas/robot**). His robot is built atop a cheap £6.58 chassis by SainSmart coupled with the Adafruit Motor HAT mounted on a Raspberry Pi 3. He has detailed the hardware assembly in the guide. Lukas installed TensorFlow and hooked it up with the Flite (Festival Lite) text-to-speech package to allow the robot to vocalise what it sees. The robot includes some cheap sonar sensors to keep it from bumping into things, but isn't autonomous and is controlled via a simple custom web server written in Python.


Above Before you enable the Donkey Car's autopilot, you'll first have to train it by SSHing into the car and driving for about ten minutes using a virtual joystick in a web browser

Credit
Adam Conway, Donkey Cars


Below Both Ogma autonomous SDCs use the Steam controller for initial training and correction

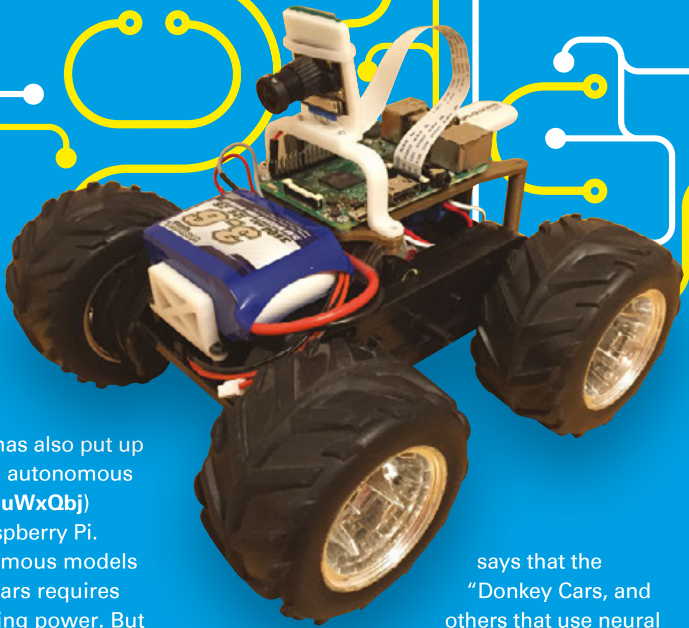
Credit
Ogma Intelligent Systems

On the other hand, Adam Conway and Will Roscoe have put together an easy-to-follow guide to building yourself a self-driving racing car powered by a Raspberry Pi (**hsmag.cc/qopEcK**). Their DIY autonomous racing cars are dubbed Donkey Cars and instead of selling pre-built kits, the duo encourage you to assemble your own.

Adam initially wanted to build an OpenCV-based rover, but Will insisted on leveraging machine-learning techniques. Their robocar now uses TensorFlow via the simplified Keras interface. You first have to train the Donkey Car by driving it around a track via a browser-based remote control. The Raspberry Pi records images and the steering angles, which are then sent to an Amazon EC2 instance that trains a TensorFlow model. It takes several laps of data accumulation to fully train the autonomous driving model, after which it can be loaded onto the car for hands-free laps around the track.

LEARNING

"Adding the words 'DIY' in front of established industries requires them to be 1) cheap, 2) easy, and 3) capable. Only with single-board computers (first Arduino for drones, and now RPi for cars, which is a much harder problem due to it requiring computer vision), can you get all three," says Chris Anderson, who's the CEO of 3D Robotics and organises the popular DIYRobocars meet-up. Autonomous car enthusiasts come together in these events to race their AI-powered robocars. The idea behind DIY Robocars is to improve AI without spending too much money.


Right
Karl built another bot, the ATOM (hsmag.cc/Teacfp), to get a hang of ROS

Credit
Karl Kangur

In fact, Chris himself has also put up a guide to building an autonomous racing car (**hsmag.cc/uWxQbj**) which runs with a Raspberry Pi.

Developing autonomous models for training the robocars requires quite a bit of processing power. But technology firms are now working to squeeze the learning aspect into tiny SBCs as well. To this end, AI research firm Ogma Intelligent Systems has created a couple of self-driving cars (SDCs), one built around the Raspberry Pi 3 and another based on the Pi Zero. "The Ogma SDCs differ from other autonomous Pi-powered cars, like the Donkey Cars, by doing all learning and subsequent inference/prediction only on the Pi (3/Zero)," explains Ogma's Senior AI Research Engineer, Richard Crowder.

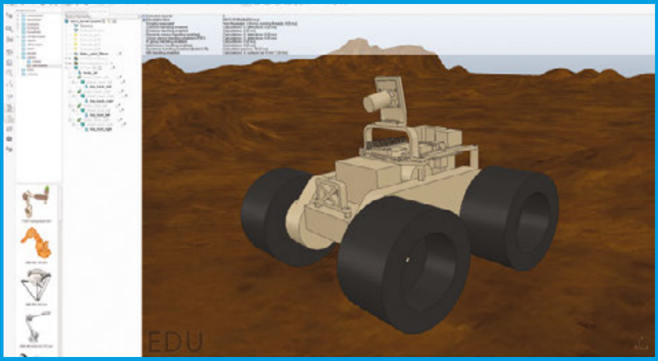
The secret sauce that powers the SDCs is the company's open source EOgmaNeo library. You can build autonomous robocars based on the EOgmaNeo library following the in-depth guides (**github.com/ogmacorp/EOgmaDrive**) that detail both the hardware assembly and the software installation for both the SDCs.

To further illustrate the unique ability of the company's SDCs, Richard says that the "Donkey Cars, and others that use neural networks, all rely on more powerful compute devices to take sensory data from the car and pass that data into more powerful hardware to train a neural network model. Those neural networks require more power-hungry computing due to the 'offline supervised learning' techniques that they use. Once their neural network model has been trained, the model learning/training is stopped and the frozen model is uploaded to the self-driving car."

CLEVERER THAN TOP GEAR

These robocars can now only use the uploaded training model to find their way around the track. The Ogma SDCs are different in that they use what Richard calls 'online unsupervised learning' techniques that help them learn and make corrections as they drive around. "The neural network model that we use can perform all the online learning and inference processing on a SBC," stresses Richard.

Despite their original objectives, boards like the Raspberry Pi and Arduino have cultivated an ecosystem that nourishes a maker's interest in advanced computing subjects like artificial intelligence. With new add-ons and sensors, and growing support from technology bigwigs, we wouldn't be surprised if the next big AI development comes from the garage of a maker like you.


Left
Karl Kangur has written a tutorial (hsmag.cc/dYjaAI) on using V-REP to simulate a robot before building it

THE ARDUINO A REALLY SPECIAL LITTLE BOARD

From an Italian student watering hole, to conquering the world

By **Jenny List**

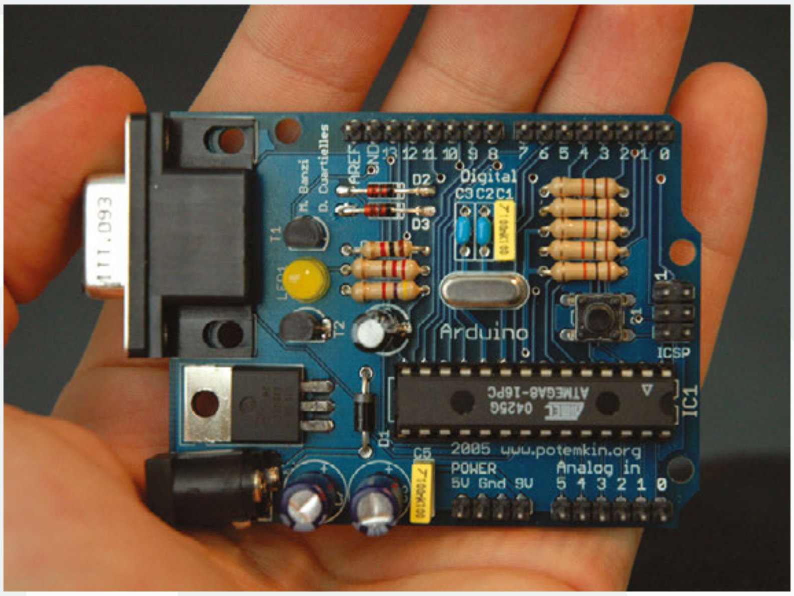
Right The Arduino R3, the latest version of the 'classic' Arduino

If you were to walk into a typical hackspace as one new to the world of making and ask for help with automating your project, the chances are you'd receive a unanimous suggestion. Use an Arduino, they'd say, and show you a small blue circuit board with a couple of rows of headers and a USB socket. Such has been the success of this board and its stablemates, that a decade plus of more able competitors haven't displaced it from its position as the go-to single-board computer for maker projects. To understand something about why that has been the case, it's worth looking back at the start of the project that spawned it.

In the early 2000s, as a reaction to the limitations of previous simple microcontroller boards, a group of Italian postgraduate students and a lecturer at the Interaction Design Institute in Ivrea came together to create the first version of what would become the Arduino project. They were lucky: they had the Wiring platform that had been the work of their fellow student Hernando Barragán to work with, a simple-to-use open-source integrated development environment and microcontroller board.

They created a fork of Wiring ported to the inexpensive Atmel ATmega line of microcontrollers, and the hardware they evolved became the ancestor of today's Arduino boards. It featured an ATmega8 microcontroller in what is now a familiar form factor with its I/O lines brought out to a set of headers designed to accept expansion boards known as 'shields'. The ease of programming from the Arduino IDE and on-chip bootloader coupled with the flexibility of the shield boards compared to the relatively small prototyping area found on previous boards proved to be a hit, and the years since have seen an ever-increasing range of official successors.

If at the moment you're wondering where the name 'Arduino' comes from, it may come as a surprise to find that a 21st-century microcontroller prototyping board has a name derived from an Italian king of the 11th century. Arduin of Ivrea was Margrave of Ivrea from about 990 until 1015, and King of Italy from 1002 until 1014. He was defeated and forced to abdicate by the German King Henry II in 1014 and would be the last King of Italy until the restoration of the Italian monarchy in the 19th century. The Arduino is a project with its origins in a university, and as with graduate students everywhere, the team had a favourite bar. Theirs was named after Arduin of Ivrea, and their project took its name from it.

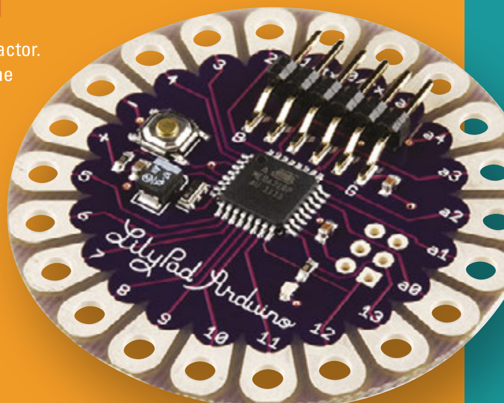


Above
An early Arduino board from 2005, featuring a serial port where today's Uno has a USB socket

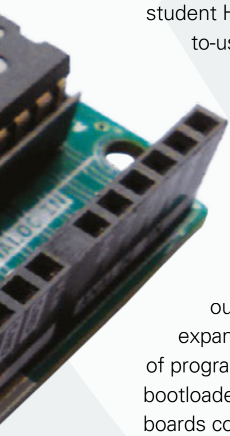
The open-source nature of the whole Arduino project has been a crucial factor in its success as a platform, meaning that instead of being a single hardware product line from one Italian source, it has become a global phenomenon with an astonishing array of products claiming some level of Arduino compatibility. In that sense, while an Arduino is generally understood as one of the official boards or a direct clone, it is safe to say that the Arduino project now transcends its origins and has instead become a platform ecosystem. When you buy an Arduino, you are not merely buying it for the board itself, because the descendants of the original 8-bit ATmega-based boards are now →

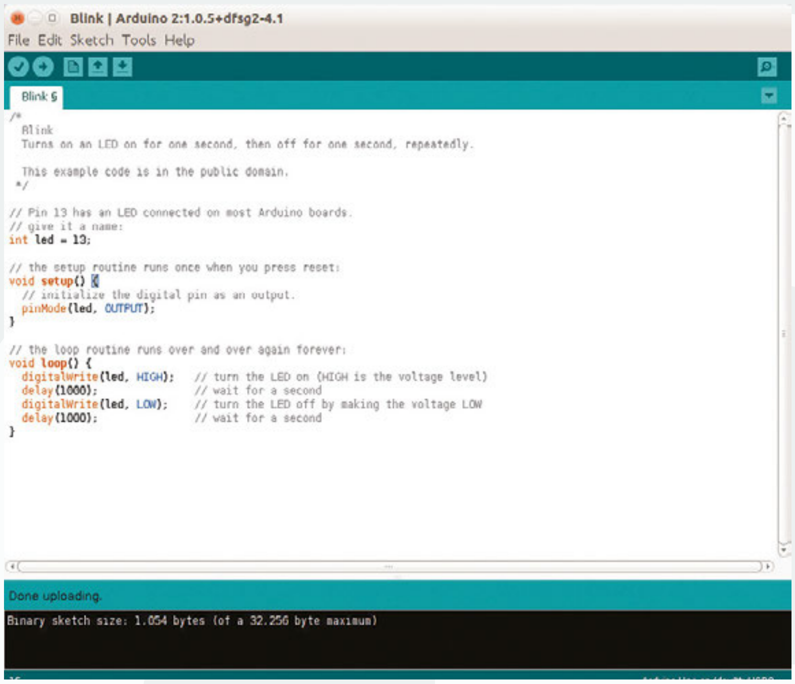
THE **LILYPAD**, AN ARDUINO FOR WEARABLES

Not all Arduino boards follow an official form factor. The Lilypad from Sparkfun for example takes the low-power version of the ATmega processor from the Uno and incorporates it into a circular printed circuit board designed for incorporation in wearable electronics. Special attention has been given to ensuring that it does not snag on fabric, and instead of header sockets for I/O lines it has large circular pads designed for sewing conductive thread to. It has even been designed to survive clothes washing.



The Lilypad's sewable contacts are ideal for conductive thread




Blink | Arduino 2:1.0.5+dfsg2-4.1
File Edit Sketch Tools Help
Blink §

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

```
Done uploading.
Binary sketch size: 1.054 bytes (of a 32.256 byte maximum)
```

Above The Arduino IDE screen

rather technically outdated. Instead you are buying into the ease of the Arduino IDE and access to the Arduino community with its huge array of software libraries and support, and it is these resources which continue to make Arduino the obvious choice for so many projects.

Aside from the official Arduino boards, you'll also see a huge number of non-official ones described as 'Arduino compatible'. These range from variants that incorporate extra hardware such as a wireless module or a motor interface, to boards with different architectures that maintain the expansion header format of an Arduino for shield compatibility. You

A CHEAP BOARD WILL WORK AND RUN YOUR COMPILED SKETCHES, BUT THE EXTRA MONEY FOR THE GENUINE ARTICLE REALLY DOES BUY QUALITY

will also find direct clones of an Arduino Uno or similar, and even outright fakes bearing sometimes comically bad reproductions of the official branding. Arduino lead Massimo Banzi has gone on record as welcoming the variant boards as bringing something extra to the project, while deploring the clones and fakes. You may find a Chinese Arduino Uno clone to be a cheap way to get started, but aside from supporting the project, when you buy a genuine board you are also guaranteeing that your board has, of course, genuine components. In 2014 the manufacturer of the USB-to-serial chip found on most Arduinos issued a driver update designed to disable counterfeit copies of its devices; among those caught were many of the cheaper non-official Arduino clones. So your cheap board will work and will run your compiled sketches, but the extra money for the genuine article really does buy quality.

CHAIRMAN OF THE BOARD

So, given that you now know some of the history behind the Arduino project and you have some idea of the ecosystem, which of the many boards should you buy, and what can you do with an Arduino?

The 'classic' Arduino, in fact the board that earns the generic title of an Arduino, is any member of the line of descendant boards from that original one with an ATmega8 microcontroller and the signature two rows of header sockets to accept shield boards. Over the years the microcontrollers have been upgraded to the ATmega168 and then ATmega328 to give more storage capacity, but broadly the same functionality has been maintained from the earliest serial-port Arduino to the latest USB-equipped Arduino Uno. This is the board that you should buy if you are new to the world of Arduino hardware, because it is the one with the most support, software, and libraries available for it. There are many clones of the Arduino Uno, but in the range of official boards it is available in variants with either a surface-mount CPU or one in a dual-in-line package mounted in a socket. Either board will give you the full Arduino experience, but of the two it's the dual-in-line version which is a little more versatile because the CPU can be removed and placed on a breadboard or other project.

Beside the Uno, there's a variety of official boards with special capabilities. The Arduino Leonardo for example looks like an Uno, but can become a fully fledged USB device in its own right, capable of emulating for instance a keyboard, or a mouse. Meanwhile, the Mega 2560 is an Arduino with a hugely expanded interface, useful for complex CNC projects. Then, beyond the ATmega line of processors is a huge array of boards that can be programmed with Arduino software. Of the official boards, there are several that use ARM processors, including the Due, which brings raw computing power to the Uno form factor, and the Yún, which has a 32-bit MIPS system-on-chip that runs a custom Linux distribution and incorporates WiFi connectivity.

If you are shopping for Arduino boards, you may also notice a variety of smaller PCBs: the Arduino

ARDUINO, GENUINO, WHAT'S ALL THAT ABOUT?

Genuino

If you are shopping around for Arduino boards, you may see genuine boards labelled 'Genuino' as well as 'Arduino'. This relates to a trademark dispute between different members of the Arduino team, with ownership of the Arduino mark in Europe resting with one camp and with another for the rest of the world. As part of the resulting legal battle, the rest-of-world trademark camp registered the name 'Genuino' for their boards when sold in Europe. The resulting confusion caused at least one retailer to stop selling the official boards entirely. Fortunately for all concerned, the competing factions were able to settle their differences earlier in 2017. So if you should find a Genuino-branded Arduino, don't worry. It's still an official Arduino, albeit one with another name on it.

Micro, Mini, and Nano. These can be available either as official boards or as clones with startlingly cheap prices, and with a set of pins fitted, can be used on a breadboard or similar. They contain the rough equivalent of a full-sized Arduino Uno on a smaller PCB, but care should be exercised by those new to the Arduino world as they are not physically compatible with Arduino shields, they are not all current designs, and different programming cables may be needed to use them.

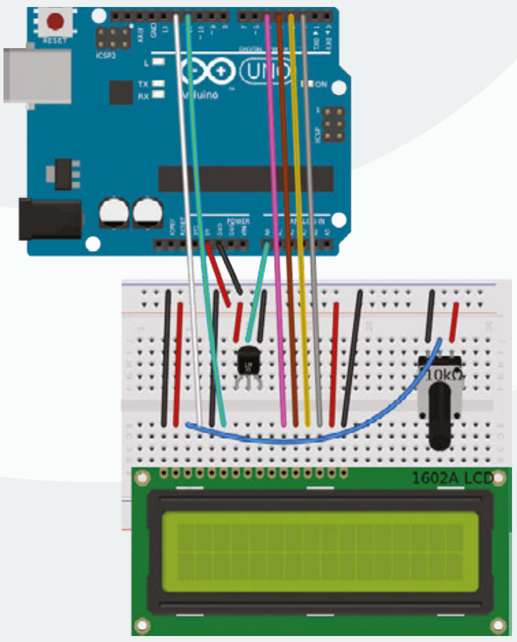
When you have your shiny new Arduino, whatever model you have bought, your next step will be to do something with it. Write some code, see it running. If you go to the Arduino website, download and install the IDE, and then plug your Arduino into your computer with the appropriate cable, you should be ready to go.

FIRST STEPS

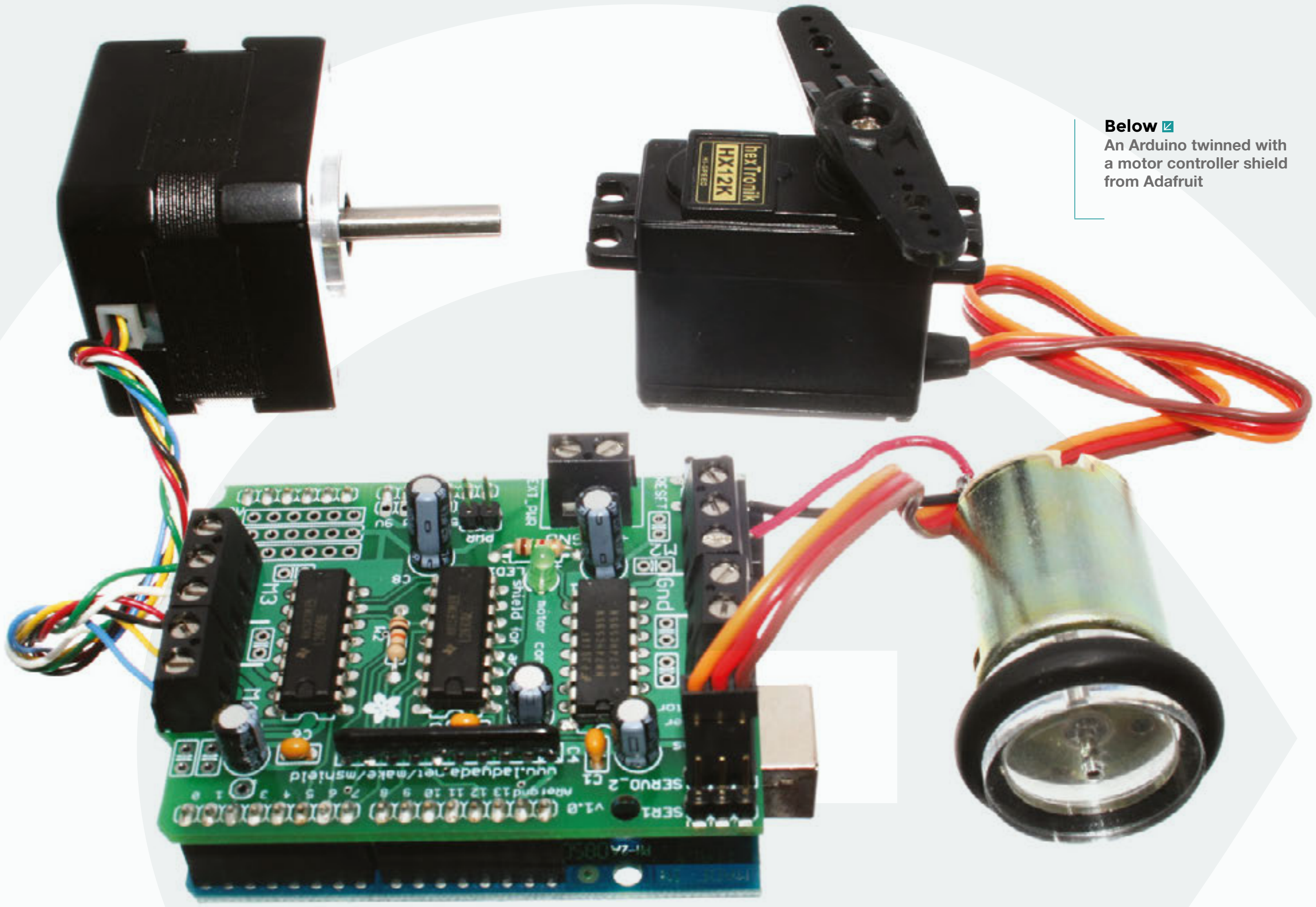
Opening the Arduino IDE puts you straight into a code editor, and if you are an experienced Arduino user, you could just start typing C code. Fortunately, in the event that you are not, the IDE ships with a library of demo code which you can load from the File menu: simply find the 'Examples' option and pick one. The easiest one to start with can be found in the 'Basics' submenu: 'Blink' simply flashes the on-board LED. If you look at the code, you'll see two main sections: **setup()**, which initialises whatever pins the code will use, and **loop()**, which contains the guts of your work and, as its name suggests, loops continuously through it. Arduino programming in depth is beyond the scope of this article, but there is plenty of information online if you are willing to spend a minute with a search engine. If you have perfected your code, or if you merely want to see the Blink example flash the LED, there are a couple more steps. The IDE must recognise the Arduino board you have, and then you must compile it to a sketch

IF YOU ARE SHOPPING FOR ARDUINO BOARDS, YOU MAY ALSO NOTICE A VARIETY OF SMALLER PCBS: THE ARDUINO MICRO, MINI, AND NANO

and upload it to the hardware. To perform the first step, on the 'Tools' menu select your board model from the 'Board' submenu, and then select the port it is plugged into on the 'Serial port' submenu. Then, to see it running on the board, select 'Upload' from the 'File' menu. You will see some status messages scroll past at the bottom of the IDE window; then, if all is well, the on-board LED will start to flash at one-second intervals. Congratulations, you have just run your first Arduino sketch! →



Left An Arduino thermometer with LM35 sensor and LCD display



Below
An Arduino twinned with a motor controller shield from Adafruit

Of course, while flashing an LED makes a good simple demo of an Arduino it's obvious that the boards are capable of so much more. If you are looking for ideas for what to connect to your board, there is a huge range of shields and breakout boards on the market, often complete with software libraries and example sketches of their own. Companies such as Sparkfun and Adafruit maintain extensive catalogues, if the official range of Arduino shields were not enough.

THE ARDUINO MEGA 2560, AN ARDUINO WITH MORE I/O LINES

There are plenty of applications for which the Uno form factor can not provide enough connectivity. For example, a large robotic or CNC device will have multiple sensors and servos to connect to it. To address this problem, there is a range of Arduino boards with many more I/O lines and a much larger shield form factor to accommodate them. The Arduino Mega 2560 uses an ATmega2560 microcontroller with all those extra lines.

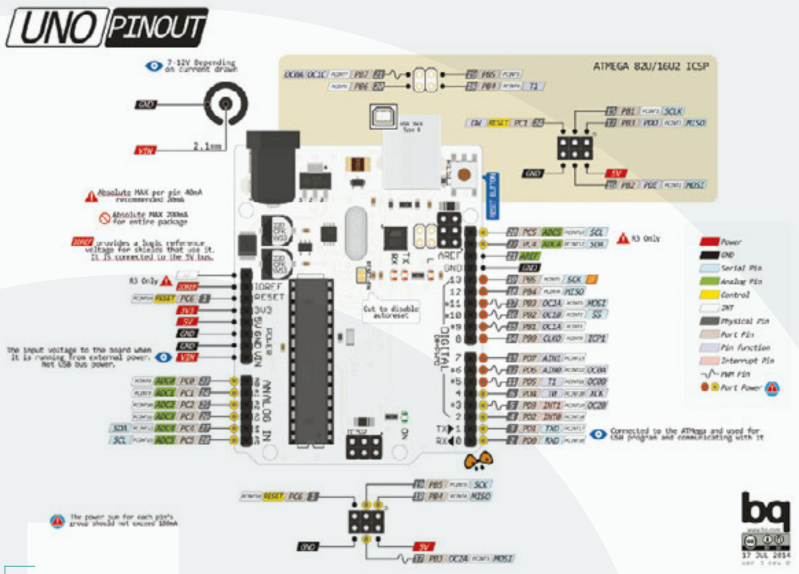
GETTING SENSITIVE

A common Arduino project, once a first-time user has moved on from flashing LEDs, is to interface one to the real world by reading from a sensor. And since temperature sensors are among the most readily available and easy to interface with, many people create Arduino thermometers and thermostats. The simplest of these temperature sensors are three terminal chips which take a 5 volt power supply from the Arduino and return a voltage on their other terminal proportional to their temperature. This can be connected to one of the Arduino's analogue input pins and read using the Arduino's `analogRead()` function for your code to calculate a reading from. Since any measurement project including a thermometer is relatively useless without a display, you will also find a lot of Arduino projects featuring LCD displays. The commonly available LCD displays using the venerable Hitachi HD44780 driver chip will interface directly with an Arduino, and can easily be controlled using the LiquidCrystal Arduino library. The example on page 41 shows an Arduino wired to an HD44780 display and an LM35 three-terminal temperature sensor to make a complete thermometer project.

Of course, the uses for an Arduino extend far beyond a simple LED flasher or a thermometer, and are far too numerous to cover in this article. But it's worth looking at one more common use for an Arduino: its ability to produce a pulse-width-modulated (PWM) output allows it to control servos, and motors when paired with a suitable shield. You'll find motor controller shields from multiple manufacturers; one of the most commonly found comes from Adafruit. And once you have a motor shield, you can bring it to a host of robotic projects, from simple two-wheeled crawlers such as the Ardubot, to fully functioning robot arms such as the inexpensive and easy-to-build MeArm.

It has been over ten years since the Arduino first saw the light of day, and the mainstay of the hardware range is now based on very outdated hardware when compared to more recent competitors such as the Raspberry Pi or the BeagleBone. This might be the cue for it to fade into insignificance, as has been the case for other darlings of the electronics hobbyist world, but for two things. First, the Uno and its ancestors may use old chips, but they still do the same job that they did in 2005 very well, namely providing an extremely accessible microcontroller platform for beginners and experts alike. Secondly, the Arduino project is not simply a collection of boards. The open-source IDE and bootloader model has outgrown the hardware, and it is this that makes Arduino special in 2017. Whenever a new and exciting processor or board enters the market, it will always be only a matter of time before someone brings it to the Arduino environment.

If those Italian postgraduates had never met in a bar all those years ago and Arduino had never happened, it's certain that we would still have had affordable microcontroller boards. What is in doubt is whether we would have had so many that are as accessible through the same open-source development environment. Instead we would probably have had a plethora of competing incompatible IDEs from board and chip manufacturers, many of which would have been difficult to use, or closed-source and proprietary. If you were that hackspace newbie we mentioned at the start, it's likely that whichever board you would have used would have had a much steeper learning curve, and if you weren't extremely dedicated to your project, you might have given up. It's fortunate then that we have the Arduino, and that its story is far from over. The future will see an ever-expanding range of boards bearing the name, with as-yet-undreamed-of capabilities. So if you've never used one, we hope you'll now have the confidence of knowing a bit more about the project to help you pick up your first Arduino and work with the platform as it further evolves. □



Above
The pinout diagram of an Arduino Uno R3

ARDUINO TIMELINE

1997	Atmel brings the first AVR microcontroller product to market.
2001	Processing computer language/IDE developed by Casey Reas and Benjamin Fry, at the MIT Media Lab.
2003	Wiring development platform developed by Hernando Barragán at the Interaction Design Institute Ivrea, Italy.
2003	Wiring ported to Atmel AVR8 processor architecture by Massimo Banzi, David Mellis, and David Cuartielles, again at the Interaction Design Institute Ivrea, Italy. This becomes a fork of the Wiring project, to be named Arduino.
2005	The first boards in the recognisable Arduino form factor are produced featuring an ATmega8 processor, after a succession of other prototypes.
2006	The Arduino NG gains an ATmega168 processor, featuring 16kB of storage.
2008	The tiny Arduino Nano appears, and the Duemilanove gains an ATmega328 with 32kB of storage.
2010	The 'classic' ATmega328 Arduino becomes the Arduino Uno.
2011	The Leonardo, boasting USB device capability, is launched.
2012	The 32-bit ARM Cortex M3-based Arduino Due is launched.
2013	The Arduino Yún is launched, featuring a MIPS processor and a custom Linux distribution.
2014	Beginning of split between Arduino LLC and Arduino SRL, leading eventually to creation of Genuino brand.
2015	Arduino/Genuino 101 launched, based upon the Intel Curie microcontroller.
2016	Reconciliation and merger between the two rival Arduino companies.
2016	Launch of Arduino MKR1000, featuring a 32-bit ARM Cortex M0 and on-board WiFi.

We Learn WELDING

From beginners to slightly burnt beginners: learn from our mistakes as we take on a new skill

By Ben Everard

Hacking things isn't so much a skill as the intersection of a lot of skills. You may have to be able to design things, do some coding, solder bits together, and build something to hold it all together. The more skills you have, the wider the range of projects you can take on.

Personally, I have a strong background in computing. I can whip up code to solve most problems I encounter. Need a virtual machine set up and managed to handle the back end of a task? I'm your man. I can have a decent crack at digital electronics, and have a large stack of boards that I've soldered together over the years. However, when it comes to physically building stuff, my skills leave a bit to be desired. When pressed, I can assemble something out of wood that serves the purpose. When it comes to metalwork… well, let's just say I have to start from scratch. I'm always keen to find new ways of expanding my repertoire, so I set out to see how far I could get learning to weld in a day. To be more specific, can I learn enough welding to make a small stick man from a metal bar? Equipped with only rudimentary equipment (an AC welder and appropriate safety equipment) and a little instruction, I set about the task.

Electric 'stick' welding is, in principle, a simple thing. You connect one bit of metal to a power

Below Our very first attempt to control the arc. You can tell from the welding splatter and occasional balling that we still need some practice

Left
By the end of the day, my joints still had some problems, but were strong enough for many purposes

LEARNING TO WELD

Basic welding isn't hard, but it does require quite a bit of kit that can be a little expensive (expect to pay around £100 for a basic welding setup), and it's best used (at first) with some guidance from someone who knows what they're doing. Vocational schools often have short courses to help you learn the basics, or you might find a willing instructor in your local hack/hacker/maker space. Some vocational schools will let you use their equipment after you've gone through the training, which can be easier and cheaper than trying to set up on your own.

> A few sparks splutter out and the welding rod is stuck firm to the surface. A few wiggles pull it free, **but it's a fairly unimpressive start**

supply capable of creating a large current, and attach the other side of the power supply to a flux-covered rod known as an electrode. Hold an electrode close to the metal you want to join and a spark arcs between the two. This arc is hot enough to melt both the metal you're welding and the electrode. This all pools together to form a cohesive mass that forms the joint you're welding. The first step, then, is to learn to create this arc.

The beginner's approach to this is to run the electrode along the metal briefly, then just move it away slightly – an action not unlike striking a match. I have a plate of scrap metal clamped to the bench in front of me on which to practise striking an arc. The view through the welding mask is completely black at first, so I line up the electrode with the mask up. I flip the mask down and strike. Nothing. I try again. Nothing. A few sparks splutter out and the welding rod is stuck firm to the surface. A few wiggles pull it free, but it's a fairly unimpressive start.

WELDING VS SOLDERING

When thinking about welding, it's sometimes useful to compare it to something it's not, such as soldering. When you solder a joint, you heat up the things to be joined, then add a filler metal (the solder) which melts into the joint. When it cools down, the solder hardens and everything is held together.

The difference between this and welding is that at the end of it, there are still three distinct things: the two objects being joined and the solder. The joint holds together because it's all stuck together, but they're still different things and could be removed from one another.

When you weld, you melt all three things and mix the resultant pool of molten metal. As such, there's no clear line between one thing and the other, they just blend together. There's no such thing as unwelding (as there is with unsoldering). You can cut the joint, but you can't separate out the constituent parts.

HELLO WELD!

Frustrated, I run the electrode along the surface again. There are a few sparks, and this time, as I pull the electrode away, there's a dull yellow glow emanating from the gap between the rod and the metal plate. In my excitement, I pull away and the arc dies almost as soon as it was created. It's not a great arc, but it was definitely an arc. After a few more tries, I can create an arc, if not consistently, at least regularly.

Just as generations of programmers have started coding by getting the computer to utter 'Hello World', so novice welders often start by guiding the arc around in their name. This gets us used to not just holding an arc, but also manoeuvring it through the dull, almost black world that we view through the welding mask.

I line up, flick my mask down, and begin the B. The arc melts both the plate of metal the arc hits and the electrode. As the electrode melts, it's deposited on the plate and I leave a ridge shaped in the letters of my name. I flick my mask up as the tail of the N is still glowing slightly from the heat. Initially the letters are black and slightly crusty, but a swift bang from the hammer and scour →

with a metal brush removes the burnt flux, leaving just the letters raised up in metal.

That's not really welding, it's just depositing metal, but I feel hugely satisfied by the feeling that metal – which in my imagination is solid and immutable – has bent to my will.

Stage one of learning to weld is done, it's now time to join some bits of metal together.

In principle, this is simple. Using exactly the same technique I used to write my name on the metal, I need to melt both sides of the joint and deposit a little metal into the gap between them. If all goes to plan, this should cool and solidify to a single piece of metal with the three parts mixed together in the middle.

LET'S BUILD SOMETHING

Mask down, arc struck, I begin to move down the joint. Immediately I see where the skill in welding comes in. The instant the arc is established, the metal starts to melt. Move it too soon and nothing's melted enough to stick together. Leave it too long and it's too melted, leaving a hole in the metal. I need to carefully glide it down, gently moving the electrode between the two surfaces to be joined as it travels down the seam. Of course, I fail spectacularly at that. I move in jerky motions, and try to cheat by moving back up the seam to a part where I'd moved too fast.

The welding rods are coated in flux that protects the joint from oxidisation (it serves a similar purpose to the gas in MIG and TIG welding – see

OTHER TYPES OF WELDING

We tried to learn AC stick welding. Equipment-wise at least, this is the simplest form of welding. You just have a coil that converts 240 V (or whatever your local mains voltage is) to a lower voltage but a much higher current than is usually available through a socket (it can be well over 100 amps). The welding electrode is surrounded by flux that keeps the joint clean, but can also cause problems. It's a bit of a rough-and-ready type of welding that can be great for hackers but isn't always the best choice. Here are some other options:

DC – Stick

Similar to the AC welding that we did, but this time using DC electricity. The equipment needed is slightly more complex, but the general process is exactly the same as with AC.

MIG

Metal Inert Gas welding also uses an electric arc to melt the metal, but rather than flux, it uses an inert gas (such as argon, carbon dioxide or helium) to protect the weld, which can result in a neater join. The filler rod is automatically fed into the joint as you weld.

TIG

Tungsten Inert Gas welding also uses an inert gas to protect the joint, but unlike MIG welding, the filler rod isn't automatically fed in and it's up to the welder to apply this as and when it's needed. TIG welding is the most versatile form of welding, but it's also the slowest and most difficult to learn.

Spot

Very easy to do, but limited in what it can achieve. Spot welding uses two electrodes close together (typically on opposite sides of the joint) to create a point of heat as well as pressure to hold the objects in place. There's no filler rod, and the two objects are just melted and pressed together. This is usually used to join two flat surfaces.

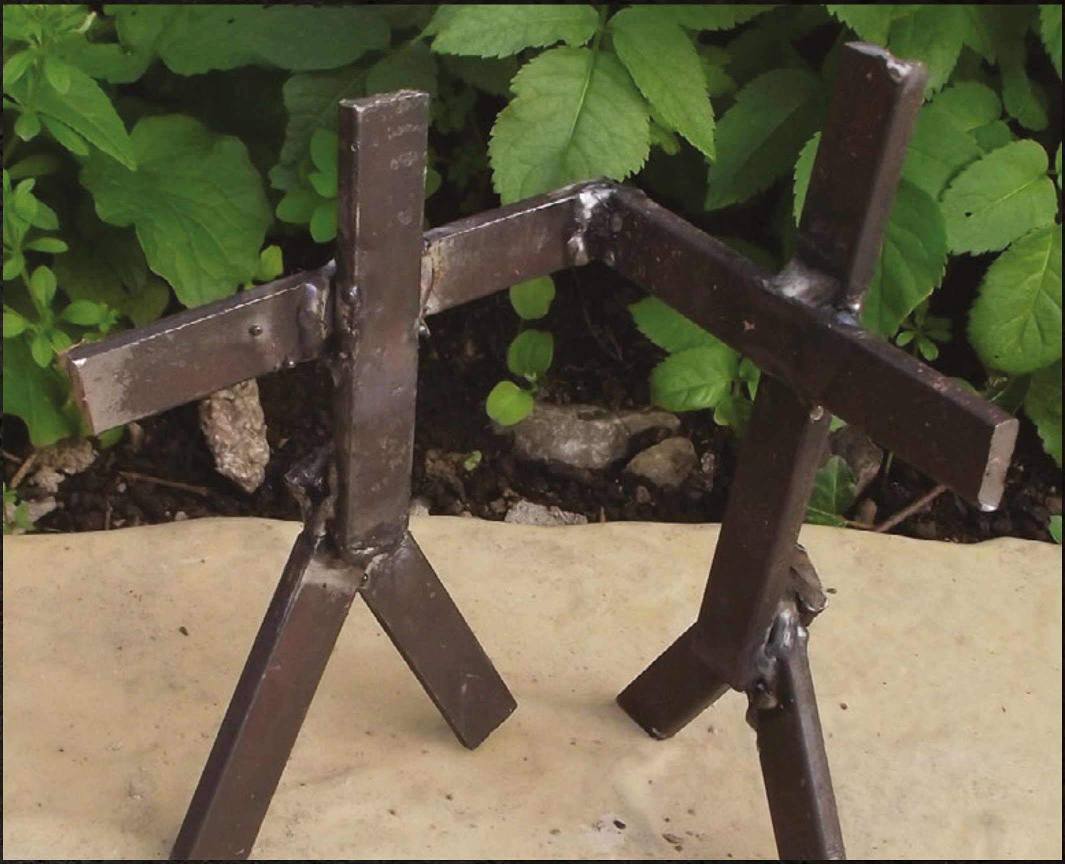
Oxyacetylene

Unlike the others we've covered, this one uses gas to heat up the metal. A combination of oxygen and acetylene is burned in a welding torch which melts metal. A filler rod can then be used where necessary.

Brazing

This one isn't really welding, but more like high-strength soldering: you heat up the bits of metal you want to join and use this heat to melt a brazing rod into the joint.

Below My equipment for the day was an aging Weldmate that used to belong to my grandfather. Basic equipment is fine for basic welding



Right ↗
The finished product. Metal sculptors may not be fearing the competition just yet, but I'm proud of it

boxout). This should melt and rise to the surface of the joint; however, if you move back up the weld, this flux gets embedded in the joint and you get a stick-shaped hole.

Ugly welding is acceptable, though. The question is, how strong is it? Time to give it a bend and see if I can break it.

Arrghnnn!

The freshly welded metal holds quite a bit of heat and even through hefty gloves, it singes my fingers. Only slightly though. After waiting another minute for it to cool down, I try to snap the joint. My first joint is ugly, but surprisingly strong. Despite it looking like someone tried to nail-gun two bits of jelly together, I can't break the joint with my hands, so I'm chalking that up as a win.

We wanted to find out if you could learn to weld in a day, and for that we needed a test. For us, it was whether or not we could weld together a simple stick-man sculpture (Maybe sculpture is a little too grand a term, but you get the idea).

The welding here is harder as it's at strange angles, and the clamps holding things in place can get in the way a little, but it's all the same basic process: strike an arc, position the arc in the seam, and move the arc to blend the metal together to create a solid joint. Some are, ahem, more successful than others, but the end result, after only a few hours from the first time I held a welding rod, is a solid structure.

