MAKE | BUILD | HACK | CREATE

HackSpace

TECHNOLOGY IN YOUR HANDS

hsmag.cc | February 2023

ROBOT ARMS ON TEST
Automatic appendages

MAKE FLIGHT

How DIY engineers are building drones, balloons, and rockets

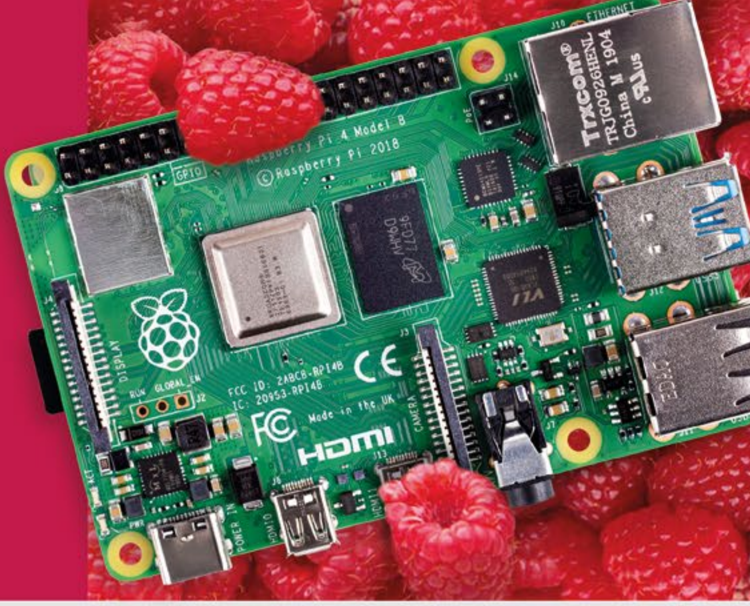
Feb. 2023
Issue #63 £6

MAKE YOUR OWN BINARY CLOCK
1s and 0s with MicroPython

DIY TROMBONE CONTROLLER
paaaarp your way to victory

FELTING **SOLDERING** SNAILS **3D PRINTING**

American **Raspberry Pi** Shop



- Displays
- HATs
- Sensors
- Cases
- Arcade
- Swag
- Project Kits
- Cameras
- Power Options
- Add-on Boards
- Cables and Connectors
- GPIO and Prototyping

Partner and official reseller for top Pi brands:

adafruit sparkfun ELECTRONICS PIMORONI HiFiBerry and many others!

Price, service, design, and logistics support for **VOLUME PROJECTS**

USA PiShop.us

Canada PiShop.ca

Raspberry Pi APPROVED RESELLER

Welcome to HackSpace magazine

Taking flight seems to be one of the universal dreams of people. To soar in the sky like the birds. For millennia, this dream had to remain just that, but with modern tools and techniques, we can actually fly. In the last couple of decades, flying has become so commonplace that we sometimes forget just how magical it is that it's possible.

Building a flying machine doesn't have to be complicated. Generations of schoolchildren have perfected the art of folding paper so that it flies. The board that distracts the teacher while they do this might have changed – from blackboard to whiteboard to interactive screen – but the folded paper remains the same.

Of course, just because it doesn't have to be complicated, that doesn't mean it can't be. This is one area where you can let your imagination soar [pun very much intended]. So, grab your toolkit, and let's make some things fly. Let's turn our dreams of flying into reality, though perhaps we'll fly by electromechanical proxy. If, like Icarus, we fly too close to the sun, at least for us it will only be some electronics that fall to the ground.

BEN EVERARD
Editor ben.everard@raspberrypi.com

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

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

GET IN TOUCH

hackspace@raspberrypi.com
hackspacemag
hackspacemag

ONLINE

hsmag.cc

Available on the App Store

GET IT ON Google Play

PAGE 30
FREE PICO W WHEN YOU SUBSCRIBE

EDITORIAL

Editor
Ben Everard
ben.everard@raspberrypi.com

Features Editor
Andrew Gregory
andrew.gregory@raspberrypi.com

Sub-Editors
David Higgs, Nicola King

DESIGN

Critical Media
criticalmedia.co.uk

Head of Design
Lee Allen

Designers
Sam Ribbits, Olivia Mitchell

Photography
Brian O'Halloran

CONTRIBUTORS

Stuart Moore, Marc de Vinck, Andrew Lewis, Rob Miles, Phil King, Nicola King, Jo Hinchliffe

PUBLISHING

Publishing Director
Russell Barnes
russell@raspberrypi.com

Advertising
Charlie Milligan
charlotte.milligan@raspberrypi.com

DISTRIBUTION

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

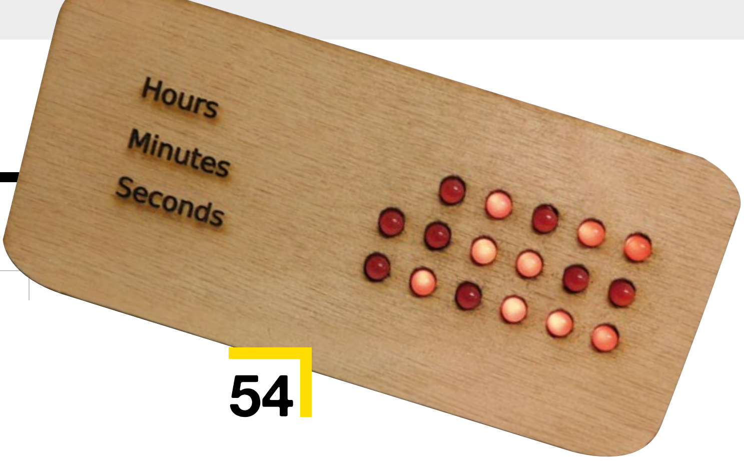
SUBSCRIPTIONS

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

To subscribe
01293 312189
hsmag.cc/subscribe

Subscription queries
hackspace@subscriptionhelpline.co.uk

Contents



54

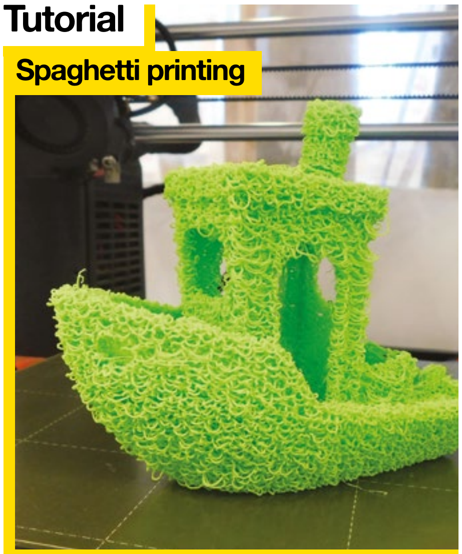
Cover Feature

DIY FLYING MACHINES
Take to the skies with homemade drones, rockets, balloons, and more