The essence of hacking is expanding your skills range, and welding is a great area to move into. It doesn't take too much time to grasp the basics and even basic skills can be useful. I can't claim to be a competent welder, but I do feel that I now have a new skill that I can bring to bear on things I make. It might be a while before I'm doing anything load-bearing or which has to look good, but a custom robot frame or a jig to hold things in place is now within my repertoire – just. □

HACK YOUR OWN WELDER

In essence, an AC arc welder is just something that can supply a lot of current (generally a minimum of 50 A). There's nothing particularly complicated in this, and we've seen plenty of home-made welding setups, both by using coils to increase the current in mains power or by amalgamating enough batteries to supply current.

It's a fascinating project if (and this is a big if) you have the skill and experience to do it safely. Remember that you need to create enough current to melt steel. That means that there's enough current to do a lot of damage to just about anything that gets in its way, including a human.

One thing that you should never attempt to make yourself is goggles. Eye damage in welding comes from ultraviolet light, so it can be hard to know if protection is adequate until after any damage has been done. Look after your eyes – invest in good quality goggles.

Ultimate Overclocking

CRAZY STUFF

Done by
Kingpincooling.com

How much extra speed can we wring out of a Raspberry Pi Model B?

Overclocking can take many forms and shapes, from a useful everyday boost of a few hundred MHz (think of tuning a car engine to get better acceleration performance) to insane cryo-cooled rigs for international OC competitions (drag-racing cars with jet engines, no less). Overclocking is not limited just to personal computers: the same concept applies well for nearly any digital system, including mobile processors and embedded SOC (system-on-chip) devices, such as the Raspberry Pi 3.

One of the common ways to get performance gains is by increasing the running frequency of the processor, memory and storage interface. More frequency = faster computation. That comes at a price: increased power consumption and possibly, reducing stability, as an erroneous operation is more likely under stress.

OVERCLOCKING THE PI

Where do we start? First, we study the overall system design: what components are present on computer's PCBA and how we can increase their performance.

The heart of the Pi is the Broadcom BCM2837 SoC, which has a quad-core Cortex-A53 ARM processor and a VideoCore IV GPU.

A separate 1GB LPDDR2 SDRAM memory chip – EDB8132B4PB-8D-F from Elpida (Micron) – is located on the underside. Memory is allocated dynamically between CPU and GPU use, depending on the settings in the raspi-config tool.

Connectivity is provided by the Broadcom BCM43438 WiFi/Bluetooth chip and SMSC LAN9514 USB/LAN hub. This is only important to us because we need to keep these chips alive during overclocking so we can keep communicating with the Pi.

Now it's time to check the nominal clocks and voltages for each of the components. The nominal max CPU clock speed is 1200MHz with active power management; the nominal max GPU 3D- core clock is 300MHz (200MHz for the 2D clock); and the LPDDR2 memory clock is 400MHz (800 MT/s), 2.5ns cycle time.

All clocks are generated internally by the PLL section of the SoC, which takes the 19.2MHz input clock from a tiny oscillator on the back of the board and multiplies that to get a higher frequency.

The Pi takes all its power from a single input – micro-USB connector 'PWR IN' – and has on-board regulators to generate low voltages required by the CPU, GPU, LPDDR2 memory, and peripherals. It's important to know what these voltages are, as higher clocks may require higher voltages to maintain stability and error-free operation.

Now, with this knowledge about Pi 3 clocks and voltages, the first step would be to install some benchmarks to establish the baseline performance, after which we can try to use existing knobs to increase clocks/voltages to compare how much performance difference we gain. It's important to test actual performance, not just reported MHz speed, because if the CPU overheats it could reduce actual running clock to lower values and we'd get worse performance than expected, often even lower than a non-overclocked result. The very same methods apply to 'traditional' PC overclocking and benchmarking.

An important note on the actual used sample results: due to variation in manufacturing processes, every piece of silicon, be it processor core or memory, has a different margin (overclocking ability before data gets corrupted). This means that theoretical chip A may overclock to 1400MHz, but chip B of the same model and in the same condition could reach only 1350MHz. Chip C on the other hand may be capable of running 1450MHz. To test this in practice, we'll use not just one Pi 3 module, but five of them to find the best specimen!

Table 1
On-board voltage topology with used controllers spec

VOLTAGE	SOURCE	USAGE	CONTROLLER	NOTE
+5 VDC Input	External PSU	Main power input	Can be alternatively sourced from pin header J8	
+3.3 VDC	+5 VDC	Main I/O voltage	Diodes PAM2306	Switching VREG Channel 1
+1.8 VDC	+5 VDC	DRAM/CPU/ GPU voltage	Diodes PAM2306	Switching VREG Channel 2
PLL, +3.3 VDC| SOC LDO	Internal PLL voltage	Internal from BCM2837	Linear regulator for PLL clock power	
CPU, +1.0 VDC| +5 VDC	Main CPU voltage	Richtek RT8088A	Switching VREG	

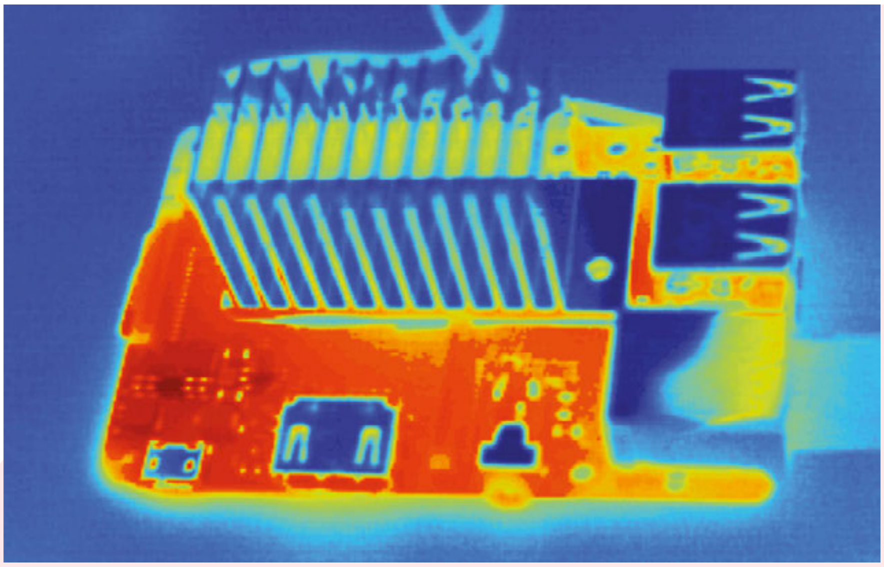
FREQUENCY AND VOLTAGE CONTROL SETTINGS

To adjust clocks and voltages in a traditional computer, you have a special BIOS Setup interface. Boot settings, low-level device configurations, various memory settings, and power management settings are often available in the PC BIOS. The Pi has much less room and power in its internal bootloader, so actual overclocking settings, like clocks, are set by a special kernel configuration file, located on the FAT section of the Pi's SD card: **/boot/config.txt**.

All these clocks are separate with their own clock generation, so they can be adjusted independently. There are also a few additional settings that can be tweaked.

- **arm_freq_min** – Minimum value of arm_freq used in low power state. Default is 600 for Pi 3.
- **core_freq_min** – Minimum value of core_freq used in low power state. Default is 250 for Pi 3.
- **sdram_freq_min** – Minimum value of sdram_freq used in low power state. Default is 400 for Pi 3.
- **temp_limit** – Thermal limit protection threshold. Sets clocks/voltages to default once reached. Default limit is 85°C.
- **sdram_schmoo=0x02000020** – Memory training tweak.
- **over_voltage** - Processor logic voltage offset, in 25mV/bit steps. Allowed range from -16 to 8 (8 * 25 = 1.200 VDD_CORE(DEFAULT) + 0.2 = 1.400V).
- **over_voltage_sdram_p** – memory cell level voltage offset. Same range maths as over_voltage.
- **over_voltage_sdram_i** – memory I/O voltage offset. Same range maths as over_voltage.
- **over_voltage_sdram_c** – memory logic level voltage offset. Same range maths as over_voltage.
- **force_turbo** – Disable dynamic low power states for RPI SoC. This setting also voids your warranty.
- **boot_delay** – Some owners have reported this to be helpful in the case of SD card data corruption when used with force_turbo.

Warning: Overriding the temp_limit and force_turbo settings will void the warranty on a Raspberry Pi 3 computer. →



Left The thermal camera shows the impact of an aluminium heat-sink

Let's download some benchmarks and run them to see if we can get the Pi to go fast, using these adjustment knobs.

Due to a PLL maximum clock range limitation at 3200MHz on the Pi 3, there is no known way of pushing real CPU frequency past 1600MHz. If one were to configure a higher value, it would result in the processor running with a much lower clock speed, as shown by a simple performance test:

```
--------- HWBOT Prime 0.8.3 ----------
Processor detected:
ARMv7 Processor rev 4 (v7l) BCM2835
Estimating speed... 4x 1,650MHz @ -86.187
C
976 MB memory
Running benchmark using 4 threads.
Starting benchmark...
Warm up phase:  ......................
Benchmark phase: ......................
All done! Current CPU temperature:
-84.035 C
Score: 304.56.
```

The score here should be around 520, if the clock 1650MHz were correct, but it's even 44% slower than the nominal 1200MHz clock – about equal to half of the desired frequency, 825MHz CPU clock.

BENCHMARK SOFTWARE SETUP

First, download the latest Raspbian Stretch image. For all testing presented here, we used the version from 7 September 2017 with kernel 4.9.41.

```
Linux rpi-oc1 4.9.41-v7+ #1023 SMP Tue
Aug 8 16:00:15 BST 2017 armv7l GNU/Linux
```

It's worth setting the CPU power governor to performance mode, to favour less switching from idle state to full-performance state:

```
echo "performance" | sudo tee /sys/
devices/system/cpu/cpu0/cpufreq/scaling_
governor
```

A handy script to measure current CPU frequency and report die temperature:

```
root@rpi-oc1:/home/pi# cat ./check_cpu_
speed_temp.sh
cat /sys/devices/system/cpu/cpu0/
cpufreq/scaling_cur_freq
/opt/vc/bin/vcgencmd measure_temp
```

It is also working well to report negative temperatures (below 0 °C), which is very handy for extreme overclocking.

```
root@rpi-oc1:/home/pi# ./check_cpu_
speed_temp.sh
600000
temp=-35.1'C
```

The first number shown is the CPU frequency; the second, **temp** value is the SoC thermal sensor reading. The frequency drops to 600MHz when the Pi 3 is idle, to save energy, but will jump to the max value once a workload is running. Now we're ready to install the benchmarks.

HWBOT PRIME BENCHMARK

There is also an overclocking guide for HWBot Prime on the **HWBOT.org** website, which we can use to measure performance in prime numbers calculation. To set up the benchmark, download the JAR file:

```
wget http://downloads.hwbot.org/
hwbotprime.jar
```

To execute this Java-based benchmark, we need to pre-install the openJDK from **openjdk.java.net**.

```
apt-get install openjdk-8-jre
```

Starting the benchmark is just a simple call for .jar from a Java environment:

```
java -jar hwbotprime.jar
--------- HWBOT Prime 0.8.3 ----------
Processor detected:
ARMv7 Processor rev 4 (v7l) BCM2835
Estimating speed... 4x 1,200MHz @ 61.224 C
970 MB memory
Running benchmark using 4 threads.
Starting benchmark...
Warm up phase: ................  done!
Benchmark phase:  .............  done!
All done! Current CPU temperature: 72.522

C
Score: 440.96.
```

Here, our example score is **440.96**. The utility also reports CPU temperature, which can be handy for checks.

Note that the OpenJDK library build version can have a very big impact on the score, so test results can be compared only when using the same package versions. You may see many faster results online, obtained using a different JDK version.

It's a good idea to run the benchmark multiple times to make sure that scores are consistent. Some small variation, of a few percent, is normal.

If the module is unstable, you can get random locks or kernel panic messages,

Clock domain	Parameter in / boot/config.txt	Minimum	Default	Maximum
CPU/logic clock	arm_freq	100	1200	1600
2D GPU/L2 cache clock	core_freq	250	400	600+
Video decoder clock	h264_freq		300	
Imaging pipeline clock	isp_freq		300	
3D engine clock	v3d_freq		300	
LPDDR2 memory clock	sdram_freq	200	450	600+

Table 2 Available clocks and kernel parameters for adjustment

or just bad scores. A few examples are presented in the logs below:

```
Message from syslogd@rpi-oc1 at Oct 25

17:41:40 ...
kernel:[   97.266669] 7fe0: 62442dfc
62442e08 00000000 76f19950 80000010
63208a94 55555d80 55555547
Message from syslogd@rpi-oc1 at Oct 25
17:41:40 ...
kernel:[   97.296319] Code: 0a00000a
f57ff05b e2853028 f593f000 (e1932f9f)
.............................. done!
All done! Current CPU temperature: 9.576 C

Score: 260.77.
```

In this case, the score is half what it's supposed to be. Often tests just crash due to processor instability.

SYSBENCH BENCHMARK

The benchmark utility sysbench (**wiki.gentoo.org/wiki/Sysbench**) allows you to benchmark processor, memory, file I/O, and mutex performance on Linux platforms. It runs in a command-line interface as a console tool. To install this benchmark in the Raspbian OS, we use the apt-get tool:

```
apt-get install sysbench
```

Once installation is successful, it can be executed using a single command with the desired test parameters. In this article, this benchmark will be used to test memory speed. Our test will allocate a memory buffer and then read/write from it. This is then repeated until the provided volume (–memory-total-size) is reached. Users can provide multiple threads (–num-threads), different sizes in buffer (–memory-block-size) and the type of requests (read or write, sequential or random).

```
sysbench --test=memory --num-threads=4
--memory-access-mode=rnd --memory-total-
size=800M run
sysbench 0.4.12:  multi-threaded system
evaluation benchmark
Running the test with following options:
Number of threads: 4
Doing memory operations speed test
Memory block size: 1K
Memory transfer size: 800M
Memory operations type: write
Memory scope type: global
Threads started!
Done.
Operations performed: 819200 (2263497.25
ops/sec)

800.00 MB transferred (2210.45 MB/sec)
```

For comparison reasons, it's important to keep the same settings across the benchmark, so we know we're comparing apples with apples.

OPENQUAKE GRAPHICS TEST

The graphics core in a Raspberry Pi is powerful enough to run a special version of the famous Quake 3 FPS! So we can use it to benchmark combined processor and graphics core performance.

```
wget http://www.berryterminal.com/dl/
ioquake3_99.1.36-rpi01_armhf.deb
sudo dpkg -i ./ioquake3_99.1.36-rpi01_
armhf.deb
sudo apt-get install openarena
sudo apt-get clean
cp /opt/vc/lib/libbrcmEGL.so /lib/libEGL.
so
cp /opt/vc/lib/libbrcmGLESv2.so /lib/

libGLESv2.so
```

Make sure you have set a GPU memory size of at least 224 MB in raspi-config, otherwise the game won't start.

To start the benchmark, just run **/usr/games/openarena**.

The Raspberry Pi 3 gets rather hot running bare metal, without any heat-sinks in still air. A few thermal images taken with a Fluke Ti32 camera reveal temperature gradients well. The memory does not get hot at all, barely differing from the board surface temperature. However, the Broadcom SoC runs around +47°C at idle, going up to a scorching +75°C under full load.

Based on these images, there is no need to have dedicated heat-sinks for the memory chip, as it would be cooled from PCB thermal conduction once we get main the SoC colder.

OVERCLOCKING RESULTS

Our initial check was to see what max frequency the Pi can boot into console. To perform all CPU and memory speed benchmarks, a plain headless configuration was used. That means the Pi was connected to a network over the Ethernet port, with sshd running to provide access to the console remotely.

To avoid limitations from power supply input, a high-end EVGA NEX 1500W PSU was used as a power source, which →

Benchmark test	CPU Frequency	GPU/L2 Frequency	DRAM Frequency	Result	Temperature
HWBot Prime	Default	Default	Default	440	+69.3
Sysbench memory 800MB	Default	Default	Default	2025.5 MB/sec	+49.3

Table 3
Baseline benchmark results without overclocking

can supply a serious 25 A on +5V output. Measured voltage was +5.120 V at the Pi pins. The connection between the Pi and PSU was made using a short cable with AWG18 wires.

Now we know which Pi is the best, we can take that one for a full modification and cooling workout. So all further testing was performed on the promising unit #4. The thermal image of a Pi overclocked to 1500MHz reveals a hot spot at +92°C! With default throttling settings, that is 7°C over the throttle limit temperature, when the CPU speed drops to reduce stress.

IMPROVING COOLING – AIR

During typical operation without overclocking, the Raspberry Pi 3 does not require special heat-sinks or additional cooling. However, with the overclocked settings, especially with increased voltage for the processor, it will get too hot for reliable operation.

Attaching a simple aluminium-finned heat-sink and additional airflow from a fan can provide much better thermal conditions, securing better stability and overclocking headroom. The design of the PCB is quite friendly to this simple modification, as there are no tall components in close proximity to the processor. A thin sticky thermal pad for the heat-sink attachment will do the job.

With this simple heat-sink treatment, our Pi was able to run around 1500MHz in loops, stable enough to pass any performance benchmarks multiple times.

Thermals are now much better, with the SoC area reduced to around +57 °C, instead of over +90 °C.

Our best score in the HWBot Prime benchmark was around 15% faster, and the memory benchmark yielded a 26% performance increase. Can it go further?

VOLTAGE MODIFICATION

To improve stability under extreme operation with a 1600MHz processor clock, VDD_VCORE voltage was supplied externally from an EVGA EPOWER V module, programmed to 1.500 V. The EVGA EPOWER V can supply up to 2.000 V, which is plenty of headroom for our purposes.

This way we are also not limited by the over_voltage range 1.400 V maximum limit (setting 8) and can apply arbitrary high voltages. To connect an external power source, you'll need to hook thick enough wire to the C163 positive terminal. It's easy to spot by looking at the connection with a little power inductor. This is confirmed by the schematic section as well.

The external source is connected to this point by AWG18 wire. Since nominal current is barely a few amps, just one wire would work well enough. Also, return ground wire is important, so we've used a large HDMI connector body to get a low-resistance ground connection.

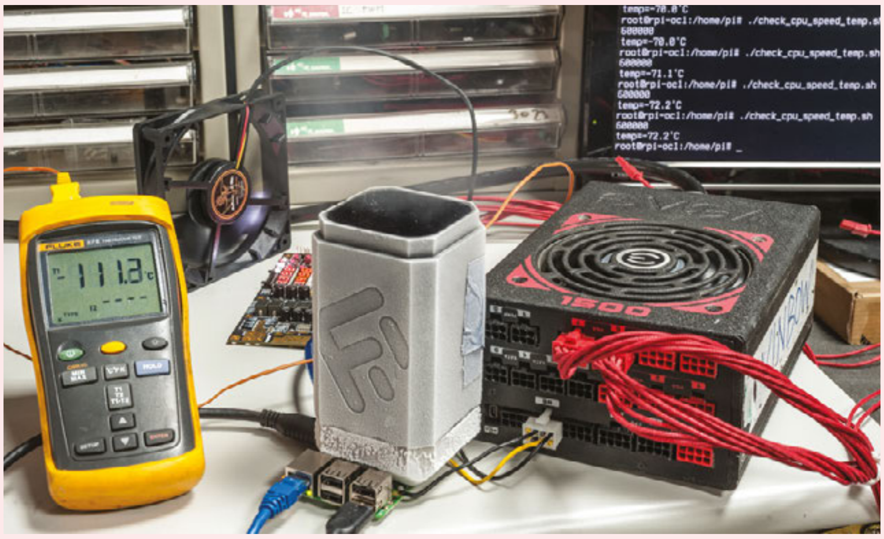
Another benefit of using an external core voltage supply is that this voltage is not controlled by the Pi's dynamic power management, so we will have constant and stable voltage on the rail, no matter whether the CPU is idle or busy crunching numbers.

EXTREME COOLING – LIQUID NITROGEN

Now the heat-sink was replaced with a massive Kingpincooling.com F1 extreme cooling evaporator block. To make it fit the Raspberry Pi, we had to remove the J8 pin header and the J3 and J4 FPC and the J7 A/V connector. We also coated both sides of the PCBA with petroleum jelly to avoid water condensation and ice shorting components on the board.

Thermal grease was applied on top of the CPU and heavy copper was just standing on top of it. The bottom side was supported by a small rubber mat to keep everything flat and steady.

The benefit of using a massive copper block for LN2 cooling instead of a smaller tube is the thermal response time of such a system. It will take minutes for a small CPU to warm such a large block of cold copper,


```
temp=-70.0'C
root@rpi-oc1:/home/pi# ./check_cpu_speed_temp.sh
600000
temp=-70.0'C
root@rpi-oc1:/home/pi# ./check_cpu_speed_temp.sh
600000
temp=-71.1'C
root@rpi-oc1:/home/pi# ./check_cpu_speed_temp.sh
600000
temp=-72.2'C
root@rpi-oc1:/home/pi# ./check_cpu_speed_temp.sh
600000
temp=-72.2'C
root@rpi-oc1:/home/pi# _
```

Left
Some serious cooling is needed to achieve the highest CPU clock speeds

Right ➡
Crank up the cooling, and the clock speed, to play the Quake 3D demo

so the operation of the whole jig is simple: no need to pour liquid nitrogen all the time to keep a stable temperature. Instead just give it a splash to keep temperatures within the desired window. Extreme overclockers use the same method to cool traditional PC processors and graphic cards, but at a faster rate, as thermal loading of a modern multi-core Intel/AMD CPU or Nvidia/AMD GPU can reach hundreds of watts.

Since the thermal image camera cannot capture temperatures below -30°C, we will have to use software temperature reporting as a base measure, together with a Fluke 52-II thermometer, to stabilise container temperature during the benchmark runs.

The Broadcom SoC chip has a cold-bug around -100°C (by software-monitoring value), which means that it will stop working once the temperature drops below that. Since the boiling point of liquid nitrogen is -196°C, we need to maintain variable temperature control. This can be done manually, by pouring an amount of liquid nitrogen onto the evaporator block. Experimentally, it was determined to keep the copper block temperature at around -120 to -140°C, maintaining a stable Pi temperature close to -80 to -90 °C.

Getting 1550MHz stable didn't require very cold temperatures: the chip was functioning fine even at just -20°C. 1600MHz was stable once temperatures were below -45 to -50°C.

The Quake 3D game demo also ran flawlessly at the maximum CPU frequency. However, due to software configurations, there was no benchmark data to compare with. Fastest run logs are presented below.

HWBOT PRIME TEST

This test completed without issues at the maximum 1600MHz clock, at a pretty impressive temperature of -88.34°C, and gave an overall score of 514.53. The memory test yielded 819 200 operations performed (2 839 246.15 ops/sec) with a total of 800.00MB transferred (2772.70 MB/sec).

Are these numbers worth the nitrogen used? Not really, but it was fun to see how can it run and what the limits are of ARM-based computer overclocking. If the CPU clock frequency could be increased higher than 1600MHz, we would be likely to see a much bigger impact from going to extreme overclocking.

But until then, that's all for now.

SUMMARY AND CONCLUSION

Overclocking is fun as hobby, but it can also be useful in practical applications. Jack Zimmermann (**hsmag.cc/heFhWO**) demonstrates an excellent example of how overclocking a Raspberry Pi 3 in the role of a Stratum-1 NTP time server helps to reduce the uncertainty of GPS time synchronisation. Another possible use for an overclocked Raspberry Pi is various cross-platform emulators and game console emulators, where performance is never enough.

With the help of a little liquid nitrogen, we managed to overclock a Raspberry Pi 3 Model B to its maximum 1600MHz limit without much trouble. This article reveals a few basic bits about overclocking theory and methods. It's not rocket science, so anyone can do it, and getting hold of cryogenic liquids is far from mandatory. In the end, it's another way to have some fun with this capable little microcomputer. ☐

Benchmark test	CPU Frequency	GPU/L2 Frequency	DRAM Frequency	Result	Temperature
HWBot Prime	1550 MHz	450 MHz	550 MHz	504	-49.0
HWBot Prime	1600 MHz	450 MHz	550 MHz	512	89.9
HWBot Prime	1600 MHz	500 MHz	550 MHz	514	-86.8
Sysbench CPU 20000	1550 MHz	450 MHz	550 MHz	71.5 sec	-25.2
Sysbench CPU 20000	1600 MHz	450 MHz	550 MHz	69.3 sec	-77.4
Sysbench memory 800MB	1550 MHz	450 MHz	550 MHz	2582.1 MB/sec	-21.0
Sysbench memory 800MB	1600 MHz	500 MHz	600 MHz	2772.7 MB/sec	-86.3

Table 4 ⬅
Extreme overclocking results with maxed out CPU frequency

Helping HANDS

Makers use their skills and networks to lend a hand

Goli Mohammadi

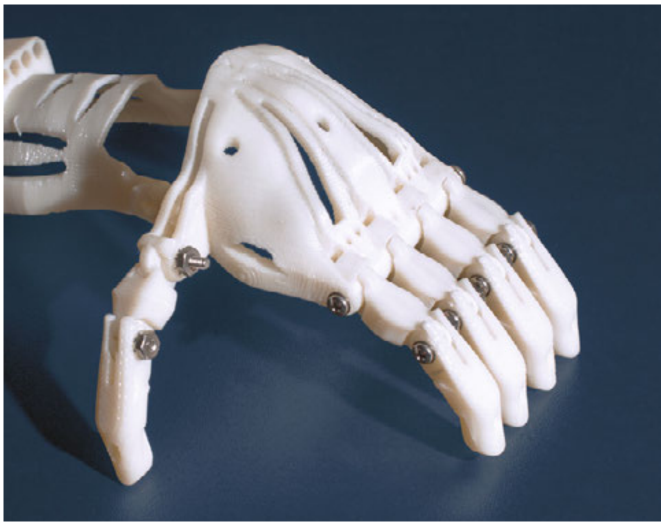
@snowgoli

Goli Mohammadi is a word nerd, highlighter of makers, and lover of mountains. When she's not staring at glowing screens, she's romping around nature. Find her at **snowgoli.com.**

Though worldwide statistics on the number of amputees are hard to come by, consider that there are currently roughly 2 million people who've lost a limb in the USA alone, with about 185,000 cases added each year. When we take into account naturally higher statistics in developing countries and map that to the global population, the numbers are staggering, and the price tags on replacement limbs are always daunting.

The causes for amputation are as varied as the cases. Naturally congenital medical conditions are among the stats, but one of the leading causes is diabetes, followed by injury from accidents and combat. Now imagine what you'd do without one of your limbs, whether a hand, arm, foot, or leg. While humans are incredibly adaptable creatures, daily tasks would be infinitely more challenging, if not impossible, depending on the specific amputation.

Enter the vast field of prosthetics, artificial replacements for missing body parts, including



Above
Even the Food and Drug Administration in the US is researching low-cost 3D-printed prosthetics, like this hand that was printed in a Center for Devices and Radiological Health (CDRH) lab

Left
Open Bionics collaborated with ILMxLAB, Lucasfilm's lab for immersive entertainment, to create a series of bionic hands for young amputees, featuring designs from *Iron Man*, Disney's *Frozen*, and this *Star Wars* one

hands, arms, legs, feet, eyes, and teeth. The ancient Egyptians are largely credited as having created the first prosthesis, evidenced by a wooden toe found on a mummy thousands of years old. Varied prostheses have been found around the globe, including an artificial leg made of bronze and iron with a wooden core, dating back to 300 BCE, discovered in Italy.

Interestingly, though popular sentiment relegates the creation of prostheses to professionals in the medical field, in the Dark Ages, making artificial limbs was the work of all kinds of tradesmen, including woodworkers, metalsmiths, and even watchmakers, who would apply their knowledge of gears and mechanisms to add functionality to the limbs. Sometimes replacement limbs lacked any function, other than being a placeholder of the missing limb. Sometimes they were made with the bare essentials, nothing more than the infamous metal hook arm and wooden peg leg.

MAKING STRIDES

But as materials and technologies advanced over thousands of years, so too did the functionality and comfort of prostheses, as well as the price. While we could go on about the specific strides that the field of prosthetics has taken over the course of history, this story instead focuses on how these advancements are being made more accessible to people who need them, through the collaboration, ingenuity, and generosity of makers.

Just as in much of the field of medicine, the availability of advancements doesn't mean they're within reach of the average person. And because in many cases, having access to good prostheses is not a matter of 'life or death', the kind and calibre of limb your health insurance will cover (if you even happen to have it) depends on the type of amputation and the deemed daily needs of the patient.

Among upper and lower extremity prostheses, some partial and some full, the average price tag is anywhere from $5,000 to $80,000 and beyond. Plus, they normally need to be replaced or repaired every few years for mechanical reasons. What's more, if the patient is a child, the prosthesis will need to be regularly upsized as the child grows. So while, →

A SHOW OF HANDS

There's currently a vast and growing array of readily available, low-cost, open-source hand designs made by interdisciplinary teams. Sample a few here and explore online for many more.

FABLE is an acronym for 'fingers activated by low-cost electronics', and this electromechanical hand offers precise finger movements generated by electrical signals from the muscles. FABLE is a project of the Open BioMedical (OBM) Initiative, a global non-profit dedicated to affordable, 3D-printed medical solutions, with a founding team based in Italy.

openbiomedical.org/fable

TINA is a different offering from the OBM Initiative, this one strictly mechanical, using a unique system of rods that move in response to movement in the wrist. A prime example of the fresh solutions devised by collaboration between disciplines, TINA was designed by the Polish jewellery designer Justyna Stasiewicz, who did the 3D modelling, while biomedical engineer Cristian Currò lent expertise on the biomechanics and 3D printing.

openbiomedical.org/wil

OpenBionics' prosthetic hand is designed with a focus on anthropomorphism, based on the belief that it's an important factor in success of use and adoption. The core structure, 3D-printable and made with readily available off-the-shelf parts, features an impressively opposable thumb comprised of a selectively lockable toothed mechanism capable of nine different opposition configurations. The bio-inspired finger actuation and transmission system can attain superior flexion and extension, and the fingers even have soft tips.

openbionics.org/affordableprosthetichands

Handiii, from the HACKberry open source community, originated in Tokyo as a project of Exiii, with the express purpose of solving three problems in existing myoelectric prosthetic hands: too expensive, not easily repairable, and few design choices. Handiii uses the electrical signals in the arm to intuitively control the hand and keeps cost down through 3D-printed parts, using a smartphone for the electromyography (EMG), and improving the design of the mechanisms in the hand. The arm is less than $300.

exiii.jp/projects/#handiii

The beautiful thing about collaborative digital design is that it can be **made, shared, and iterated upon across the globe**

MAKERHEALTH: BRINGING DIY TOOLS TO THE FIELD

While touring medical facilities in Nicaragua, MIT's Jose Gomez-Marquez and Anna K. Young made an important observation: Even though nurses there and around the world may not recognise themselves as 'makers', they most certainly are.

Often equipped with only off-the-shelf solutions that either don't serve the needs of the patient or do so in an ill-fitting fashion, nurses regularly perform customisations and repurpose materials to better suit each patient's unique needs. From hacking prescription bottles for visually impaired patients to modifying grips to create better walker handles, nurses have been innovating creative makeshift solutions to aid patient comfort and safety for at least the last century.

The problem is that a brilliant solution that one nurse devises may never make it past the walls of that hospital. Imagine the progress that could be made in health care if providers, nurses and beyond, were deputised to create custom solutions in their in-hospital makerspace and were encouraged to share solutions with the greater community through online project repositories. Welcome to MakerNurse and the umbrella organisation MakerHealth, projects of MIT's Little Devices Lab.

One of their initiatives is to open makerspaces in hospitals, and the very first of its kind was opened at the University of Texas Medical Branch, posing the question, "What if we gave creative people better tools to innovate?" The goal is to follow this space up with many more. In addition, the network offers online tutorials, a repository of medical hacks and projects, as well as workshops. As the mission statement reads, "We believe everyone can be a medical maker."

Above
MakerHealth co-founder Jose Gomez-Marquez helps nurses at the Mayo Clinic prototype patterned floor lights to assist fall-risk patients

yes, the future has arrived in terms of fully functional robotic arms and legs, they're also incredibly cost-prohibitive to the average person.

SOLUTIONS WITHIN REACH

Now let's overlay this scenario with the maker movement – with its massive brain trust and ethos of sharing, collaboration, and open source ideals. Add the proliferation of desktop manufacturing and affordable electronics, and you get a much brighter picture. While the traditional process for creating a custom prosthetic is generally to mould, cast, vacuum form-fit, assemble, adjust, and repeat, the maker answer is to scan, print, and adjust.

Much of the cost of manufactured prosthetics is rooted in medical mark-up and proprietary designs. But as one maker-made organisation, Limbitless Solutions, espouses, "We believe no family should have to pay for their child to receive an arm." There's now a virtual cornucopia of freely available open source designs for prosthetics online, most being 3D-printable hands, some robotic, others not. In the burgeoning realm of democratised prosthetics, making hands and arms is perhaps more accessible than feet and legs for the time-being, both in form factor and functionality.

There are a number of organisations, including the originator e-Nable, that encourage, support, and organise teams of interdisciplinary (and often international) makers to tackle the need for low-cost, easily reproducible prostheses. And the beautiful thing about collaborative digital design is that it can be made, shared, and iterated upon across the globe. Teams can work across national borders, and digital files can be made freely available to download and print from anywhere. These agile teams also enable rapid iteration and intelligent design.

The future looks much brighter when there are entire networks of makers, health care professionals, designers, and technologists dedicating time and talent to furthering the field of low-cost prosthetics, and they're showing no sign of slowing down. What's more, most are actively looking for volunteers to get involved. Maybe you can consider lending your skills and 3D printers to the cause…? □

PROJECT

Enabling the future

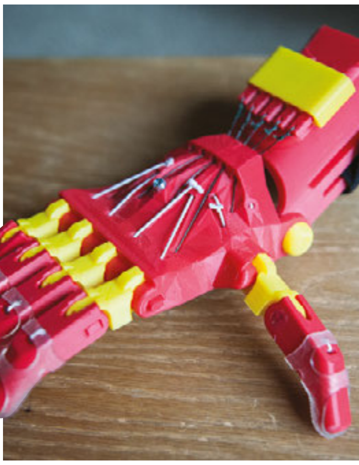
What started with a steampunk mechanical hand prop documented on YouTube has become one of the largest volunteer networks of makers of 3D-printed upper limbs, mostly hands. Back in 2011, Ivan Owen posted a video of an oversized mechanical hand he made as a costume prop for a steampunk convention. He received many requests for blueprints, but one request in particular changed his life. The mother of a five-year-old boy named Liam in South Africa reached out to see if Owen could create a miniature version of his hand for her son, who was born with no fingers on his right hand.

Owen began doing research, and using the National Library of Australia's online archive, Trove, he drew inspiration from a prosthetic hand design developed by an Adelaide dentist in 1945 for a corporal who had lost most of his hand to a gun accident. The hand was made using whalebone, cables, and metal pulleys, and reportedly served the wearer for 30 years until his death.

The first version of Liam's hand was modelled after this classic, but Owen realised Liam would quickly outgrow it, so he began looking into creating a 3D-printed version. With the help of some collaborators, the first 3D-printed prosthetic hand was born, giving Liam an impressive amount of new-found dexterity.

Owen posted a video of Liam and his new hand on YouTube and offered the design files on Thingiverse with the hope that the community would help streamline the hand and that anyone anywhere in the world could make one for someone who needed it. From those humble beginnings, the e-Nable community was born. In the first year, they amassed 3,000 volunteer members and collectively built hands for more than 750 people around the globe. The second year roughly 2,000 people from 45 countries got a hand from the e-Nable community, and the movement has continued to grow.

On their site, you can sign up to be the recipient of a hand or to volunteer to help make hands. There are also how-tos and other resources if you want to build your own hand, with nine designs to choose from, with names like Raptor Reloaded and Osprey Hand. □



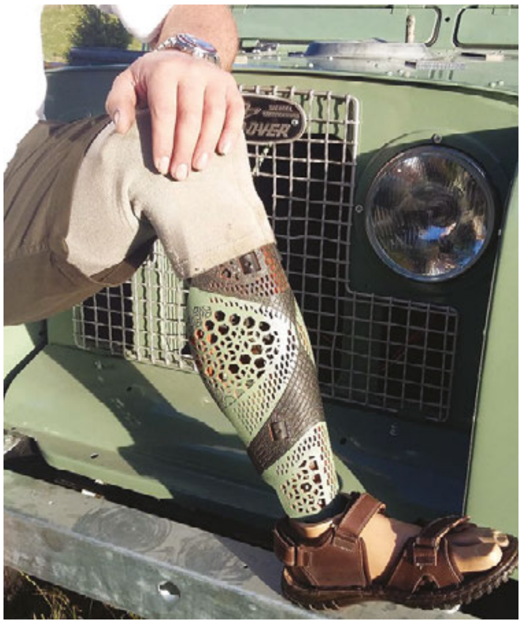
Above The first 3D-printed prosthetic hand opened up a world of possibility to five-year-old Liam

PROJECT

Leg bling

Let's put functionality aside here for a minute and talk about aesthetics. Most traditional, standard prosthetics are devoid of character, intentionally designed to detract attention. Industrial designer McCauley Wanner and architect Ryan Palibroda of Canada's Alleles Studio set out to flip that paradigm on its head – or rather leg.

They make incredibly stylish 3D-printed and hand-painted prosthetic leg covers designed to complement the wearer's wardrobe and serve as outlets of self-expression, offering a robust colour palate and healthy variety of designs to choose from. Their noble mission is to "do for prosthetics what a previous generation of fashion designers did for the eyeglass industry." □



Left Leg covers come in many designs, or your own custom artwork

HackSpace meets...

BECKY STERN

Electronics, knitting and how to be a professional maker

For the past decade, Becky Stern has been one of the leading American voices in the maker subculture. She's made light-up trainers (OK, fine, sneakers), maintained a 1975 Honda CB200T motorcycle, created a jumper to turn off errant TVs... well, she's made far too much to go through here. What's more, she's meticulously documented almost all of these builds so you can craft your own projects using the same tools and techniques. In case you need more evidence of her commitment to inspiring other makers, she teaches Making Studio at the School of Visual Arts (New York) as part of its Products of Design Masters of Fine Arts (MFA).

This former Director of Wearable Electronics at Adafruit has been featured on CNN, BBC, Forbes, Vice, Engadget and just about every other major tech news outlet. When lunar legend Buzz Aldrin needed an illuminated jacket for an appearance on *The Late Show With Stephen Colbert*, there was only one woman for the job.

HackSpace's Ben Everard caught up with Becky Stern to chat about what it means to be a professional maker in the modern world, and see how she's getting on with her current job at Instructables. →

Left
"This old camera brings something fun and personal with your own interests rather than just being a piece of consumer of technology"

You make things from a huge range of disciplines. Do you have a favourite tool or technique among them?

There's not a favourite, there's just familiar and less familiar. I learned how to knit when I was about 15 and I learned how to sew when I was eight, so those types of handcrafts always make me think of my family and being a kid, whereas I only learned how to do some motorcycle repair in the last two years – they all have different types of endearing qualities.

I guess I like things where I don't have to get too messy. I like getting dirty with jewellery and stuff, but I don't like getting oil on my hands. Working with luxurious materials always feels good. Wool and leather, yarn and fabric are always nice to touch whereas when you have to wear protective gear or get toxic chemicals on your hands … it's a different mindset.

Protective gear can feel like a barrier between us and the thing that we're making – it changes the feel of the process. Do you make stuff because you want things that you can't get another way or is it the process of making things that you like?

I really like making and sharing. Often, I could just as easily buy something but I'm interested in having a conversation about how that thing is made or how it works. [For example] it's an IoT device and you want to talk about the security vulnerabilities – making an IoT device yourself is a great way to have a discussion on the internet about the security issues.

I think most of the things I make you can get, but some things you can't. This is a vintage camera [see images]. I have a collection of vintage cameras because I've always been interested in photography. I don't take film photos any more – they're expensive to develop and this camera never took good film photos in the first place – and so I upgraded it with a Pi camera. It takes three photos, makes a GIF and then uploads it to Tumblr. Of course that's something you can't get in the store (a camera that uploads GIFs to Tumblr) but you can take GIFs on your phone and upload. The novelty is that you made it yourself and it's ultra custom – it's not so much about the object as how you feel after having built it. How do you feel after you build something compared to how you feel after you buy something. Yes, there are multiple ways to use your phone for the same net effect, but it doesn't bring a smile to your face to have your picture taken by a phone any more, whereas this old camera brings something fun and personal with your own interests rather than just being a consumer of technology. There's some empowerment there that I think is the point, more so than finding features that don't exist yet because features will always exist soon.

> "Making an **IoT device yourself** is a great way to have a discussion on the internet about the **security issues**"

Do you think that you got a dramatically different set of photos (OK, GIFs) from that camera than you would have done with a phone?

I think so because the subjects were performing for a special purpose. The audience is different than for a typical phone. It's like oh, they don't know quite how it works because I'm the one who programmed it to do what it does. You kind of have to have a more personal relationship with discovering it so for sure I got more smiles and more people had their photos taken.

One thing we've always found interesting about your work is that you bring a really wide variety of skills into your makes. You said before about sewing and knitting from when you were young and you were working at Adafruit for a while working with electronics. Many people come to making from the programming side of things. What skills would you recommend are worth learning for someone with a computing background?

I see a lot of computer scientists and people who know Linux really well, but are just dipping their toe into electronics. They should certainly learn more about electronics and physical computing, and I would say even try out Arduino stuff just for comparison's sake because it really helps understand the low-level logic for components and sensors and stuff.

It's clichéd to say now, but the knowledge transfer density is high when electronics folks try their hand at 3D printing. If you can make an enclosure for your project, it makes your project more real. To that end, I would say vector drawing skills – Adobe Illustrator or Inkscape – are really important even before you learn CAD tools. There are simple tools like Tinkercad that I think are really useful for people who are doing coding stuff.

Also, call it Product Design for lack of a better word, although that term is also being used now for apps and that sort of thing, but I mean old-school industrial design – usability. If you're going to design a device that someone's going to hold in their hand, the shape of the object is affected by all sorts of psychological factors that are subject to the [content] of whole PhD degrees and Don Norman's *Design Of Everyday Things* [see review, page 129] and how our intuition affects our ability to interact with the everyday world. How do you know to push on a door? That sort of thing is really important. It's not just one skill. It's a whole discipline of interaction design.

Above
"My advice to anyone who wants to be a professional maker of anything is to be constantly publishing stuff"

You can see some of the images taken with the camera here: **hsmag.cc/kBcNMt**

I would also say woodworking is really important, laser cutting is really important. Instructables has free classes on all of these skills. You can go to Instructables.com and find an intro electronics class, an intro Raspberry Pi class, an intro Arduino class, a woodworking class, a table saw class.

One thing we've noticed is that you're often described as a DIYer rather than a maker or a hacker. Do you prefer this?

DIY, maker, whatever! Look it up on Google Trends and whichever one is higher is the one people are searching for more. I've never been into labels – they're both fine. I think DIY is a more old-school term. I gave myself a title before the word 'maker' really caught on.

I've been trying get across to people what I'm doing recently and it's a little hard to get across to anyone outside of the subculture what it's about. "It's kind of about making stuff", but that seems too broad unless you're used to the term 'maker'. A lot of people don't get the word 'hacker'. Maybe I'll adopt DIY.

It's the '70s term. That's when my parents were starting to do home improvements, so to them it's always been DIY. My parents have renovated an old farmhouse into a bed and breakfast and they do all the framing and the plumbing and the electrical and the tile work, so I've always been in a house of the do-it-yourself attitude. Growing up watching TV shows like *This Old House*, that kind of home improvement stuff was always called DIY. The kind of thing I was working on up through college was always an extension of that. I don't know if 'maker' resonates more with a younger audience because they know what the scene is now, but I'm sticking with 'DIY'. It's fewer characters. →

Left
You can just see the Raspberry Pi Zero in Becky's smart camera peeking out of the case

In your career, you've been involved with quite a few of the major organisations in the maker scene. Do you have any advice for people coming in who'd like to be a professional maker or follow in your footsteps?

My advice to anyone who wants to be a professional maker of anything is to be constantly publishing stuff – not to wait for someone to ask you to do something. It's to be constantly pushing out something that you believe in, even if you're not being paid for it at first. I was hired as a media manager, or a video producer, at Make and Adafruit and my skills in video production and photography and project management were what set me apart from someone who was just making a tutorial. I could manage others who were doing the same work as me, and also see how my work fitted into a larger editorial vision, but that's because I had a strong point of view.

I think it's important to not let potential sponsors [control what you do]. Just be genuine to what you want to make and shape the opportunities for [...]. I don't know if it's product sponsorship or freelancing for some of these publications. Let your ideas drive the relationship and see how things could fit in, rather than saying, "I have these skills as a maker, what do you want?" because often when you're being hired for your social influence it's because you have good ideas, not because you're special and unique in the way that you make a video. Anyone can make a video or a tutorial; it's about consistency, breadth, and vision.

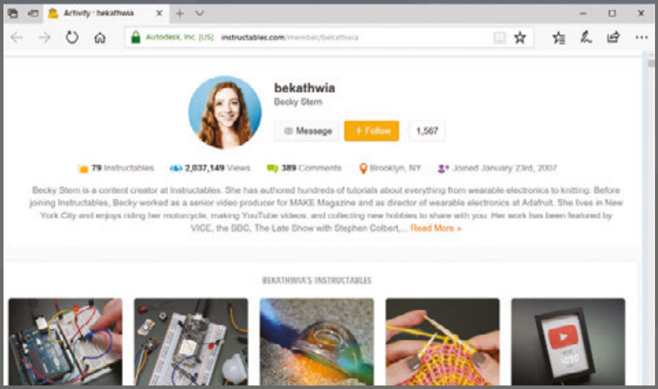
Logistically, it can be hard to get a job at these big companies now. I see the industry and the market changing a little as there are more independent makers like Bob Clagett (I Like to Make Stuff) and The Sorry Girls on YouTube (the Toronto-based duo). There's a lot more single entrepreneurs – individual people not working for a large company – trying their hand at the business of being a maker. It really comes down to what you have to say and your ability to produce high-quality [content], and frequently. I got offered the opportunities, I think, in my career mostly because of my tenacity in publication. I was just relentlessly always publishing something. I was never taking a six-month long hiatus, I was never taking a year off to write a book, I was constantly publishing all the time. If you want to make a living doing maker projects, you have to be able to show people that you have a workflow that's going to be sustainable.

> "The change in the marketing industry and the blowing up of the maker subculture have really **changed the landscape** for what it means to be a professional maker"

You mentioned the way that individuals are creating opportunities for themselves. Is that how you see the future of professional makers – where you may have a partnership with a company but it's much more about your personal brand and the things you've built up?

My observation has been that more individual personalities are succeeding in the maker space as their own business entities through private product sponsorships rather than my situation, which is becoming a full-time employee of a maker company. I've just seen it happen a lot more this decade compared to last decade. I don't know if that's because the maker subculture is bigger now.

There's also a bigger overlap with the bigger non-maker culture of observing makers as entertainment. A lot of non-makers are observing makers and thinking, 'oh, I want to do a project someday.' I feel like that audience has grown by orders of magnitude in the last ten or so years, which enables a platform along with regular social media influencer marketing where the future of marketing has changed. People looking to do marketing with young people who are excited about making things are reaching out to people who are influencers in the community already and sometimes it's easier for those companies to have a direct



bekathwia
Becky Stern

79 Instructables · 2,037,149 Views · 389 Comments · Brooklyn, NY · Joined January 23rd, 2007

Becky Stern is a content creator at Instructables. She has authored hundreds of tutorials about everything from wearable electronics to knitting. Before joining Instructables, Becky worked as a senior video producer for MAKE Magazine and as director of wearable electronics at Adafruit. She lives in New York City and enjoys riding her motorcycle, making YouTube videos, and collecting new hobbies to share with you. Her work has been featured by VICE, the BBC, The Late Show with Stephen Colbert,... Read More »

BEKATHWIA'S INSTRUCTABLES

Left
You can see all of Becky's creations at hsmag.cc/xJVxsk

sponsorship with those people than it is to buy a big campaign with a big magazine or a huge booth at Maker Faire. They could support sponsorships for YouTubers who are already going to have that embedded audience who will show off their product to exactly the right demographic. But it also means that there are a lot more people trying to be just that. Who wouldn't like to be their own boss and do whatever projects we want to? I think that there's a lot more people with that goal now too but at the same time, it's never been easier.

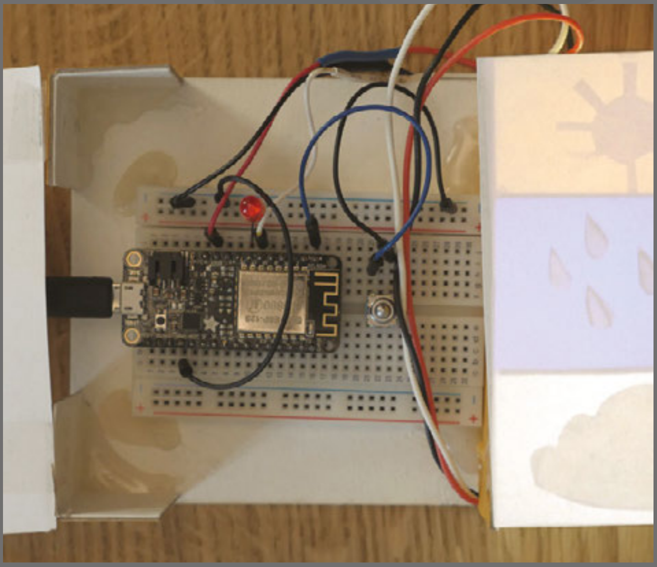
You're now at Instructables. I don't want to call it a social network, but it's sort of a network of people showing off what they've done. There are a few companies in this sector: Instructables, Hackster, YouTube, etc. What is it that you think makes Instructables the best place?

It's the breadth of subject matter that makes it unique. You go to a site like ravelry.com – it's the internet's pre-eminent site for knitting and crochet patterns and it's a great community. It's a great website, but anyone who doesn't knit or crochet has never heard of it. There's not a lot of bleed-through of people who are just interested in simply making stuff on that site. On Instructables, you could go there because you're interested in knitting and crochet and see somebody's project that includes electronics and get inspired to try a whole new genre of making things that you never knew that you would have been interested in. The community is really nice. There's a team at Instructables whose job it is to enforce the 'be nice' comment policy that's been there for all of Instructables' more than 15-year history. I think the quality of the community coming to look at each other's projects and support each other is really huge.

I think the 'classes' content there that's still relatively new – it's just gone up in the last couple of years – has more honed, curated, and introductory-level classes for individual maker skills. Those cross over really well, so if you're dipping into a really cool project but you don't know how it was even conceived of, you can do a class that can give you the background information to then pull off that cool project that inspired you in the first place.

Do you have any tips for people looking to win Instructables competitions?

The contests are judged both by user votes and by a judging panel, so the user votes impact which projects get selected for review as finalists for the judges, and anyone can become a judge as well. If you go to the site footer, you can write in about becoming a contest judge and that could potentially be a good way to see what projects are coming in as finalists. There's no secret to winning a contest when you just have to create good content, so writing good instructions with good photos and with empathy for the person who's reading – who's trying to create your project. That's how you create a good Instructable, and good Instructables win contests. It doesn't hurt to campaign among your friends and get them to vote for you or to write into big sites and blogs that talk about projects like yours to get them to look at your project during the voting window, so that any spikes in traffic you get would allow you to get people to vote for your projects. So if it's a technology project, sites like Hackaday, BoingBoing, and Engadget. Those kinds of places garner a lot of traffic for DIY-type technology projects [and] could really help you move the needle on votes, but you have to have a good quality Instructable to start with. I did an Instructable recently called Five Tips for Better Build Videos and it included tips about scriptwriting and how you can turn the script into the draft for your Instructable, and just some tips on documenting your projects in general. □



Left
An internet weather station is a great way of helping you decide what to wear in the morning

DUCT TAPE

No toolbox should be without this ubiquitous fix-all

Despite the fact that duct tape, in its current incarnation, wasn't even invented until the mid-twentieth century, it's hard to imagine what the world did without it. The winning combination of water-resistant layer plus fabric plus adhesive makes it a wonder material, equally capable of repair and creation.

In the early years, its predecessor, adhesive-backed duck cotton fabric, was used primarily for medical purposes. As the story goes, in 1943, during the tail end of World War II, a woman named Vesta Stoudt was working at the Green River Ordnance Plant in Illinois, inspecting and packaging rifle grenades. A mother of two Navy soldiers, she worried that the thin paper tape and waterproofing wax used to seal the ammo boxes was too difficult and potentially time-consuming to open in the field. She suggested creating and using a water-resistant adhesive-backed cloth tape, but when her words fell on the deaf ears of her supervisors, she hand-wrote a letter to then-President Franklin D Roosevelt, whose own sons were also fighting in the war.

Much to her delight and relief, her suggestion was received and deemed important enough by the American leader for the government to call in the Johnson & Johnson company, known for its fabric medical tapes, charging them with inventing what was to be the first duck tape, named for its ability to repel water like a duck and for the fact that it was made from duck cotton. It was originally army green. Quickly recognised for its versatility, it also became dubbed '100 mile an hour' tape because it could be used on virtually anything, including vehicles, aircraft, and guns, not to mention its emergency medical usages in the field. So in this case, the mother of invention was actually somebody's mother.

After the war, the sticky wonder was promoted for civilian purposes, notably as a way to hold ventilation ducts together. It was then offered in silver to match the ducts and referred to as duct tape. Ironically, the standard variety has since been deemed unsafe for use on HVAC ducts because of its flammability and toxicity when heated, but it has found plenty of uses the originators could never have imagined, from building kayaks and clothing to repairing spacecraft. As the old saying goes, if you can't fix it with duct tape, you haven't used enough. But before we get to that, let's unpack this magic tape and take a look at what makes it work.

Sealing the duct tape sandwich is a top coat of low-density polyethylene (LDPE), the most common plastic on earth, which provides its resistance to water, abrasion, and friction, all while staying flexible. Below that, duct tape gets its superior tensile strength from its fabric mesh foundation. The tighter the weave and higher the thread count of the mesh, the stronger the

Goli Mohammadi

@snowgoli

Goli Mohammadi is a word nerd, highlighter of makers, and lover of mountains. When she's not staring at glowing screens, she's romping around nature. Find her at **snowgoli.com.**

Get the Grade

With many grades to choose from, how do you pick the right duct tape for your project? Even though there are lots of classifications, such as 'all purpose' and 'advanced strength', each manufacturer has its own way of defining these. Basically, you want to look at these four specs.

Thickness

Generally, duct tapes range from 3 mils to 17 mils. Thicker tapes are naturally stronger but also less flexible, which affects how they wrap around objects. A good general-use tape should be about 11 mils. You can also layer thinner tapes to increase their strength.

Adhesive

Quality duct tapes all have adhesive made of natural rubber. The more natural rubber, the better the performance of the tape. There's also the 'hot-melt' adhesive variety, but these tend to be less reliable in extreme temperatures and are weaker.

Threads

Looking at the threads of the cloth grid (or scrim) can give you an idea of the tape's quality. Threads that run the length of the tape lend strength, while threads that run across show how easily it is to tear by hand. So, the closer those threads, the stronger it is and the easier to tear by hand.

Manufacturing Technique

Duct tapes are either made by co-extrusion or lamination, with the former being far superior. In co-extrusion, the scrim is melted into backing, forming a cohesive whole, before the adhesive is added. With lamination, all three layers are pressed together, causing a risk of bubbles and weakness.

Price can also be a good indicator of how high-quality the tape is, and consequently, how long it will last and keep its structural integrity. You really do get what you pay for here. There are also classifications such as consumer, industrial, and military grades. Industrial is the variety that professionals use for actual HVAC ducts, and is more waterproof, heat-resistant, long-lasting, and adherent to metals than the consumer variety. Military grade then ups the ante to the next level of strength and durability, often reflected in the price.

Above Duct tape, shown here attached to some reinforced concrete rebar, is ridiculously versatile

tape and the higher the rip strength. No longer restricted to just cotton, thread options now include nylon, rayon, and polyester. Lastly, the adhesive is applied as the base layer, unique in that it's made with rubber compounds and caked on to the mesh in a much thicker coat than used on most other tapes.

Duct tapes are classified in grades, determined by the type of adhesive and strength of the fabric. An

"Duct tapes are classified in grades, determined by the type of adhesive and strength of the fabric"

inexpensive, standard duct tape may have a 9 kg (20 lb) rip strength, while military grade could easily be twice that. The varieties and grades are as varied as the uses. There's even a nuclear-grade duct tape that's certified for low leachable halogens and sulphur, can withstand temps up to 93°C (200°F), is UV-resistant for up to a year, and doesn't leave a residue when removed.

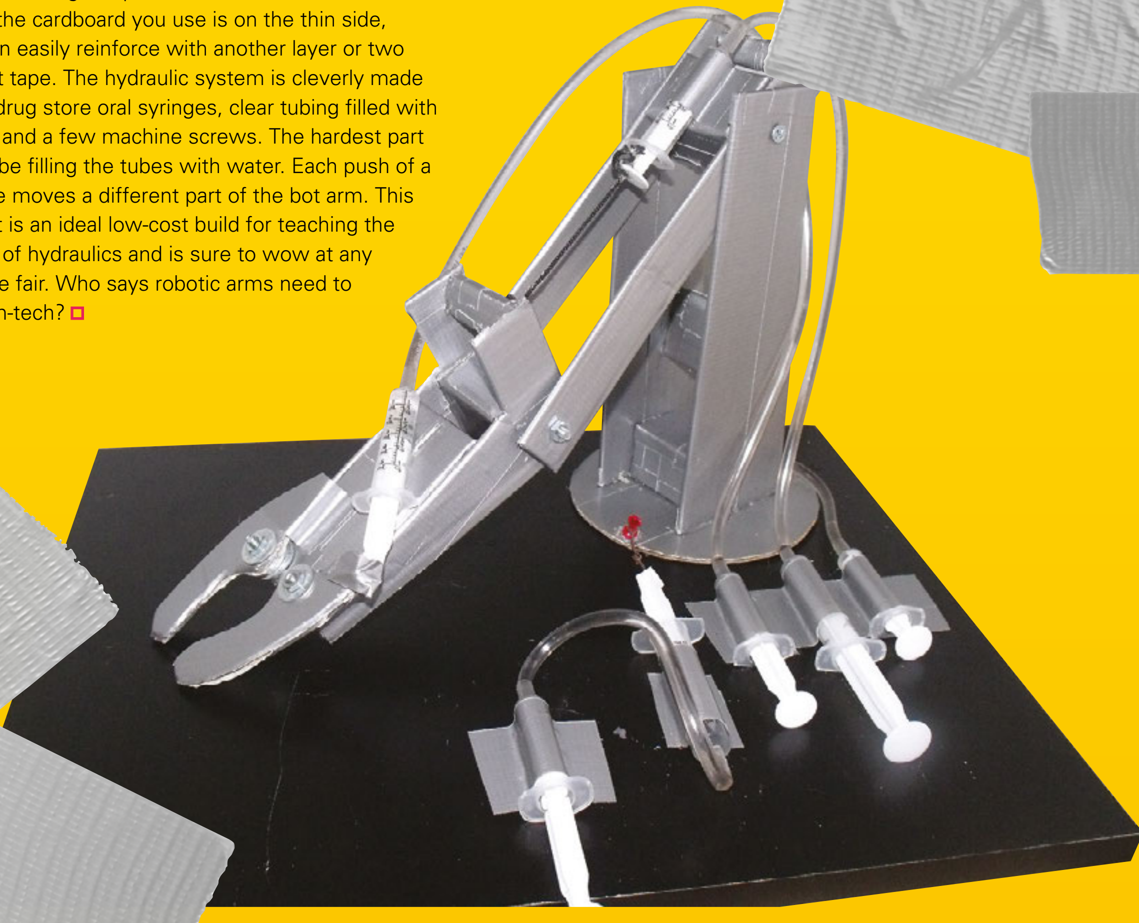
Worth mentioning is the disambiguation between duct tape and gaffer tape. Though the words are commonly used interchangeably, these are two different products with key differences. Typically used by theatre, film, and photography professionals, gaffer tape (named for the chief electrician on television and motion picture crews) is matte to deter light reflection, has a cloth top layer, and employs a heat-resistant adhesive that is easily removed without leaving stickiness. →

BOTS + BOTS OF TAPE

Project Maker
NELSON YEPEZ

Build Your Own
hsmag.cc/EqnXPe

Nelson Yepez always wanted to make a hydraulic robot arm, but most of the projects he saw online required a way to cut wood or metal. Having neither, he decided to go super low-tech by using reclaimed pizza boxes reinforced with duct tape, giving the bot a "cool metal-like look" as well as added durability. Yepez's cutting template makes the build a breeze, and if the cardboard you use is on the thin side, you can easily reinforce with another layer or two of duct tape. The hydraulic system is cleverly made using drug store oral syringes, clear tubing filled with water, and a few machine screws. The hardest part might be filling the tubes with water. Each push of a syringe moves a different part of the bot arm. This project is an ideal low-cost build for teaching the basics of hydraulics and is sure to wow at any science fair. Who says robotic arms need to be high-tech?

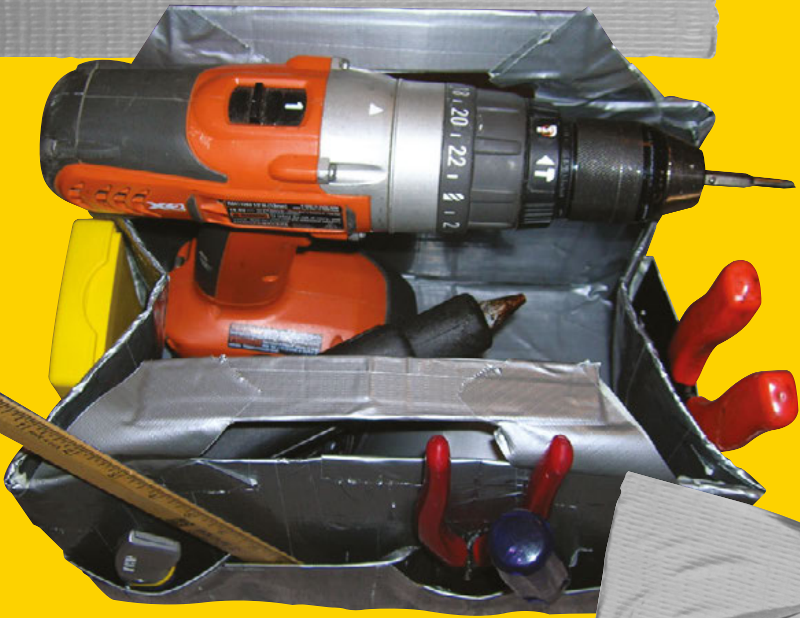

Right
When you don't have metal, fake it with duct tape

TOTING TOOLS IN STYLE

Inspired by Craftsman and AWG tool bags, Tristan Laughlin designed this duct tape version with many of the same useful features. The only ingredient is cardboard, which makes this project so inexpensive that you can save your money for the tools to fill it. This handy little carrier is water-resistant, full of useful side pockets, and is a prime example of how strong cardboard and duct tape can be when they join forces.

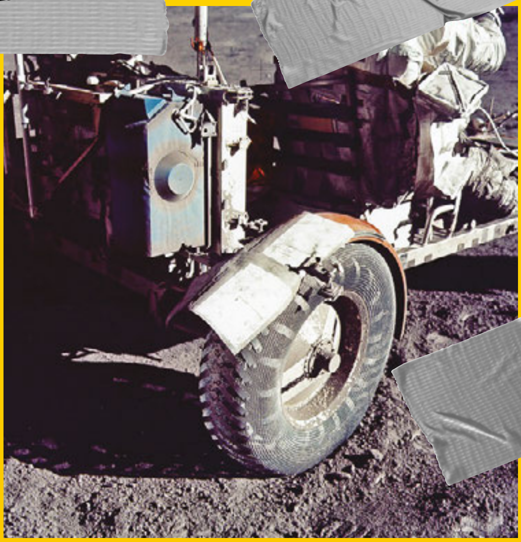
Project Maker
TRISTAN LAUGHLIN

Build Your Own
hsmag.cc/iNKdMj


Right
Like getting soup in a bread bowl, the container may end up being as useful as the contents

SPACE TAPE

We'd be remiss to not take a moment to appreciate how duct tape has been aboard every major spacecraft mission in recent history. It's not only the fastest way to keep floating items secure in zero gravity, but it's saved missions and lives in a pinch. When a fender on Apollo 17's Lunar Roving Vehicle accidentally got caught on a hammer in the shin pocket of astronaut Gene Cernan's spacesuit, half of it ripped off. Normally, a broken fender would be no big deal, but on the moon's surface, the rover was kicking up plumes of dark, abrasive moondust. Crew members used duct tape, lunar maps, and clamps from the optical alignment telescope lamp to create a makeshift fender. Apollo 13 astronauts also used it to MacGyver a carbon dioxide filter modification. When an explosion occurred on the Apollo 13 service module, all three crew members had to transfer to the lunar module, which was only designed to contain two astronauts for 36 hours. With carbon dioxide levels rising at an alarming rate, they had to find a way to make the square filters they had work with the lunar module's round holes. Using found objects and duct tape, they were able to modify and survive.


Left
Duct tape on this makeshift fender repair seems somehow right at home on the moon

Project Maker
NASA

Build Your Own
hsmag.cc/FtkRBo

hsmag.cc/aCcOmx

SUBSCRIBE AND SAVE!

NEW HACK | MAKE | BUILD | CREATE
HackSpace
TECHNOLOGY IN YOUR HANDS
hsmag.cc | December 2017 | Issue #01
BECKY STERN
How to get paid for making fun things
SMOKE YOUR OWN BACON*
*or anything else. But mostly bacon
BUILD THINKING MACHINES
Harness the power of artificial intelligence
HANDHELD CONSOLES FOR HACKERS
BOLDPORT: BUGS, BITS & BLINKENLIGHTS
TRANSISTORS 3D PRINTING DUCT TAPE CHANGING THE WORLD

SAVE UP TO 35% on the cover price

FROM JUST £4 per month

Visit: hsmag.cc/subscribe **Call:** +44(0)1202 586848

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 anytime
- No long-term commitment
- No large one-off cost

12-month subscription from £55:

- **UK:** £55 per year
- **EU:** £80 per year
- **US:** £99 per year
- **RoW:** £100 per year

PLUS **Digital subscriptions from £2.29 per month**

Available on the App Store

GET IT ON Google Play

Visit: hsmag.cc/subscribe **Call:** +44(0)1202 586848

HackSpace
TECHNOLOGY IN YOUR HANDS

FORGE

HACK | MAKE | BUILD | CREATE

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

School of making: Woodworking 101

Bring traditional techniques to your Raspberry Pi setup

Jenny Smith

hsmag.cc/msbristol

Jenny is a woodworking enthusiast with a background in bespoke furniture making, particularly reclaimed woods. She tutors at The Makershed Bristol.

As a woodwork tutor teaching traditional joinery using hand tools, I meet a lot of people from the technology industry looking to take time out away from the screen and gain a more balanced skill set.

This Raspberry Pi box project brings woodwork and technology together. Being small, it is fairly fiddly to make, but the end result is hugely satisfying. The corners of the box are joined with a whole and half dovetail, and the lid and base sit in grooves. The half dovetails enable the top grooves to continue to the ends of the box so that the top can slide in and out. This box is for a home server, so there's only a hole for the power cable and we've used the WiFi network connection.

I like to use reclaimed timber, and the maple and mahogany of my box are floorboards planed down to 10 mm and 5 mm thick respectively. Commonly available 10 mm pine for the sides and thin plywood for the base and lid would work just as well but if using pine, add an extra millimetre to the internal box height to allow for it to shrink across the grain as it dries out. Glue narrow lengths of pine together edge to edge to create wider lengths.

MARK OUT

Accurate marking out is essential to well-fitting joinery so take your time with it. Use the try square to score knife lines straight across the grain of the length of wood you are using for the box sides to show the ends of the sides, the shoulder lines of the joints, and the sides of the power source hole. Position any knots or defects in between the sides.

Run the marking gauge along the edges to show the edges of the grooves that the base and lid will sit

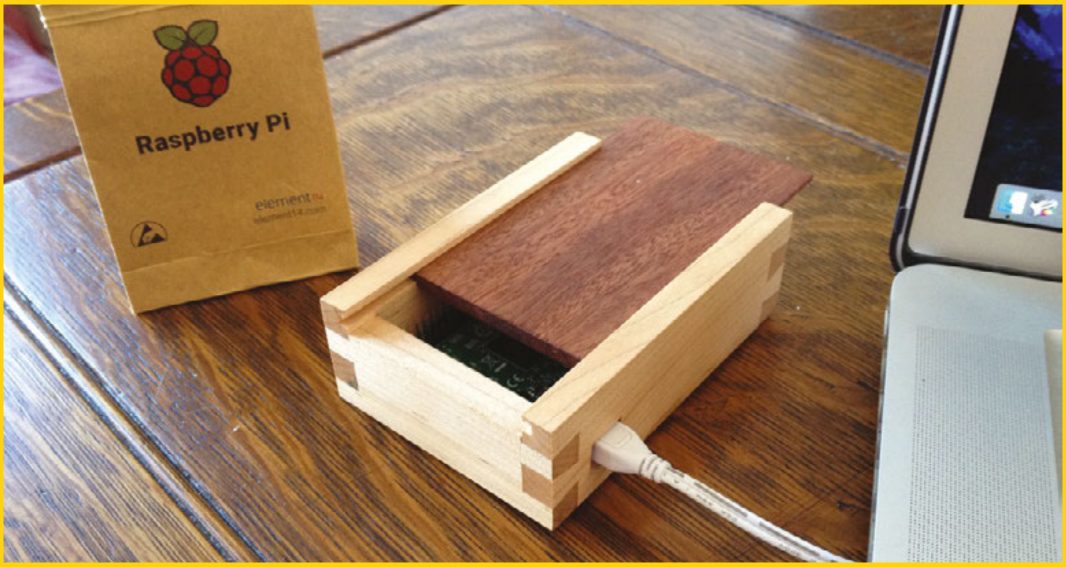
TOOL LIST

ESSENTIAL TOOLS

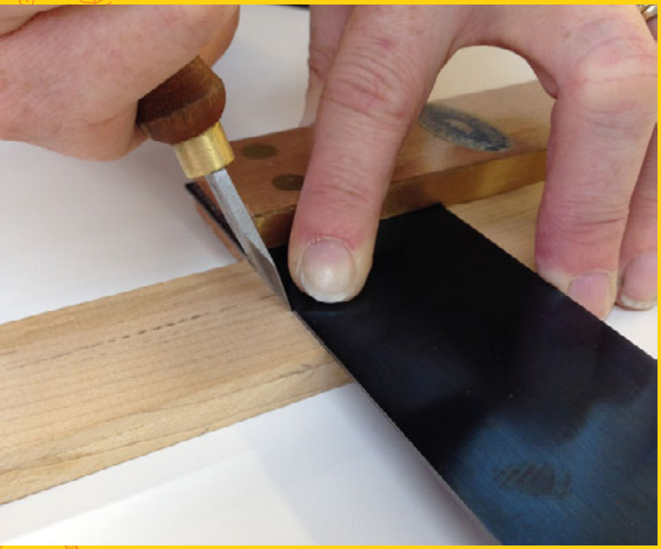
- **Ruler**
- **Try square**
- **Pencil**
- **Fine-tooth tenon/ dovetail saw**
- **Marking gauge**
- **Narrow bevel-edge chisel**
- **Clamps**

DESIRABLE TOOLS

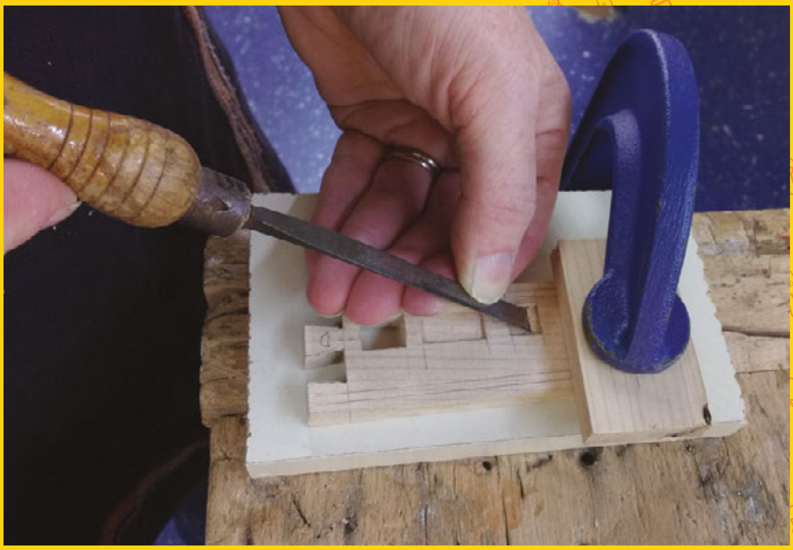
- Marking knife
- Wide bevel-edge chisel
- Plough plane
- Block plane
- Screwdriver
- Sliding bevel
- Coping saw



Above Who doesn't want to give their Raspberry Pi a snazzy new home?



Above
Chisel across the grain to cut the wood fibres



Above
Carving out holes for the Pi's sockets

in and the top and bottom of the power source hole. The grooves should be as wide as the thickness of your plywood.

When marking out the position of a saw cut, score two lines, 1 mm apart. You can then scratch out the fibres in between those lines to create a channel for the saw teeth to sit in. This will help to keep your saw on track when you make the cuts.

Tips for accuracy:

- Maintain firm consistent pressure on the stock of the try square and the marking gauge, pushing it against the edge of the wood
- Place the flat side of the marking knife against the blade of the try square

DRILL THE HOLE AND RECESSES

Carefully drill holes (on a low speed setting to avoid splitting the wood) inside the lines around the edges of the power supply hole to remove some of the waste wood. Then, deepen your knife lines across the fibres with a narrow chisel before paring away the waste wood that remains layer by layer. Have a piece of hardwood or laminate underneath the box side to support the fibres of the wood as you remove the last layers and stop tear out underneath.

Use the chisel to also remove the waste wood from the HDMI and audio recesses. When chiselling in line with the grain of the wood, take care to use light pressure on the chisel to avoid splitting.

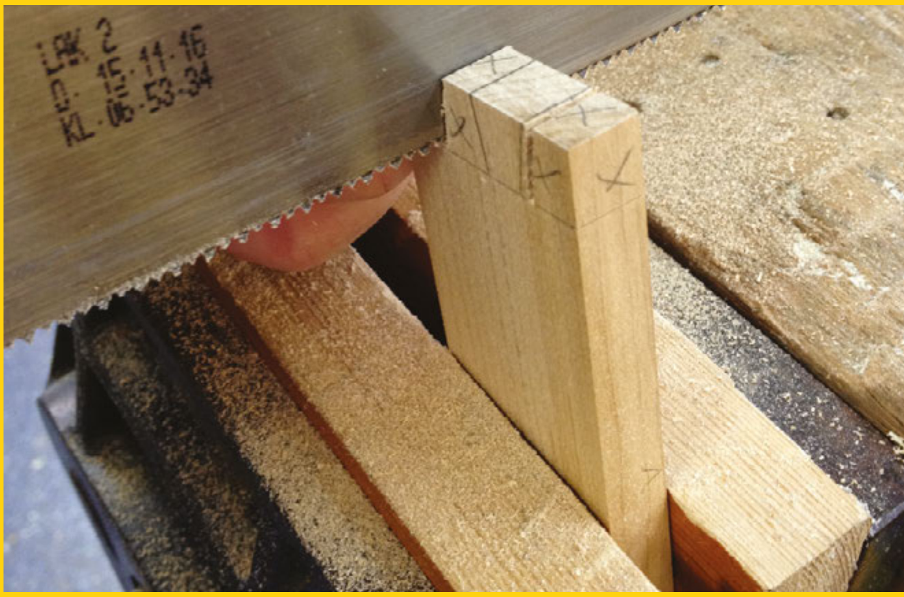
Now cut the lengths for your box, placing your wood on top of a scrap piece that you're happy to sacrifice to stop the wood tearing out underneath as you cut.

MARK OUT THE DOVETAILS

Mark out the shape of the dovetails onto the end grain and inside/outside faces of the long box sides using a pencil, try square, and sliding bevel or ruler.

Don't yet mark out the dovetails sockets/pins – you will do this to precisely match the shape of your dovetails once they have been cut. Mark the waste side of all of your lines with a pencil cross so that it's →

Below
Before you pick up your chisel, saw along the lines you marked



really clear which sections of wood you are keeping and which you will remove.

Use a dovetail or fine tenon saw to cut on the waste side of all of the lines. Use a try square to position the lines you are cutting at 90 degrees to the bench – it's far easier to saw straight down than to try and lean a saw to the correct angle.

Remove the waste wood from between the dovetails, either with a coping saw or narrow blade (that can turn the corner at the bottom of one of your saw cuts) or by using a series of chisel cuts.

CHISEL THE DOVETAILS AND SHOULDERS

Use a bevel-edge chisel to pare away thin slices of wood from first the shoulders (the gaps in between the dovetails) and then the cheeks (sides) of the dovetails, back to your pencil lines.

Use the try square to check that all your surfaces are flat and square, marking high points in pencil to be chiselled away – squareness is more important than paring exactly back to your lines of the dovetail shape, as you will mark out the sockets to fit the resulting dovetails.

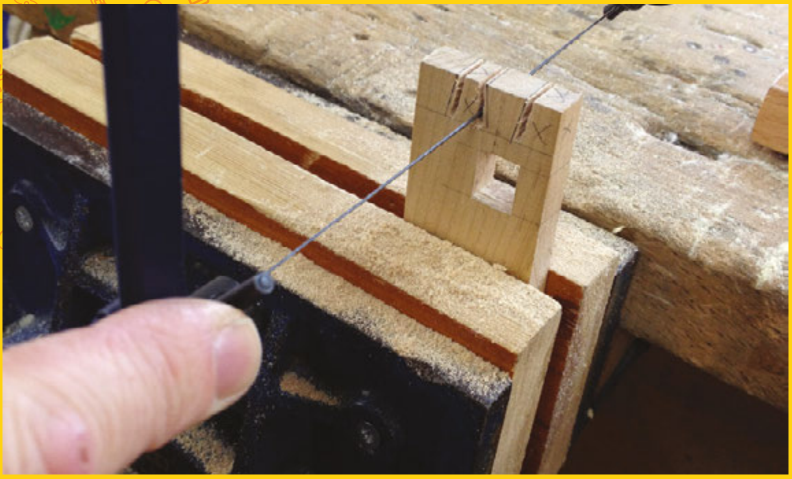
A good way to chisel square shoulders is to use a scrap piece of wood (fence) clamped in line with the shoulder line to position your chisel. Clamp the box side and fence flat to the bench and slide the chisel down the fence. It's difficult to check if the shoulder in between two dovetails is square using a try square due to the restricted view. If the blade of the try square touches the edge of the wood when the stock is held against both inside and outside faces, though, you can deduce that the shoulder is square.

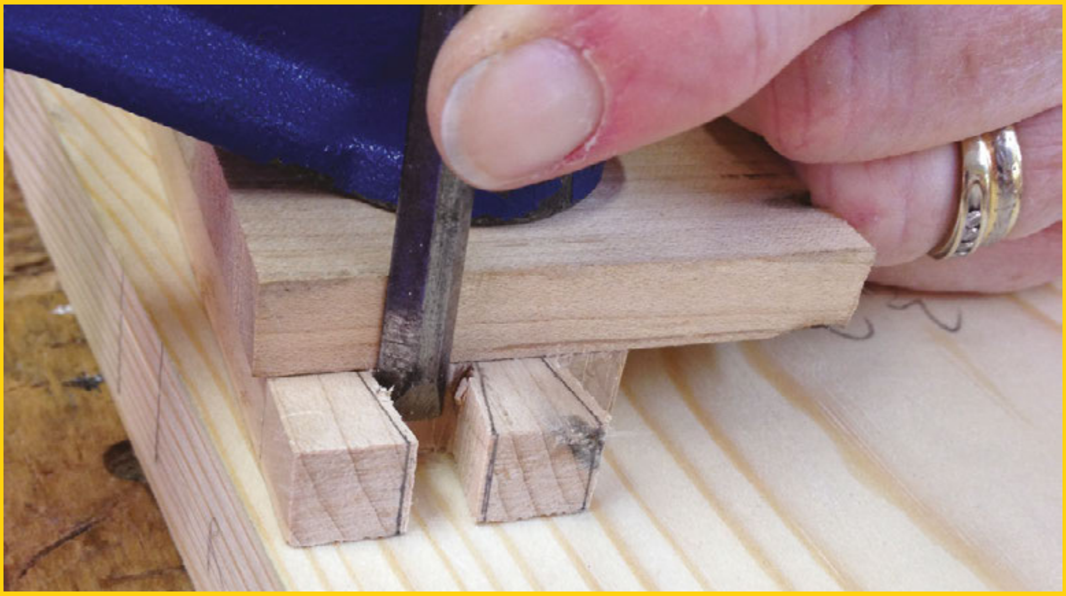
When paring the cheeks, position the surface at 90 degrees to the bench so that you are chiselling straight down.

MARK OUT AND SAW THE SOCKETS

Label both sides of each corner with the same letter A/B/C/D so that you can be certain that you are always putting the same ends together. Then, draw the shape of your dovetails onto the end grain of each matching piece, making sure that the inside faces are correctly positioned facing each other and the edges of both pieces are flush.

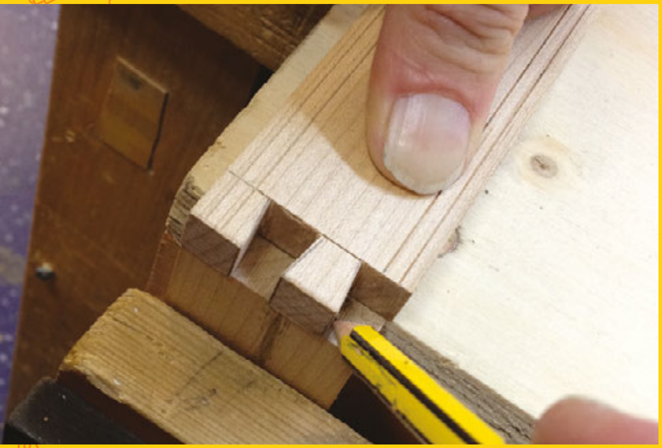
Continue those pencil lines straight down to the shoulder lines on both the inside and outside



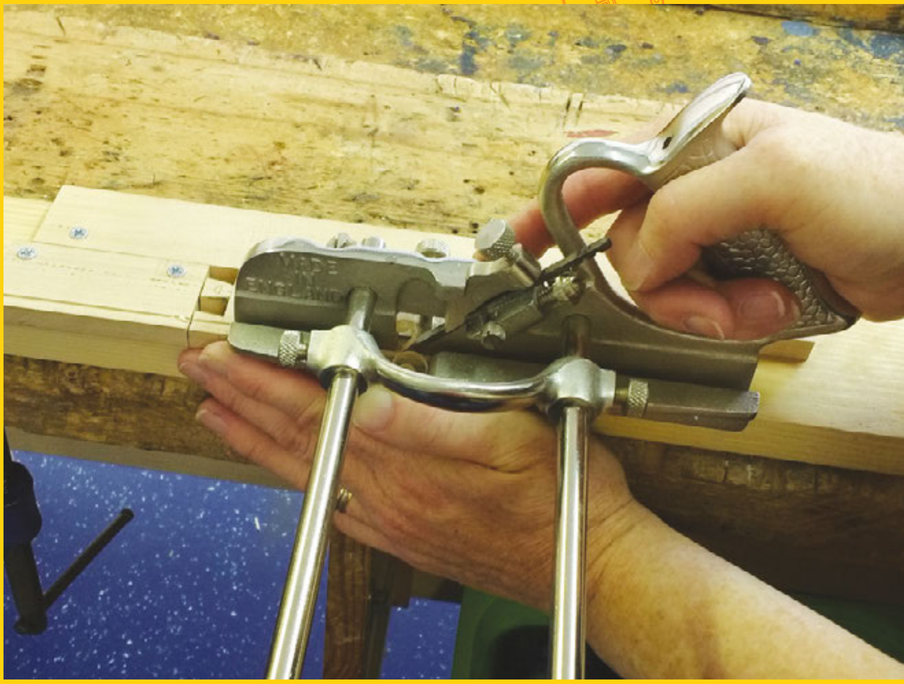


Above Mark pencil crosses on the waste side of each line to clearly show the areas to be removed

Right Cut on the waste side of all the lines you've marked



Above
Marking out the sockets, following the lines of the dovetails



Above
This is a plough plane: use it to carve grooves for the box lid to slide into

faces. As with the dovetails, position the saw cut at 90 degrees to the bench, saw on the waste side of the pencil lines, and remove material from between the pins with a coping saw.

CHISEL THE SOCKETS

Chisel the shoulders and then the cheeks of the sockets flat and square, as you did with the dovetails.