22

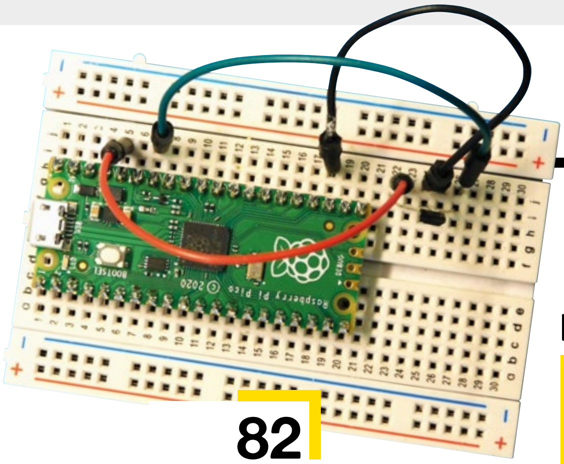
Tutorial

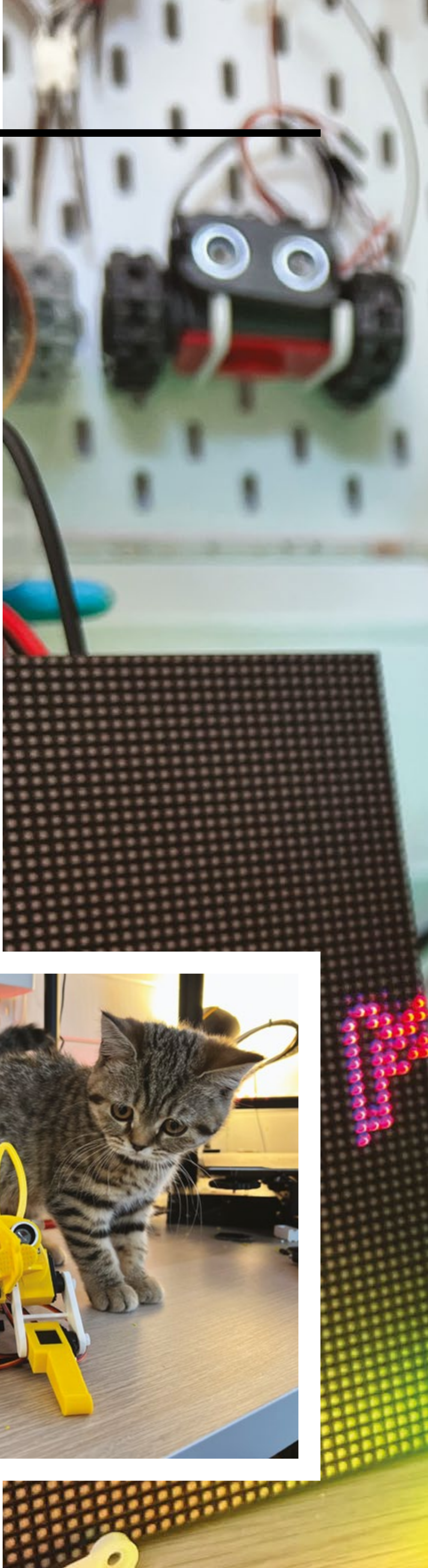
Spaghetti printing



52 How can something so wrong feel (and look) so pleasantly right?

96

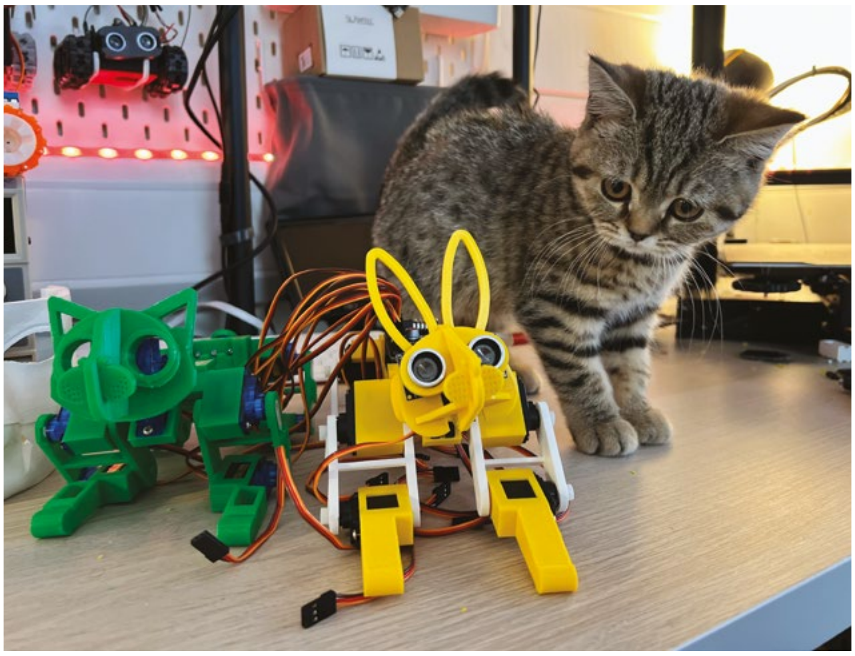




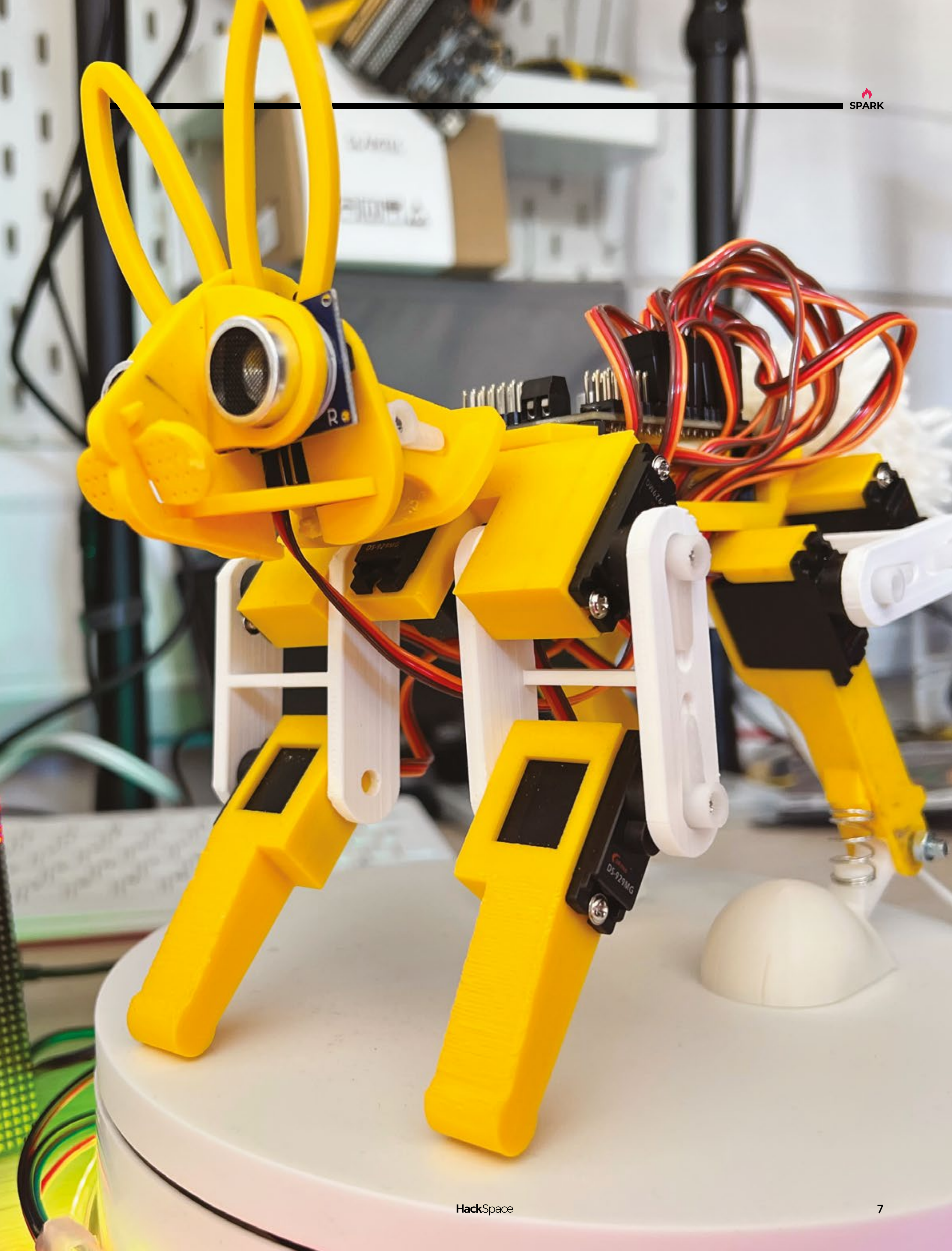
Bugs the Robo-Bunny

By **Kevin McAleer** kevsrobots.com

We're used to the concept of open-source hardware, but it's very often more seen in theory than in practice. This Robo-Bunny, by Kevin McAleer, is based on PicoCat v2, which was itself based on OpenCat. It's had a new head, new legs, and a fluffy tail added, and it's controlled by the Pimoroni Servo 2040, which powers up to 18 servos. Turn to our interview with Kevin on page 38 for more on **kevsrobots.com** and the brilliant things he's doing. □



Right Robo-Bunny, seen here with PicoCat, and a flesh-and-blood cat (Tigger)



PI-FAN

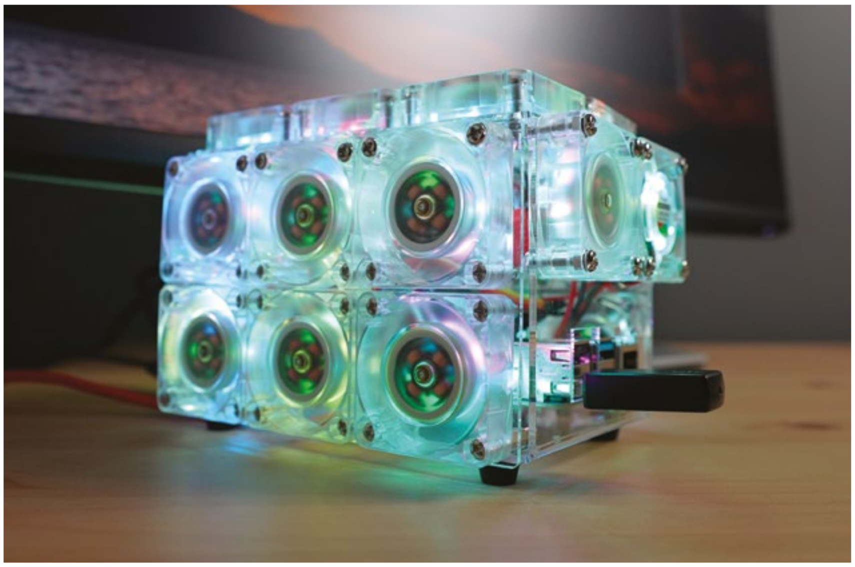
Only Fans Case

By Michael Klements

hsmag.cc/OnlyFansCase

"Every time I've made a new case for my Raspberry Pi, there are always a few comments suggesting I add another fan or make improvements to the cooling." So says Michael Klements in the introduction to his latest build video. Michael's taken this idea and ran with it, creating a case with every surface (except the bottom – there's no point trying to blow air into your desktop) covered in a total of 24 5V fans. The case itself is made of two pieces of laser-cut acrylic, bent into two U-shaped pieces and connected with brass inserts.