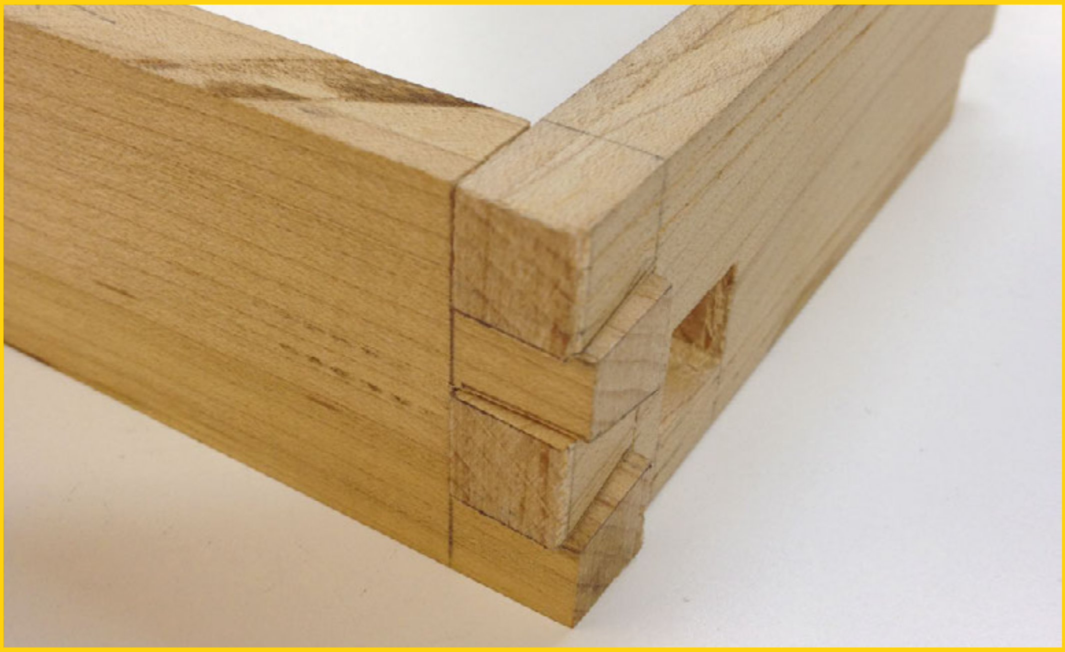
Make sure that you don't chisel away the pencil lines on the end grain – these lines were marked on the outside of your dovetail shapes and if you lose them, your joint will be loose.

Keep trying to fit the two sides of the joint together as you chisel back to the lines – it will often go together before you think it is ready.

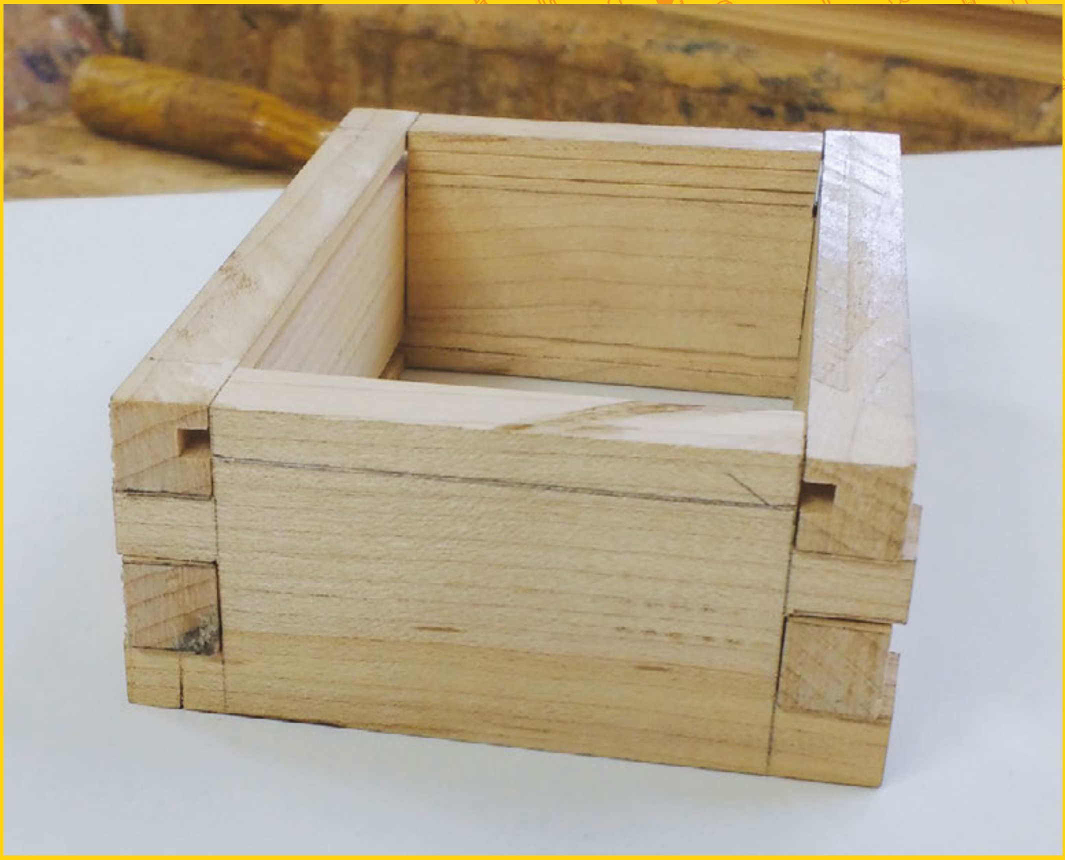
PLOUGH THE GROOVES FOR THE BASE AND LID

On the dovetail sides, the grooves for the base and lid can be cut using a plough plane. This plane has a narrow cutter and a moveable fence that can be set at the required distance from the cutter. When pressed against the edge of the wood, the fence ensures the groove is straight.

Mark the depth of the groove in pencil on the end grain at both ends of the dovetail sides. →



Left
A strong joint depends on corresponding surfaces touching each other on assembly so that the glue can form a bond



Right
Assemble the box once before you glue it together, just to make sure it all fits together

You'll need to set up a jig using offcuts of your wood screwed down onto a base to surround the box side and hold it in position, as it is too small to use clamps directly – they would be in the path of the plane.

Set the distance between the plane fence and the cutter to 3 mm. If the plane has a depth stop, set this also to 3 mm. Press the fence firmly against the edge of the wood to ensure accuracy and try and remove even layers along the whole length. The lid grooves will run through the half dovetails.

If you don't have a plough plane, you can pare away the waste wood from inside the groove with a narrow chisel. Be careful!

The base grooves of the socket sides need to be chiselled out, as opposed to plough-planed, because they need to stop short of the end. Ensure you chisel across the fibres first to break them and then gradually pare away thin layers to minimise the risk of splitting. A depth of 1 mm is sufficient to hold the base in position.

MAKE THE LID AND BASE

We chiselled a rebate into the edges of the base and positioned the raised section inside the box in order to reduce the internal dimensions. This meant that the power source hole could be positioned further away from the bottom edge, giving it more support from the surrounding wood. If you're using solid wood, mark the rebates out with the marking gauge set to 3 mm and use a wide bevel-edge chisel to pare away thin layers down to the line. Make sure the inside corners are clear of all fibres so that the lid runs smoothly inside the grooves.

If you're using plywood, you can build up a rebated lid and base by gluing two layers together, with one layer slightly smaller than the other.

The base needs to extend into grooves on all four sides. The lid only needs to fit into grooves along the dovetail sides and can be cut flush at either end.

The socket sides of the box need to be reduced in height to allow the lid to pass over the top of

" If you're using plywood, you can build up a rebated lid and base **by gluing two layers together,** with one layer slightly smaller than the other "

them. Assemble your box and mark the required height. You can remove the unwanted sections with saw cuts and then sand or plane to a smooth finish.

GLUE UP

Before you glue up any project, you should fit all the pieces together without glue to make sure there are no adjustments to be made.

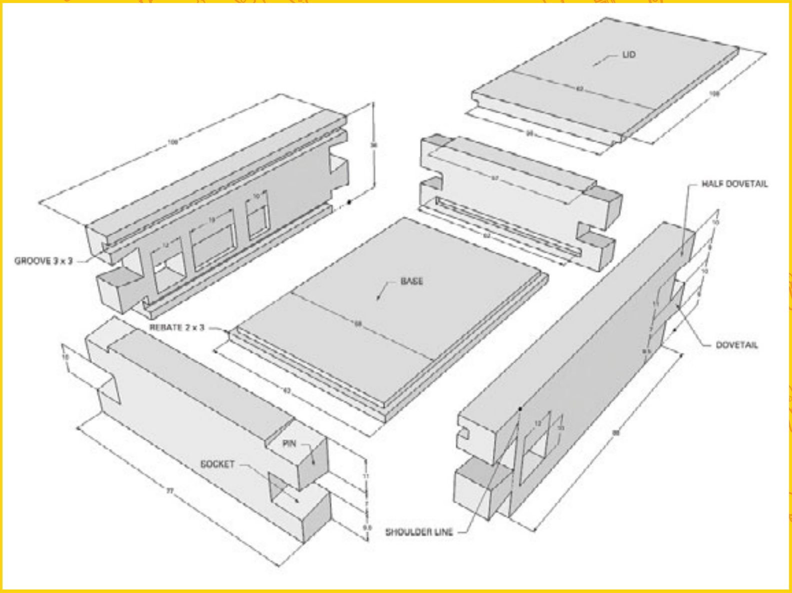
Once you're happy with the fit, lay out your pieces and apply wood glue to the inside faces, cheeks, and shoulders of the dovetails only – no glue is required on the socket sides or the base and lid.

Clamp the box together in both directions, using small sections of scrap wood to protect it from being damaged by the clamps. Allow 24 hours for the glue to dry.

FINISHING

Your box will need to be either planed, scraped or sanded to clean up the surfaces ready for finishing. If you use a plane, take care not to plane off the edge of any areas of end grain, as they will splinter out. If you use sandpaper, always sand in the direction of the grain to disguise scratches. Start out with a coarse paper to remove marks and then work up through the grades to a fine finishing paper, removing the scratches of the previous coarser grade as you go.

Danish oil is a quick, easy, and durable finish. Apply this sparingly with a soft lint-free cloth, using a small brush to get into corners. Remove any excess Danish oil, particularly in the grooves and the edges of the lid. This will ensure that the box opens and closes smoothly after finishing. Well done! Now your Raspberry Pi projects can have a nice new home.



Above
Here are the measurements if you want to make your own Pi box

Left
The finished article: lending gravitas to our Pi web server

3D CAD modelling for beginners

Start printing your own designs with this phone stand modelling tutorial

Cameron Coward

cameron_coward

Cameron Coward is a mechanical designer, writer, and author specialising in hacker and maker tech. Find out more about him and his work at **cameroncoward.com**

YOU'LL NEED

- **Windows, Mac, or Linux computer**
- **An internet connection**
- **Autodesk Fusion 360** (free for students or hobbyists)
- **Callipers, a measuring tape, or a ruler**
- **Access to a 3D printer** (if you'd like to print your design)

You've had that 3D printer for a little while now; it was so exciting when it showed up in the mail, chock-full of gleaming metal and possibilities, wasn't it? But, like most people, you've likely started to get bored with printing pencil holders, coasters, and figurines that you find online. Those trinkets are fun for a little while, but the real potential of your 3D printer is in printing your very own designs.

Designing your own 3D models means using CAD (computer-aided design) software. Unfortunately, CAD software has a very steep learning curve, and is difficult for beginners to get started with. After all, there are entire college degree programmes devoted to becoming proficient in CAD. But, that doesn't mean there isn't hope! Learning to use CAD is much less about where all of the individual commands are, and more about understanding the concepts involved.

In this article, you'll learn how to use those basic concepts to model your own smartphone stand. With these skills, you'll be able to design your own ideas. But, more importantly, you'll gain an understanding of how to approach 3D modelling so that you can continue to expand your knowledge. Everyone needs a starting point, and if you've been looking to get started with 3D modelling, this is it!

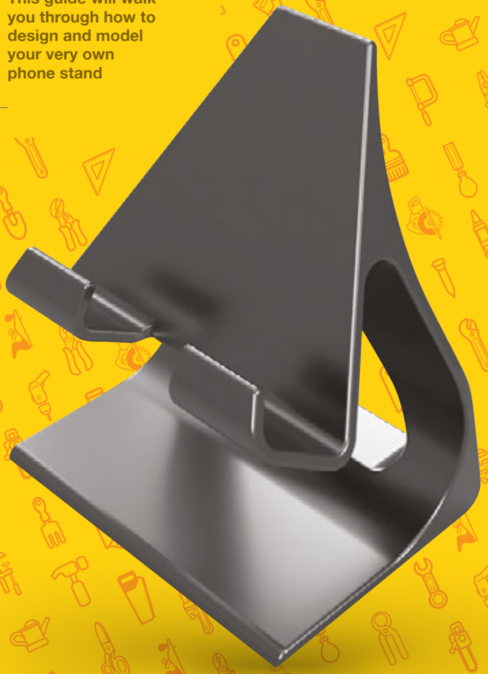
The first thing you'll need to do is download Autodesk Fusion 360, which can be found with a Google search. Fusion 360 is on par with professional CAD software, but is free for students and hobbyists to use. Unlike 3D sculpting programs, like Blender or Maya, parametric CAD software is intended for engineering purposes and all of the modelling is handled mathematically.

With a 3D sculpting program, you mould your design like you're working with digital clay – it's a visual process. Parametric CAD software requires that you define features with dimensions and relationships (the parameters). The benefit is twofold: the resulting model maintains the precise dimensions that you specify, and it stores the entire design history so you can modify individual features later.

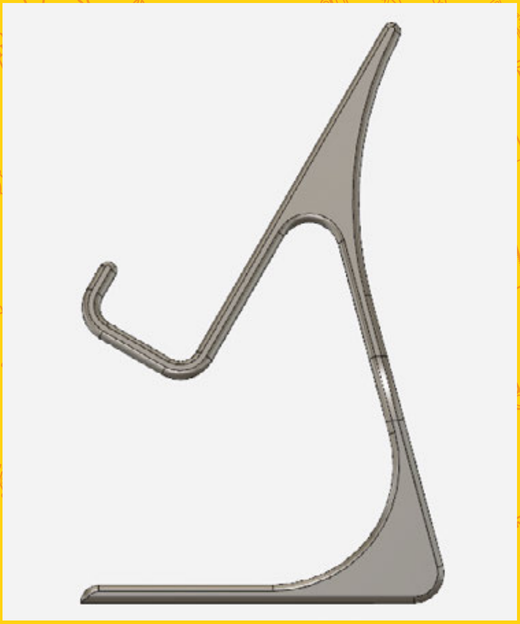
CAD LOOKS SCARY; TAKE YOUR TIME

Once you've downloaded Fusion 360, take some time to familiarise yourself with the layout of the interface. The primary part of the window is the viewport, where you'll interact with the model. The viewport is fully three-dimensional, and the model can be rotated and viewed from different angles while you're working on it. The View Cube at the top right of the viewport lets you quickly rotate the model to predefined angles.

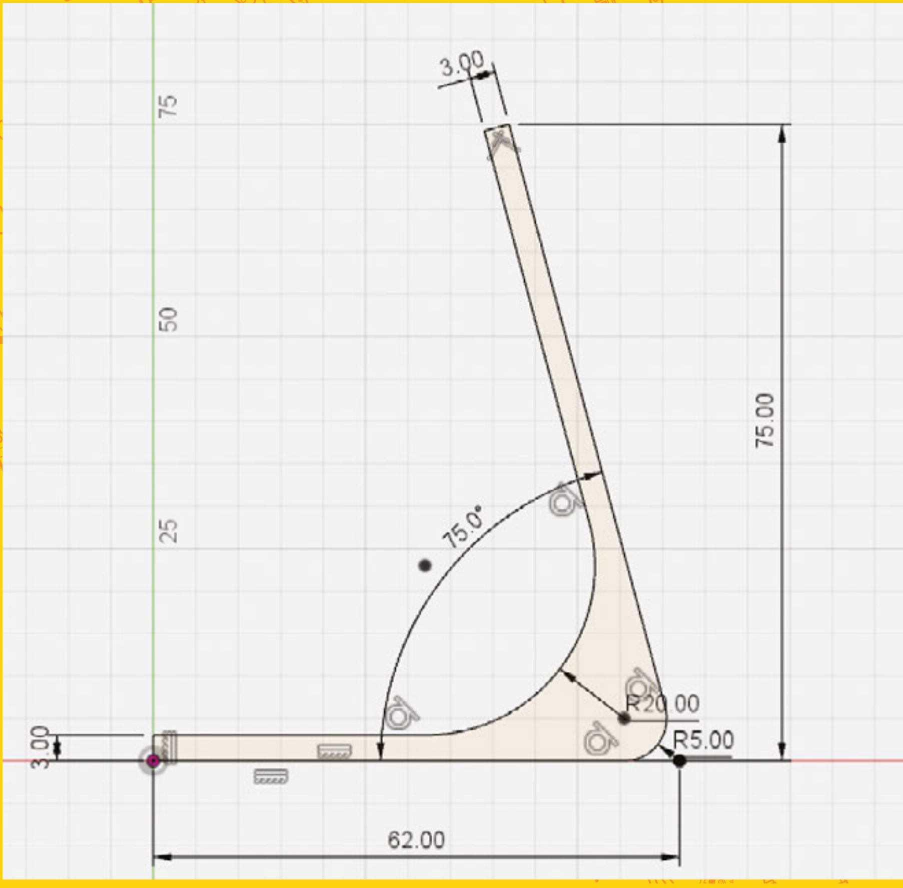
The main toolbar along the top of the viewport contains all of the tools you'll need for modelling.



Below This guide will walk you through how to design and model your very own phone stand



Above When you begin modelling a new design, start by finding the angle that gives you a basic feature for your first sketch



Above Keep individual sketches and features simple. Overreaching by trying to put everything in a single feature will only make the process more difficult

On the left-hand side of the main toolbar is a drop-down for switching workspaces. Each workspace is set up for different tasks; for instance, you use the Render workspace to set up a realistic rendering of your model for presentation purposes. You'll find everything you need for this guide in the Model workspace.

Running down the left-hand side of the viewport is the component browser, which has information and settings for units, views, and buttons for toggling object visibility. The bottom of the window will show the design history as you create features, and you can access each step to modify it. Finally, the top left-most button opens the Data Panel, where you can manage your projects and design files.

You're modelling this stand for your actual phone, so you'll need to take some measurements of it. A pair of callipers are going to give you the most accurate measurements. If you don't have callipers, a ruler or measuring tape will work. You'll need to measure the length, width, and thickness of your phone. This tutorial will use millimetres, but you can work with inches if you'd prefer.

BREAK THE DESIGN DOWN INTO PARTS

The key to CAD modelling is breaking the model down into a series of individual features. For instance, if you were modelling a cup you might make the first feature a cylinder, and use a second feature to make that cylinder hollow. While there is no right way to model something, there are some best practices. Multiple simple features are generally better than a single complex feature, as there is less opportunity for error and the features are easier to modify later on if need be.

We'll cover a good way to break the phone stand down into a series of features. But, first try to visualise the stand as a 3D object, and consider how it can be reduced down into simple steps. Looking at the end result might make the model seem complex, but each individual feature is actually quite simple. Think of it a bit like building a LEGO set: each block is very basic, but when added together they can make an intricate design.

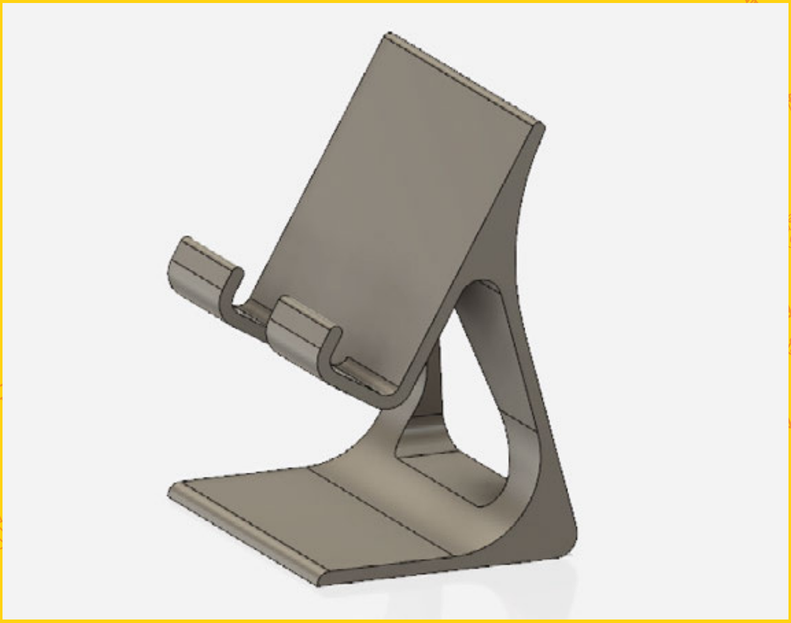
The first modelling step is to create an extrusion to act as the base of the model. The Extrude feature is definitely the most commonly used tool for the vast majority of designers. It takes a 2D sketch profile and give it thickness to form a 3D solid. It can also do the opposite, and cut that sketch profile out of an existing solid. If you picture the phone stand from the side, you'll notice that it has a consistent silhouette, so starting from that angle is a good idea.

KEEP SKETCHES SIMPLE

Sketching that entire silhouette, however, would make this first feature pretty complex. Instead, →

QUICK TIP

Often, when modelling for 3D printing, it's a good idea to take into account the need for supports and to avoid overhangs when possible.



Above Don't settle for just 'functional' in your designs. Take the time to give it some finishing touches to make it look good!

THE ORIGIN

A number of predefined references are automatically created when you start a new model. The centre of all of these is the Origin, which everything else references. The X, Y, and Z axes all intersect at the Origin, and three planes are created by these axes. In Fusion 360 the XY plane is usually called the Front plane, the XZ plane is called the Top plane, and the YZ plane is the side (or right/left) plane. These correspond to the views named on the View Cube.

start with only the base of the stand and the upright support. Click the Create Sketch button from the main toolbar, and select YZ (side) plane as the sketch plane. Then, use the drawing tools under the Sketch menu to draw lines to form the outline of this first feature. Both the base and upright should be 3 mm thick, the base should be about 62 mm long, and the upright should be about 75 mm tall. The angle between the two should be roughly 75°. These dimensions can be set using the Sketch Dimension tool.

Adding fillets to the corners where the angled lines of the upright meet the horizontal lines of the base is a good idea. Fillets round those edges, making the model stronger and more professional-looking. Unless you have a reason to leave them sharp, you should round most corners and edges with fillets, or blunt them with chamfers. A fillet is defined by the radius of the arc connecting two lines, while a chamfer is defined by the length and angle of the chord.

"Your sketches should always be fully defined, **which means the lines can't be moved**"

Your sketches should always be fully defined, which means the lines can't be moved. Geometric constraints can help with this, and are in the Sketch Palette. The bottom line of the base, for example, needs to be horizontal. So, to lock that, you'll select that line and then click the Horizontal/Vertical constraint. When a line is fully constrained, it will change from blue to black. Make sure all of your lines are black before continuing, and the sketch should be constrained to the Origin point.

TRIAL AND ERROR

Almost every feature will have a dialogue box that contains a variety of options that affect the resulting 3D solid body. The most common options are the selection (like what you use to choose the sketch profile), and the primary dimension (like the distance of an extrusion). Try changing these options to see what change they make. And, if a feature ever gives you an error or acts unexpectedly, check these options to see if they fix the issue.

EXTRUDE IS A POWERFUL TOOL

Once you have completed the sketch, choose the Extrude tool from the Create menu to give it thickness. This is where the width measurement you took of your phone comes in. You'll select the closed profile that you sketched, and extrude it to match the width of your phone. It's best to make this Symmetric, so that the part is centred on the origin. Make sure the Measurement option is set to overall distance, and make the Distance equal to the width of your phone.

The next feature is an extruded cut through the upright support. This will reduce the amount of material in the final model, and adds visual appeal. The sketch for this feature needs to be perpendicular to the last one, so choose the XY (front) plane as the sketch plane. Then, draw a shape to cut out (like a triangle), making sure to leave enough material on all sides of the upright to keep it strong. When you extrude your sketch, set the operation to Cut, and make the Distance Through All.

To make the actual platform that your phone will rest on, you'll create yet another extrusion. This will be on the same plane as your first extrusion, and needs to form a J shape to hold your phone. The angle of this platform should be around 60° from horizontal. Make the height about half the length of your phone, and the space between it and the front flip should be slightly larger than the thickness of your phone. This will allow you to easily rest your phone on the stand, while still keeping it good and secure.

The final feature, to finish the base structure, is a cut-out for your phone's charging port. Cut this out of the bottom support and lip of the platform. Sketches don't have to be on the predefined planes, and the sketch for this extruded cut can be the outer face of the lip. Simply draw a rectangle that's the same height as the slip, make it wide enough to access the charging port, and extrude the cut to the face of the platform.

There is one more cut to consider, and that's from the platform. There is no reason to have a big, bulky rectangle for your phone to rest on. Removing material will speed up the printing process and reduce how much filament is needed. Create a sketch on the platform, and cut out the superfluous material. The shape can be whatever you'd like, so make it aesthetically pleasing to you.

AESTHETICS ARE IMPORTANT

Now add some finishing touches, which will be fillets and/or chamfers. The most important ones are where the platform meets the upright support, to give the stand rigidity and strength. When you're adding this to an existing solid, there is no need to draw a sketch. Just choose the Fillet or Chamfer tool from the Modify menu, and select the edges you want to smooth out.

For visual appeal, it's good practice to add a fillet or chamfer to any edge that doesn't need to be sharp. Breaking these edges gives the model a more finished appearance, and removes sharp edges that might be uncomfortable to handle. Keep adding fillets until you're satisfied with the overall look of your phone stand.

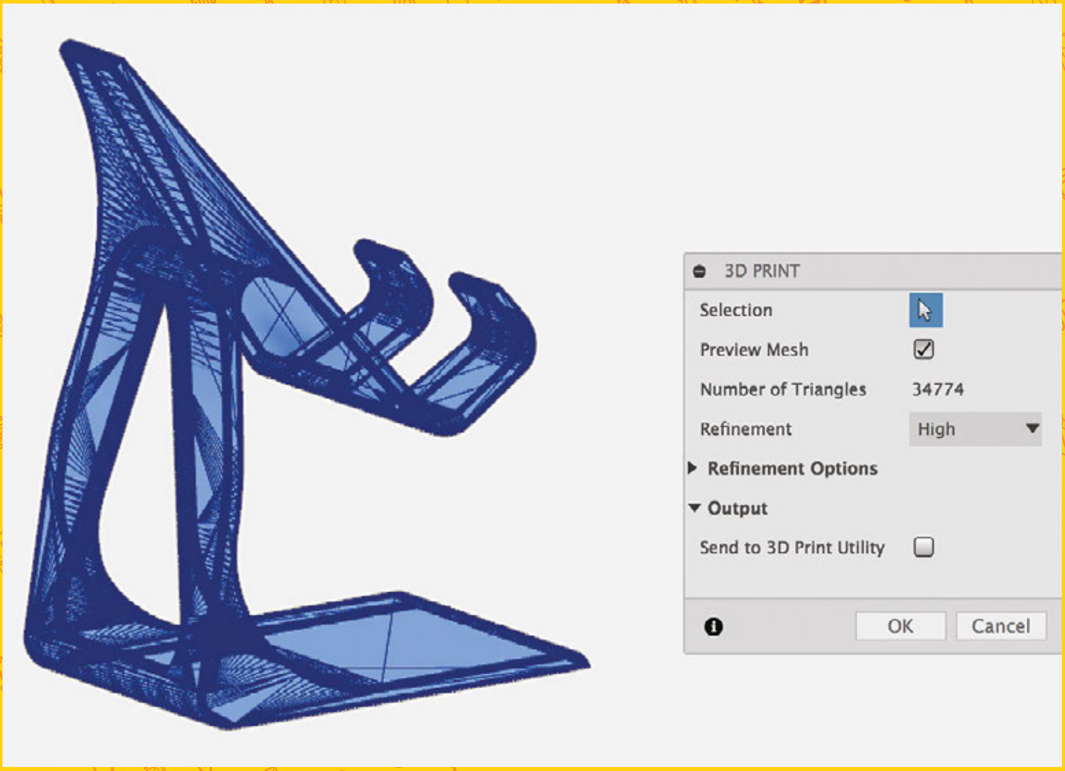
FILLETS AND CHAMFERS

The radius of a fillet (pronounced 'fill-it') cannot be larger than the space available, so if you get errors, try making the fillet smaller. A well-placed fillet can make all the difference in the perceived quality of the final design. Chamfers work well in place of fillets when a more modern and angular aesthetic is desired. Most designs will probably benefit from some combination of both fillets and chamfers.

PRINT THAT MODEL!

With the model done, you can prepare it for 3D printing. From the Make drop-down menu, click 3D Print to bring up the STL settings dialogue box. Select the model, and set the options to your liking. Unless you're trying to make the STL file small, it's a good idea to use the High settings to make the resulting STL mesh as detailed as possible. You can then either output the STL to your 3D printing software, or save it somewhere on your computer.

And that's it, you're done! While there are lots of other types of features and tools that weren't covered in this guide, you can do a great deal of modelling with what you've learned here. The next time you want to 3D-print something, try using these tools to create the model yourself! □


3D PRINT
Selection
Preview Mesh ✓
Number of Triangles 34774
Refinement High
Refinement Options
Output
Send to 3D Print Utility ☐

QUICK TIP

Always consider how your creation will be used. For instance, you might want to add a small clip or guide to keep your charging cable in place.

Left
Once you're done modelling your design, saving an STL for 3D printing is trivial. For quality, keep the refinement high

Add Arduino power to your projects

Get started with coding for the Arduino platform

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**

So you want to start programming microcontrollers and doing some cool projects with the hardware. You've selected Arduino as your starting platform, purchased a popular Arduino board, and you're ready to get started. What's next? In this short article, we'll show you how to get started coding for Arduino.

Arduino (**arduino.cc**) is a very popular hardware platform for computer-controlled hardware projects. Arduino is a small, inexpensive, programmable microcontroller that exposes a multitude of input and output (I/O) connections you can use to create computer-controlled circuits, wiring in switches, lights, sensors, and more. It's an open hardware platform, which means that the hardware specification is open source, so anyone with the means can design and distribute their own Arduino-compatible hardware. Therefore, there's a series of devices made by arduino.cc and a bevy of 'compatible' devices from other vendors as well.

To program an Arduino device, you'll code applications in a language similar to very old programming language called C; these applications are called sketches. Because the Arduino is basically a small computer system, although with limited processing speed and memory, the platform supports a subset of the capabilities of C. You'll code your Arduino applications in an integrated development environment (IDE); Arduino offers both a locally installed IDE or a cloud IDE to use for your projects. There are alternative IDEs available as well; you can find a list of options at **hsmag.cc/aQJqkJ**.

When creating sketches, you'll code your sketch in an IDE, then connect your Arduino compatible device to your PC using an USB cable. With that in place, the IDE compiles your sketches into executable code, then downloads them to the Arduino device over the cable. As your sketch runs, you can pass data between the IDE and your Arduino device over a serial communication channel enabled in the IDE (shown in **Figure 1**). Once compiled code is deployed to the device, the device resets and, once the device completes initialisation, it executes the sketch.

An Arduino sketch consists, at a minimum, of two parts: code that runs once, and code that runs repeatedly. Let us show you.

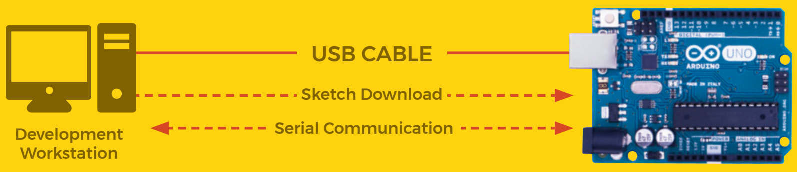
In the Arduino IDE (described later), an empty Arduino sketch looks like this:

SERIAL COMMUNICATION

The serial communications capabilities of the Arduino platform expose additional capabilities for your sketches. At a minimum, you can use serial communication to send data back to the IDE while you're troubleshooting your sketches. To do this, open the IDE's Tools menu and select Serial Monitor. A new window opens, and any data written using the Serial commands (described at **arduino.cc/en/Reference/Serial**) will appear in the monitor window.

You can also use serial communications to transfer collected data (from sensors connected to the Arduino board) to another computer system like a Windows PC or a Raspberry Pi. Makers often do this since the Arduino supports analogue inputs and the Raspberry Pi does not. In this scenario, the Arduino becomes merely a data collection device, and the Raspberry Pi does whatever number crunching is appropriate for the project, potentially even displaying data on a connected screen or uploading the data to a remote server for processing.

Figure 1 Arduino development architecture



```
/*
*/
void setup() {
}
void loop() {
}
```

The first part of the sketch is a comment block. Anything, absolutely anything you enter between the **/*** and ***/** characters is ignored by the Arduino compiler.

```
/*************************************
 My First Arduino Sketch

 by John M. Wargo
  December, 2017

Meatloaf meatball pork ground round fatback
kielbasa cow porchetta pork loin ball tip. Spare
ribs picanha drumstick pork jerky cupim alcatra
meatball beef ribs. Ball tip ground round
pastrami pancetta shank kevin.
**************************************/
```

In your sketches, you'll use this commenting approach when you have multiple lines of content you want displayed within the sketch. At a minimum, use a comment block at the beginning of the sketch to describe the sketch, as we've done in the example, using dummy content from the Bacon Ipsum generator (at **baconipsum.com**). You should also use block comments like this to describe important parts of your sketches.

You can also add single-line comments to your sketches. To do this, start any line in your sketch with double forward slash characters (**//**) or after any of your code. All content following the double forward slashes is ignored by the Arduino compiler. In the following example, a single-line comment precedes the definition of the **numCols** variable. The comment and the executable code are on separate lines, so we started the comment line with the double forward slashes.

```
//Number of columns in the table
int numCols;
```

Or something like this where the comment follows the definition of the **relayStatus** variable:

```
bool relayStatus;  //The current status of
the relay (on/off)
```

The sketch's **setup** function is defined with the following code:

```
void setup() {
}
```

Any code you add to this function (you'll add your code between the curly braces **{}**) is executed by the Arduino device as soon as you power it up and the hardware finishes initialising. This function is executed only once; you'll use this function to set up your sketch and execute the things that only need to be done when the sketch starts.

You'll normally use this to define the configuration of your hardware; as many input/output (I/O) connectors on the Arduino can be used for either input or output, you'll have to tell your sketch how you intend to use them. We'll show you an example of this in a little while.

The final component of a minimalist sketch is the **loop** function:

```
void loop() {
}
```

In this function, put any code that you want to run repeatedly on the Arduino. The Arduino executes the **setup** function once, then executes the **loop** function over, and over, and over again until either the Arduino explodes (it won't, we're just kidding) or you disconnect power from the device. You can put all your code in the loop, or break your code into smaller functions and call those functions from the **loop** function.

> **"Arduino is a small, inexpensive, programmable microcontroller** that exposes a multitude of input and output (I/O) connections"

To see all of this in action, look at the following example. By default, the Arduino developer tools include a simple sample sketch called Blink.

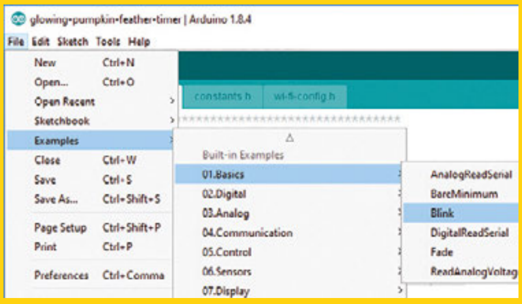
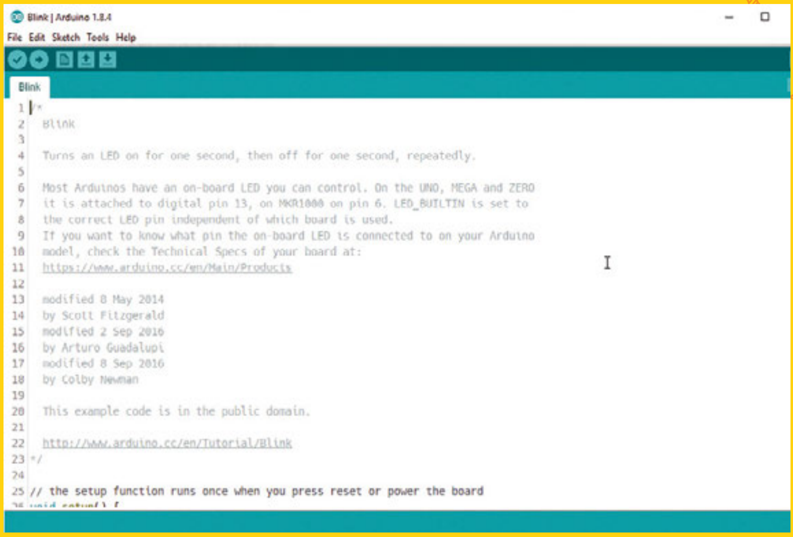


Figure 2 Opening the Arduino Blink sketch

YOU'LL NEED

- **An Arduino-compatible board**
  An actual Arduino device is preferred as there's extra setup required for many Arduino compatible boards. The recommended starter board is the Arduino Uno (hsmag.cc/QKaKXM) or the newer, and more capable, Arduino Zero (hsmag.cc/KGJbVd)
- **Microsoft Windows, Apple macOS, or Linux**
- **Universal serial bus (USB) cable**
  To connect the Arduino device to your computer system. Arduino on-device connectors vary; most use a micro-USB connector, but the Uno uses a USB A/B cable



```
/*
  Blink

  Turns an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
  the correct LED pin independent of which board is used.
  If you want to know what pin the on-board LED is connected to on your Arduino
  model, check the Technical Specs of your board at:
  https://www.arduino.cc/en/Main/Products

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Blink
*/

// the setup function runs once when you press reset or power the board
```

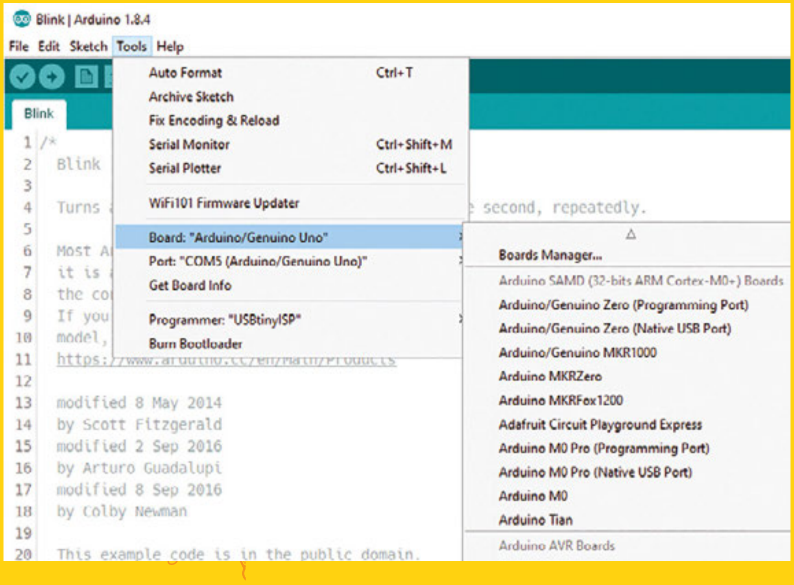
Above The Arduino Blink sketch

Most → Arduino devices include an LED on board, hard-wired into one of the Arduino's I/O ports. The included Blink sketch enables you to quickly accomplish something with the Arduino – turning that on-board LED on and off repeatedly.

Note: The Blink sketch starts with a long and thorough introductory comment block that we're omitting here for brevity's sake. We'll show you how to open the sketch soon, so you'll be able to study the whole sketch in detail.

```
// the setup function runs once when you press
reset or
// power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an
output.
  pinMode(LED_BUILTIN, OUTPUT);
}
```

Below Configuring the IDE for the connected Arduino board


Blink | Arduino 1.8.4
File Edit Sketch Tools Help
Blink
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
Board: "Arduino/Genuino Uno"
Port: "COM5 (Arduino/Genuino Uno)"
Get Board Info
Programmer: "USBtinyISP"
Burn Bootloader
Boards Manager...
Arduino SAMD (32-bits ARM Cortex-M0+) Boards
Arduino/Genuino Zero (Programming Port)
Arduino/Genuino Zero (Native USB Port)
Arduino/Genuino MKR1000
Arduino MKRZero
Arduino MKRFox1200
Adafruit Circuit Playground Express
Arduino M0 Pro (Programming Port)
Arduino M0 Pro (Native USB Port)
Arduino M0
Arduino Tian
Arduino AVR Boards

```
/*
  Blink

  Turns                                    second, repeatedly.

  Most A
  it is
  the co
  If you
  model,
  https://www.arduino.cc/en/Main/Products

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.
```

```
// the loop function runs over and over again
forever
void loop() {
  // turn the LED on (HIGH is the voltage level)
  digitalWrite(LED_BUILTIN, HIGH);
  // wait for a second
  delay(1000);
  // turn the LED off by making the voltage LOW
  digitalWrite(LED_BUILTIN, LOW);
  // wait for a second
  delay(1000);
}
```

In the **`setup`** function, there's only one executable line:

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

Calling **`pinMode`** sets the configuration for one of the Arduino's I/O pins. In this case, its configuring the I/O pin defined in **`LED_BUILTIN`** for output mode. Remember, most Arduino boards have an on-board LED; the Arduino team has preconfigured the Arduino development environment to store the I/O pin associated with each Arduino board in a variable called **`LED_BUILTIN`**. Any time the sketch references **`LED_BUILTIN`**, the compiler replaces the reference with the actual pin number to which the LED is connected. The Arduino Zero has its LED wired to I/O pin 13, so for the Zero, the code is essentially:

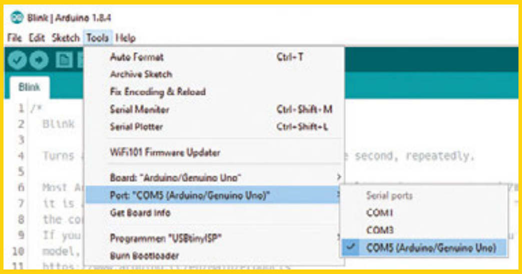
```
pinMode(13, OUTPUT);
```

With this in place, the sketch knows that when working with pin 13, it will be outputting (sending a voltage) to the pin, not receiving input.