It works, yielding a noticeable but undramatic 4 degrees cooling compared with one of Michael's single-fan designs. More importantly, it's very, very silly.



Left
This many fans consume more power that the Raspberry Pi can spare, so Michael's had to add an external 2.5A, 5V power supply

REGULAR

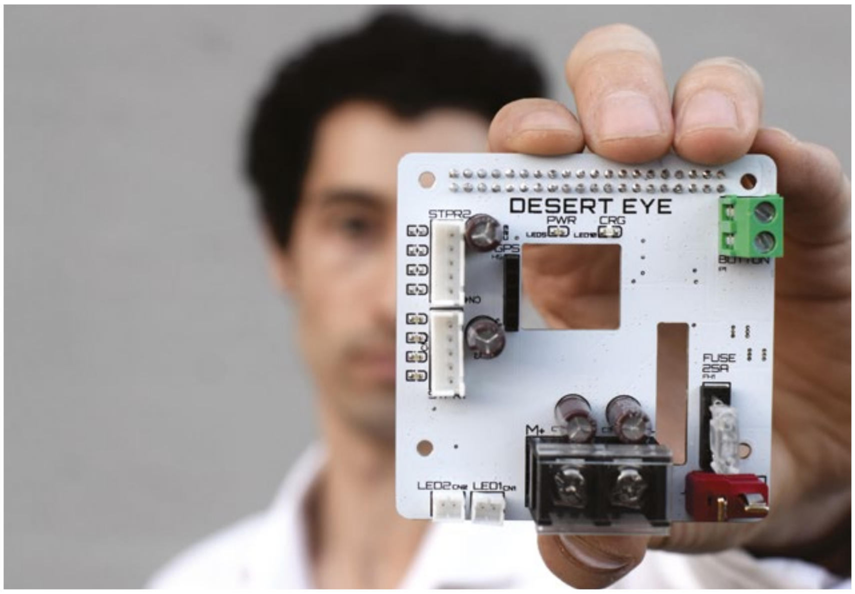
Tracked robot

By **Aviv Butvinik**

hsmag.cc/DesertEye

If you like awesome stuff, you'll love this: it's a tracked robot, controlled from anywhere via the internet, featuring a camera, built on a Raspberry Pi and a custom PCB. It's also 3D-printed, and designed from scratch by maker Aviv Butvinik.

This is the second version of his Desert Eye robot; the first iteration used bicycle chains connected with 3D-printed parts as the tracks. This proved fiddly and time-consuming to assemble, so Aviv switched to using belts printed in one piece out of flexible filament. If you're interested in high-performance 3D-printed robots, check out his videos.



Right
This design is truly a masterclass in Fusion 360

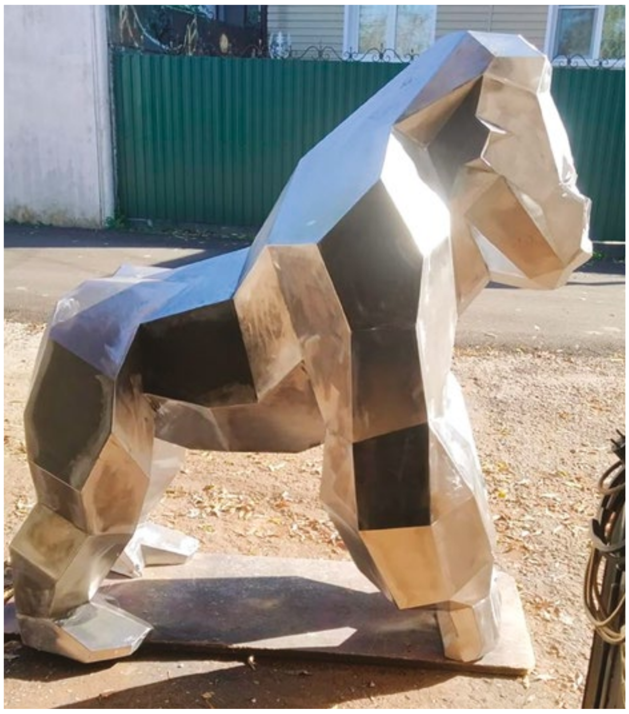
REGULAR

Low-poly Gorilla

By **Gleb Taburkin** Instagram.com/ta6urkin

Squint at this sculpture and you'd be forgiven for thinking it's come off the bed of a 3D printer… not a bit of it. This low-poly gorilla sculpture is actually a big old chunk of metal – over 70kg of sheet metal to be precise, welded together by Vladimir Kuznets, implementing a design by sculptor and automotive engineer Gleb Taburkin.

Sheet metal is a different beast to hot plastic, so even though the design looks similar to the sort of design you'd send to a printer, the process is completely different. It takes Gleb about a week at the computer to create a design, including the instructions for the welder – you can read his process here: **hsmag.cc/DesigningForSheetMetal**.


Right "Kuznets" is Russian for Smith, or one who makes things in metal. We enjoyed very much this example of nominative determinism

REGULAR

Time circuits clock

By Stephen Holdaway

hackaday.io/stecman

We suspect most readers will recognise this: it's the middle section of the DeLorean's time circuits clock, from the film *Back to the Future*. The internet is awash with implementations of this movie prop, but Stephen's version is so lovely that we chose it to grace these pages. By stripping it down to only one section, he's made it seen more accessible to anyone out there who wants to have a go themselves, and by pointlessly but brilliantly adorning the back of the PCB with the DeLorean's 'OUTATIME' number plate, he's added a touch of love. ◻

Right The year only changes once a year; the rest of the time, it's useless, so Stephen plans to make it display something useful, like "temperature, humidity, or a badly rendered version of the game Snake"

YEAR
AM
PM
HOUR
MIN
1985
01
22

Objet 3d'art

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

The object shown here is a world first: a print-in-place articulated model, printed in sweet, sweet chocolate.

It's the work of Ellie Weinstein, creator of the Cocoa Press 3D Chocolate Printer, and the file (a charming goldfish named Pepito) was cleverly designed by MakeitMichael (**hsmag.cc/ArticulatedGoldfish**).

We spoke to Ellie a few months ago while she was working on the current iteration of her printer; the clever thing about it is that it's food-safe, and it gives the user extremely fine control over the temperature of the chocolate. We're amazed at this print, and apparently it tasted good too. □

Letters

ATTENTION ALL MAKERS!

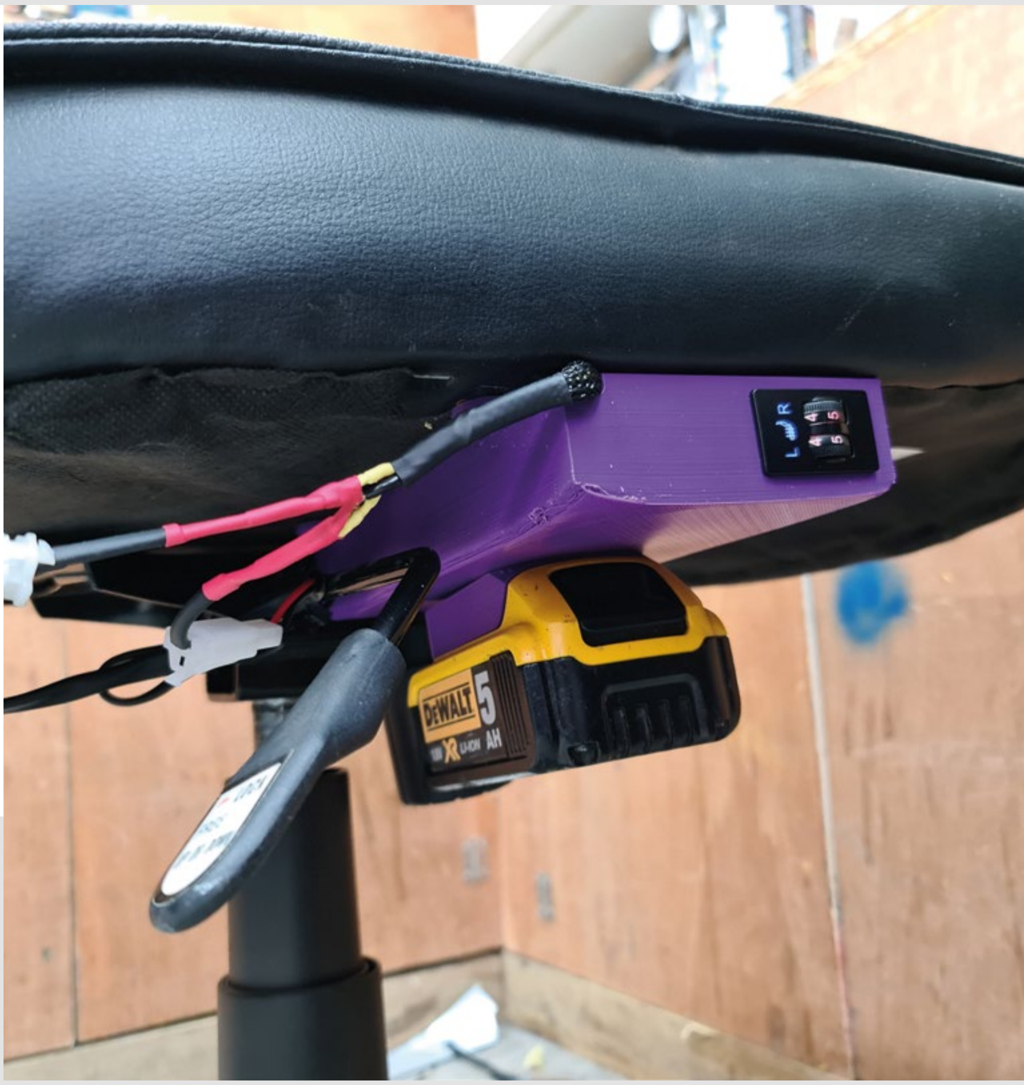
If you have something you'd like to get off your chest (or even throw a word of praise in our direction), let us know at **hsmag.cc/hello**

STAY WARM

Your heated seat came at just the right time – I didn't think I needed to make my workshop any warmer, but then the polar vortex forgot where it was supposed to be, and I found I had to put through-hole components into PCBs with fingers that would hardly move. Unfortunately, it's warmed up now and I can move around fine, so I'm going to forget all about it until the next time, when, alas, it will be too late.

Paul
Gateshead

Ben says: Hindsight is a wonderful thing, but even better than that is foresight. You know it's going to get cold at some point, so do yourself a favour and winter-proof your workspace now. Heating your seat rather than the whole room is going to save you a ton of money, so if nothing else, you should do it to spite the gas companies.



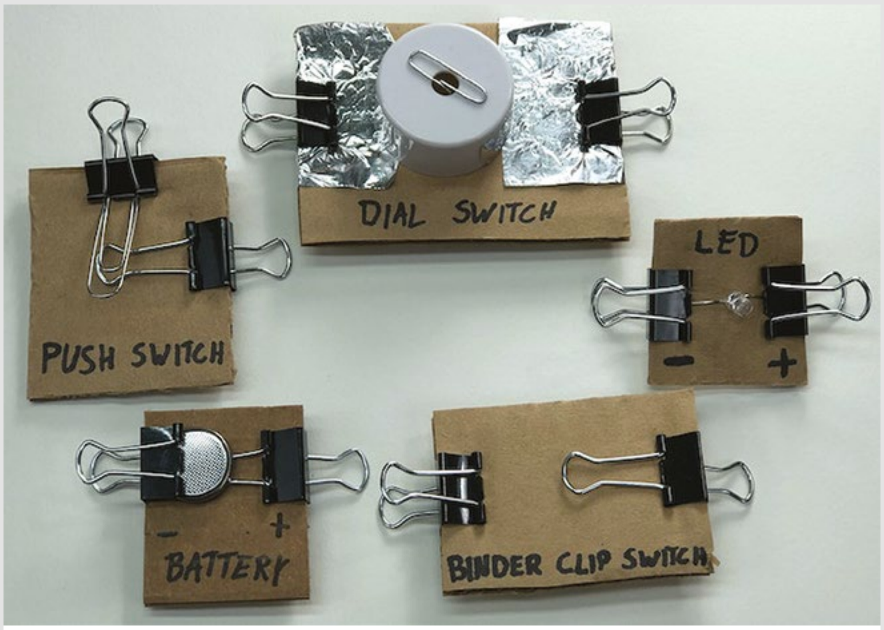
PI PROJECTS

Last issue, you put Raspberry Pi Projects on the cover – what's the point when you can't get hold of the things? I've got things I want to do: where are the units?!

John
Birmingham

Ben says: It turns out that our look at Raspberry Pi builds last issue was timely, as it looks like the flow of Raspberry Pis into the market is going to return to normal in 2023, starting with an extra 100,000 units set aside for individual consumers (namely the Zero W, 3A+, and the 2GB and 4GB variants of Raspberry Pi 4).

So keep an eye on rpilocator.com, get your bill of materials together, and you'll be building with Raspberry Pi again faster than you think.



(ALMOST) FREE FUN

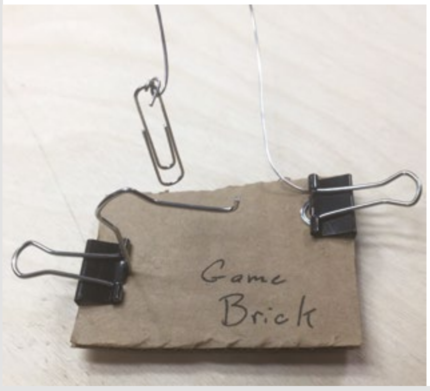
I remember a few years ago looking at the so-called innovation in tech education. An awful lot of it was augmented reality, where you'd have to hold up an iPad in front of a textbook to get additional content, such as videos. It kind of worked, but I'm not sure how many schools have the budget to buy books that don't work properly without a £500 computer.

Compare and contrast to what you can achieve with a bit of cardboard, some bulldog clips, batteries, and LEDs. Michael Carroll's *Scrappy Circuits* is exactly what teachers want: cheap, useful, cheap, accessible and above all, cheap. Hats off to him!

Jane
London

Ben says: Moore's law, the unstoppable force that makes computing power cheaper and better all the time, doesn't always translate into making new things more affordable. But one of the best things about being makers is that we can find ways around obstacles, share ideas, and build things that don't break the bank.

***Scrappy Circuits* is a perfect example of this. Mike's taken an idea and codified it so that teachers and students can replicate it anywhere they have access to a dollar store and, in doing so, he's guaranteed that it'll provide more value than iPads and AR textbooks ever will. It's brilliant, and we salute him.**



INSPIRATION

STARTS HERE

From millions of in-stock parts to cutting-edge technical resources—we've got everything you need to turn inspiration into innovation.

Get inspired at digikey.co.uk or call 0800 587 0991.

Digi-Key ELECTRONICS

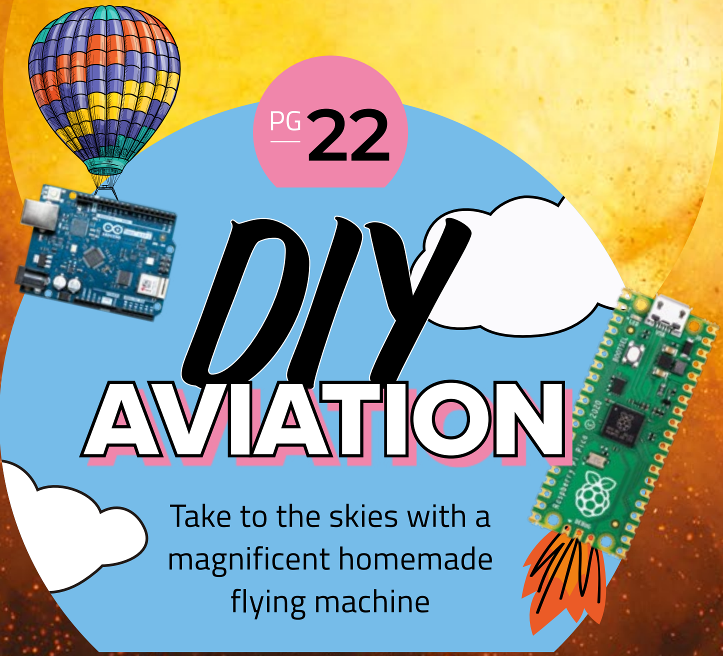
ECIA MEMBER
Supporting The Authorized Channel

HackSpace
TECHNOLOGY IN YOUR HANDS

LENS

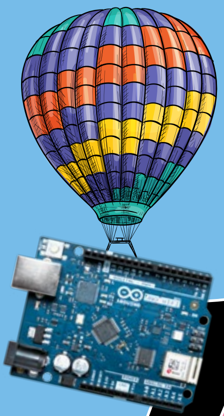
HACK | MAKE | BUILD | CREATE

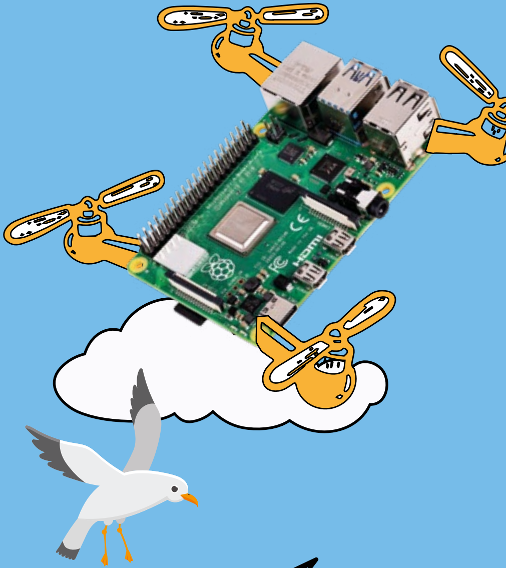
Uncover the technology that's powering the future



FEATURE

MAGNIFICENT FLYING MACHINES

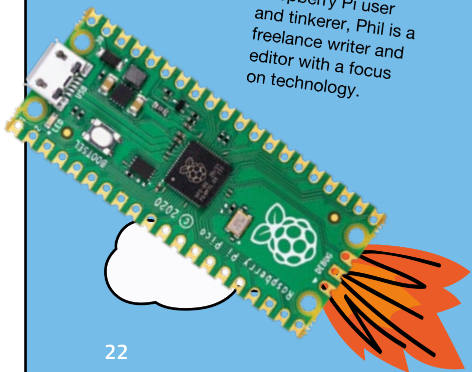




Up, down, flying around, looping the loop and defying the ground...