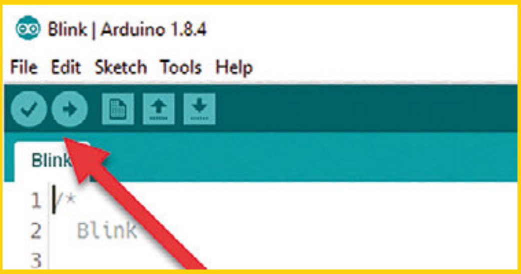
In the `loop` function, the code completes the following steps:

- Uses the **`digitalWrite`** method to set the output voltage on the **`LED_BUILTIN`** pin to **`HIGH`**. This means that the pin gets a voltage equivalent to the current operating voltage of the Arduino. Some Arduino devices operate at 3 V and others at 5 V; all that's important to know here is that with execution of this code, the Arduino is now powering the LED connected to the I/O pin at its brightest.
- Waits for 1000 milliseconds (1 second) using the **`delay()`** method.
- Uses the **`digitalWrite`** method to set the output voltage on the **`LED_BUILTIN`** pin to LOW. This translates to no voltage (0), essentially turning the LED off.
- Waits for 1000 milliseconds (1 second) using the **`delay()`** method.

Below
Setting the IDE's communication port



Below
Compile and Deploy buttons



When the code runs, it will turn the LED on for 1 second, then off for 1 second, repeating the process until you remove power from the device or deploy a different sketch.

Now it's time to see the sketch in operation. To do this, you'll start by installing the Arduino IDE on your computer system. Open your browser of choice and navigate to **arduino.cc**. On the site's top menu, click the Software link, then, on the page that opens, download the latest version of the Arduino IDE for your system's operating system. Once the download completes, launch the downloaded file to begin the software installation.

Once the installation completes, start the Arduino IDE. In the Arduino IDE, open the File menu, select Examples, then 01.basics, then Blink, as shown in **Figure 2** (page 83).

```
Archiving built core (caching) in: C:\
Users\JOHNWARGO\AppData\Local\Temp\arduino_
cache_950966\core\core_arduino_avr_uno_
c3bfe3f79ffbeab93536a1a484b588d9.a
Sketch uses 928 bytes (2%) of program storage
space. Maximum is 32256 bytes.
Global variables use 9 bytes (0%) of dynamic
memory, leaving 2039 bytes for local variables.
Maximum is 2048 bytes.
```

If the verification fails, the IDE will display information about any errors and reference the sketch line number where the error was found. You'll need to fix any errors before continuing to the next step.

Finally, click the Upload button; the IDE will repeat the verification step, then deploy the compiled sketch to the connected Arduino device. When the upload process completes, the Arduino device will immediately reset, then begin executing the new sketch. In this example, the Arduino will turn its on-board LED on and off repeatedly until power is removed from the board or a different sketch is uploaded.

> "To program an Arduino device, you'll code applications in a language similar to **an old language called C;** these applications are called sketches"

Now it's time to play around with the code. If you remember from earlier, the sketch uses delay statements to control the amount of time the LED is on and off. Right now, they're coded to pause 1 second (1000 milliseconds); modify the code so the LED stays on for half a second (500 milliseconds) and pauses for two seconds (2000 milliseconds) in between. Upload the modified code to the board and see what happens. □

NEXT STEPS

We've only lightly brushed the surface of what you can do with the Arduino platform. To make it easier for Arduino developers to get started, the IDE includes a whole catalogue of example applications you can study and use to expand your skills. To access these examples, in the Arduino IDE, open the File menu, select Examples, then look for a sketch category that appeals to you. The Basics category offers some simple sketches you can use to expand from where we've started here. There's a simple sketch to fade the on-board LED up and down (instead of turning it on and off, as in the Blink example). There are also sketches for reading analogue or digital signals; you'd use these with the appropriate analogue or digital output device connected to the Arduino. The other sketch categories offer more sophisticated sketches that work with different hardware devices and more.

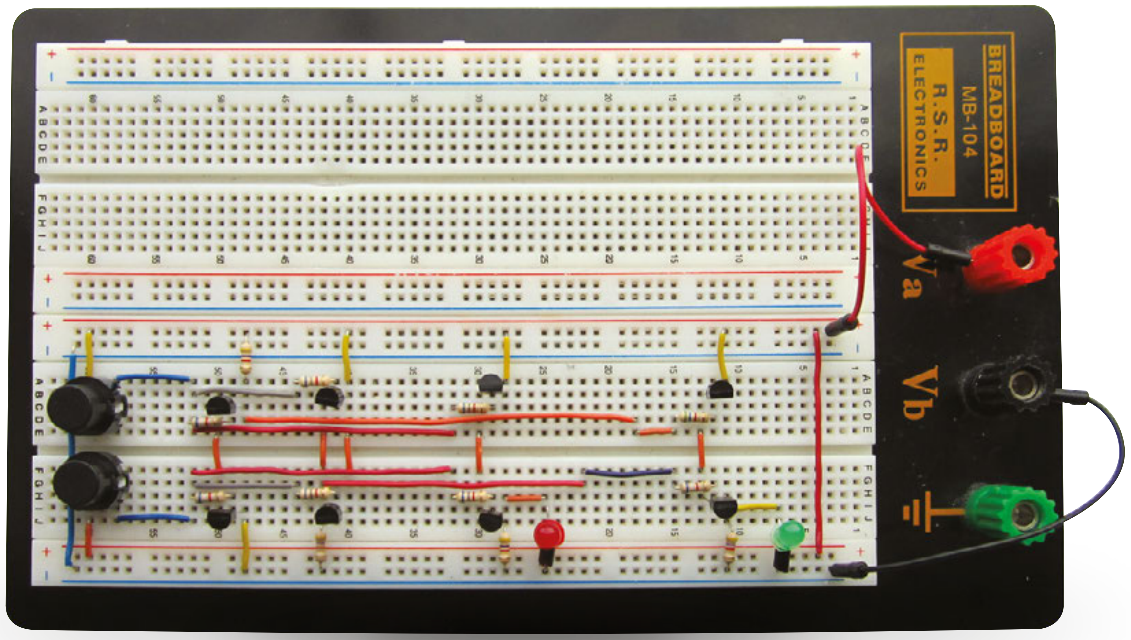
Making things add up

Eric Coates explains the basic electronic circuits used in computer arithmetic

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**



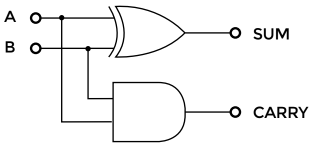
Right The complete half adder circuit built on a breadboard

Computers can carry out an awesome amount of mathematics; we all know that. So how come all of this can be performed on a machine that can only add 1 and 1; no subtraction as we know it; no multiplication and no division? The answer is that an electronic adder circuit that just adds 1 + 1 and uses a few tricks of the binary system can, with some help from a little firmware code, carry out any arithmetic we need at an amazing speed. The simple electronic calculator circuit at the heart of this wizardry is known as the Half Adder and is shown, made from just a couple of logic gates (an Exclusive OR gate and an AND gate), in **Figure 1**.

The 'half' in its name is because, while it can add 1 plus 0 and even add 0 plus 1 to get the correct answer of 1, as shown in the 'Sum' column of the truth table in Figure 1, if it adds 1 plus 1 in binary, this produces the (decimal) answer of 2, which in binary notation is 10, so a second column is needed in the form of a 'Carry' output to hold double the value of the single-digit sum output, producing the answer of 10 (decimal 2). This is fine if we only need to add a single column of binary, but the half adder cannot cope with multi-column numbers where a carry may be produced.

THE HALF ADDER'S BIG BROTHER

Because the half adder can only add two 1-bit numbers in a single column, it does not usually work alone. However, if the carry output produced by the half adder can be used as one input to another



A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Figure 1 The half adder is at the heart of all electronic calculators

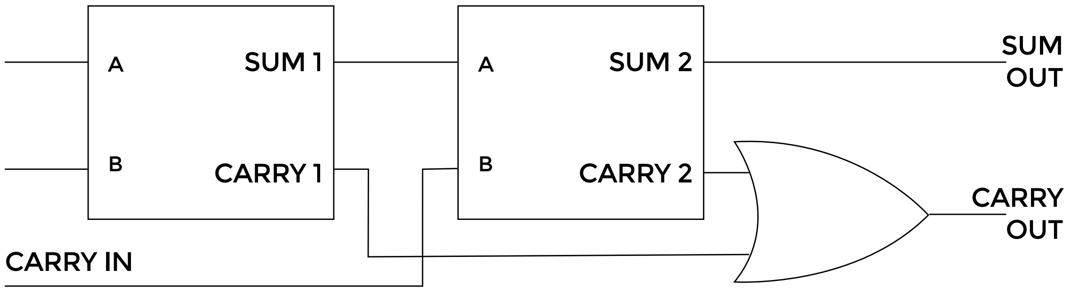


Figure 2 Two half adders make one full adder that can cope with multi-column addition

half adder, the two circuits combine to form the full adder shown in **Figure 2**, which will now have three inputs and so can add two binary bits in any one column using inputs A and B, plus any 'carry in' from a previous column, making multi-column addition possible.

So, the half adder may not carry out an impressive amount of arithmetic, but as it only takes around 30 nanoseconds to produce its answer, this basic adder could therefore carry out around 100 million similar calculations per second, which is how, when coupled with another half adder and other bits of electronic circuitry, many simple 1+1 additions are made to look like powerful maths, when it's really simple arithmetic, only at a blazing speed!

THE HALF ADDER DISSECTED

The most complex part of the half adder is an Exclusive OR (XOR) gate, which produces an output of logic 1 when its two inputs are different (01 or 10), but an output of 0 when the two inputs are the same (00 or 11). XOR gates comprise rather more electronics than the other more standard gates, which makes them slower, therefore less popular and more expensive than the other standard gates. However, the Exclusive OR logic function can be, and very often is carried out by a combination of more regular logic gates, as shown in **Figure 3** where the XOR gate is replaced by a combination of AND, OR and NAND gates, and this is the approach we shall use in our transistor/resistor version of the half adder.

So why, when logic gates are readily available in integrated circuit form, and you can easily simulate their operation on a computer screen, would you want to build them from basic electronic components such as resistors and transistors? Well, these days when the Internet of Things is becoming more and more important, the skill of combining computers such as the Raspberry Pi or Arduino with external electronic circuits and devices is becoming vital. It's one thing to move things about on a computer or smartphone display, but different skills are needed to drive output devices and make things change in the real world. Therefore it's more vital than ever to understand the operation of electronic circuits at component level, and to develop the knowledge and skills needed for constructing working circuits such as the half adder.

THE AND GATE

Our half adder circuit will contain two AND gates, both working to produce the truth table in **Figure 4**. →

YOU'LL NEED

- **Transistors**
  8× 2N3904
- **Resistors**
  4× 4.7 kΩ (4K7) for R1, R10, R11 & R12
- **8× 6.8 kΩ (6K8)**
  for all other resistors
- **Switches**
  2× Mini push to close
- **2× 5 V LEDs**
  1× Kingbright L-53GD (5 mm green) and 1× Kingbright L-53ID (5 mm red) are ideal and need no limiting resistors
- **Breadboard and wire links**
- **5 V Power supply**
  (only 20 mA current needed)

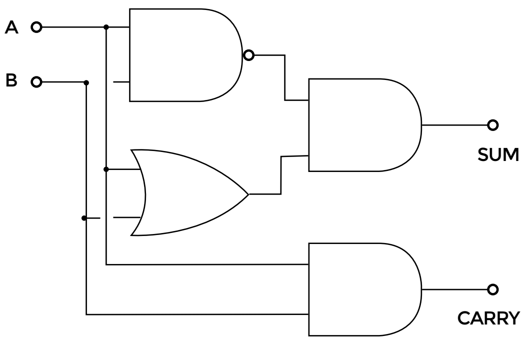
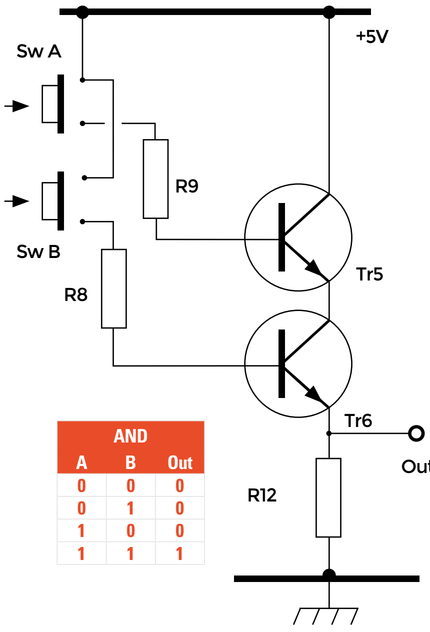


Figure 3 Equivalent half adder circuit using standard logic gates

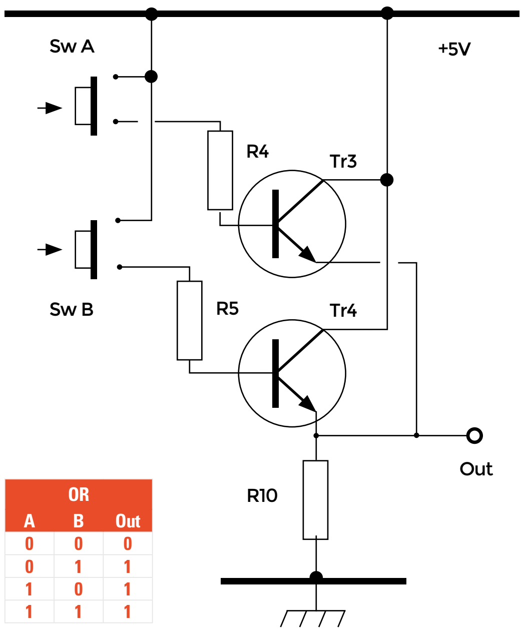


AND		
A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

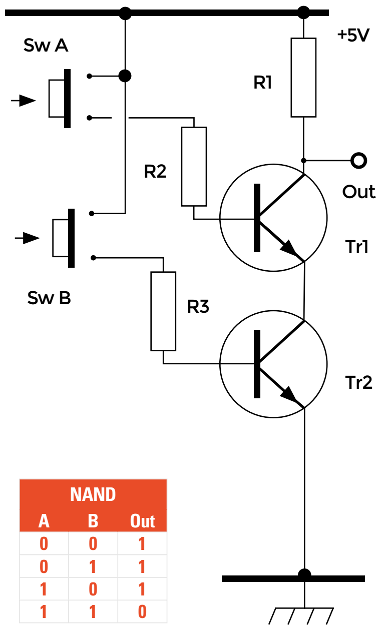
Figure 4 The AND gate output produces a logic 1 when, and only when, both inputs are at logic 1, otherwise the output is at logic 0

Figure 5 (Right)
The OR gate output produces logic 1 when either input or both inputs are at logic 1, otherwise the output is at logic 0

Figure 6 (Far Right)
The NAND gate output produces logic 1 when both inputs are at logic 0, otherwise the output is at logic 1



OR		
A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1



NAND		
A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

QUICK TIP

Before you build the circuit shown in Figure 7, it's a good idea to build each of the individual gate circuits in Figures 4 to 6 separately to begin with, so you can test each circuit to make sure they give the results shown in their respective truth tables. If you fit the two input switches to the breadboard first, you can use these to test each of the gate circuits. You'll also need to include the two output LEDs. Building these three logic gate circuits separately will enable you to find the best layout for each gate circuit and help you to understand how each gate circuit works before attempting the complete half adder.

Notice in this circuit that the two transistors Tr5 and Tr6 are effectively connected in series; that is, the same current will be flowing through Tr5 (collector to emitter) and through Tr6 (collector to emitter). Therefore, for this current to develop a voltage across R12, both transistors must be conducting. All the transistors in the whole of the half adder circuit are operated in switch mode; that is, they will either be switched on, by a large enough current flowing into their base connection to cause the transistor to conduct heavily, or switched off by removing the base current and therefore preventing any collector/ emitter current. The base current for Tr5 and Tr6 depends on the values of R9 and R8, which are both 6.8 kΩ, and the switches Sw A and Sw B are used to simply connect R9 and R8 to the +5 V supply.

So, when both transistors are switched on, the current through Tr5 and Tr6 will develop an output voltage across R12 (4.7 kΩ) equal to the supply voltage, minus a small voltage drop due to the voltages across the PN junctions in Tr5 and Tr6.

As shown in **Figure 3**, one AND circuit will be used to drive the Sum output of the half adder and the other AND circuit will drive the Carry output.

"The half adder does not carry out an impressive amount of arithmetic, but it only takes around **30 nanoseconds** to produce its answer"

THE OR GATE

As in the AND gate, the transistors here are being used as electronic switches and are activated by the currents flowing into their bases from Sw A and Sw B via the 6.8 kΩ resistors R4 and R5 (**Figure 5**). However, in this circuit the transistors Tr3 and Tr4 are connected in parallel, with the collector of Tr3 connected directly to the collector of Tr4 and the emitters of both transistors also connected together. Therefore if either Tr3 or Tr4 is made to conduct, a voltage almost equal to the +5 V supply will be developed across the output resistor R10. This therefore produces what we shall call logic 1 at the output if Tr3, Tr4 or both transistors are made to conduct, fulfilling the requirements of the OR truth table and in the total half adder circuit, driving one of the inputs of the Sum AND gate as part of **Figure 3**'s XOR function.

THE NAND GATE

Comparing the truth tables for the NAND gate in **Figure 6** and the AND gate in **Figure 4**, it can be seen that the output columns of each table are the opposites of each other. This tells us that the circuit for the NAND gate will be similar to that of the AND gate, but the outputs will have opposite logic values. Compare the circuits in **Figure 4** and **Figure 6** to see how this is done. The 4.7 kΩ output resistor R12 connected from Tr6 emitter to ground in **Figure 4** is simply moved to become R1 in **Figure 6**. Now if both transistors in **Figure 6** are made to conduct at the same time, the voltage at the bottom of R1 will be almost 0 V so producing logic 0 at the output; if either one or both transistors are switched off, however, the current path from R1 to ground will be interrupted and the output terminal will remain at logic 1, fulfilling the logic requirements of a NAND gate.

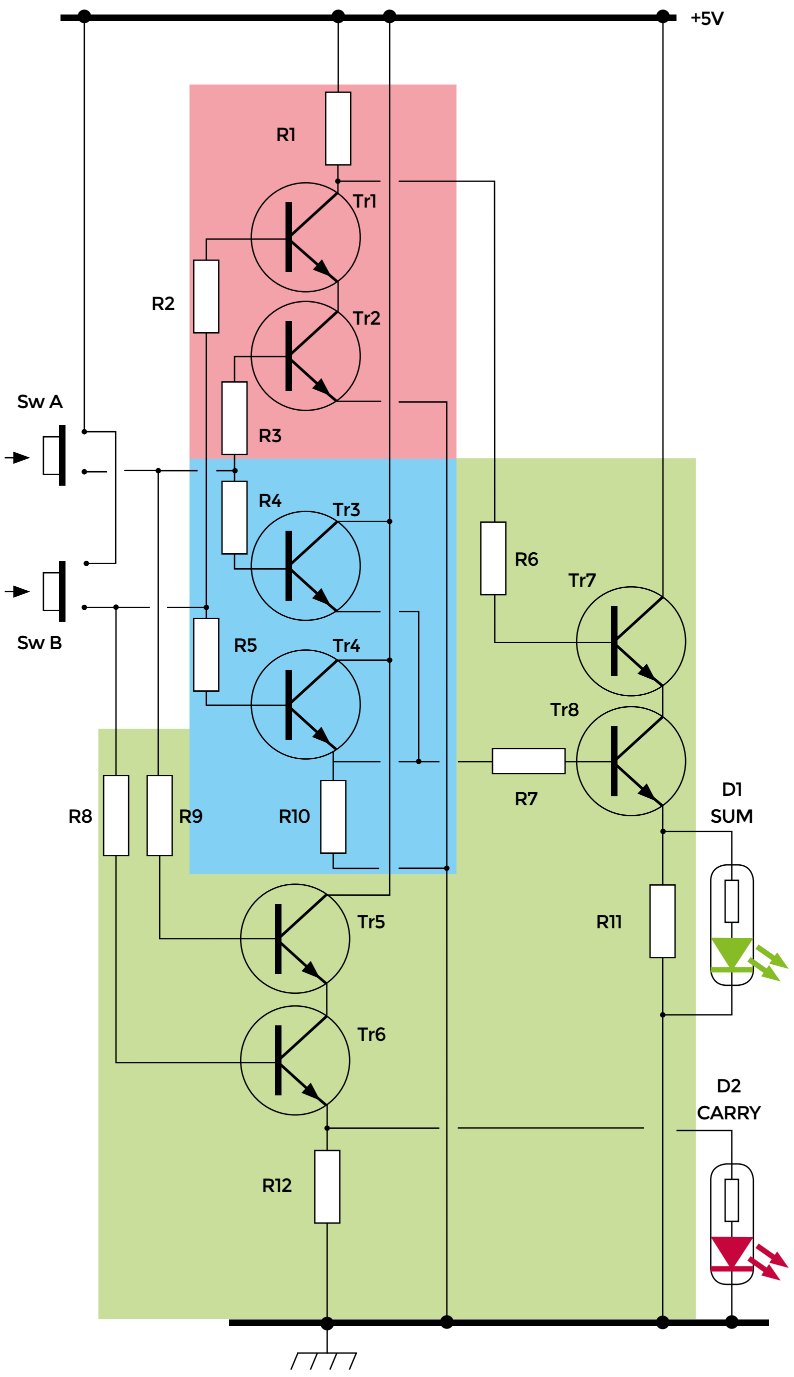
Figure 7 illustrates how two AND functions (shaded green), as well as the OR function (shaded blue) and NAND (shaded pink) logic functions needed for a half adder can be interconnected to drive two 5 V LEDs representing the Sum and Carry outputs. The LEDs used in this project are designed for +5 V supply circuits and have built-in current-limiting resistors so extra current-limiting resistors are not needed – unless you decide to use standard LEDs, in which case you will need to fit an appropriate current-limiting resistor in series with the LED.

The photograph (page 86) shows a practical layout of the half adder circuit on a dual breadboard. You can follow this layout to make a working half adder circuit or develop your own layout, depending on the breadboard and wire links you have available. Notice that only one half of the double board space has been used, so if you are feeling brave why not build a second half adder in the remaining board space and link the two circuits using another OR circuit, as shown in **Figure 2**, to make a full adder?

The half adder may not be as familiar as other more recognisable parts of a computer system such as disk drives, sound cards and touchscreens, but computers can work quite well without these, as proved by devices such as the Raspberry Pi and Arduino. However, no computer – not the Raspberry Pi, not the Arduino, nor even your pocket calculator – can work without the half adder, one of the unsung heroes of computing!

If studying adders in this article has inspired you to find out more about them, and how binary arithmetic really works to give the basic adders described here the power to carry out some awesome arithmetic, take a look at **hsmag.cc/qnFyCp**, where millions of people dedicated to learning about electronics go to study these fascinating combinational logic circuits essential to computing, as well as many more digital and analogue topics. □

Figure 7
This is the complete half adder circuit, combining Figures 4 to 6



Build a cold smoker

Add delicious flavours and create unique dishes

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

- **Kettle BBQ with lid**
- **Heat-proof ducting**
- **Hardboard** (4× 610 mm × 550 mm and 2× 610 mm × 610 mm)
- **Thermometer**
- **Galvanised fencing wire**
- **200 mm × 300 mm sheet of 1 mm stainless steel mesh**
- **Aluminium foil tape**
- **2.5 mm × 20 mm screws**
- **18 mm × 38 mm × 1200 mm planed pine**
- **8× M6 50 mm bolts with nuts**

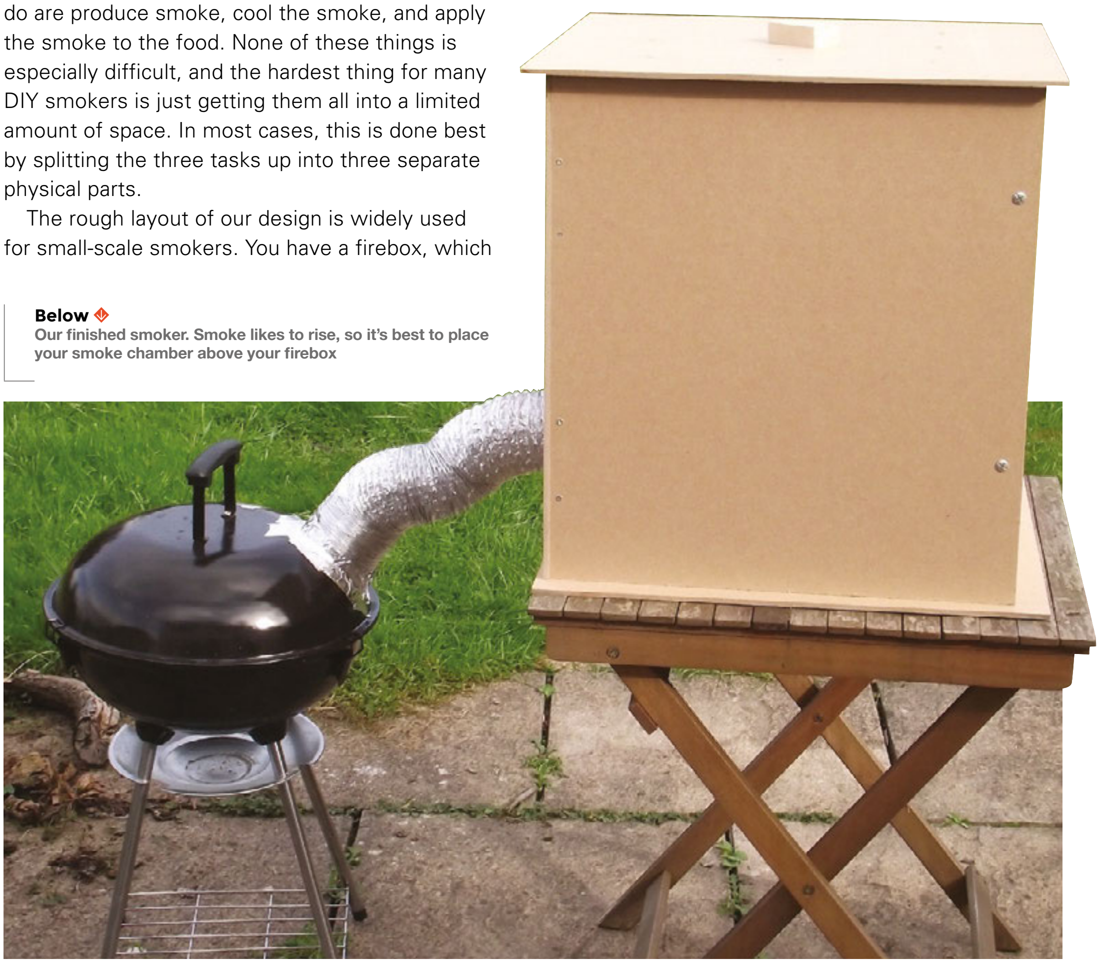
There are few things that you can turn your hacking skills to that can give you as much enjoyment as food. Most people eat around a thousand meals a year, so that's a thousand chances to use your knowledge and skill to increase your happiness. To us, that sounds like a great area to focus on. We'll start with a brilliant way of adding flavour to a wide range of foods such as meat, cheese, and fish: cold smoking.

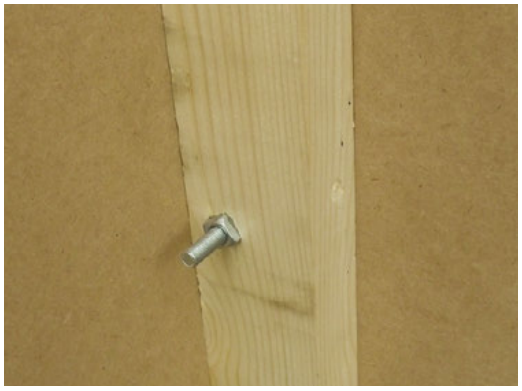
The three main things you need your smoker to do are produce smoke, cool the smoke, and apply the smoke to the food. None of these things is especially difficult, and the hardest thing for many DIY smokers is just getting them all into a limited amount of space. In most cases, this is done best by splitting the three tasks up into three separate physical parts.

The rough layout of our design is widely used for small-scale smokers. You have a firebox, which contains some smouldering wood that's used to generate the smoke; there's then some ducting that the smoke travels along, cooling as it goes. This duct deposits the smoke into a smoke chamber that holds whatever it is you want to smoke.

First, let's take a look at the firebox. This obviously needs to be fireproof, it needs to be fairly well sealed (so the smoke doesn't leak out everywhere), and it needs to connect to the ducting; but other than that, it can be almost anything that's large enough.

Below
Our finished smoker. Smoke likes to rise, so it's best to place your smoke chamber above your firebox





Above
The joints holding the smoke chamber together. The bolt through the long side makes it easy to take apart for easy storage

We've used a barbecue (because last year's miserable summer in the UK led to lots of discounts at garden centres). Garden incinerator bins are a popular alternative, and almost anything metal that you can attach a duct to will work.

The trick with producing smoke is to control the burn properly. If the wood burns too well then you'll get flames but not much smoke and you'll burn through your wood quickly. If it burns too slowly, it'll go out. There are maze smoke generators commercially available that are just tracks that allow wood to burn, but, unlike hacked-together options, these don't allow you to adjust the width of the burning sawdust, so it's hard to control the amount of smoke.

There are a few options for building your own smoke generator, but the easiest is to create a long, thin pile of sawdust that you light at one end and it slowly burns along. How long this pile should be is defined by the length of time for which you want to generate smoke, and how thin it is depends on how much smoke you want. There aren't any formulae for calculating this, as it also depends on the airflow of the smoker, the moisture content of the wood, and a myriad other factors. Essentially, getting the right size comes down to trial and error.

We've created our pack of sawdust using 1 mm wire mesh. You can buy this in sheets and it's easy to work with. Fold it into an M shape by hand and pile the wood dust in the middle groove. We found that 1 mm mesh was fine enough to hold commercial cold-smoking wood dust, but we wouldn't recommend anything with larger holes than this. Our 30 cm-long sheet of mesh burned for about four hours. If you need a longer burn time, you can either refill it or get a longer mesh.

We found that we needed a pile of wood dust about an inch and a half (38 mm) deep and about the same width to sustain a smouldering fire, and two inches (51 mm) deep provided a thicker smoke.

To get smoke, just set fire to one end of this dust trail. A blowtorch is the easiest option, but a candle underneath also works (just remember to remove →

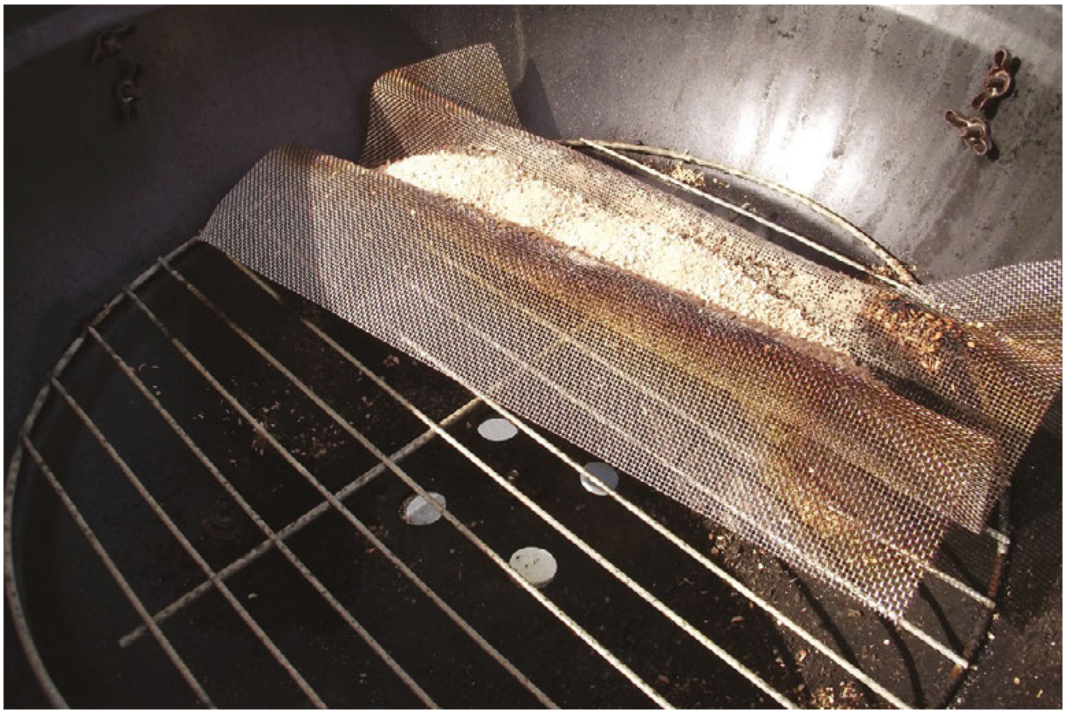
TOOLS

- Drill
- Jigsaw
- Wood saw
- Metal saw

HOT SMOKING

There are two types of smoking possible: hot and cold. In most of this article, we've looked at cold smoking, where the smoke is chilled and applied to the food without it cooking. The alternative is hot smoking, where the smoke is applied to the food as it cooks. This is as simple as chucking some damp wood on the burning coals of a BBQ, then using some form of lid to stop too much steam escaping.

You smoke the food as it cooks, which means that the smoking time is more limited, as is the range of foods you can smoke.



Left
We bent the corners of our M-shaped smoke generator to make it fit closer to the ducting taking the smoke out of the firebox

Above
The lit smoke generator will provide several hours of smoke from a single trace

SELECTING WOOD

There are a few different types of wood you can buy for smoking. Larger chunks (often around a centimetre cubed) are for hot smokers as they won't smoulder in the right way for cold smokers. There are also various types of pellets and pucks that are designed for specific commercial smokers. For a home-built cold smoker, you want to get wood dust or fine wood chips designed for cold smoking. There are a few different sources available, but make sure you get some that are for food.

There are loads of different tree species available and some people will make you think that it's essential that everything gets smoked in exactly the right species. Different woods do have different flavours, but the density of the smoke in the chamber and the length of time it's smoked for has a far bigger effect. We'd recommend starting with a fruit wood like apple or cherry. These are a bit milder than some woods such as oak, which means it's a little more forgiving of oversmoking so you're less likely to end up with something that tastes like an ashtray.

If you create your own wood for smoking, remember that it has to be uncontaminated with oil. This means no chainsaws or other mechanical devices that use oil. Cold smokers don't get hot enough to cleanly burn this oil, so you'll just end up with an oily taste in your food.

the candle before smoking the food). A cigarette lighter can work, but you're likely to burn your fingers once or twice.

We were able to hacksaw a square hole in the metal lid of the BBQ and attach the duct (see below) to the firebox using heat-resistant tape. It wasn't the neatest join, but we found that this was a situation where a neat finish would have taken much more time for very little gain. It does, however, need to be fairly smoke-proof, so you might need to pile on the tape. If anyone asks, tell them you were aiming for the rustic look. The bigger the hole you create, the better the smoke will flow through into the duct (provided you haven't made it bigger than the duct, of course).

DUCT TALES

You'll be glad to know that the firebox is the most complex part of the build. Now we've dealt with that, everything gets easier. The duct just needs to be long enough for the smoke to cool down sufficiently. There aren't any hard-and-fast rules on this as it depends on your firebox design, the amount of heat produced by the smouldering wood, and the ambient temperature in the smoke chamber. The end result we're looking for is a chamber under 26°C (79°F). As a general rule of thumb, we'd say around 1 to 1.5 metres should be fine, but we'll use a thermometer to ensure that the smoke chamber is at the correct temperature when smoking.

Provided that you control the burn in your smoke box well, the duct shouldn't get too hot, but it's always best to err on the side of caution and make sure that it's heat-proof. There are a few options here. Solid metal pipe can work provided it has a large enough diameter to let the smoke flow freely, but it can be a little awkward to work with. Perhaps the easiest option is a flexible chimney liner. There are also other flexible high-temperature ducts available. Remember that plastics can give off toxic chemicals even if they're not burning, so we'd highly recommend staying away from anything non-metallic. We used metallic ducting designed for channelling smoke in oven hoods.

CHAMBER MADE

The final part of the build, the smoke chamber, can be as simple or as complex as you like. It just has to be reasonably (though not completely) airtight and be connected to the duct. Many people reuse something they have spare. We've come across folks using tea chests, wardrobes, cider barrels, fridges, and filing cabinets (do be careful that there aren't any toxic paints or other substances, though).

We opted to build our own because we want it to be collapsible for easy storage. The design is about as simple as it's possible to be. We used four sides of 61 cm × 55 cm hardboard with the top and bottom made of 61 cm × 61 cm hardboard (the sizes were determined by the stock available at our local hardware store).

The four sides were joined with sections of 18 mm × 38 mm planed pine cut into roughly 150 mm lengths. The 18 mm wide edge was screwed into one of the box sides, leaving a slight lip where the hardboard poked out beyond the wood by about 1 mm. We joined the adjacent hardboard side to this by drilling a 7 mm hole through both the hardboard and pine and bolting it in place. Two of these joints in each corner held the box together. With this style of joint, you can simply unbolt the sides and the box comes apart. The small lip on the side pulls the sides together and minimises any gaps, though if you do find a little leakage, you can always tape the sides shut. It's worth numbering the corners and writing this on both sides so you can reassemble it easily.

You'll need some venting on the top of the smoke chamber to allow the smoke to escape, because if it can't escape, then it won't pull through from the firebox. We found that three 7 mm holes were sufficient to ensure a good airflow. Unfortunately, we discovered this after we'd drilled six 7 mm holes. If, like us, you find yourself over-ventilated, a little tape will solve the problem.

Beyond the bare box, the only things you need in your smoker are the thermometer, somewhere to put the things you're smoking, and the end of the smoke duct. We pushed our thermometer through a 3 mm hole drilled into the side of the chamber to allow us to read it without opening the chamber (which lets a lot of smoke out). We made a rudimentary rack using galvanised fencing wire threaded through holes in the chamber, and placed an oven rack on this. Finally, we used a jigsaw to cut a circular hole for the duct. We made this larger than the duct, pushed the duct through and taped it on the inside. This was neater and more secure than taping to the outside, as we had to do with the firebox.

That was all there was to it. It took a little experimentation to get the right amount of wood dust and airflow holes, but within an hour of finishing the build, we were cold-smoking our first batch of food.

WHAT TO SMOKE

Now you've got your smoker, the next question is what to put in it. Unlike a hot smoker (which is hot enough to kill bacteria), a cold smoker carries some risks. Essentially, you'll be holding the food at around room temperature for a few hours, which can allow bacteria to grow, so you need to make sure that what you smoke isn't going to harbour nasties that can cause problems.

Cheese is a great option to start with, and the flavours of different types of cheese with different types of smoke can be fantastic. It tastes best a few days after it's smoked, as this gives the flavour time to mellow. Most vegetables can be smoked, and smoked garlic, onions or peppers give a great depth of flavour to cooked dishes.

Meat can be safely smoked if it's been properly salted first. This salting slows down the bacteria enough for it to be smoked without becoming contaminated. In principle, properly made bacon is safe to smoke, but we would stay away from commercial bacon as it's hard to know what preservatives have been used. A good butcher will be able to tell you if their products are safe to cold-smoke. Alternatively, you can make your own. The process takes a little time (up to two weeks), but the results are delicious. Our favourite recipe comes from River Cottage: **hsmag.cc/cGAzON**. Just be sure to cure it for around four to five days before smoking.

There's nothing to stop you smoking many different things at a time, provided they all fit into your smoker of course. The best way of finding out what smoked things you like is to try lots of different things out!

Below
Meat thermometers are simple and cheap, but you could hook up a digital sensor and collect the data

Make your own 8-bit synths with Arduino

Getting started with the Mozzi library to get your Arduino wailing

Chris Ball

@ChrisBallMidi

Chris Ball is a technologist working in Manchester, UK. He has worked on a variety of interactive art installations. You can visit his site at **chrisballprojects.co.uk**

YOU'LL NEED

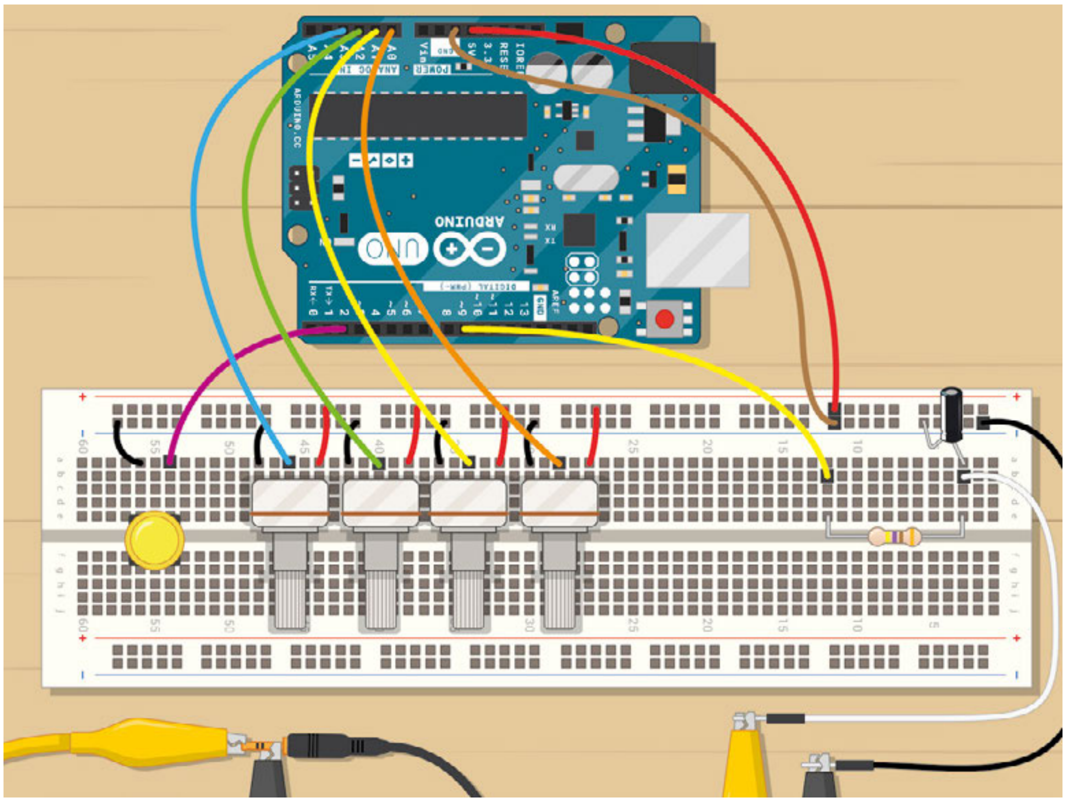
- **An Arduino** (Preferably Uno, although others are possible)
- **Breadboard**
- **470 Ω resistor**
- **Tactile button**
- **4 × 10 kΩ linear potentiometers** (usually marked B10K)

So, you've got your first Arduino, and you've tried a few basic projects. Maybe you've got an LED blinking and now you're struggling to find a project that's a little more creative. Look no further, we've got you covered! You may have achieved some basic bleeps and bloops with the built in Tone() function, but we'll be doing some much more advanced digital synthesis.

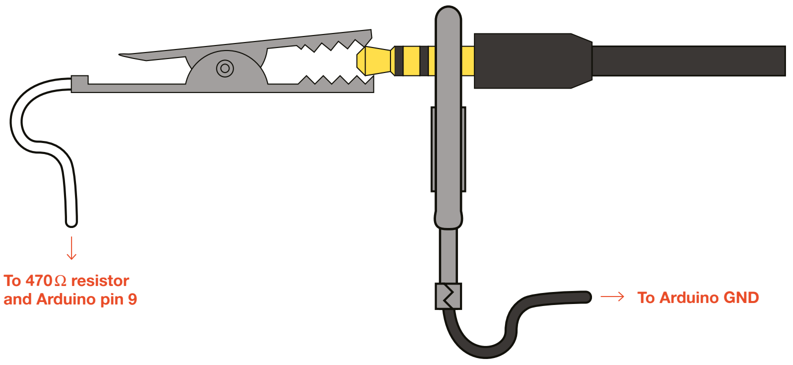
Digital synthesizers are very different from their analogue counterparts. Instead of a complex collection of diodes, amplifiers, oscillators, and other esoteric audio electronics, they mainly use processing power to generate waveforms and effects. Digital synths have other benefits too, but their main strength is that once set up, they're extremely reconfigurable; you don't need to rebuild your synth to change its sound, just reprogram it.

Throughout this tutorial we'll be using the Mozzi library to create a variety of sounds. The library is capable of generating complex waveforms, audio effects, and playing short samples, all from the modest hardware in an Arduino. We'll be using it to create a basic FM (frequency modulation) synthesizer.

We'll get started with the absolute bare minimum for a Mozzi-based sketch. Make sure you've installed



Right Our final synth, with the four potentiometers we need to play with to create the sound of the future



Above How to connect crocodile clips to an audio jack

the Mozzi library, then start your Arduino environment and open the example under File > Examples > Mozzi > Basics > Sinewave. This is a sine wave generator, which is pretty much the digital audio equivalent of a 'Hello, World!' program.

Here you'll see the basics of a Mozzi program, and you might notice it has a slightly more complex structure than your usual Arduino sketch. Let's ignore that for now, and get making some sound. Upload the code to your Arduino. If all is well, a sine wave will be generated on pin 9, and we just need to listen to it.

To connect the Arduino to our amplifier/earphones we need to connect the following:

Arduino pin 9 → 470Ω resistor → Audio jack tip (the resistor is to help protect pin 9)
Arduino GND → Audio jack base

If all's well, you should hear a sine wave at 440Hz. If you have no sound, check your volume, connections, and that the sketch has uploaded successfully.

If you've had some success, we'd recommend at this point that you take a look at some of the other examples the Mozzi library has to offer. This will give you an idea of what it's capable of, but bear in mind that some examples expect extra hardware.

Back to the sine wave generator: we'll be breaking down the elements of this sketch fairly thoroughly, as knowing the basics of how Mozzi works will enable you to make more exciting changes later.

First, we'll take a look at the includes. This is where we add in the required files from the Mozzi library, and you should see three includes: 'MozziGuts.h', 'Oscil.h', and 'tables/sin2048_int8.h'.

MozziGuts.h is the main library required for doing anything with Mozzi. This file will adapt your Arduino for use as a synth, by taking over some timers and setting up some fast sampling methods.

Oscil.h is simply a template for an oscillator. Any sound requires a repeated change in voltage, or air pressure; an oscillation. This file tells Mozzi how to create an oscillator from a lookup table.

tables/sin2048_int8.h is the lookup table we'll be using to make a sine wave. A lookup table is often used where calculating the values of a function (in this case, a sine wave) would take too long. We simply pre-calculate all the values and store them in memory. When we need them, we can simply 'look them up', hence the name lookup table.

We then have a line:

```
Oscil <SIN2048_NUM_CELLS, AUDIO_RATE>
aSin(SIN2048_DATA);
```

This is a little like saying, "Create a sine wave oscillator called aSin, using the table I mentioned before." We also have the line:

```
#define CONTROL_RATE 64
```

Which means we intend to update our controls (our potentiometers and buttons) 64 times per second. Mozzi asks for control rates to be powers of two (e.g. 2, 4, 8, 16, …)

To continue to our main functions: in **`setup()`** you'll see two commands. The first, **`startMozzi(CONTROL_RATE)`**, will start the Mozzi engine, and the second, **`aSin.setFreq(440)`**, will set the frequency (or pitch) of our oscillator. 440Hz is middle A (so if you only get this far, at least you can get your band in tune).

Typically, when writing a Mozzi sketch, you'll avoid putting anything in **`loop()`**, except the function **`audioHook()`**. This function will calculate samples (little chunks of audio data) ready to be written to our output. So where do we put our code? You'll notice, apart from the usual **`setup()`** and **`loop()`** functions that we have two more: **`updateControl()`** and **`updateAudio()`**.

`updateControl()` is where we put the changes we want to happen at our control rate (64 times per second). This will be things like reading our potentiometer values, button states and other tasks that don't need to happen too often. In this sketch, nothing like this is required, so the function is empty.

`updateAudio()` is the function that **`audioHook()`** will run repeatedly – it calculates our audio samples and stores them in a buffer to be sent to pin 9 later. You can see within this sketch the code: →

INSTALLING THE MOZZI LIBRARY

From **sensorium.github.io/Mozzi** you can click on the download link. This will take you to an optional donation page where you can help the author of the library out, if you choose. You'll then be taken to the GitHub project page for Mozzi. Click the 'Source Code .zip' link, download the zip file, and extract the contents to your **Arduino/libraries** directory (this is usually in your **documents** folder).

Your file structure should then look like this:
Documents/Arduino/Libraries/Mozzi-1.0.3 (although you may have a different version).

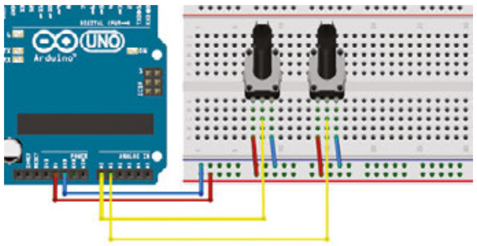


Figure 1 Connecting two pots to your Arduino

`return aSin.next();` which simply means to send the next sample for this oscillator to the buffer.

Let's make a couple of changes to the way this works. We'll add one pot (potentiometer) to control frequency, and a second pot to control volume.

Connect two pots to your Arduino (**Figure 1**). Each pot will have one side connected to 5V, the other side connected to GND and the middle (wiper) to an analogue input. We'll use analogue inputs A0 and A1.

Add the following lines of code before `void setup()`:

```
int pot0, pot1;
int volume,frequency;
```

These will be the variables where we'll store the pot values, and the frequency and volume values they will control.

Add the following lines of code inside your `updateControl()` function:

```
pot0 = mozziAnalogRead(A0);
pot1 = mozziAnalogRead(A1);
frequency = pot0 + 50;
volume = map(pot1, 0, 1023, 0, 255);
aSin.setFreq(frequency);
```

The first two lines will store our pot voltages as variables, `pot0` and `pot1`.

The third stores the value of `pot0` + 50 to a variable called `frequency`. We've added the +50 to prevent the frequency becoming too low to hear.

The fourth line will store the value of `pot1` to a variable called `volume`, but will scale it in the process to be between 0 and 255 (instead of 0 and 1023).

The last line will set the frequency of our oscillator to the value in the `frequency` variable

This covers changing our frequency, but we need to make one last change in `updateAudio()` for the volume control to work.

Change the line:

```
return aSin.next();
```

to:

```
return (aSin.next()*volume)>>8;
```

This line may look confusing, but it's very similar to multiplying the output by a value between 0 and 1. It's good to get used to calculating this way as it's significantly faster with integer values on an Arduino, and we need speed to calculate all our sample values.

If you upload these changes, you now have a basic synthesizer! You should be able control pitch with pot 0 and volume with pot 1.

So perhaps you've played that for a while and become bored already. This was bound to happen – it's only a simple synthesizer. Let's try adding another sine wave oscillator, and another potentiometer to control it. To add another potentiometer, you can repeat the connection pattern as before, with our middle wiper pin wired to A2 on the Arduino. We already have the sine wave lookup table we need, so we can do this simply by duplicating the line:

```
Oscil <SIN2048_NUM_CELLS, AUDIO_RATE>
aSin(SIN2048_DATA);
```

You'll need to give our oscillators distinct names, so we should change this to:

```
Oscil <SIN2048_NUM_CELLS, AUDIO_RATE>
aSin1(SIN2048_DATA);
Oscil <SIN2048_NUM_CELLS, AUDIO_RATE>
aSin2(SIN2048_DATA);
```

We'll add and change some variables too:

```
int pot0,pot1,pot2;
int frequency1,frequency2,volume;
```

Our `updateControl()` function will become:

```
pot0=mozziAnalogRead(A0);
pot1=mozziAnalogRead(A1);
```

QUICK TIP

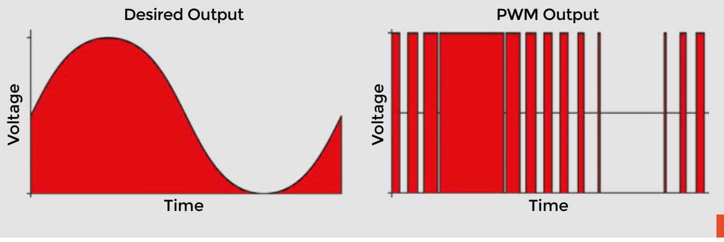
The '>>' and '<<' symbols are called bitshift operators, and they are a very fast way of dividing or multiplying by 2. The '>>8' is a little like saying "divide by 2, 8 times". If our volume value was 200, you could think of this line as Output × (200/256).

DIGITAL TO ANALOGUE WITH PWM

You might have realised that we are using pin 9, a digital pin, to do the job of an analogue output – how does this work? We are using pulse-width modulation (PWM). Simply put, if we want to approximate 2.5V with a 5V digital output, we switch the digital pin high for 50% of the time. 1V would be 20%, 2V 40%, and so on.

PWM is most commonly used for making lights (particularly LEDs) appear at different brightnesses or motors run at different speeds, all by switching a constant voltage on or off.

This approach does have significant downsides, though – mainly that it will introduce a lot of noise at your modulation frequency. Not a problem for motors or LEDs, but your ears will probably notice straight away.



```
pot2=mozziAnalogRead(A2);
frequency1=pot0+50;
frequency2=pot1+50;
volume=map(pot2, 0, 1023, 0, 255);
aSin1.setFreq(frequency1);
aSin2.setFreq(frequency2);
```

And our **updateAudio()** code will be changed also:

```
return volume*((aSin1.next()+aSin2.next())>>1)>>8;
```

Our two sine waves, when added together, could add up to a number higher than our PWM output can reproduce. In audio circles this is called 'clipping' and is generally avoided (unless you're intentionally after a distorted sound). We've prevented this here by dividing the output by two.

The above changes should result in two controllable sine waves on pots 0 and 1. You may even be able to get some interesting 'throbbing' if you pitch the notes close together – this is called 'beating' and is caused by interference between the two frequencies.

To develop the synth further, we'll introduce frequency modulation (FM). This means we'll use the output of one sine wave to control the frequency of another, resulting in varied timbres.

We'll also be making some changes to our hardware: adding another potentiometer; and introducing a push button to trigger the audio.

If you make these changes to the circuit, and upload the code from **hsmag.cc/JPNNBP**, you should have yourself an FM synthesizer!

The magic happens in two lines. This one, in **updateControl()**:

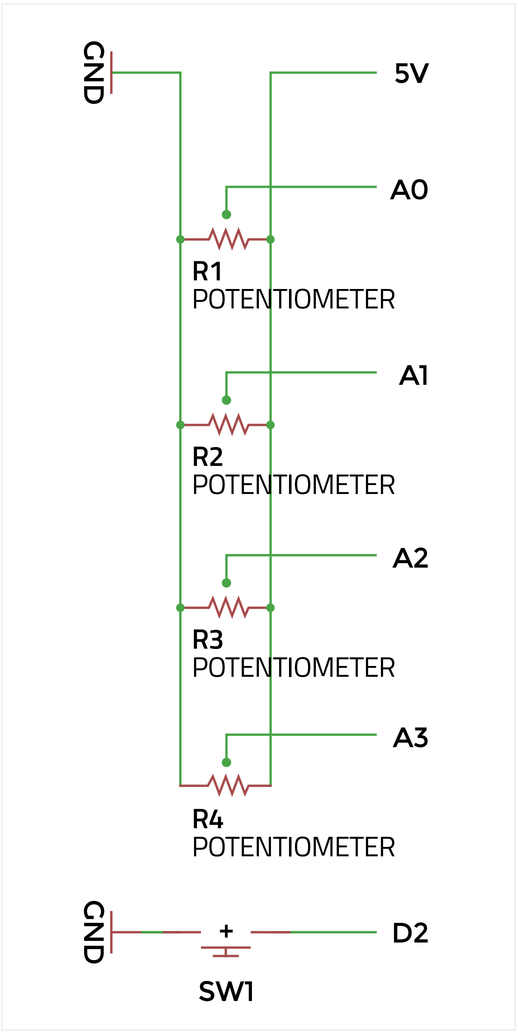
```
aSin2.setFreq(frequency2);
```

And this line, in **updateAudio()**:

```
aSin1.setFreq(frequency1+(amount*(aSin2.
next())>>8));
```

The first sets the frequency of our modulation, and the second uses that to control the frequency of our main waveform. There is also an **amount** control that will multiply our modulation, with some interesting effects. Remember, now you'll need to push the trigger button to hear sound! Try changing some of the numbers in this code and see how they affect the output.

So, you should have a basic 8-bit synthesizer, but more importantly, an idea of how to use the Mozzi library to develop it further. Mozzi has a huge selection of basic waveforms, some audio effects, and it's extremely well documented, with great examples. If you feel lost at any point, you can always check on the website.



Left The final circuit diagram for the breadboard

QUICK TIP

Having problems getting the first example working? There's a much more thorough walkthrough at **hsmag.cc/AwksSP**

OTHER ARDUINO AUDIO PROJECTS

ElectroSmash PedalShield: This is a kit designed to sit on top of an Arduino Due and turn it into a general-purpose guitar effects pedal. It has some basic examples available, and a forum with many more. **Electrosmash.com/pedalshield**

Ardutouch: International hacker Mitch Altman has created an Arduino-based synth project called Ardutouch, built on a fantastic library by himself and Bill Alessi. The library by itself is great to mess around with, although it may require an experienced Arduino user. **cornfieldelectronics.com/cfe/projects.php**

Teensy Audio Board: This hardware for the Teensy 3.1/3.2 and the accompanying audio library get an honourable mention simply because it's so fully featured. Not strictly Arduino, but Arduino-like. **pjrc.com/teensy/td_libs_Audio.html**

There are many more useful libraries in the Arduino Library List (playground.arduino.cc/Main/LibraryList) under the 'audio' section.

Going straight with PID

How to make your Raspberry Pi robot drive in a straight line

Martin O'Hanlon

@martinohanlon

Martin is the co-author of *Adventures in Minecraft*, a Raspberry Pi trainer, and blogger at **stuffaboutco.de**

YOU'LL NEED

- **Raspberry Pi wheeled robot**
- **Two motor/wheel encoders**

There is more to making a robot go in a straight line than just turning the motors on full power – in this tutorial you'll learn how to add encoders to your robot and implement a PID controller to regulate the power.

Anyone who has ever built a wheeled robot will know that driving in a straight line is a lot more difficult than you first think. Sure, holding a true heading for 1, 2 or maybe 3 metres is possible, but keeping it up past 10 or 20 metres without a veer to the left or right becomes astonishingly tricky.

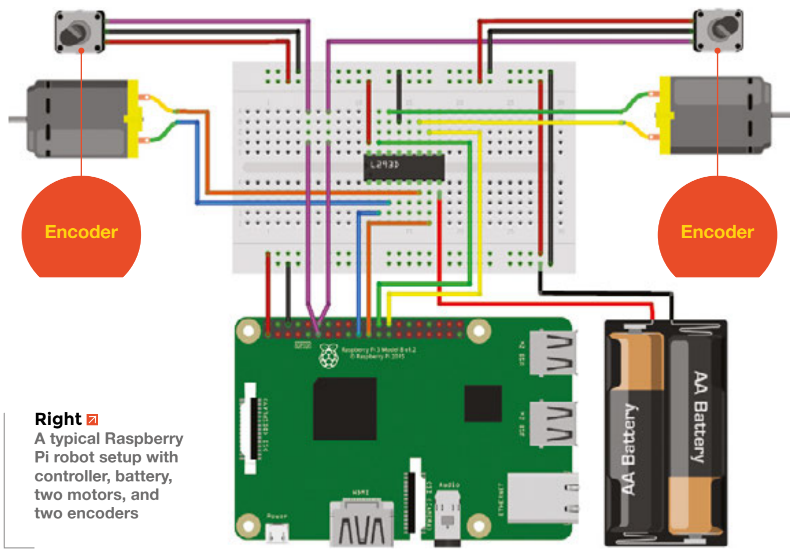
There are many reasons why this happens – uneven surfaces, differences in wheel size, bent axles and, most significantly, the fact that no two motors turn at the same speed! Minor differences in manufacturing and materials result in minor differences in output, and as a result, one motor will spin more quickly than the other. This difference may well be very small, but over time (or distance), it will show as your robot beginning to veer. If the right motor is moving quicker, your robot is going to turn in an arc to the left, and vice versa.

To counter this problem, a solution is required that can accurately measure how fast each motor is moving and then use this feedback to adjust the motor's speed at run-time so that each motor spins at the same rate.

Encoders are typically used to measure motor speed; these devices provide an output (or pulse) multiple times per revolution.

A PID (proportional-integral-derivative) controller is then used to continuously monitor and adjust motor speed to keep them in sync.

This tutorial steps through adding encoders to a Raspberry Pi-powered robot, using Python to create a PID controller, tuning it to work with your robot, and using the GPIO Zero (**gpiozero.readthedocs.io**) library to interact with the hardware.



Right A typical Raspberry Pi robot setup with controller, battery, two motors, and two encoders

ENCODERS

Encoders come in all shapes, sizes and accuracy. They can be incorporated into motors themselves or as add-ons that connect to the motor shaft or the wheel, but fundamentally they all work in the same way – a consistent signal is provided as the motor turns; the faster the motor is turning, the faster the signal.

A typical robot setup includes a motor controller (or maybe a dedicated HAT), two motors, and a battery pack. In addition, you will need an encoder per motor connected to your Raspberry Pi.

Most encoders will have three or four pins (power, ground, and one or two signal pins); typically the power and ground pins will be connected to a 3.3V and a ground (GND) pin on your Pi; one of the signal pins should be connected to a spare GPIO pin. It's important to check the specifications of your encoders before connecting them up to the Raspberry Pi.

Any Python IDE will do

1. Open up a Python 3 editor (e.g. Thonny) and create a new program.

2. Import the required Python modules:

```
from gpiozero import Robot, DigitalInputDevice
from time import sleep
```

3. Create a constant for sample time – this is how often (in seconds) your program will read the values

from the encoders – it's likely that you will need to change this value later to get the best result from your setup:

```
SAMPLETIME = 1
```

4. Create an **`Encoder`** class to monitor your encoders; this will increment a value each time the pin turns on and off.

```
class Encoder(object):
   def __init__(self, pin):
        self._value = 0

        encoder = DigitalInputDevice(pin)
        encoder.when_activated = self._increment
        encoder.when_deactivated = self._increment

   def reset(self):
        self._value = 0

   def _increment(self):
        self._value += 1

@property
   def value(self):
        return self._value
```

5. Use the gpiozero **`Robot`** class to connect to your motor hardware; each motor will connect to two GPIO pins (one forward, one back), specified as **`((left_forward, left_backward), (right_forward, right_backward))`** – our robot uses the pins **`((10,9), (8,7))`**:

```
r = Robot((10,9), (8,7))
```

6. Create two **`Encoder`** objects passing the GPIO pin the signal connects too; we've used GPIO pins 17 and 18:

```
e1 = Encoder(17)
e2 = Encoder(18)
```

TWO-PIN ENCODERS

Your robot maybe fitted with 'quadrature' encoders; these encoders use two pins, significantly increase the resolution, and allow the direction the motor is spinning to be determined.

This tutorial assumes you are using simple one-pin pulse encoders, but there is a code example at **github.com/martinohanlon/RobotPID** which should allow you to modify it. There's also an excellent write-up at **robotoid.com/appnotes/circuits-quad-encoding.html** which explains how they work and how to interpret the signals from them.

7. Start the robot by making the value of both motors 1.0 (forward at full speed):

```
m1_speed = 1.0
m2_speed = 1.0
r.value = (m1_speed, m2_speed)
```

8. Start an infinite loop and print the encoder values:

```
while True:
   print("e1 {} e2 {}".format(e1.value, e2.value)
   sleep(SAMPLETIME)
```

9. Run the program.

View the complete **encoder.py** code listing at **github.com/martinohanlon/RobotPID**.

The **`SAMPLETIME`** value should be changed to reflect your hardware; you need to find a balance between reading it frequently enough to get good results and slow enough to capture sufficient encoder ticks – try values between 0.1 and 1.0 seconds and aim to capture more than 20 ticks per sample.

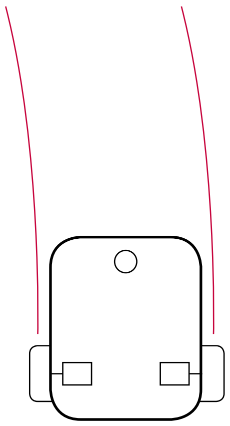
Make a note of approximately how many encoder ticks per sample your robot makes.

PID CONTROLLER

A PID controller continuously calculates an error and applies a corrective action to resolve the error; in this case, the error is the motor spinning at the wrong speed and the corrective action is changing the power to the motor. It is this continuous testing of the motor's speed and adjusting it to the correct speed which will make your robot's motors spin at the correct speed and go straight.

PID is a 'control loop feedback' mechanism

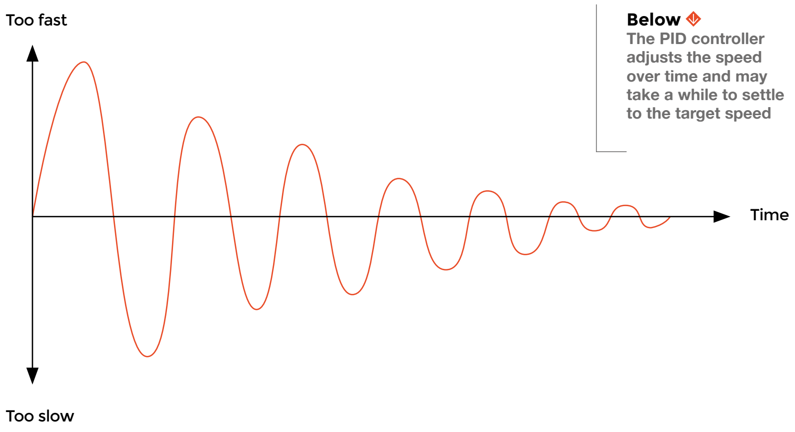
The controller will have a target motor speed that it wishes to maintain; each time the encoder values are sampled, it will calculate the difference (or error) →



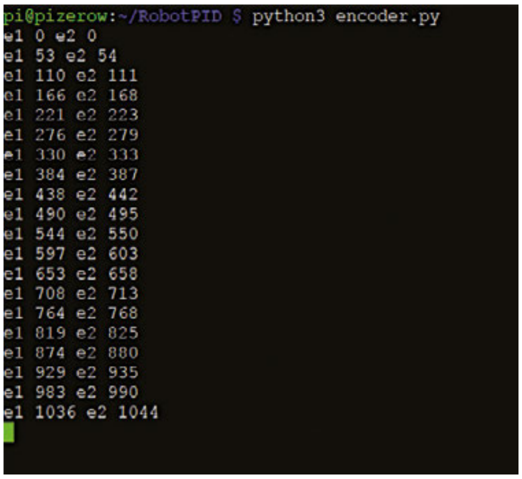
Above As the right motor spins quicker than the left, the robot always turns left

QUICK TIP

Motors with built-in encoders tend to be more expensive, but they also have much greater accuracy and precision than add-ons.



Below The PID controller adjusts the speed over time and may take a while to settle to the target speed



```
pi@pizerow:~/RobotPID $ python3 encoder.py
e1 0 e2 0
e1 53 e2 54
e1 110 e2 111
e1 166 e2 168
e1 221 e2 223
e1 276 e2 279
e1 330 e2 333
e1 384 e2 387
e1 438 e2 442
e1 490 e2 495
e1 544 e2 550
e1 597 e2 603
e1 653 e2 658
e1 708 e2 713
e1 764 e2 768
e1 819 e2 825
e1 874 e2 880
e1 929 e2 935
e1 983 e2 990
e1 1036 e2 1044
```

Right
Our encoders tick about 50 to 60 times per sample and motor 2 runs slightly faster than motor 1

OTHER PID USES

The input and outputs of a PID controller don't have to be an encoder and a motor; the controller can be applied to any situation where something needs to be constantly monitored and adjusted. This could be:

→ Using a magnetometer to make a robot move in a certain direction

→ Keeping a camera on a powered mount pointing at the same place

→ Making a robot follow a wall by measuring the distance to it with ultrasonic sensors

PID controllers are universal devices and the rules can be applied to solve many different problems.

between the target speed and the actual speed and apply an adjustment to the motor speed. If the adjustment overshoots the next time the encoders are sampled, a smaller opposite adjustment will be made. Over time, the adjustments will even out and the motors will run at a constant speed (or at least that's the theory!).

You will be changing the program you created to read encoder values to calculate an error and apply an adjustment using proportional, derivative, and integral control.

PROPORTIONAL

Proportional control is adjusting the motor speed by adding the value of the error – the value of the error (the difference in encoder ticks between the target and the actual speed) will need to be converted to the motor speed (a value between 0 and 1) by multiplying a constant (KP) to get a 'proportional' change:

adjustment = error x KP

Time for maths

Modify the program you created earlier to read encoder values:

1. Add a constant for the target of encoder ticks you want the motors to achieve; make this value about 75% of the 'encoder ticks per sample' value you made a note of earlier (in our case 60 × 0.75 = 45):

```
TARGET = 45
```

2. Add a constant (KP) for the proportional change which will be multiplied by the error to create the motor adjustment. This constant will need tuning, but a good starting point is 1 divided by the 'encoder ticks per sample' (e.g. 1 / 60 = 0.0166~)

```
KP = 0.02
```

3. At the start of the infinite loop, calculate the error for each motor by subtracting the encoder value from the target:

```
while True:
    e1_error = TARGET - e1.value
    e2_error = TARGET - e2.value
```

4. Calculate the new motor speed by adding the error and multiplying it by the proportional constant:

```
    m1_speed += e1_error * KP
    m2_speed += e2_error * KP
```

5. The motor speed needs to be between 0 and 1, so clamp the value using **max** and **min**:

```
    m1_speed = max(min(1, m1_speed), 0)
    m2_speed = max(min(1, m2_speed), 0)
```

6. Update the robot's speed to the new motor values:

```
    r.value = (m1_speed, m2_speed)
```

7. Add some debugging code to print the motor speed after the encoder values; this will be useful for tuning:

```
    print("e1 {} e2 {}".format(e1.value, e2.value))
    print("m1 {} m2 {}".format(m1_speed, m2_speed))
```

8. Before the program sleeps for the sample time, you need to reset the encoders:

```
    e1.reset()
    e2.reset()
    sleep(SAMPLETIME)
```

9. Run your program – you will see the motor's speed being adjusted each time the encoders are sampled, based on the error.

How different are your motors?

View the complete **proportional.py** code listing at **github.com/martinohanlon/RobotPID**.

Proportional control should be enough to stabilise your motors' speed and keep them turning at about the correct speed, but when there is a large error or you want the speed to adjust quickly, you will get a large overshoot and your robot will react erratically, swinging left to right – this is where derivative control helps.

DERIVATIVE

Derivative control looks at how quickly or slowly the error is changing, creating a larger error if it's changing quickly and a smaller one if slowly. This will help to smooth out the rate of change and prevent erratic changes in speed.

This is achieved by taking the previous error into account when calculating the adjustment and again multiplying by a constant (KD):

adjustment = (error × KP) + (previous_error × KD)

Modify the program to implement derivative control…

1. Create a new constant (KD) for the derivative control. Again, this value will need to be changed to get the best results for your setup; a good starting value is half the value of KP:

```
KD = 0.01
```

2. Create two variables to hold the previous errors and set them to 0:

```
e1_prev_error = 0
e2_prev_error = 0
```

3. Modify the code which calculate the speeds for motor 1 and 2 to taken into account the previous error:

```
    m1_speed += (e1_error * KP) + (e1_prev_error
* KD)
    m2_speed += (e2_error * KP) + (e1_prev_error
* KD)
```

4. At the end of the loop, set the previous error variables to be that of the current error:

```
    sleep(SAMPLETIME)
    e1_prev_error = e1_error
    e2_prev_error = e2_error
```

5. Run your program. Again you will see the motor speed change in relation to error and over time, it should stabilise to a more consistent speed.

Proportional and derivative (PD) control should provide a good level of performance but may not provide consistency of speed over time – integral control can help to bring this stability.

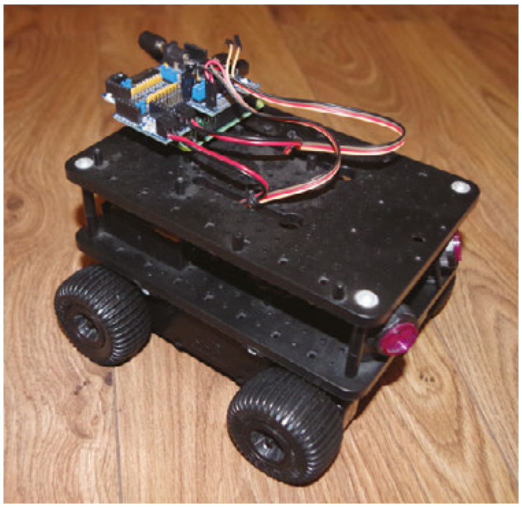
INTEGRAL

Integral control helps to deliver steady state performance by adjusting for slowly changing errors. It does this by keeping a sum of all the previous errors and applying a constant (KI) to the adjustment:

adjustment = (error × KP) + (previous_error × KD) + (sum_of_errors × KI)

Modify the program to implement integral control:

1. Create a constant for the integral control (KI); a good starting point is half the value of KD:



Left Thanks to a pair of controllers and PID, our robot now runs in a straight line

```
KI = 0.005
```

2. Create two variables to hold the sum of all previous errors and set them to 0:

```
e1_sum_error = 0
e2_sum_error = 0
```

3. Modify the speed calculation to take into account the sum:

```
    m1_speed += (e1_error * KP) + (e1_prev_error *
KD) + (e1_sum_error * KI)
    m2_speed += (e2_error * KP) + (e1_prev_error *
KD) + (e2_sum_error * KI)
```

4. At the end of the loop, increment the sum variables by the current error value:

```
    sleep(SAMPLETIME)
    e1_sum_error += e1_error
    e2_sum_error += e2_error
```

5. Run the program. You should see over time that the motor speeds start to stabilise. □

QUICK TIP

You may not have to implement proportional, integral and derivative (PID) control to get your robot to go straight: P or PD might be good enough.

TUNING YOUR SETUP

To get PID control working for your setup, it will need to be tuned; this will involve modifying the constants KP, KD, and KI. There is no exact science to this and there will be a certain amount of trial, error, and intuition required before you find the right setup for your robot.

The following tips however should improve your tuning:

1. Start by modifying the KP constant and get the performance as good as you can before moving onto KD and then finally KI.

2. If the motor adjustments are too aggressive, swinging between too fast and too slow, reduce the constant.

3. If the motor speed isn't changing fast enough, increase the constant.

4. Make any change in small increments; even a very small change can have a dramatic effect.

Once tuned, each motor should settle down to a speed which is close to the encoder target.

Dreaming of a multicoloured Christmas

Build a Bluetooth remote-controlled Christmas light show

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**

This project combines simple woodworking with electronics to produce a remote control Christmas decoration. So you can get yourself in the Christmas spirit from the comfort of your own armchair!

The decoration is in the form of the four letters of XMAS (yes, we know that's cheating, but it saves on the number of LEDs we needed to buy). These are filled with green and red LEDs which are controlled by an Arduino.

The first step is to build the backboard. You will need a piece of 6 mm plywood sized 300 mm by 100 mm. Use paper templates to transfer the shapes of the letters on to the board, then mark out the positions of the LEDs, remembering to leave space between each of them and the edges of the letters. Drill a 10 mm hole for each LED. For best results, use a piece of scrap wood below and clamp your work to the bench. Use sandpaper to clean up the front face and holes. The sides of the letters are made from 3 mm plywood. Use a knife and metal ruler to score a line on the wood. Repeat this many times until you have cut all the way through. Cut strips 15 mm wide, then cut each of them to length with a junior hacksaw.

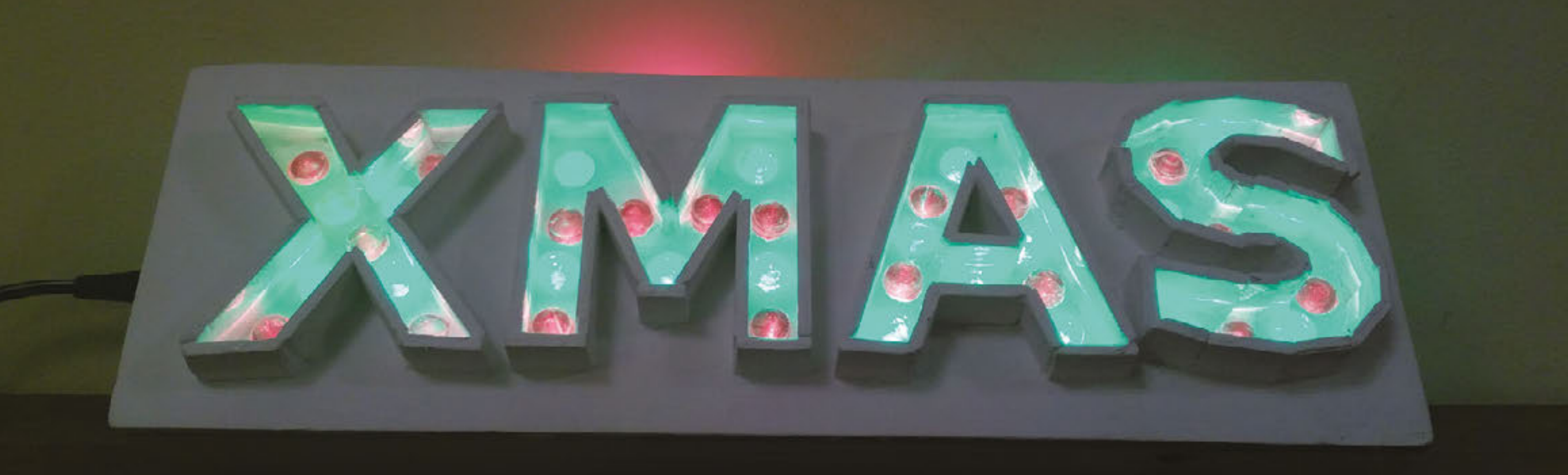
For the complex curves of the 'S', you will need to use many short pieces. Use sandpaper to form a bevel on each piece so that they fit together in a curve.

Glue your strips in place with wood glue and hold in place with pins and tape. Glue a block of wood 300 mm × 20 mm × 10 mm along the bottom edge of the back, ensuring that the narrow edge is attached. Wood glue is best left overnight to dry.

NOW IT'S TIME FOR THE ELECTRONICS

Fill any holes with a light-coloured filler and paint with white paint to reflect the light. Try to keep the paint out of the holes so that the LEDs will fit.

To give our decoration some Christmas pizazz, we are using red and green LEDs. These can't be run straight from our 12 V supply, so we are using resistors to limit the current. Solder a short length of wire to the



Above
Marking out the backboard

anode (long leg) of the LED and a 2.2 kΩ (2K2) resistor to each of the cathodes (short legs). Solder a short length of wire to each of the resistors.

For each letter, solder all of the anode wires together, then attach the resulting bundle to a long wire. Solder each of the resistor wires together and attach that to a long wire. Now connect up each of the anode wires to a single wire. Use heat-shrink or tape to stop the wires shorting on each other.

To test this has worked, connect the anode wire to the positive connection on the power supply and the cathode/resistor to the negative.

DO YOU WANT CHIPS WITH THAT?

The Arduino Uno is not capable of providing enough current to drive all of the LEDs. However, luckily for us there's a handy chip called the ULN2003, which contains an array of seven Darlington transistors; this often comes on a breakout board with four channels, designed for stepper motors. We can use this chip to switch or drive the LEDs.

Connect one of the GND connectors to the GND on the Darlington array chip. Connect up four of the inputs of the chip to the Uno's pins 2, 3, 4, and 5. Connect the corresponding output to the cathode wires for our four letters. The anode wire connects to the Uno's Vin connection.

We can test that the light works before continuing with the Bluetooth module later. The test program below demonstrates turning the lights on and off.

The first section sets up the variables.

```
int sequence;
//Pattern of LED outputs
unsigned char pattern[4][4] = {
                        {HIGH,LOW,LOW,LOW},
                        {LOW,HIGH,LOW,LOW},
                        {LOW,LOW,HIGH,LOW},
                        {LOW,LOW,LOW,HIGH}
                              };
```

Left
Forming the S

The **setup** section configures the pins 2 through 5 as outputs.

```
void setup() {
  // Configure all of the pins that control the LEDs
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  // Initial variable values
  sequence = 0;
}
```

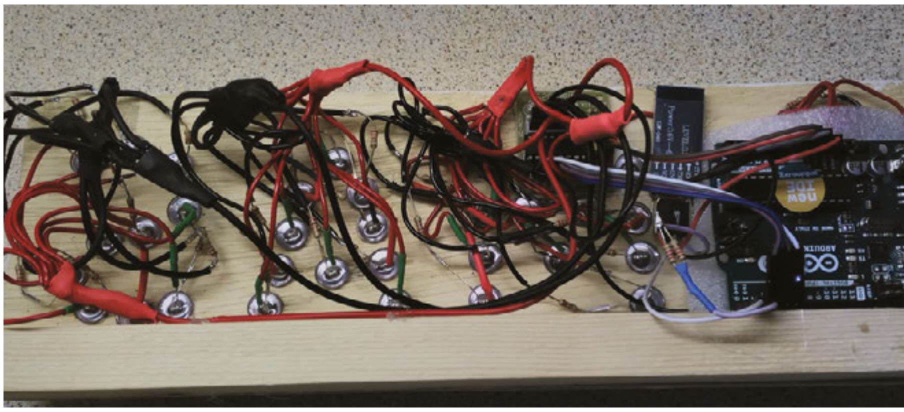
The **loop** sets the values of the outputs based on each row in the pattern array. Then, on the next iteration of the loop, the next row will be used. When it gets to the last row it resets back to 0, the first row. This tests that our wiring for the LEDs and Darlington array is working correctly. You can change the values in the matrix and see how it affects the LEDs.

```
void loop() {
  for (int ledpin=0; ledpin <= 4; ledpin++){
      digitalWrite(ledpin + 2, pattern[sequence]
[ledpin]);
   }
  sequence = (sequence + 1) % 4
  delay(500);
}
```

YOU'LL NEED

- Arduino Uno
- HC05 Bluetooth module
- ULN2003 Darlington Driver
- 20 × Red LEDs
- 20 × Green LEDs
- 43 × 2K2 resistors
- 9 × Header pins
- Wire
- Solder
- 9-12 V power supply 1 A
- 6 mm plywood
- 3 mm plywood
- Paint
- Wood filler
- Wood glue

Below
Wiring the letters

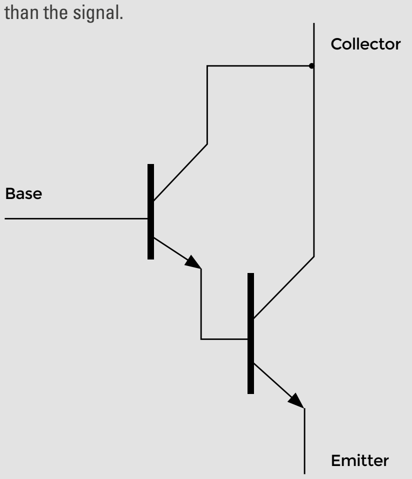


TOOLS

- 10 mm drill
- Saw
- Junior hacksaw
- Soldering iron
- Sharp knife
- Steel ruler
- Sandpaper

DARLINGTON TRANSISTORS

A bipolar transistor uses a small signal current to switch a load current hundreds of times larger. In 1953 Sidney Darlington from Bell Labs realised that if two transistors were put together then combined they could switch currents that were thousands of times bigger than the signal.



UNPLUG THE POWER

The Bluetooth module acts as a wireless serial port for the Arduino Uno data sent from the phone, which arrives at the Uno one character at a time. Although the Uno supports a hardware serial port, this is shared with the programming cable. So that we don't have to keep disconnecting the Bluetooth module to change the programming, a software serial library is used. This allows us to define pins 7 and 8 and receive RX and transmit TX.

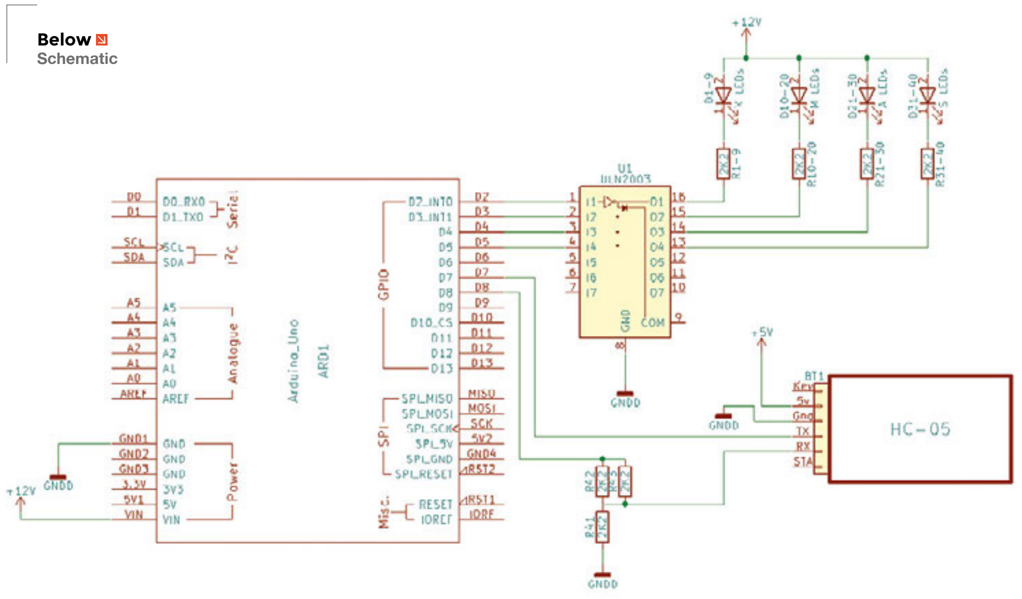
The Bluetooth module requires a 5 V power connection, GND, and two data lines – RX and TX. However, the data lines use 3.3 V logic while the Arduino Uno employs 5 V logic. Therefore, a simple level shifter will be required; it is possible to build this using a simple combination of resistors. The TX output of the Uno connects to a 2K2 resistor and this, in turn, connects to the pair of 2K2 resistors in parallel that are connected to GND. The RX input of the Bluetooth module connects to the junction of these two resistors – see the schematic below. The TX output of the module connects straight to the RX input of the Uno. The Uno will correctly process the lower logic levels of the module.