Phil King

A long-time Raspberry Pi user and tinkerer, Phil is a freelance writer and editor with a focus on technology.



There's no fear of flying here: we're showcasing some of the best airborne creations we've found floating around the makersphere. We start off down the runway with some incredible aircraft, including an insane model VTOL jet and a flapping winged dinosaur.

Taking off, in vertical fashion, we explore the world of drones and copters, from a tiny Pico-based flight system to a wildfire-spotting drone, a spinning UFO, and a crazy contraption that looks more like a combine harvester, but somehow flies!

Next, we soar skywards in more graceful fashion with some buoyant balloons and dirigibles, including a remote-control hot air balloon, a robotic airship that can fly autonomously, and a rigid-frame Zeppelin-style craft.

Finally, we boost ourselves even higher with a range of rockets. There's even one that can land safely back on terra firma, SpaceX style.

With such a wide variety of airborne projects that can be made, some of them using household items and materials, it's hard not to feel inspired to get one in the air.

RC VTOL F-35 PARKJET

hsmag.cc/F35Parkjet

This is one versatile remote-control flying machine. Just like its full-scale inspiration, it can zoom through the air at breathtaking speed, but also hover mid-air and perform a perfect vertical take-off and landing (VTOL). It was created by prolific drone maker Nick Rehm, who tells us it's really just a tricopter hidden inside a foam F-35 body, with some clever motor mounts on the ailerons. At its heart is a Teensy microcontroller running Nick's custom dRehmFlight software (see box overleaf) which manages the aircraft's active stabilisation and VTOL capability.

"The biggest challenge with any VTOL [aircraft] is the transition between hover and forward flight and ensuring that your flight stabilisation code seamlessly fades between these two very different flight modes," explains Nick. "For example, in hover, differential thrust provides roll control, but in forward flight the same differential thrust gives yaw control. Managing the changing dynamics of the vehicle through transition is the key."

In hover mode it flies like an acrobatic racing drone, while in forward flight it's like a high-performance aerobatic RC airplane. Nick says neither mode is easy to fly for a beginner, but a whole lot of fun for more experienced RC pilots with experience flying both planes and helicopters. "I've publicised build plans, parts list, flight control code, and tutorial videos online so others can build one as well. I can't be the only person having this much fun."

ARDUINO GLIDER

hsmag.cc/ArduinoGlider

An aircraft doesn't always need to have motors, as this Arduino-controlled glider proves. It was designed and built by Rich, aka Thumblegudget, who believes that "as humans, we learn by doing." With no prior knowledge of electronics, he was impressed by the variety of sensors that could be attached to an Arduino. The inspiration for this project was a mini IMU breakout with a gyroscope, accelerometer, and magnetometer to monitor its orientation, including pitch, roll, and direction.

Rich mounted the Arduino and IMU breakout in an old model glider as a "primitive autopilot" to control servos attached to its ailerons and flaps; this enables it to achieve the ideal pitch angle for gliding and follow a programmed heading.

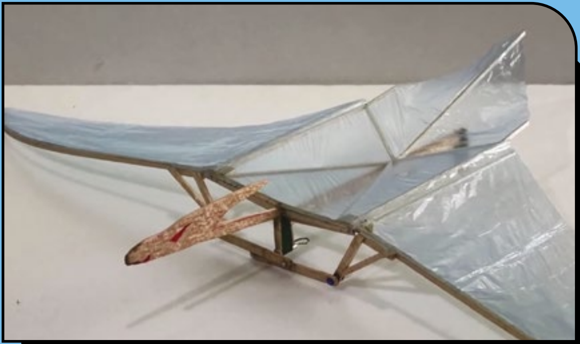
The only way to test the glider out was to take it outdoors and launch it from a local hillside. As you'll see in the video, the first few attempts didn't go so well, as it needed to reach a decent launch speed to fly reliably, but he eventually achieved a successful flight… before setting the heading for hillside and crashing it – one of the perils of making flying machines.

PTEROTHOPTER

hsmag.cc/Pterothopter

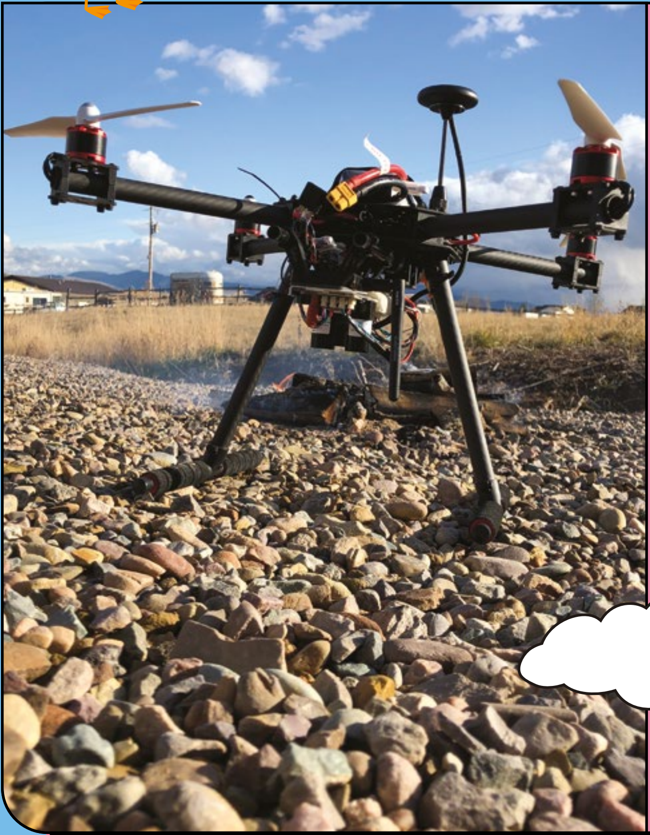
Some of the earliest attempts at human flight were ornithopters based on the flapping wing action of birds – Leonardo da Vinci even sketched a design for one. While full-size ornithopters were notable failures, smaller-scale models have latterly been made to fly.

Powered simply by a twisted rubber band, Hobi Cerdas's ornithopter is a pterodactyl based on a design by Colin Taylor. It's constructed from lightweight balsa wood and thin plastic with ice lolly sticks used for the wing-flapping control rods. →



DRONES & COPTERS

Powered by spinning rotors, these flying machines are a whirlwind of fun



HOT SPOTTER

hsmag.cc/HotSpotter

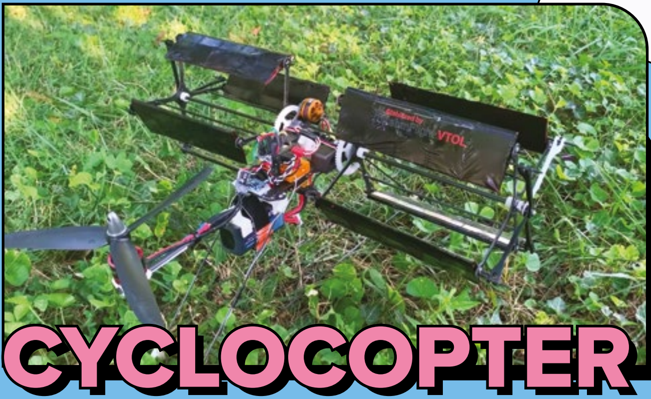
Drones are fun to fly, but they also have their uses. Based on off-the-shelf hardware controlled by a Raspberry Pi 3A+, Jason McDonald's autonomous drone can detect and report smouldering wildfires.

"I wanted to create something practical with the [NXP HoverGames drone] kit that filled a niche not already being targeted by the commercial drone industry, and/or lower the cost of a current application," he says.

Equipped with a thermal sensor, GPS, and a lidar system to measure the distance to the ground, the Hot Spotter can survey an area to create a heat map. "As the drone flies over an area, it records average temperatures of spots on the ground," explains Jason. During a ten-minute flight, the drone can survey a 1600m^2 area twice.

Once the heat map is generated, all the readings for a given point are averaged together. "This method helps to localise heat sources and distinguish between large warm regions versus hot spots," says Jason.

While disaster struck when the drone crash-landed due to a malfunctioning rotor, Jason has rebuilt the drone to be "better… stronger… faster."



CYCLOCOPTER

hsmag.cc/Cyclocopter

Traditional drones and copters have vertically mounted propellers to produce upward thrust, but this is something different altogether! Two multi-bladed cycloidal rotors spin on a horizontal axis to generate lift. It's the creation of Nick Rehm, who has been designing "crazy flying contraptions" for many years and first experimented with cyclocopters in the lab at the University of Maryland.