To connect your phone to the Bluetooth module, we need to pair them. This process uses the unique identifiers of the devices and a pin number so that the module trusts your phone to send commands. When you turn on Bluetooth, you can scan for nearby devices and select the correct one; you should find this is called 'HC-05'.

The other thing needed is some kind of remote-control software. There are hundreds of Bluetooth remote-control applications available in the Apple App Store and Google Play store. Here are two that work with this project:

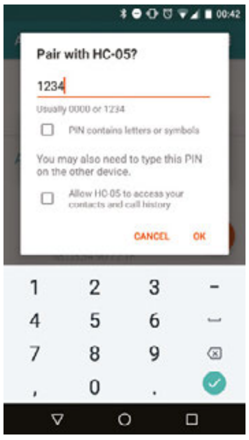
iPhone – Handy Bluetooth Arduino Controller by Paul Shelley: **hsmag.cc/RxvlXA**

Below Schematic



QUICK TIP

Sawdust mixed with glue makes a good substitute for filler.



Pair with HC-05?

1234

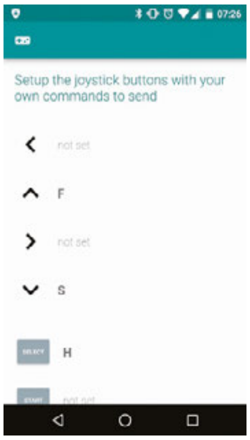
Usually 0000 or 1234

☐ PIN contains letters or symbols

You may also need to type this PIN on the other device.

☐ Allow HC-05 to access your contacts and call history

Above
Pairing the HC-05



Setup the joystick buttons with your own commands to send

<	not set
^	F
>	not set
v	S
SELECT	H
START	not set

Above
Configure the buttons

Android – Arduino Bluetooth controller by Giumig Apps: **hsmag.cc/wCRMha**

You only need three buttons to control the speed and turn the lights on and off. Configure the app to send a single letter when the buttons are pressed: **F** = Faster, **S** = Slower, **H** = Toggle Halt/Run.

We can use these single-letter instructions to determine how our lights flash. The second code example expands on our first. The first part now includes the library for the software serial port and defines a variable that uses that library. There are also variables for the speed and to determine if the sequence is running.

```
#include <SoftwareSerial.h>
char command;
bool running;
int speed;
int sequence;
// software serial RX = digital pin 7, TX =
digital pin 8
SoftwareSerial BTserial(7, 8);
//Pattern of LED outputs
unsigned char pattern[4][4] = {
                   {HIGH,LOW,LOW,LOW},
                   {LOW,HIGH,LOW,LOW},
                   {LOW,LOW,HIGH,LOW},
                   {LOW,LOW,LOW,HIGH}
                       };
```

The **setup** section also gains a new command, **BTserial.begin**, which tells the serial port to communicate with the BlueTooth module at a speed of 9600 bits per second.

```
void setup() {
  // Configure all of the pins that control the
LEDs
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  // Initial variable values
  waitTime = 200;
  running = true;
  sequence = 0;
  // Configure serial port for BlueTooth module
  BTserial.begin(9600);
}
```

The **loop** code adds logic in order to check whether there is data from the Bluetooth module and, depending on the value, to speed up, slow down or stop the flashing.

```
void loop() {
  if (BTserial.available() > 0) {
      char command = BTserial.read();
      switch (command) {
        case ‘S’:
          if (speed <  1000) {
                  speed += 20;
          }
              running = true;
          break;
        case ‘F’:
          if (speed > 50) {
                  speed -= 20;
              }
              running = true;
          break;
        case ‘H’:
          running = false;
          Break;
    }
  }
  if (running) {
//Write the pattern to the LEDs
  for (int ledpin=0; ledpin <= 4; ledpin++){
      digitalWrite(ledpin + 2, pattern[sequence]
[ledpin]);
   }
  delay(speed);
  sequence = (sequence + 1) % 4;
  }
}
```

FINISHING UP

It is a good idea to secure all of your components to the backboard. We used hot glue and also added a piece of insulating foam to stop the Uno shorting on the LED wiring. Enjoy your programmable, remote-control Christmas decoration! Happy Xmas!

QUICK TIP

There are smaller and cheaper Arduino-compatible boards that can work with this project. Check first that they have enough pins for the lights and a serial port.

IDEAS FOR ENCHANCEMENT

- Use PWM to control the light’s brightness
- Add more flashing sequences, try using a 3-dimensional array
- Build your name in lights
- Bigger letters, more LEDs

Build a three-foot trebuchet

Makers – dust off your drill and prepare for glory!

Everyone knows what a catapult is. It throws stones and flaming projectiles at castles. It is famous – and rightfully so. But for some reason the trebuchet, which is a different type of siege engine, is not so famous. Yet, it is much more powerful than a catapult. It can launch heavier projectiles longer distances and is much easier to build! Rather than using complicated twisted ropes, a trebuchet uses the power of gravity to launch its projectile. In this tutorial you will learn how to build one that is 3 ft (91 cm) tall and capable of launching projectiles at some pretty amazing distances.

Everything in this build is very common except for two small parts: a welded ring and several push-on external retaining rings. However, even these are easy to get and you can even improvise something as a replacement for them. We will take a closer look at them as they are needed in the build. And we will offer you possible alternatives.

This build is well within reach of a person who has minimal DIY skills. Can you use a drill and a hammer? Then in about six hours you will be launching tennis balls at your neighbour's house. But don't do that! Your neighbour might also read this tutorial and build one for themselves. Then you will have a right proper medieval warfare campaign right in your own back yard...

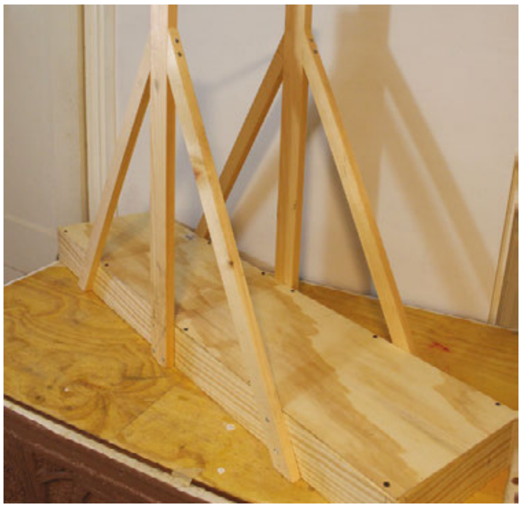
We purchased all the parts at a local (US) home improvement store. The wood cost $44.62; the various other parts, paracord, steel bar, nails, screws and everything else cost $28.26, for a total of $72.88. You can reduce this price significantly if you have some of the materials on-hand. A good example is the screws: if you buy full boxes, you'll have plenty for future projects. If you already have an assortment of nails and screws, you can reduce the cost of this project by $15.

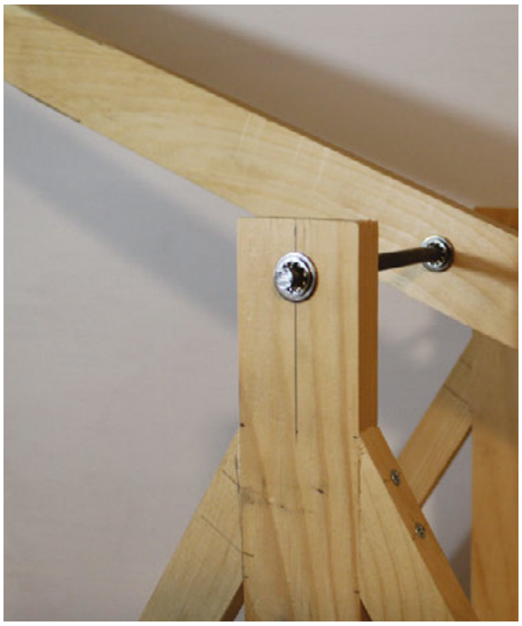
A trebuchet works the same way as a see-saw. If you sit down very quickly on one end of a see-saw, the other end pops up equally quickly. And the heavier the weight, the faster the pop-up. If you shift the fulcrum closer to where you push, the other end will travel even faster. A trebuchet capitalises on this. It uses the principle of a fulcrum and the power of gravity to launch projectiles rapidly into the air.

We will build this trebuchet in four major parts: the base, the swing arm, the ballast box, and the string/pouch assembly.

PART 1
THE BASE OF THE TREBUCHET

Measure and cut the 1 × 4s to form a base for the trebuchet. It is 48″ (122 cm) in length and 12″ (30 cm) wide. Notice that the 1 × 4s are on their sides. Screw them all together using #8 wood screws that are 2″ (51 mm) long. Cut one of the sheets of plywood in half and nail or screw these halves to the base. →





Above
The swing arm is mounted onto the trebuchet

Left
Creating a sturdy base for the trebuchet is important. It keeps it stable during the strong centrifugal force that shoots the projectile

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**

YOU'LL NEED

- **2 × sheets of plywood 2′ × 2′ (61 × 61 cm), 7⁄16″ thick**
- **12′ (366 cm) pine wood** 1″ × 4″ (25 × 101 mm)
- **12′ (366 cm) pine wood** 1″ × 3″ (25 × 76 mm)
- **12′ (366 cm pine wood** 1″ × 2″ (25 × 51 mm)
- **1 × length of steel rod ⅜″ (9.5 mm) thick × 36″ (91 cm)**
- **8″ × 16″ (20 × 41 cm) piece of leather or cloth**
- **8′ (2.4 m) paracord**
- **1 × 6d (51 mm) nail**
- **1 × 1″ (25 mm) eyebolt**
- **Ballast** Anything heavy, such as stones, sand or weights
- **1 × ⅞″ (22 mm) Welded ring** (a keyring will work as a substitute)
- **8 × ⅜″ (9.5 mm) push-on external retaining rings** (optional)

TOOLS

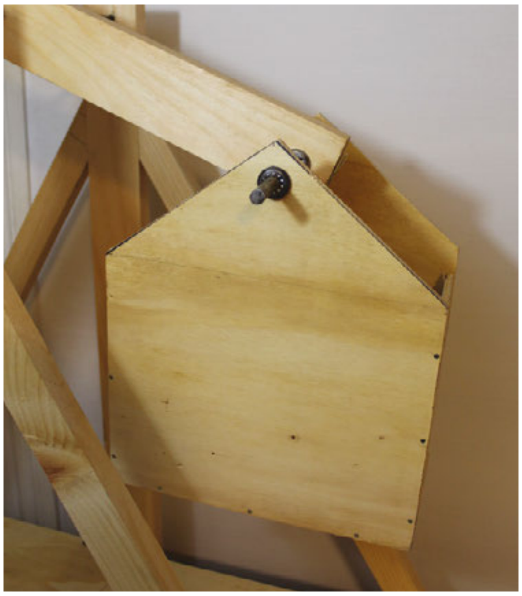
- **Electric drill**
- **⅛˝ (3 mm) drill bit for pilot holes**
- **⅜˝ (9.5 mm) drill bit** (match this drill bit size to the diameter of your steel rod)
- **Screwdriver or screwdriver bit for drill**
- **Hand saw**
- **Hack saw**
- **Hammer**
- **Measuring tape**
- **30/60 triangle** (optional)

It is important to have this smooth and flat surface because the sling will ride on top of this when the trebuchet fires.

Measure and cut two lengths of 1 × 3 to 30˝ (76 cm) in length and screw them vertically to the base, one on each side and each right in the middle of the base. Drill a ⅜˝ (9.5 mm) hole 1 inch (25 mm) from the top of each of these. Cut your steel bar to 14˝ (35.5 cm) and insert it through these two holes right across the trebuchet. This is the axle (fulcrum) for the swing arm. These vertical supports are not strong enough to sustain repeated use of the trebuchet because there is significant centrifugal force applied in a front-to-back direction. We brace against this force by adding four support pieces that are at an angle to the uprights. Measure and cut four of your 1 × 2s to 30˝ (76 cm) then cut an angle on the ends of each. All four pieces are identical and each piece has a 30-degree angle cut on one end and a 60-degree angle cut on the other end. Screw these to the uprights.

PART 2
THE SWING ARM

To make the swing arm, cut one of your 1 × 3s to a length of 52¼˝ (133 cm). At one end of it, drill a ⅜˝ (9.5 mm) hole 1 inch (25 mm) from the end. This hole is for the ballast basket we will make. Drill a second hole 12˝ (30 cm) from that same end of the swing arm. This hole is for the fulcrum. Mount the swing arm onto the structure of the trebuchet using the fulcrum hole and the steel rod.



Above The ballast box should be sturdy and strong in order to hold the weight of the ballast

QUICK TIP

The plywood is thin. Use 1˝ (25 mm) finish nails for assembly of the ballast box.

KEEP THE BUILD CLEAN

Always drill pilot holes before putting in screws. This will ensure that the screw goes into the wood straight, and will help prevent any cracking or splitting of the wood. Use a ⅛˝ (3 mm) drill bit for pilot holes.

PART 3
THE BALLAST BOX

Our final piece of wood is the second piece of plywood. Measure out a box to make. The box is made up of five pieces and it has a ⅜˝ (9.5 mm) hole 1 inch (25 mm) from the top. When making this box you should be aware that it needs to swing freely without hitting anything on the trebuchet, including the swing arm and the base of the trebuchet. Attach it to the swing arm hole one inch from the end. Cut a piece of your ⅜˝ (9.5 mm) steel bar for this.

The house shaped ballast box is 10˝ (25 cm) wide and 12˝ (30 cm) tall. You can trim the top 4˝ (10 cm) of it to form the triangular roof of the house shape. Measure and cut two of these. The sides of the box hold those house shapes 6˝ (15 cm) apart. You have a lot of freedom in making a ballast box. Just be sure it doesn't rub on any part of the trebuchet base or swing arm throughout its full swinging motion.

PART 4
THE STRING AND SLING

This is the most important part of the build, but also the easiest. The sling itself is a piece of cloth or soft leather 8˝ × 16˝ (20 × 41 cm), folded in half. You can trim the cloth of the sling so the top of each end is triangular, similar to the house shape of the ballast box, but triangular-shaped on both ends of it. Cut two lengths of paracord each to 48˝ (122 cm) and tie them to the tips of the sling. Then attach one to an eyehook at the end of the swing arm. And on the other length of paracord, attach a metal ring. We used something called a welded ring, bought at a hardware store for about a dollar. You can use a keyring for this. It is important to use a metal ring that will easily slide over the nail without catching or getting stuck.

In the final picture you can see a nail sticking out of the end of the swing arm. This nail is critical. Use

FINE TUNING

Want to improve the look of your trebuchet? There are a few things you can do, including tapering down the swing arm so it's thinner at the string/pouch end. This will also increase the efficiency. We tapered the swing arm down to a width of 1½˝ (38 mm) at the end. You can also paint or stain your trebuchet any colour or theme you desire.

THE GOLDEN RATIO

Want to build a larger, or smaller, trebuchet? The most important thing to consider is the ratio of the swing arm on both sides of the fulcrum. Keep this at 3.75 to 1 and you will get maximum efficiency. The length on the sling side is the 3.75 and the length on the side of the ballast is the 1.

a 6d (51 mm) nail and file the head off of it. Hammer it into the end of the trebuchet so it sticks out.

Locking things in place

For this project we use two lengths of ⅜″ (9.5 mm) steel rod. You can just feed these lengths through the holes and the trebuchet will work. But with repeated use, things will start to slip. To prevent this you should secure the parts in place on the rod with washers and some kind of locking rings. The easiest and cheapest way to do this is with push-on external retaining rings. They are simple little metal rings that you push by hand right over the steel bar. They have small internal teeth on them and stay locked onto the bar wherever you place them. We purchased some at a hardware store for 40 cents each.

FIRING THE TREBUCHET

Fill the ballast box with weight, put a projectile in the sling and pull the swing arm down. Place the ring over the release pin. Inspect everything to ensure the paracord is laid out straight on the base of the trebuchet and not twisted. Fire it by letting go of the swing arm. As the swing arm moves in its arc, it pulls both strings evenly, bringing the pouch along the base of the trebuchet. At a point in its swing near vertical, the ring will slide right off the nail/pin. This opens the sling and releases the projectile.

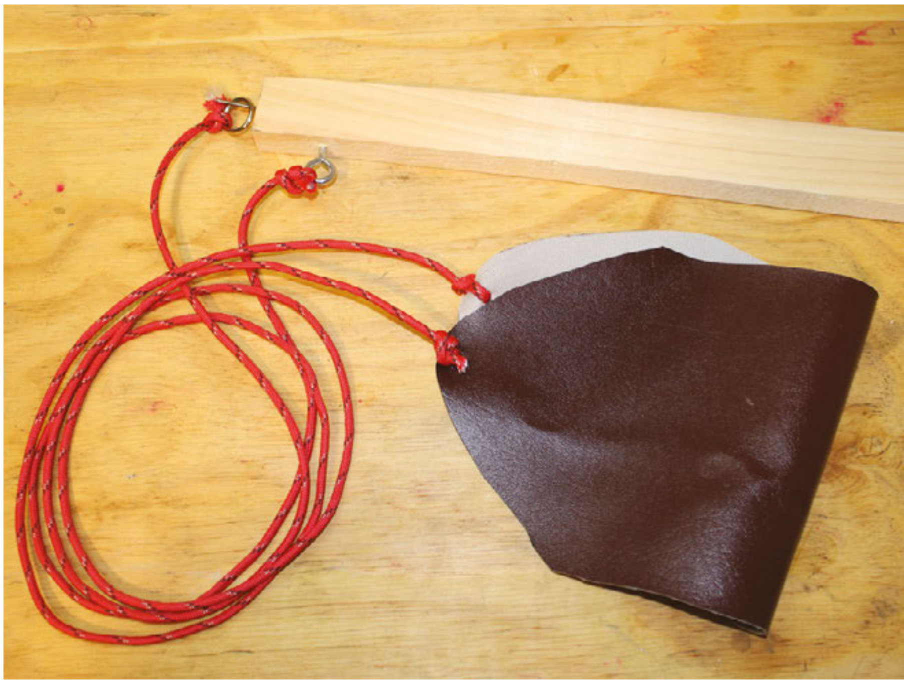
This trebuchet is a simple machine with only two moving parts. If you have trouble launching projectiles, here are some things to look for:

First, you should look to the ballast basket. Is it rubbing or hitting on something? Does it not allow for smooth motion of the swing arm? Adjust it, resize it, or trim it as needed. If your trebuchet fires its projectile straight up or backwards, your nail needs to lean more forward. If your trebuchet fires its projectile at the ground or not too far then your nail needs to lean less to the front. But just make these adjustments to the nail very slightly – and test it.

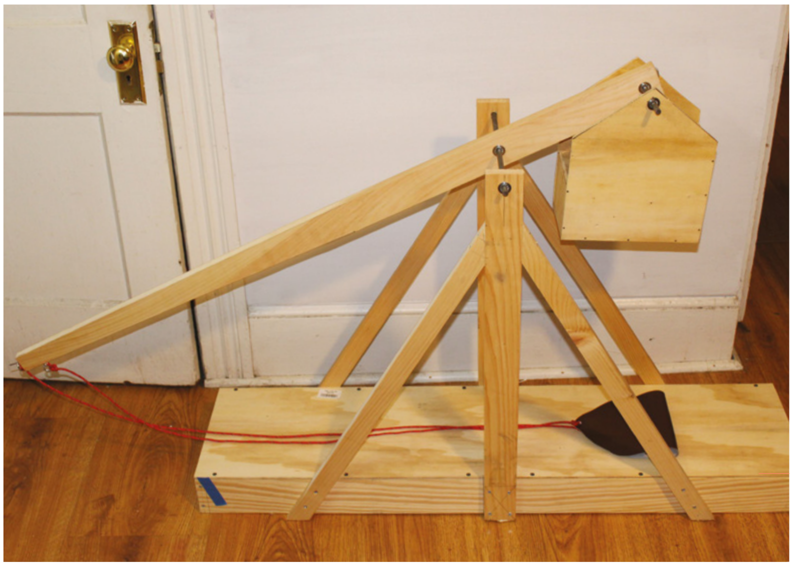
MORE POWER!

One of the best things you can do to make your trebuchet even more powerful is to add a bearing around the steel rod at the fulcrum of the swing arm. This will take a lot of the friction out of the motion.

How heavy your ballast is depends on your trebuchet. Start with 10 lb (4.5 kg) of weight and give it a try, then add more weight. It will get to a point where adding more weight will not throw your projectile further. Want to control the distance it shoots? You can do this by controlling the weight in the ballast box. An easy thing to do is pre-make bags of sand and label them. Then you can just add or remove bags of sand to the ballast box, keeping track of the weight inside and the distance it throws. □



Above This photo shows you the shape of the sling, where to put the eyebolt, how to install the pin, and how to tie it all together with the two pieces of paracord



Below The completed trebuchet as built in this tutorial. Be careful where you launch your projectiles!

pi-top

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

The modular laptop with sliding keyboard

8HR BATTERY LIFE	14" FULL HD 1080P SCREEN	180° HINGE	CUSTOM PASSIVE COOLING BRIDGE	MODULAR RAIL

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.

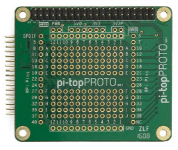
The modular desktop

14" FULL HD 1080P SCREEN	MODULAR RAIL	ADJUSTABLE VIEWING ANGLES

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-top**CEED as their solution to Computer Science and STEAM-based learning.


pi-topPROTO

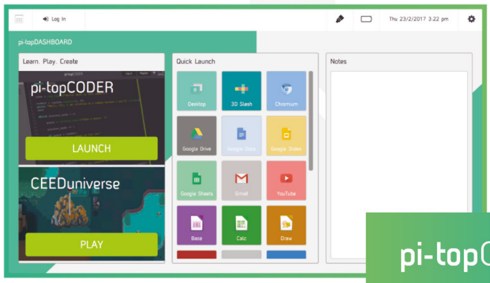
pi-topSPEAKER

pi-topPULSE

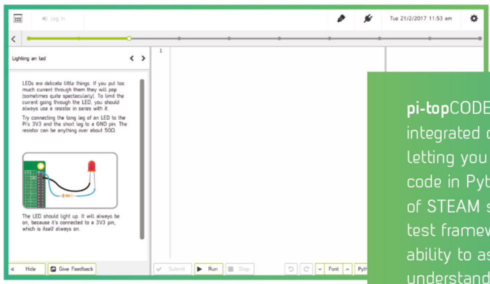
Modular **Accessories**

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

The OCR* endorsed **pi-top**OS (Operating system) platform comes pre-installed on the SD card shipped with every unit. **pi-top**OS 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-top**CODER and have fun learning to code with CEED**universe**!



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.

CEED**universe**

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

HackSpace
TECHNOLOGY IN YOUR HANDS

FIELD TEST

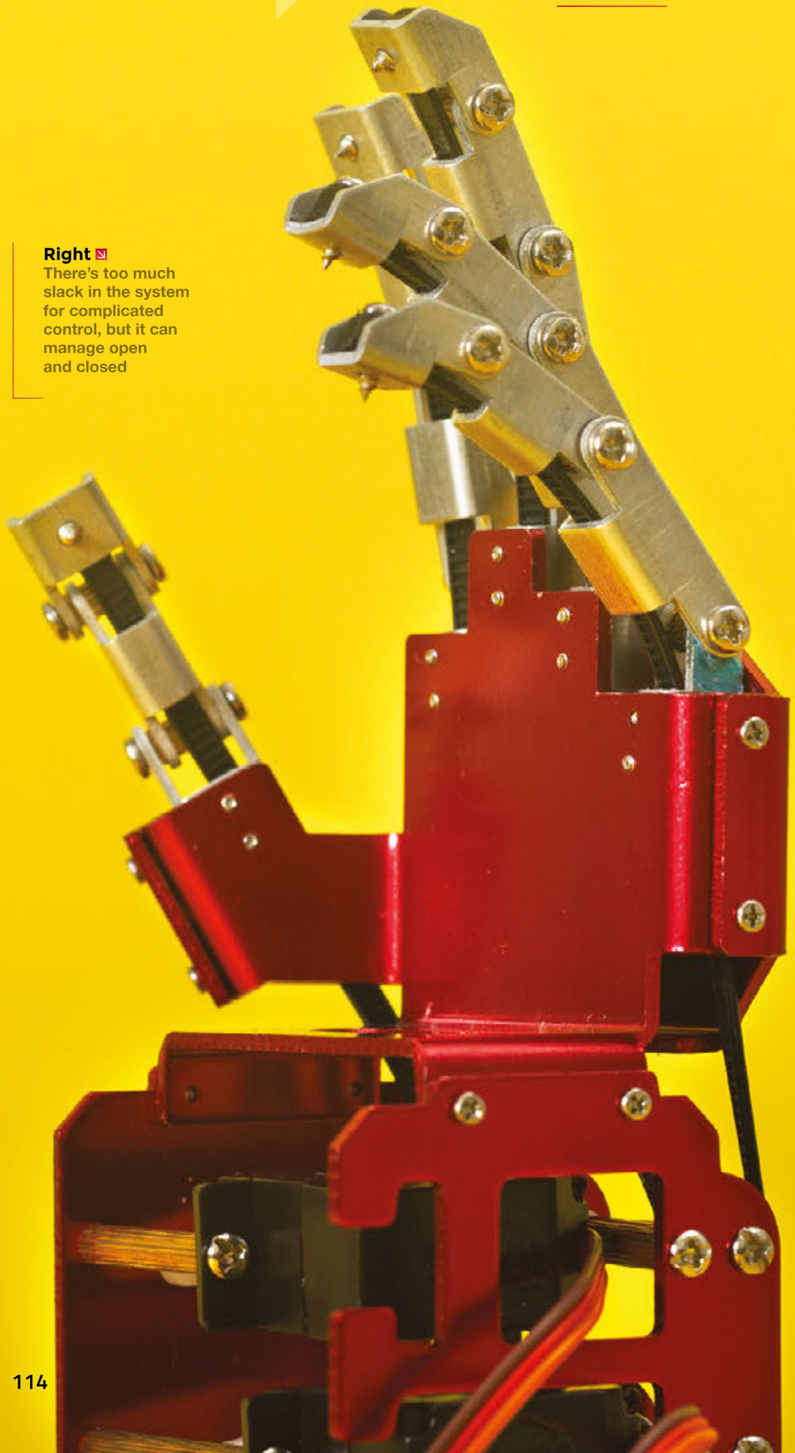
HACK | MAKE | BUILD | CREATE

Hacker gear poked, prodded, taken apart, and investigated

DIRECT FROM **SHENZHEN**

Robotic hand

Ben Everard looks further afield in his quest to build a robotic army



Right There's too much slack in the system for complicated control, but it can manage open and closed

There's something iconic about a robotic hand. It seems far more technologically advanced than, say, a wheeled robot, despite the fact that you don't really need that much technology to build one. The hand straddles the border of human and robot in a way that few other body parts do. It's a staple of science fiction, both utopian and dystopian. Naturally, we wanted one, so we bought one direct from China.

Buying things from Chinese manufacturers is a quite different process from getting things from a supplier in Europe or America. For starters, brand names are often non-existent and shops come and go. The actual stock is often put together by an unnamed factory and then sold via several distributors on several websites. Without a brand-name, it's impossible to know for sure whether you're getting the same item, but if they look the same, they probably are. We bought ours from the XuQi Hobby Store on AliExpress, but you can get the same item from several Chinese-based sellers: **hsmag.cc/CNLYCC**.

These direct-from-China stores are an integral part of the hacking world because they offer products that either aren't available elsewhere, or at vastly lower prices. They provide individual hobbyists with a range of stock that, just a few years ago, was only available to large corporations.

YOUR BOOTS. GIVE THEM TO ME

Shipping from China is rarely quick, but it is cheap. Our hand was £44.78 including free shipping, and it arrived in about three weeks. Before it arrived we got a message on AliExpress linking to details of how to download the instructions from Baidu – not an entirely trivial process for an English speaker as the website was in Chinese, but the instructions worked.

The package arrived as a slightly battered box containing a few metal plates, a myriad of screws and connectors, and six small servos. The instructions were easy to follow and assembling the hand took a couple of hours of quite fiddly work.

It could have been easier if all of the screws had been magnetic.

Robotic hands come in many different forms, and one of the key differentiators is the number of degrees of freedom (DOF). Essentially, each DOF is a part that can be moved independently. The most basic hands have one DOF, which means that they can be opened or closed, but everything opens or closes at the same time. 5 DOF means that each finger can be moved independently, but there's only one movement in each finger. Actual human hands have far more DOF than this since each finger can be moved side to side and forwards and backwards as well as opened or closed. There isn't a 'best' number of DOF, and it all depends on your needs. Plenty of useful robotic hands have 1 DOF, and this enables them to pick up objects and manoeuvre them. Perhaps the biggest advantage of 5 DOF is that it allows more human-like gestures.

"**The finished result is quite wobbly,** but each servo does control a finger, and it does legitimately have 5 DOF"

COME WITH ME IF YOU WANT TO LIVE

The finished result is quite wobbly, but each servo does control a finger, and it does legitimately have 5 DOF. With quite a lot of slack in the system, accurate positioning isn't really possible, but you can broadly open and close each finger (and the thumb) independently. It takes about 50 degrees movement of the small servos driving the hand to move a finger from closed to open. This is easy to control from any standard servo driver (though one isn't supplied with the hand), but bear in mind that your power supply will have to be sufficiently powerful to drive all servos at once.

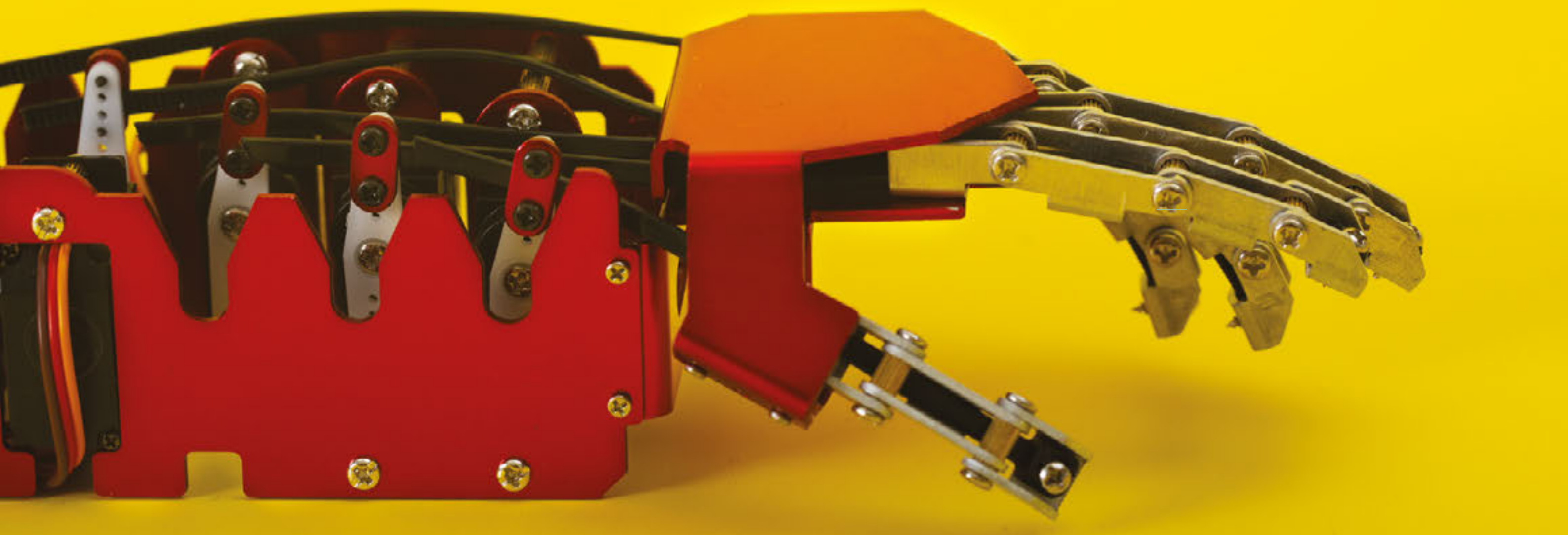
There's no feedback system in the hand, so there's no sense of touch. If you tell a servo to move a finger to a certain position, it will use all its power (which is very little) to try to do this. If you try to use this hand to pick things up or manipulate objects, you'll most likely just stretch all the linkages in the hand or burn out the servos.

Without accurate control, any sort of feedback about pressure on the fingers, or a more powerful grip, this hand isn't really useful for picking things up. Some gestures, however, are quite possible. The hand can manage a passable thumbs up, fist, open hand, and a few other simple gestures. The lack of side-to-side movement in the fingers does limit the sign-language potential, particularly for vulgar-minded Brits.

That isn't necessarily a problem because at this price there's plenty of possible uses for it, from costumes or props to novel ways of visualising data (a finger-counting clock, anyone?). Grasping of small items just about works, but only if the item is the correct shape and in the right place. It you actually want to move things, you'll have more success with a simple pincer, but let's be honest, no one actually wants a robotic humanoid hand because it's practical. □

DIRECT FROM **SHENZHEN**

Below
The screws poke through each finger. This can increase grip, but may also cause problems



Can I Hack It?

Novelty Christmas tree

Learn how to hack Christmas decorations to do your bidding!

Les Pounder

@biglesp

Les Pounder is a maker and author who works with the Raspberry Pi Foundation to deliver Picademy. He also helps teachers/learners to become creative technologists. He blogs at **bigl.es**

YOU'LL NEED

Christmas Decorations Xmas Tree
Battery-operated dancing Xmas tree 30 cm

COST

£11.49

WHERE

hsmag.cc/QcqgNg

Christmas is here and with it comes a slew of lights, musical cards, and animated figures that dance to holiday songs. So what can we hack and is it worth it? We take a look at a dancing Christmas tree that spins, shakes and sings… for a very long time! We look at the components and the circuit board that connects them to understand how it works and offer suggestions as to how this cheaply purchased decoration can be hacked using boards such as the Raspberry Pi and Arduino. Let's start with the most accessible part of the tree, the batteries!

BATTERY BOX

Powering the tree we have three AA batteries (1.5V per AA) in a common battery compartment, which has a space for an on/off switch but sadly none is present. This can be easily hacked in place using a common sliding switch, and this may just save your family's Christmas! Powering the unit from an external supply is possible as the 4.5V battery connection is direct to the main PCB (red and black wires.) So using a USB supply is possible, either through a computer or a power bank. USB is 5V so a little over 0.5V difference to the stock voltage, not too different to fresh alkaline batteries which can be up to 1.6V each. You can either create a voltage dropper or run the risk of running the unit at 5V, which may shorten the life of the tree, but not by much.

CONTROLLER PCB

The controller PCB is quite small, but it contains the chips for motor control and music playback. The input that triggers the motor drivers to power the motors occurs when the user presses one of the tree's hands (cunningly labelled 'Press Here'). The button is connected on one side of the PCB to the 4.5V VCC connection and when the button is pressed, the button connects to a Ground pin, which drops the voltage briefly to 0. This input is processed by an anonymous chip on the circuit board, sadly, and as is common with many cheap mass-produced electronics of this kind, the chip is covered in epoxy which protects the identity of the chip. This is called 'COB', 'Chip On Board' and is used to protect the investment made in the board. You can dissolve the epoxy, but for this object it isn't worth the time as we can see how the input and output works from the PCB.

As mentioned before, the board also controls music playback, and we get a 'lovely'

Right This Christmas tree may not look like much, but inside there are plenty of components that we can hack and control using boards such as Raspberry Pi and Arduino rendition of 'Jingle Bell Rock'. The speaker is directly soldered to the PCB which means that the COB has an amplifier built in, lowering the cost of production.

MOTORS

The Christmas tree has two DC motors and across the terminals of the motors are a 100 nF capacitor (printed 104 on the capacitor) used to smooth the current going to the motors. Each motor is offset using plastic guides. The motor at the top controls the 'wiggle' and 'shake' of the tree; this includes a 90 degree connection that converts vertical motion to horizontal. The lower motor is also offset and is used to spin the tree around; it connects to a central point using a gear. These plastic guides provide simple movement for simple motors.

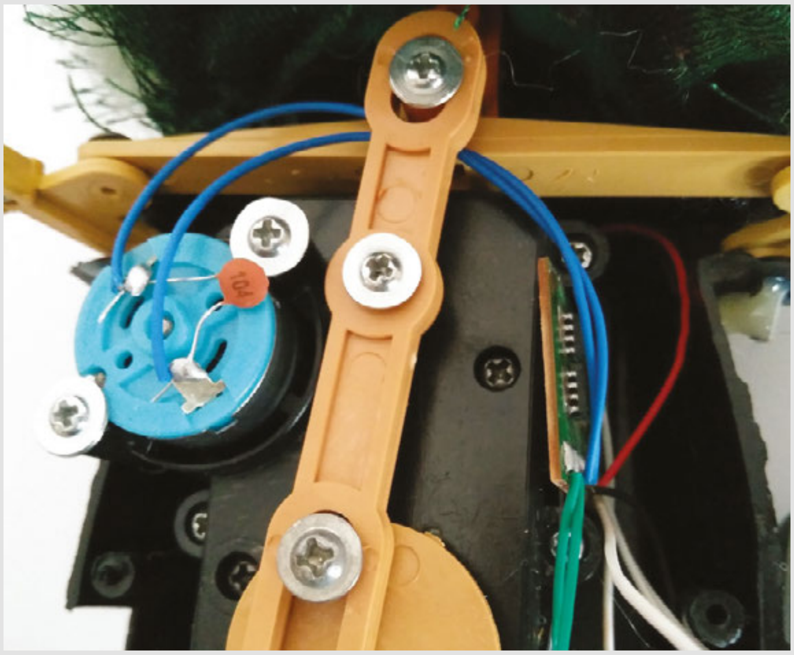
The motors are controlled by two NY9M006A single-channel motor drivers in an 8-pin plastic SOP (Small Outline Package) chip, and a quick look at their data sheet shows that they are capable of driving motors up to 6V at an absolute max output current of 1.2A; ideally these chips would never be pushed that hard and from the data sheet, the typical output current at 4.5V is 0.11 milliamps. If the chips are pushed too hard then they will generate heat, but luckily the chip has a thermal shutdown circuit that will protect it from harm.

HACKING

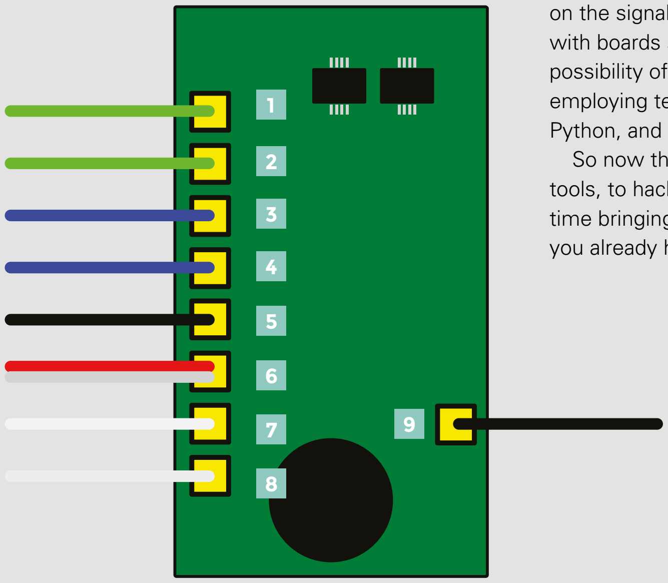
So can this board be hacked? Short answer: yes. If using an Arduino or another board with 5V logic, you can connect straight up to the PCB and independently control all aspects of the Christmas tree. So, triggering the tree using an ultrasonic distance sensor, PIR or photo resistor is possible, which would be great for an embedded ambush trap to scare your friends, or for interactive displays that run on demand.

If you are using a board with 3.3V logic, such as Raspberry Pi, micro:bit or ESP8266 then you cannot directly connect to the PCB, as using 5V logic with the 3.3V GPIO will cause damage to the pins, and possibly the board itself. To use a 3.3V board with this Christmas tree we will need to use a bidirectional logic level converter, which can be found online very cheaply. Place one in the path between the PCB and your 3.3V board and it will buffer and forward on the signal at the correct voltage. Using the tree with boards such as the Raspberry Pi opens up the possibility of internet-enabled Christmas decorations, employing technologies such as Node-RED, MQTT, Python, and even Scratch.

So now that you have the knowledge, and the tools, to hack a Christmas tree, have a great holiday time bringing your tree to life using technologies that you already have in your workshop/home. □



Above The circuit board that controls the tree is easy to interface with: all you need is a soldering iron and some hot glue to hack this simple decoration



1, 2	**Bottom motor**
3, 4	**Torso motor**
5	**Battery GND**
6	**Battery VCC & button**
7, 8	**Speaker connection**
9	**Button** (in hand of tree)

INTERNET OF TREES

There is plenty of space inside the Christmas tree for us to install a Raspberry Pi Zero W and a USB power bank. Using a remote connection (SSH), you can control the tree using Python and MQTT, a lightweight message system that enables devices to send and receive data over vast networks. Or you can use Node-RED to control the tree over the internet!

ONLY THE BEST

Handheld console for hackers

Build yourself a portable boredom eradicator

Gaming gets a bad rep. Despite numerous studies disproving links to increased violence and others demonstrating how regular gaming sessions can boost hand-eye coordination and problem-solving skills, there are those who still think that a go on an Xstation or PlayBox will rot your mind.

Learning to build electronic equipment or write your own computer programs, though, are endeavours against which nobody could argue – which is why it's great to see the two worlds combine in the form of do-it-yourself console kits.