"The blades need to be very lightweight to reduce centrifugal forces on them as they spin," he explains, "but they also need to be incredibly stiff to prevent chord-wise and span-wise bending." He therefore made them from carbon fibre ribs and rods wrapped in monokoat film. The pitch of the blades is adjusted as they spin, to generate net upward thrust. A vertical propeller on the nose is used to control pitch.

Upon returning to the cyclocopter concept with his dRehmFlight flight controller firmware, Nick was "pleasantly surprised to see it flying so well" so decided to make a video about it. After attempting a low-altitude flip, he crashed and damaged it, but it has since been repaired and is flying again.

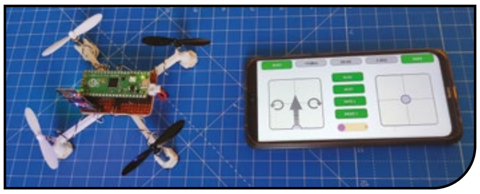
PIWINGS

hsmag.cc/PiWings

Drones come in all sizes and, by using a tiny Raspberry Pi Pico microcontroller, educator and maker Ravi Butani was able to create a miniature, low-cost flying machine system designed for STEM educational purposes.

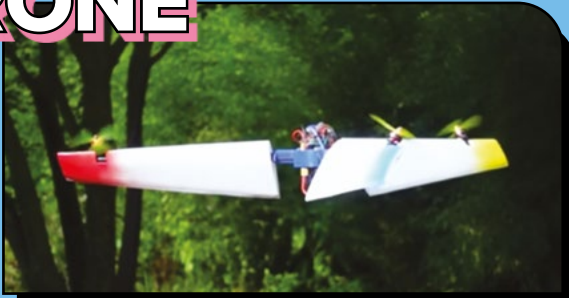
A standard Pico board is mounted on a custom-designed PiWings PCB which can drive up to six DC motors and four servos. It also features a six-axis IMU for auto-levelling purposes, while an ESP-01 module provides Wi-Fi connectivity.

The system seems very versatile and Ravi has already demonstrated using it with tricopter, quadcopter, and hexacopter configurations, all of which seem remarkably stable in flight when controlled via Wi-Fi from an Android smartphone app – or, for longer range, an RC radio controller. Ravi says the PiWings platform could also be used for fixed-wing aircraft and even ground- or water-based vehicles. He is currently designing the final version of the board for launch and also plans to make everything open-source.



CYCLONE-ROTOR UFO DRONE

hsmag.cc/UFODrone



While Nick Rehm's Teensy-based drone resembles a standard tricopter at first glance, its rotors can be tilted to make the whole drone spin in the air; with the addition of tilting blades on the arms, this can generate lift with far greater power efficiency.

"I've always loved the spinning maple-seed drones," Nick tells us, "but didn't quite know how to implement a control strategy to actually make them controllable from a fixed frame of reference." While building his experimental UFO drone, however, he soon realised that it could also be made to fly forwards like a fixed-wing plane, by pitching the blades over 90 degrees and flying on the symmetrical blades/wings. "I was inspired by the SUTD AIR Lab's THOR design, which is a similar take on the spinning wing VTOL drone design." Nick says there are plenty of applications that could benefit from the increased flight time and payload of this versatile design. "You could even use the spinning nature and a lidar to map the environment as you fly" and maintain a set altitude, as he shows in one of his videos. →

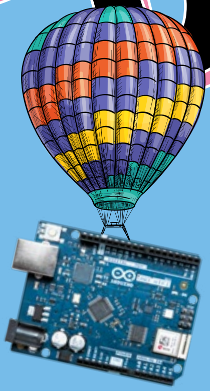
dRehmFlight

While working on a Raspberry Pi autonomous quadcopter project for a student competition, Nick Rehm was frustrated by the time needed to learn the ArduPilot software architecture to make the changes needed, so he decided to write his own flight control code on an Arduino.

After using it for his prototype VTOL F-35 Parkjet project and posting a video of it, he found that lots of people wanted to get their hands on his code. "So I cleaned it up, gave it a name, and spent 3–4 weeks writing comprehensive documentation and filming supporting tutorial videos." dRehmFlight was born (**hsmag.cc/dRehmFlight**) and has since been used by many people around the world to get their flying machines in the air: "airplanes, helicopters, tiltrotors, tailsitters, ornithopters, you name it."

BALLOONS & AIRSHIPS

Soar skyward gracefully under the power of hot air or gas



RC HOT AIR BALLOON

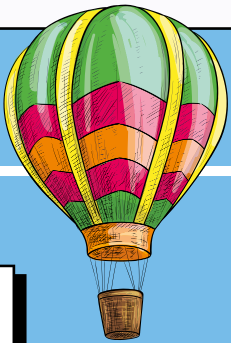
hsmag.cc/RCHotAirBalloon

A collaboration between Matt Barr and real-life hot air balloon pilot Josh Thurman, this remote-controlled, propane-fuelled hot air balloon can reach heights of 100 ft (30m) or more – although a tether line is used to prevent it floating away in the wind.

"We thought this balloon might have a practical application for balloon demonstrations (local schools, for example)," says Matt, "Setup of a normal balloon requires a team of people and fuel cost is an issue. One person can set up and fly our RC balloon. It's large enough that it draws lots of attention when it is in the air and is still able to demonstrate adequately the physics of balloon flight."

Matt opted to use two Arduino Uno boards for the project: one located in the transmitter box and another in the receiver box on board the balloon's basket. Both are connected to nRF24L01+ wireless modules to communicate at up to 800m.

Controlling the balloon is fairly simple: turning a potentiometer on the transmitter results in the receiver rotating a servo to pull the handle of a weed burner in the balloon basket. Naturally, this heats the air in the balloon to make it rise. Matt has also programmed a failsafe to turn off the burner if the wireless signal is lost, to prevent fly-away if the balloon were to become untethered.



ROBOTIC AIRSHIP

hsmag.cc/RoboticAirship

Fancy making your own low-cost airship using a Raspberry Pi for control? Then take a look at this open-source project from the New Dexterity research group at the University of Auckland in New Zealand. All of the instructions, CAD files, and software are available on GitHub: **hsmag.cc/RoboticAirshipGH**. Designed for robotics education and research purposes, it offers extended flight times and should only cost around $100 to build.

The airship's 3D-printed gondola is built around a Raspberry Pi Zero W, which forms the central control and communication unit for the aircraft. Three DC motors and propellers are used: one for lift and the others for thrust and directional movement. Power is supplied by a 500mAh Li-ion battery.

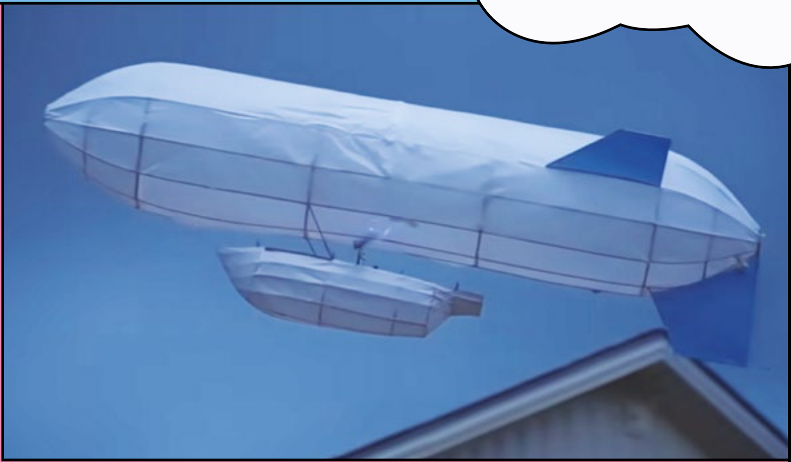
Once assembled, the gondola is attached by Velcro to a 91 cm helium-filled Microfoil party balloon – although the team have also experimented with other balloon types such as latex and bubble. According to the makers, using a Raspberry Pi makes this airship more flexible than a comparable kit such as the Arduino-based Blimpduino. The addition of a Pi Camera Module enables the airship to follow waypoints (marked on the floor) autonomously and could also provide a video feed. In addition, the airship can be controlled manually via Wi-Fi and is compatible with the ROS robotics platform.

Up, up, and away

Jo Hinchliffe detailed how he made a hot air balloon in HackSpace issue 61 using household items such as tissue paper, thread, thin wire, and a bin liner. Hot air was provided by a heat gun. You can download the PDF plans from the issue page: **hsmag.cc/issue61**.

RC RIGID AIRSHIP

hsmag.cc/RigidAirship



When most people think of rigid airships, they can't help but recall the notorious Hindenburg disaster of 1937; caught on film newsreel shown around the world, the tragic fire single-handedly ended the golden age of ocean-crossing passenger airships. So perhaps it's not a good idea to fill your model airship with highly flammable hydrogen – unless, like Peter Sripol, you plan to destroy it on purpose with a firework.

Prolific maker and YouTuber Peter is best known for his flying contraptions, including a full-size homemade electric aeroplane. Fortunately, given its eventual fiery fate, this smaller-scale rigid airship is remote-controlled and unmanned.

The first step was to build a rigid ellipsoidal frame from a series of spoked wheel-shaped pieces of balsa wood linked together with narrow struts. The key, of course, is to make it as lightweight as possible. He then coated the frame with tissue paper; since it wasn't airtight, a separate gas balloon was needed to go inside it, so Peter custom-made one from thin plastic drop cloth – lighter than foil.

Some larger propellers were added to the gondola to provide extra lift. Even that wasn't enough, though, so Peter opted to fill the gas balloon with hydrogen. Finally, with some calm winds, he managed to get it flying fairly smoothly. As it was getting beaten up by then, however, and thinking he could build a better version, he decided to set it alight mid-air. If you're planning on building one, we strongly advise filling it with helium instead! →

HOMEMADE ROCKETS

Reach new heights with a DIY rocketry project

UNITY

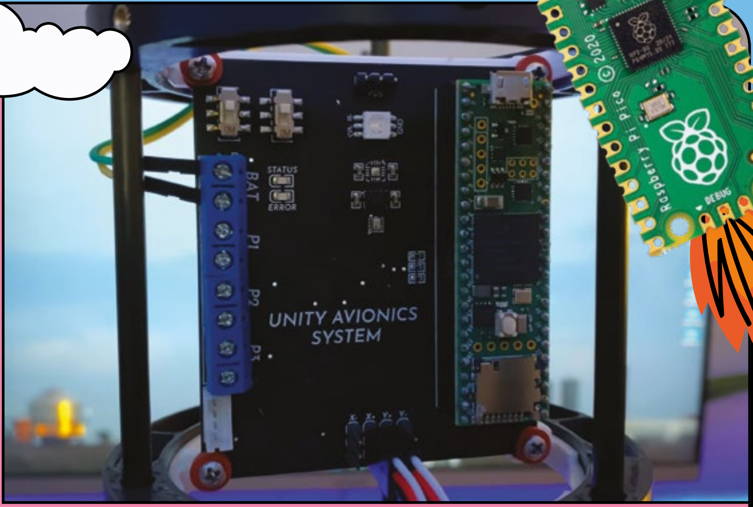
hsmag.cc/UnityRocket

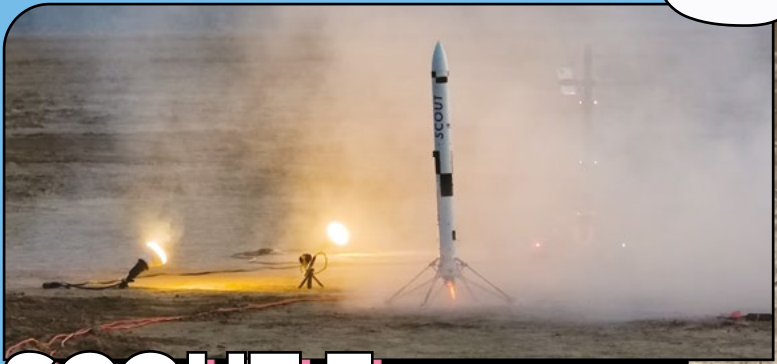
A rocket maker keen to make use of thrust vector control (TVC), Cole Purtzer heads up Delta Space Systems. Improving on his previous Frontier rocket, the one-metre-tall Unity is constructed from carbon fibre. It stabilises itself in flight with the use of a servo-actuated TVC mount – made with machined shoulder screws pivoting on ball bearings, this has a range of ±12°.

To ensure a gentle return to earth, a parachute is automatically deployed by a servo-driven mechanical ejection system – which can also be tested on the ground, pre-launch.

In addition, Unity features a new avionics unit that incorporates a high-res gyroscope and accelerometer package, along with a BMP280 barometer to determine vehicle state. A 4K camera can be mounted on the back of the PCB for in-flight footage.

After several successful test flights, Cole has moved up to a larger G25 rocket motor to reach higher altitudes – he's now aiming for 250 metres.





SCOUT F

hsmag.cc/ScoutF

When Joe Barnard set out to build a rocket that could land safely back on terra firma, SpaceX style, little did he know that it would take him seven years. "I wouldn't have done it if I knew it'd take so long," he admits, "but I'm glad I did it anyway. There's this weird benefit to being naive about a daunting project because the thought that you're 'almost there' keeps you motivated when things aren't working."

Shunning the fins found on many a model rocket, Scout F is built around the concept of thrust vector control (TVC). "Landing a rocket requires some type of attitude control at slow speed and fins only work when the rocket is moving fast. Right when the rocket is about to touch down you need a way to keep it pointed upright, so thrust vector control seemed like a straightforward way to solve that problem."

To this end, Joe created a thrust vectoring mount from machined aluminium to fit around the rocket motor, rotated by servos to control the direction of thrust. He also ended up designing a custom flight controller board, Signal R2 (available from his **BPS.Space** website).

While the project involved a bunch of elements, Joe tells us the most challenging part was the software. "Determining the rocket's orientation on the ground and in flight is a rabbit hole of problems, knowing exactly how high and how fast the rocket is moving is way more complex than just having good sensors, and all of the logic related to when the motor needs to light, how to throttle the motor [using a pair of ceramic pincers], and how to move through different flight phases is really tricky."

Reach for the stars

After finally landing a rocket, Joe's next big goal is to soar a whole lot higher than Scout F's 30-metre altitude: he aims to build a 'space shot' rocket over the next few years. "Going above 100 km is a goal for lots of amateur rocketeers as it puts you above the Kármán line, which is the internationally recognised boundary of space. It's sorta the final boss of amateur rocketry. It's a major challenge for lots of reasons, but some of the big issues are the speeds the rocket needs to survive, and having powerful enough rocket motors to get you there. For a rocket using solid motors with fins for stabilisation, it's hard to make that happen without going at least Mach 4, sometimes closer to Mach 5, which puts a crazy amount of stress on the vehicle.

"As for the motors, depending on how much your rocket weighs, there aren't many commercial options, which means you have to manage building a controlled explosion yourself, without, you know… exploding yourself in the process."

Initial launch

Joe has come a long way since the days of launching model rockets with his dad when he was a kid, and the thrill never fades. "You get this really cool feeling seeing something you built fly. Even when it doesn't work perfect, it's very rewarding."

For rocketry beginners, he advises starting out with a kit from Estes Rockets (**estesrockets.com**). "Rockets are expensive, but you can grab a small kit from a hobby shop with everything you need for under $50. You can do a ton of experimentation and learning at the small scale without spending hundreds or thousands of dollars, and it's easy to fly those rockets at most public parks. I did lots of small projects getting started that way, and it'll be a lot more fun and rewarding than trying to build something massive right off the bat." □

Flat-pack rocketry

Rockets don't have to be high-tech or expensive. In HackSpace #60 (**hsmag.cc/issue60**), Jo Hinchliffe took us through how to build a mini 'flat-pack' rocket from bits of balsa wood and plywood. He used a similar lo-fi approach to make a swing-wing rocket glider in issue 56 (**hsmag.cc/issue56**), too.

Build a rocket

If you feel inspired to launch your own rocket, check out the big feature back in HackSpace #12 (**hsmag.cc/issue12**). It'll give you lots of useful information on how to get started, including how to design your own personalised rocket using OpenRocket.

SUBSCRIBE TODAY

FOR JUST £10

Get three issues plus a **FREE Raspberry Pi Pico W** delivered to your door (UK only)

hsmag.cc/FreePico

Subscription will continue quarterly unless cancelled

HOW I MADE

By Stuart Moore

A CYBERNETIC SNAIL'S EYE

Taking a peek with a stalk-based vision sensor

In my work, I call myself an artist who works with technology, or a creative technologist, and a lot of my work is about the relationships that humans and living creatures share with machines.

This project is to build a very simple vision system inspired by a snail's eye. The exploration of organically inspired vision has led me to build versions of a clam's eye, a fly's eye, and a crab's eye. The key takeaway for me: simple is good. In a sense, vision starts with plants, and it all begins with the question, 'what are you trying to see?'