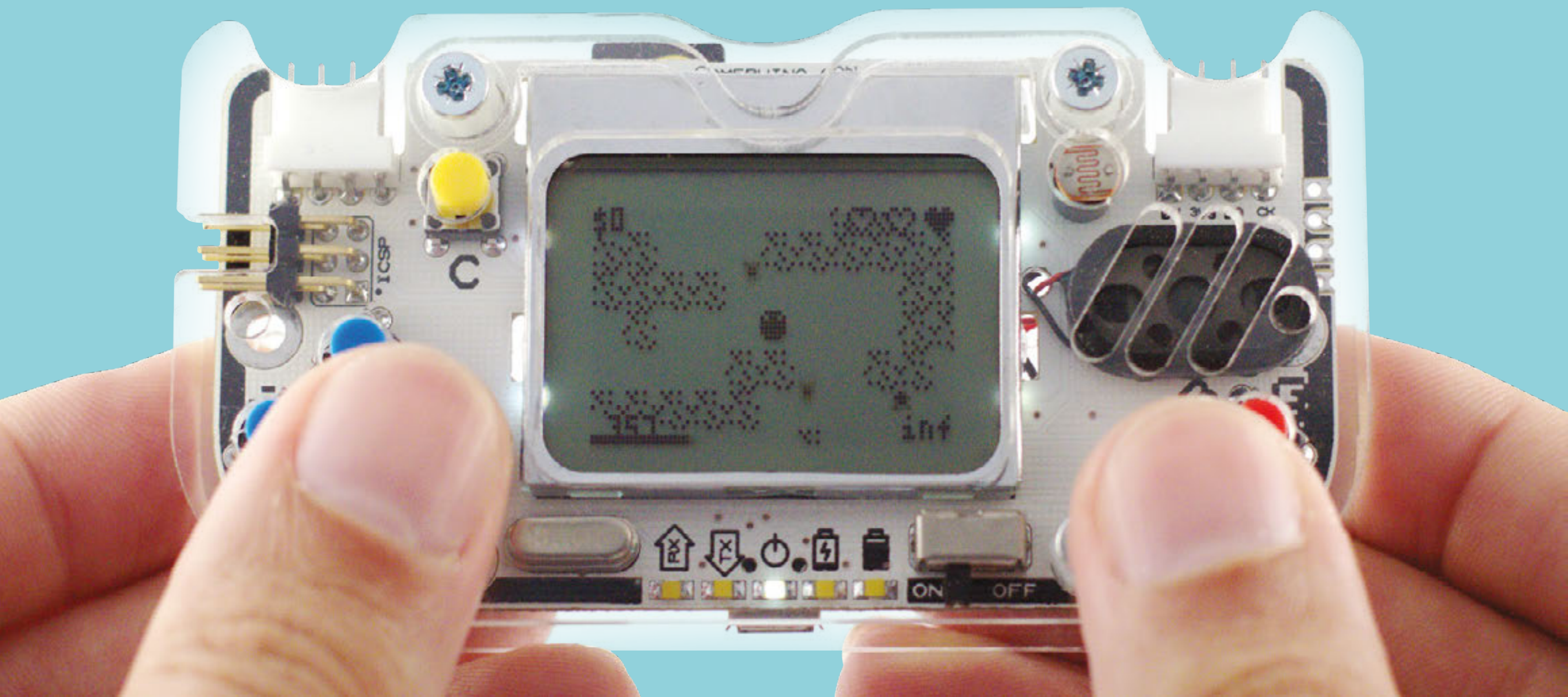
Ranging from compact, low-cost devices which encourage you to write your own games to more complex kits that need to be carefully soldered from individual components, DIY consoles are a great way to learn new skills and to get younger makers interested in electronics and coding. Now, too, is a great time to get started: over the last few years a wide range of devices and kits have hit the market, and none requires anything in the way of prior experience.

THE FUN FACTOR

Boasting low-resolution displays and kilobytes, rather than gigabytes, of memory, the consoles on test aren't going to give the latest triple-A titles a run for their money. They will, though, guide you through making the most of the hardware to produce some impressive creations which can then be shared with the world.

With four of the most impressive kits on test – the Gamebuino, MAKERbuino, Arduboy, and Creoqode 2048 – it's time to see which machine is king of the hill and which are nothing but cannon fodder.

Below If you want to exercise your brain as well as your thumbs, why not build your own games console?



BEST OF BREED

Gamebuino vs Makerbuino

GAMEBUINO ◈ **£49.72** | **gamebuino.com**

MAKERBUINO ◈ **£43.47** | **makerbuino.com**

It's near-impossible not to compare the Gamebuino and the MAKERbuino, for at their hearts they are one and the same. Based on an Atmel ATmega328P microcontroller running at 16MHz and an 84 × 48-pixel single-colour front-lit liquid crystal display (LCD) salvaged from old Nokia mobile phones, the two devices are designed to be entirely interoperable: a game written for the Gamebuino will work unmodified on the MAKERbuino and vice versa.

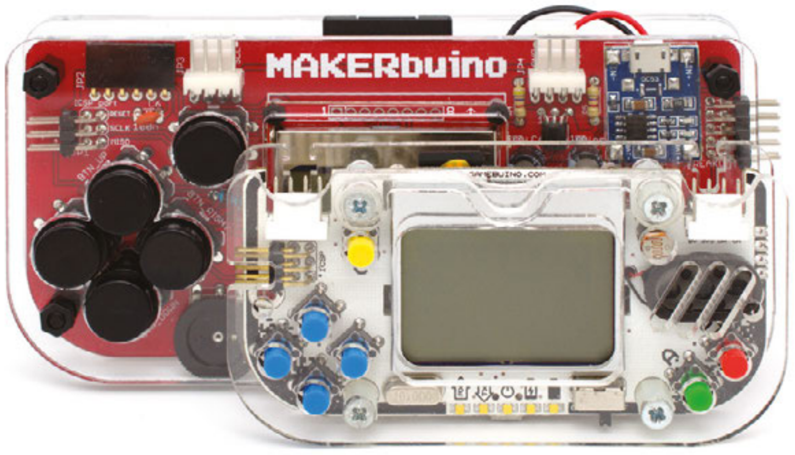
The Gamebuino is the original design, and the brainchild of Aurélien Rodot. An easily pocketable design measuring just 95 mm by 49 mm and 19 mm in thickness in its bundled casing, the Gamebuino comes ready-to-use out of the box.

The MAKERbuino came later, when tinkerer Albert Gajšak approached Rodot to discuss a spin of the open-source design which would turn it into a truly do-it-yourself console soldering kit. The most obvious result of the change in design, aside from receiving it as a box of loose components: a major increase in size at 139 mm by 66 mm and 26 mm in thickness.

HANDS-ON

As a soldering kit, the MAKERbuino works well. The instructional webpage is clear and the through-hole components easy for a beginner to handle, though mounting the charger board for the bundled lithium-polymer battery can be a little tricky. The design benefits from a few other changes, too, including a headphone jack for private listening and double the battery capacity – enough for around 24 hours of active play time, compared to 12 for the Gamebuino.

The two consoles operate in exactly the same manner. Games are written in the Arduino Integrated Development Environment (IDE) using a framework written by Rodot, then compiled and stored on an SD card. This SD card – full-size in the case of the MAKERbuino, microSD for the Gamebuino – is inserted into the console and a loader running on the microcontroller allows you to pick which game you'd like to play next.



The loader is the key feature of the -buino family: as well as allowing you to carry your entire game collection in your pocket and switch between them at will, it is smart enough to support save files – dumps of the electrically erasable programmable read-only memory (EEPROM) portion of the ATmega microcontroller – allowing progress and high scores to remain even as you load and unload games from memory.

As for which of the two consoles deserves your attention, it entirely depends on the sort of experience you're after. If you want something truly portable and are looking to concentrate on the coding side of things, the Gamebuino is the obvious choice; if you want to flex your soldering muscles, the MAKERbuino is a better option at the cost of pocketability. In either case, you're unlikely to be disappointed. ◻

VERDICT

Gamebuino

Neat, quick to get started, and plenty of hacking potential.

5/5

Makerbuino

All the fun of the Gamebuino, but for DIYers.

5/5

BEST OF BREED

Arduboy

ARDUBOY £50.50 | arduboy.com

If your key consideration in a handheld games console is portability, the Arduboy is about as small as you're likely to find and still be able to use. Measuring 53mm by 85mm and just 6mm thick, the Arduboy is like a small stack of credit cards — but credit cards you can use for some on-the-go gaming.

As with the other devices on test, the Arduboy uses an Arduino-compatible microcontroller, the Atmel ATmega32U4. This is connected to an impressive 128×64 single-colour organic light-emitting diode (OLED) display which looks best in the dark but, sadly, does suffer from banding effects when displaying horizontal lines – as demonstrated in the bundled platform game Mystic Balloon.

There's no assembly to worry about with the Arduboy, and once charged via a micro-USB cable, a simple flick of a switch is enough to get you started. The device holds a single game at a time, a major disadvantage over the larger Gamebuino and MAKERbuino, and new games are loaded directly from the Arduino IDE. There's no permanent storage here, either: any progress you've made in a game is lost when you load a new game in its place.

Writing your own games for the Arduboy is similar to doing so for the Gamebuino and compatibles: libraries are provided to handle things like sprites and sound effects – though this is limited to single-channel beeps and boops, a disappointment compared to the four-channel polyphonic capabilities of the Gamebuino family – and development takes place in the Arduino IDE.

Given its small stature, there's little surprise to see that the Arduboy is missing some of the features of its larger stablemates. In addition to the disappointing single-channel audio, the Arduboy lacks any easily accessible expansion headers to make use of the spare pins on the ATmega microcontroller, and there's no way to connect multiple Arduboys together for multiplayer gaming.

The biggest issue, though, is with the single-game nature of the device. While a Gamebuino compatible will allow you to carry your entire library wherever you go, switching games on an Arduboy takes a laptop or desktop with a copy of the Arduino IDE installed – a major hit to an otherwise extremely portable pocket-size device.

The Arduboy can't be faulted on quality, however. The chassis includes an acrylic front which protects the circuit board and OLED screen and a metal back, giving it surprising heft and a very solid feel. The buttons are responsive and the layout will be immediately familiar to anyone who remembers Nintendo's classic Game Boy machines.

As a device to impress your friends, the Arduboy can't be beaten; as a tool for learning electronics and programming, though, the Gamebuino family is a better choice.

Left The Arduboy is highly pocketable, taking up the same space as a small stack of credit cards

"**As a device to impress your friends, the Arduboy can't be beaten;** as a tool for learning electronics and programming, though, the Gamebuino family is a better choice"

VERDICT

If you're looking for something to slip into a pocket, the Arduboy is a great choice.

4/5

BEST OF BREED

Creoqode 2048

2048 £200.90 | creoqode.com

Nobody could ever accuse the Creoqode 2048 of being pocket-friendly. Measuring 293 mm by 108 mm and 33 mm thick with an overall weight of 538g, the 2048 is an absolute beast of a machine in an eye-catching smoked acrylic finish.

The size of the 2048 is almost entirely down to its choice of display, a 64 × 32 matrix of programmable red, green, and blue (RGB) light-emitting diodes (LEDs) originally designed for use in industrial signage applications. Once assembled, the display is an incredible thing to behold: such a low resolution as to make each individual pixel highly visible, but bright and colourful as it shines through the front acrylic panel.

Sadly, getting to that stage is a painful process. The 2048 is supplied as a self-assembly kit, which should supposedly require no soldering. Before our sample proved stable, however, we were forced to follow a guide on the Creoqode website to remove the too-thin wires from the battery holder and replace them with thicker-gauge versions – an extremely fiddly process no beginner should be expected to carry out.

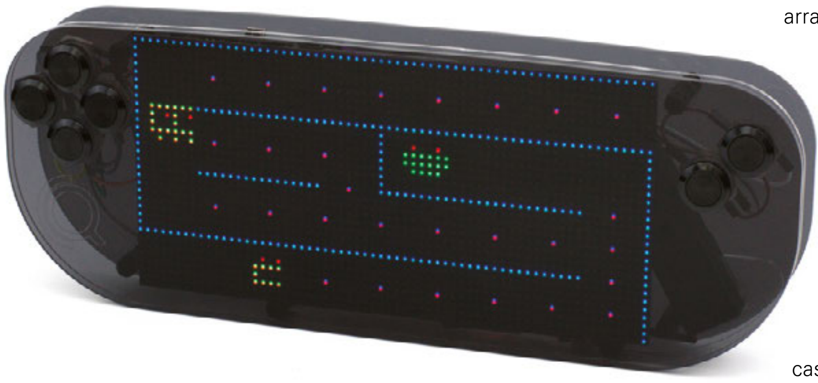
Even ignoring this issue, the 2048 is an awkward kit to put together. The bundled instruction manual has black and white assembly pictures which are too low-quality to be of use, and the web-based version isn't a whole lot better. A wire loom is provided which proves to be too short to connect all the components, and although extensions are included it's never clear when they should be used – meaning you reach the end with extensions to spare and some very taut wiring.

The heart of the 2048 is a rebadged Arduino-compatible microcontroller based around the Atmel ATmega2560, and unlike the other devices on test it's full size: you can easily remove the microcontroller and use it for other projects, should you so choose, and accessing the spare pins is a breeze.

With 256kB of program memory to the 32kB of the rival devices, you'd expect the 2048's games to be the most impressive. Sadly, they're not: The sample games available from Creoqode's website are simply demonstrations of its rough capabilities. You'll find no high score tables, no levels, nothing but simple animations you can control with the six face-mounted buttons, and no framework around which to build your own games.

The buttons, too, disappoint. Sturdily made from metal, the anti-vandal style buttons look absolutely stunning but are extremely difficult to use thanks to their metal collars. The four on the left, arranged to form a four-way direction control, are particularly awkward to trigger.

It's the price that finally does the 2048 in: at £200.90 delivered to a UK address, you could easily purchase all three of the other devices on test and have cash left over.



Left
While eye-catching, the Creoqode 2048 would be a terrible first introduction to programming or electronics

VERDICT

Poor design, poor documentation, and a sky-high price put the 2048 at the bottom of the league.

2/5

HEAD 2 HEAD

Tinkercad Circuits VS Fritzing

Plan circuits with **Ben Everard's** pick of beginner's software

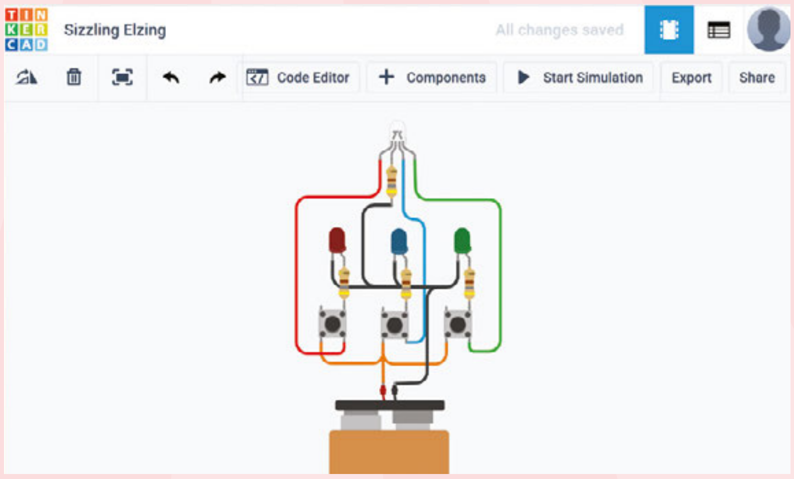
Below The simulator in Circuits allows you to push buttons and see what would happen without actually building the circuit

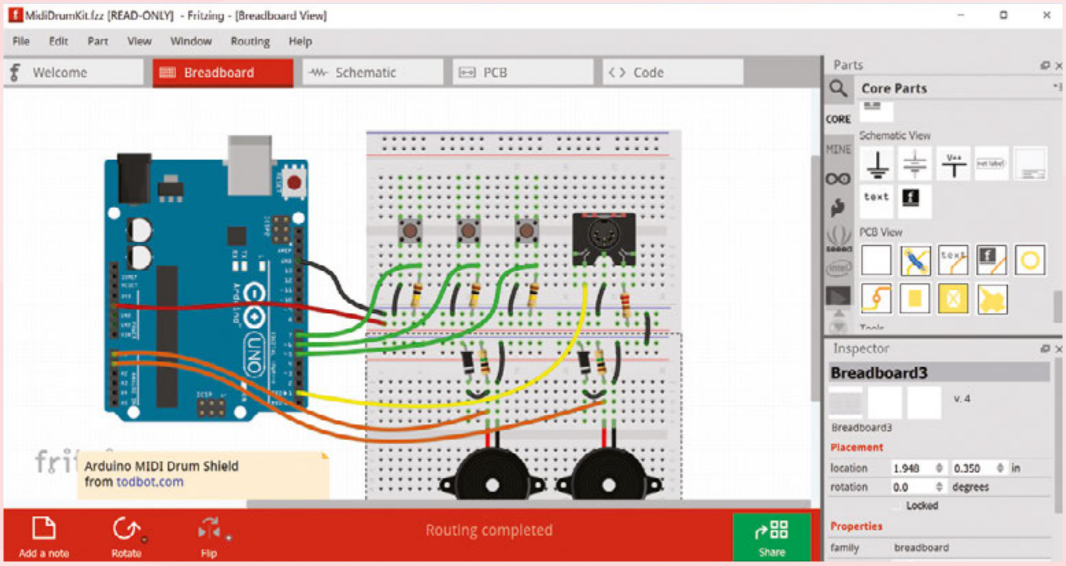
For a long time there was one bit of software leading the field in beginner's circuit design: Fritzing. However, there's now a pretender to the throne: Tinkercad Circuits (formerly circuits.io).

Tinkercad Circuits is a web-based tool that helps you design circuits by dragging and dropping components and wiring them up in different ways. Previous versions of Circuits (hosted on **circuits.io**) included the ability to edit the schematics and PCB layout of the design, but these have been removed and there's only the physical view left. If you want to take your project to PCB, you'll need to export the Eagle files and open them in that (more complex) design tool.

The main feature of Circuits is the simulation mode that allows you to see what would happen if you built the circuit and powered it up. This is complete with multimeter readings, oscilloscope outputs, and code running on programmable devices (such as Arduinos). This is impressive and, in our experience, worked well. Perhaps we're just curmudgeonly sticks-in-the-mud, but we feel that the best way to learn electronics is by doing electronics with real components. Surely it's better to build the circuit and see what happens, rather than simulate it and see what a computer tells you it should do? Still, this simulation is a quick and easy way of trying things out.

There's a reasonable range of components for basic projects, but pretty soon you're likely to find yourself searching for something that's not there, and this is perhaps the biggest let-down of Circuits.





Left Despite its simple interface, Fritzing allows you to develop some quite advanced circuits

You will probably hit the limit of what you can do with it quite quickly.

Tinkercad is far more than just Circuits, and the 3D object design tool has a wide variety of objects that other users have built and shared to help inspire you. As yet, the number of circuits shared is quite limited, but if this increases, this could be a useful resource for people learning electronics.

" The main feature of Circuits is the **simulation mode** that allows you to see what would happen if you built the circuit and powered it up "

BACK TO BASICS

Fritzing is a much more clear-cut design tool. You can take components and place them in a schematic, breadboard or PCB design, and see if these three views are showing the same circuit. You can choose from an almost endless supply of components to do this with (and many more third-party component libraries are available online). You can order PCBs directly from the application, or export Gerber files that can be fabricated by most PCB manufacturers.

The one thing you can't do is see what the circuit does – you'll have to actually build it to do that.

There is a code editor as part of Fritzing, but since there's no simulation, this is for writing code to upload to other boards. It's not very powerful and is most useful for keeping projects tidy by including the code in the same set of files as the design.

There is also a sharing platform on Fritzing, but it's a little hard to use. There's no web-based view, so you have to download a project before you can see what it looks like, and comments aren't widely used, so there's not much feedback on what is there.

With the demise of the schematic and PCB modes in Circuits, the latter becomes a much weaker proposition as Eagle is a significantly more complex tool for a beginner to use. We're also a little uncomfortable relying on a web-based tool for what is fundamentally an offline activity. Despite all these weaknesses, the simulation mode could be a boon in some circumstances, and the fact that it's available on any machine with a web browser is a plus.

However, if you're actually interested in designing and building real circuits using physical components, Fritzing is a far more capable tool, yet still accessible for beginners. □

OTHER OPTIONS

When it comes to PCB design, there are two tools that really stand out for serious users: KiCad and Eagle. KiCad is open source and free for any use. Eagle is free for personal use with some restrictions. They're both serious, professional-grade tools that are powerful enough for most uses, but that power comes with a far more complex user interface and a much steeper learning curve. We'd strongly recommend getting to grips with the basics using a simpler tool such as Fritzing before moving on to one of these.

There's a wide range of circuit simulation tools; however, almost all of them focus on schematics rather than physical layout (as Tinkercad Circuits does). Circuitlab.com is a good online option, KTechLab works well offline on Linux systems, and iCircuits works on Windows.

VERDICT

Tinkercad Circuits

A good circuit simulator, but less useful now the schematic and PCB tools have been removed.

3/5

Fritzing

The definitive beginner's tool for designing circuits.

5/5

MeArm Pi

£70 | mime.co.uk

Some say the robots are coming, if not quite to take over the world then perhaps to snaffle a few human jobs at the very least. Certainly, if you're in the market for a mechanical arm, you'll find yourself rather spoiled for choice. Whether or not you'll get one of this quality for this price is an entirely different matter.

We've been impressed by the MeArm Pi – an affordable, open-source robot arm brought to life with the aid of £56,376 worth of Kickstarter cash earlier in the year. Aimed at children aged 11 and over, it has been designed to be simple to assemble and a cinch to operate. In reality it's a tad fiddly, yet the resulting build is quite robust and there's no doubting that it's a lot of fun.

Everything you need is in the box. That's the acrylic pieces, the screws, a Raspberry Pi HAT complete with twin-joysticks, and even a small hex key for connecting many of the parts together. Okay, you need a Raspberry Pi and a power supply too, but if you're buying something with the Pi name in it, then the assumption is you'll already have these knocking around.

So what is it like? Well, the pieces initially feel a little brittle as you snap them out of their holdings, but you soon realise they're actually rather sturdy. They've been cleverly created so that they generally fit only one way, making the build more intuitive and the potential for forcing them together much less nail-biting. That said, there is a bit of second-guessing involved, mainly because the paper instructions can feel a little confusing. Fortunately, explanatory videos online help clear up any misunderstandings. Two pairs of hands come in handy, too.

Mostly, though, the build is straightforward. The servos are pre-calibrated so they only need to be slotted into place (a great step forward from previous MeArms). Meanwhile, tight moving parts can be rectified by simply loosening the screws. The fiddliest part, for us at least, was screwing the arm assembly to the centre of its base. But we liked how neatly the servo cables wrap through the parts and the end result was entirely worth the effort.

Bringing the arm to life is satisfyingly straightforward. You only need to set up the SD card with a special OS image based on Raspbian, insert it into the Pi or Pi Zero, connect the HAT, and make use of a tool called Headless Pi which lets you get going simply by plugging in the power.

It takes just over a minute to twitch into action, and the basic setup allows you to use the twin joysticks to open the claw and move the arm up, down, left, and right. More fun can be had by connecting the arm to its local web server using a computer or tablet, however. This lets you directly program it using a range of languages including Snap! and Blocks, but Python will probably be the most popular for HackSpace magazine readers.

This programmable control takes it from a toy to a hackable tool that could find its way into our projects. Alternatively, you can also get the arm without the HAT and control the servos directly using any hardware. The hackability and robustness mean it's a great choice for any time you need to pick things up and move them about.

Above The MeArm showing off its neat cable management

Above The construction is simple yet robust

VERDICT

The easiest robotic arm for hacker projects. Suitable for beginners and masters alike.

5/5

Hologram Nova

$49.99 | hologram.io

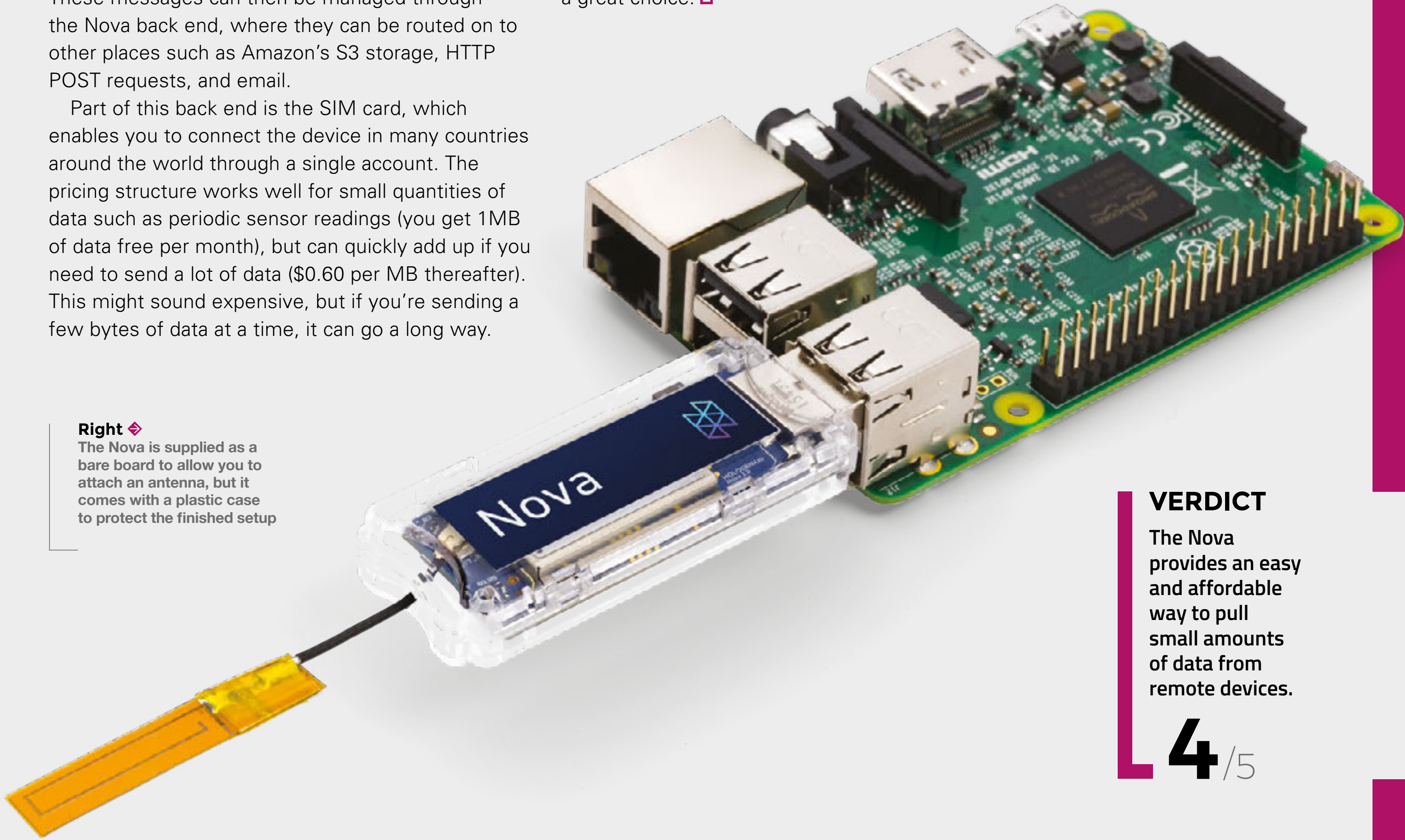
The Hologram Nova is two products combined into one. There's the hardware itself, and the back-end platform that supports data transfer to various different services. The hardware is a 2G and 3G USB modem based on the u-blox wireless module. In principle, this should work with most single-board computers, but it's designed for and tested using Raspbian on the Raspberry Pi. It also comes with a pair of external antennae that should make it easier to get a connection in a remote area (other U.FL antennae can be used as well).

The software runs via a command-line Python tool that enables you to publish messages to a topic. These messages can then be managed through the Nova back end, where they can be routed on to other places such as Amazon's S3 storage, HTTP POST requests, and email.

Part of this back end is the SIM card, which enables you to connect the device in many countries around the world through a single account. The pricing structure works well for small quantities of data such as periodic sensor readings (you get 1MB of data free per month), but can quickly add up if you need to send a lot of data ($0.60 per MB thereafter). This might sound expensive, but if you're sending a few bytes of data at a time, it can go a long way.

The end result works really well if you want to aggregate sensor data in a way that's supported by the back end. It's far less versatile than some other options for connecting to the mobile network, and this lack of versatility allows Hologram to focus on doing what it does in a simple and straightforward manner. A single command run on the terminal can send data straight into your back end with no additional setup required. What's more, you can manage multiple devices in multiple countries from a single webpage.

If you're after a general mobile web connection for a single device, the Nova is probably not the best option. However, if you're looking to pull in data from lots of devices to a single back end, then the Nova is a great choice. □



Right
The Nova is supplied as a bare board to allow you to attach an antenna, but it comes with a plastic case to protect the finished setup

VERDICT

The Nova provides an easy and affordable way to pull small amounts of data from remote devices.

4/5

Frog Board

$13.20 | tindie.com

When it comes to wireless microcontrollers, there's nothing quite as small, connected and cheap as the ESP8266 module. The most basic versions of this board – the 12E and 12F – are typically under £2, but they can be hard to use, as they come without USB connectors or pins and aren't breadboard-friendly. There are loads of boards built up around this to make it easier to connect to and use, but all of these add size and cost. The Frog Board is designed to take a simple ESP8266 module and add all the bits you need to make it easy to program. It can then be detached and used again and again.

You just need to press a 12E or F into the flexible pins and you can then program it via the USB connector and use the broken-out I/O pins. Once it's set up, you can pop it out of the Frog Board and solder it into your project. For your next project, you just need a new ESP8266 module and you can pop it into the Frog Board to program/prototype it.

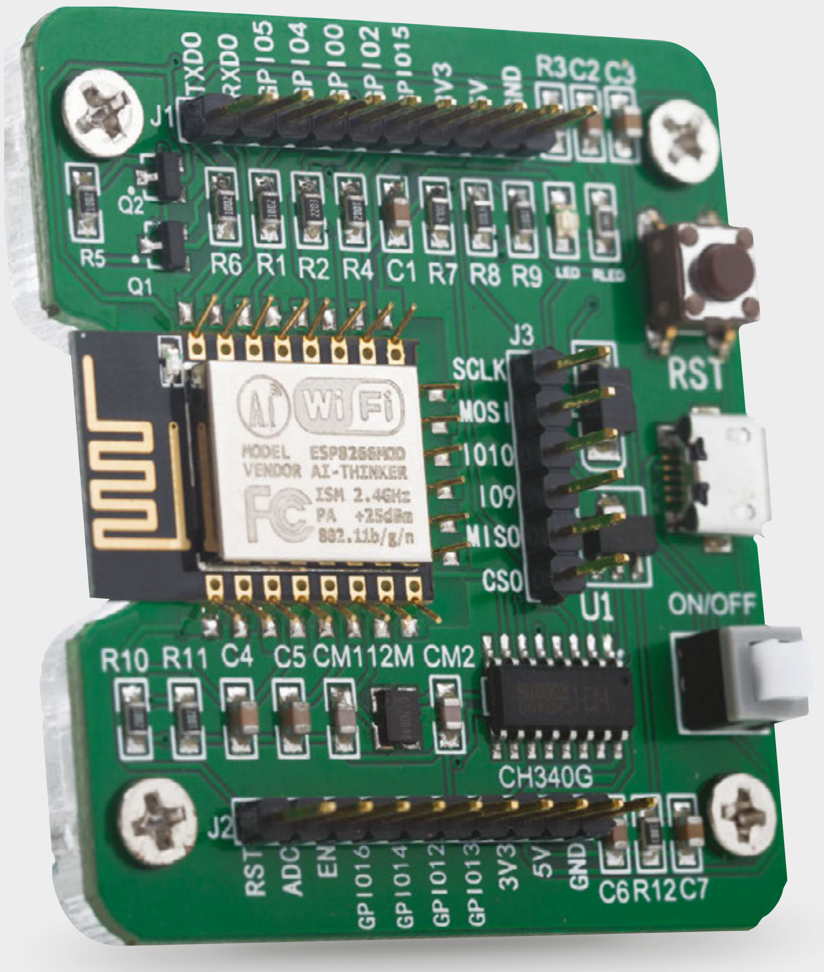
The USB connector for communication, power and pin breakouts are all quite standard. What really sets this product apart are the spring-loaded pins that both hold the module in place and create an electrical connection. It really is as simple as pushing the module in and out.

The Frog Board comes bundled with a 12E module so you can get started straight away, but it really comes into its own as a reusable tool for when you have multiple 12E modules that can all be programmed from the same Frog Board.

The Frog Board PCB is mounted on an acrylic base and feels sturdy. We've been busy popping the module in and out of the Frog Board and, while the pins probably won't stand up to abuse, they should last well if treated kindly.

Once mounted in the Frog Board, the ESP module can be programmed from the Arduino IDE (provided you download the ESP8266 addon – details are available on the Frog Board web page) or one of the other tools available for the board as easily as fully-integrated boards. ◻

Below The springy pins create a reliable but temporary connection for programming and prototyping



Right The board is mounted on a solid base for extra strength

VERDICT

The Frog Board makes it trivial to use ultra-low-cost microcontrollers in projects.

4/5

Lectrify: Very Useful Circuits

$10 | lectrify.it

While there are loads of ways of adding electronics to projects, Arduinos and Raspberry Pis to name but two, the vast majority use programmable chips to provide the functionality. These work well, but are often far more complicated than they need to be when you just need a simple LED controller.

What we love about the Very Useful Circuits boards is that they strip a feature down to the bare minimum of essential components. The Blinker, for instance, works using just two transistors, two capacitors and four resistors, while the NiteLight is a transistor, potentiometer, phototransistor and resistor wired together. There is also a touch sensor and a whetstone bridge. These circuits are simple enough that you can see what's going on. There's nothing hidden inside silicon chips or obfuscated with code. The schematics are printed on each board so you can compare what's on the board with what the circuit looks like.

The counterpoint to this simplicity is that each board does exactly one thing: the Blinker only blinks, the NightLight is only a night-light, and so on. You can't program them to do anything else.

Each project comes as a set of modules on a single board. In this setup, everything is connected via PCB traces so that it all 'just works'. However, if it's not the right shape for your project, you can snap out the modules and wire them in different orientations, for example if you want the light sensor further away from the LED or the dimmer of the two blinking lights further apart. These modules are easy to connect with crocodile leads or soldered wires. They're even compatible with LEGO, so you can connect them into your plastic brick-based projects.

Each Very Useful Circuit comes with a card telling you how to assemble the circuit (they're available both as kits and pre-built) and with a basic explanation about how everything works. We would have appreciated a little more information about how the circuits work, though. For example, the Blinker circuit is based on a multivibrator with two transistors. This is quite a common circuit, but it's not immediately obvious how it works to someone not familiar with electronics. A little more detail would go a long way to helping people understand the circuit (and to potentially expand it with their own designs).

If you have an interest in electronics, the Very Useful Circuits provide an alternative to the programmable controller route for adding some electronics to your projects. They're achievable projects for almost anyone, and easy to integrate with physical builds. □

VERDICT

A great way for beginners to incorporate electronics into their projects, but more documentation would be useful.

4/5

Below The snap-out modules make it easy to get started, but are still flexible enough for physical projects

EspoTek Labrador

$29 | espotek.com

An oscilloscope displays variations in a voltage over time, usually within a two-dimensional plot. Back in the '70s, '80s and '90s, they were huge and heavy and the two-dimensional plot was beamed onto a CRT screen, set alongside chunky knobs to dial in a suitable set of values. Size and weight changed dramatically when LCD replaced CRT, and they're changing again with computers replacing the screen, the knobs, and the logic circuitry.

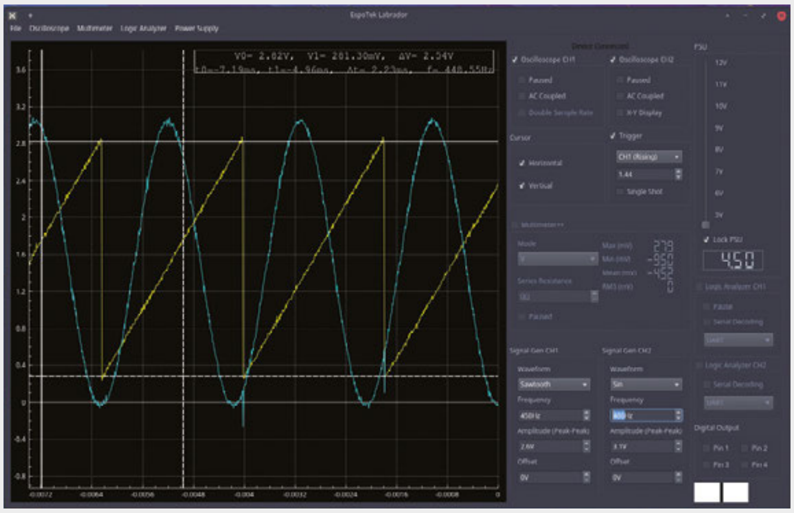
EspoTek's Labrador is one such device, only it's not just a dual-channel oscilloscope, but also a logic analyser, multimeter, power supply, and waveform generator built from a handful of surface-mount components sitting atop a 35 mm × 37 mm PCB. It's about the same size and weight as a pastry canapé, and metaphorically speaking, tastes just as good.

SIMPLE SOFTWARE

Getting started is as simple as downloading, installing and running the accompanying software, followed by connecting a micro-USB cable (included) between the PCB and your computer. The application software is available for Linux, Windows, Mac OS and Android, and like the hardware, it's 100% open source. With the PCB connected, a red LED flickers into life and the main output area within the application will update to show random noise going from the oscilloscope input channels. You can test everything is working correctly by connecting the DC output from channel 1 of the signal generator to the DC input of channel 1 of the oscilloscope. As soon as you ramp up the amplitude in the signal generator section of the software interface, the random noise will transform itself into your chosen waveform.

The application itself is easy to use and well designed, especially if you've not used an oscilloscope or made electronic measurements before. It doesn't fill the screen with too many details, and limits settings and configuration options to the most useful and common. However, it's also capable of some serious circuit and microcontroller analytics; various trigger values can be used to synchronise waveforms, serial messages can be decoded in real time from the logic analyzer, and grabbed values can be saved as a CSV text file.

Labrador's PCB is designed to be connected to a breadboard. The ten pins beneath the long header will fit nicely into the tenth column of a standard board, enabling the horizontally oriented power supply pins to connect to the negative and positive rails running the length of most boards. These rails can then be used to power your own components, from 4.5 volts to 12 volts, in 50 millivolts increments, which is brilliant for powering devices such as an Arduino. When tested with a multimeter, the output was also reasonably accurate, going from 4.67 V to 12.13 V. The long header provides convenient access to the four digital outputs (3.3 V), great for turning on LEDS, alongside the signal generator and the separate 3.3 V output. Not all of these functions can be used at once, such as the multimeter and the oscilloscope, but many can be. The only thing really missing is PCB annotations, but creating your own solution with a breadboard is perfectly in fitting with both the device and its remarkable price.



Above The accompanying software is well designed, quick, and easy to use

Below Yes, that is a Labrador dog on the back of the PCB

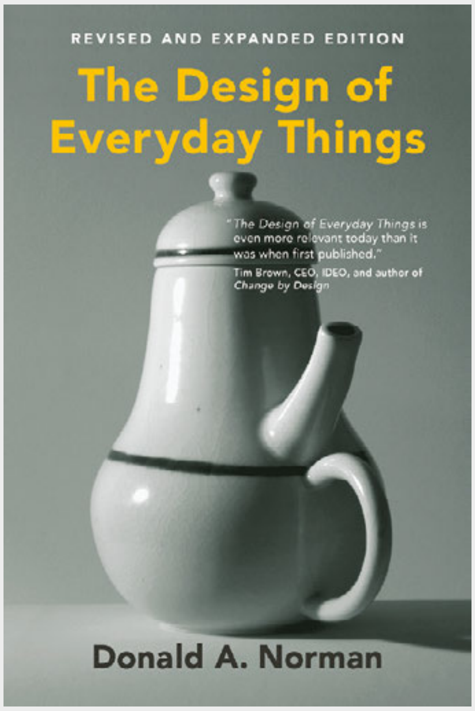
VERDICT

Tiny and inexpensive, the Labrador really is an 'electronics lab in your pocket'.

4/5

The Design of Everyday Things

£14.95 | jnd.org


REVISED AND EXPANDED EDITION
The Design of Everyday Things
"The Design of Everyday Things is even more relevant today than it was when first published."
Tim Brown, CEO, IDEO, and author of Change by Design
Donald A. Norman

In our interview with Becky Stern, she mentioned that a key skill for hackers and makers is industrial design, and we agree. As makers of things, we want it to be as easy as possible for other people to understand how to use the things we create.

In *The Design Of Everyday Things*, Don Norman looks at the various psychological factors that influence how our puny brains try to comprehend the objects around us. The more you understand how this process happens, the easier your users will find your devices. The aim is to make things instinctive to use so that people can just pick the thing up and know how to get it to do what they want. This isn't very noticeable when it works properly, but it's glaringly obvious when this process fails: people get stuck trying to pull or push a door that actually sides or a user gets frustrated pushing a button that they should be turning… It's the sort of thing that makes us feel stupid when it happens to us, but in reality is a failure of design.

BUILDING FRIENDLY MACHINES

This isn't a new book (the first edition came out in 1969, and the revised and expanded version in 2013), but that doesn't matter because although various technologies have come and gone in that time, people's understanding of physical things hasn't changed.

As well as being packed with useful information, *The Design of Everyday Things* is an interesting and enjoyable read for anyone who builds physical objects. You don't need to be a designer to understand what the problems are or to benefit from a better understanding of the subject matter. This is a book for anyone who builds things that people use.

If you're building stuff for yourself, you probably instinctively understand how to use your creations anyway. However, for anyone building things for other people to use, *The Design of Everyday Things* will help you understand the finer points of transferring information between the squishy meat brain and the engineered perfection of your device. □

"The aim is to **make things instinctive to use** so that people can just pick the thing up and **know how to get it to do what they want**"

VERDICT

The definitive book for understanding the interface between people and machines.

5/5

WE NEED YOU

Got an idea for an article? Want to write for us?

GET IN TOUCH

We want your tips, comments, and questions!

hackspace@raspberrypi.org

@HackSpaceMag

HackSpace

TECHNOLOGY IN YOUR HANDS

hsmag.cc

CanaKit™

CanaKit Raspberry Pi 3 Complete Starter Kit

SanDisk
32GB
micro SD HC
CanaKit Raspberry Pi Quick-Start Guide
CanaKit
2.5A Power Supply for Raspberry Pi

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

Raspberry Pi APPROVED RESELLER

$74.99 US DOLLARS **£59.99** EXCLUDING VAT **€64.99** EXCLUDING VAT

ELECTRONIC KITS • ELECTRONIC PARTS • RASPBERRY PI • ARDUINO