It began when I needed a vision system for a project, and the more obvious AI approach wasn't working well enough. I realised the problem was the camera acting as an 'eye' was far more sophisticated than the AI 'brain'. In a real creature, that would never happen – the eye and brain develop together as a single system. So I drastically simplified the camera, rethought the logic, which got simpler too, and it worked far better! Afterwards, I realised that the new arrangement was very like a clam's eye…

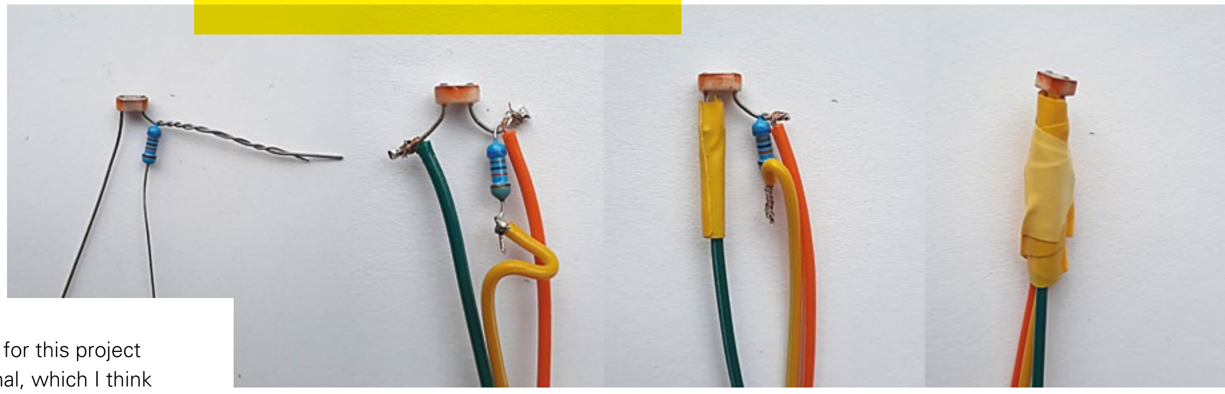


Figure 2 →
How I wired everything up and kept it tidy

I've chosen the snail for this project because it's very minimal, which I think is interesting. I would quite like to know, what are the fundamental elements or the atoms of vision? The snail comes close. It's so simple it doesn't even deal in terms of image. Vision here is really feeling at a distance.

It's not quite pure snail, either. We'll borrow a cheeky bit of fly for our convenience. Snails, as you probably know, have eyes on the end of stalks. A pupil is visible as a tiny black dot on the end of the bulb. Behind this is a fixed lens, and next are light-sensitive cells. The lens can't focus like a mammal eye. Its most important job is to narrow the field of view, so it makes a difference what direction the eye is pointing in.

A fly has a wonderfully efficient approach to the same issue. Each facet (ommatidia) you see in a fly's eye is the end of a tube. A few light-sensitive cells sit at the bottom of each tube, achieving the same directionality with minimum fuss.

"SNAILS HAVE EYES ON THE END OF STALKS"

OVERVIEW OF THE BUILD

There are three modules. One is the light-sensitive 'eye' itself. One is a Raspberry Pi Pico in the role of nervous system. One contains a vibration motor of the type that makes your phone buzz. Think of the buzz as communicating the snail's feelings about what it sees.

Parts list:

- Raspberry Pi Pico with header pins
- MX1508 Motor Driver board
- Light-dependent resistor (LDR). Part number GL5539 or GM5539
- 1 × 680 Ω resistor
- Vibration motor – the 3 V coin-style
- Small piece of black paper
- Some wires with female DuPont connectors

CONSTRUCTION 1: THE SENSOR

For this we'll use:

- LDR part number GL5539 or GM5539
- 1 × 680 Ω resistor
- DuPont wires
- Small piece of black paper
- Electrical tape →

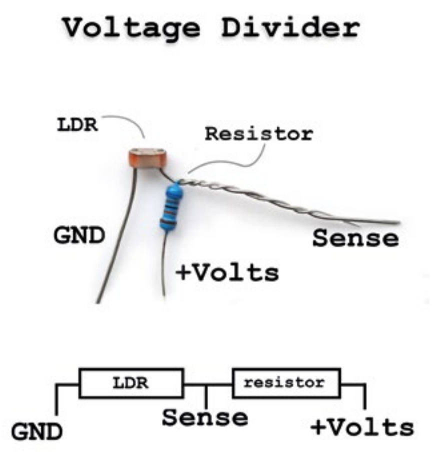
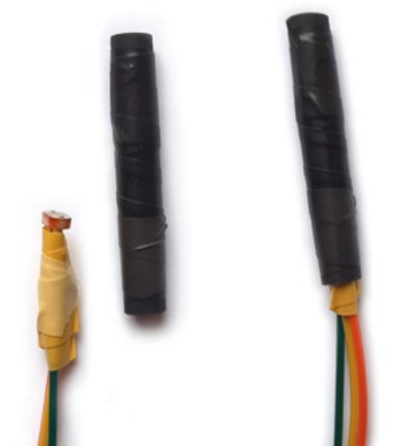


Figure 1 ↑
LDRs vary in resistance. To convert this to a voltage we can sense, we need to build a divider

Figure 3 ↓
A tube helps narrow the field of vision



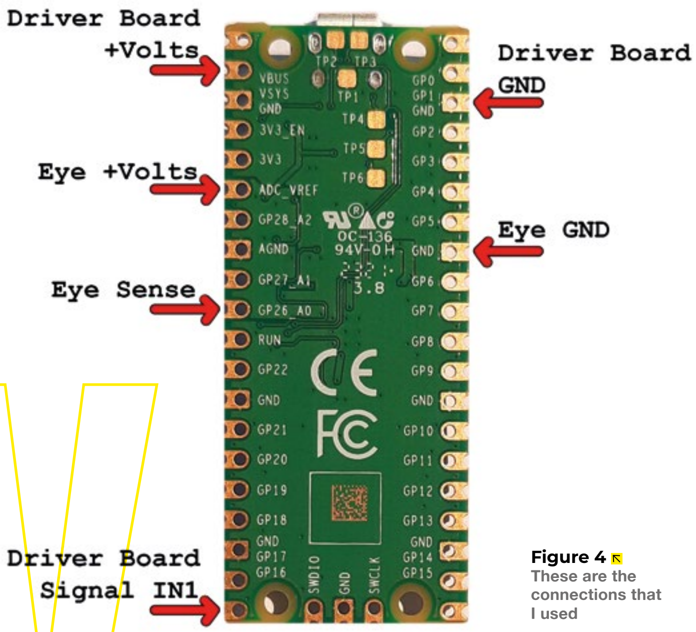


Figure 4 These are the connections that I used

First the theory:
An LDR is a type of resistor where its resistance varies depending on how much light falls on it. The GM5539 (GL5593 or just 5539 are the same part) has a resistance of around 700 Ω in the dark and 30 Ω in daylight. The Pico will be able to read this changing resistance via an analogue input pin – that is, once we've made the LDR into one half of a voltage divider circuit.

The Pico can't read resistance directly. It reads voltage, which is closely related but slightly different. Thus we need one more resistor to make a voltage divider (**Figure 1**). At one end of the voltage divider, we need a known voltage source. We'll use 3.3 V from the Pico VREF pin. At the other end, we connect to a Pico ground (GND) pin. The Pico analogue input pin connects to the middle point marked 'Sense'. It reads the voltage between this sensing point and ground. We would often call that the voltage over the LDR. The 3.3 V gets split between the two resistors according to the ratio of their resistance. If the resistors had a ratio of 2 to 1, twice as much voltage would appear over the larger resistor. For us, that would be 2.2 V over the big one and 1.1 V over the smaller one. Easy but essential. It's actually very fundamental to the idea of voltage.

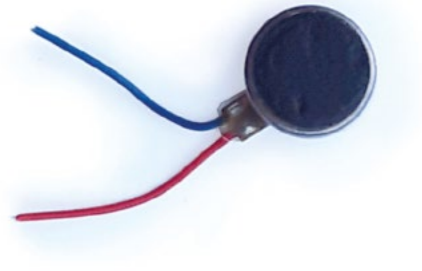
"IT NEEDS TO BE ROBUST SO WE CAN HANDLE IT"

Now to build it:
Twist together one leg of the LDR and one leg of the resistor. Now you have three points. Solder a DuPont wire to each of these. Be sure to use three different colours and make a note of which colour goes where or you'll have to undo it again later!

It needs to be robust so we can handle it and so it doesn't short out. Wrap a little electrical tape around the lone LDR leg, then you can tape up the rest in a manageable little package (**Figure 2**). The conductive parts shouldn't touch each other, and the light must still be able to get to the top surface of the LDR.

Next, take some black paper approximately 40 mm square and roll it into a tube the right size to fit neatly over the assembly. I rolled mine around a piece of 6 mm dowel. Once it's rolled, tape it and fit the tube over the sensor package (**Figure 3**). The LDR needs to 'look' down the tube, so make sure it's not bent. My LDR sits around a third of the way into the tube. The greater length of tube the LDR looks down, the smaller and more focussed its dot of vision becomes, trading off against lower light sensitivity.

Tape it if you need to, and the part's complete. Did you make a note of the wire colours? **Figure 4** shows the connections to the Pico. These are LDR to GND, Resistor to VREF, and Sense point to pin



GP26_A0 – one of four pins with analogue input facility. Headers on the Pico accept female DuPont connectors, and your modules will plug together. You can solder directly if you like, of course.

CONSTRUCTION 2: THE MOTOR DRIVER

We'll use:

- MX1508 Motor Driver board
- 3 V coin-style vibration motor
- DuPont wires

First the theory:
We shouldn't connect the vibration motor directly to the Pico. Its signal pins can't supply enough power, though they'll try anyway and probably burn out. The motor driver board adds the power behind the signals. At root, there are only two signals we can send: 'go' and 'nothing', translating into motor 'full speed' and 'stop', but we can do better. By sending rapid little pulses of power, we can get graduated speeds – a technique called pulse-width modulation (PWM). We choose the Frequency, which is how small to slice sections of time into. Then we decide what percentage of time each slice is on and off for. That's the 'duty cycle'. A 100Hz frequency is 100 slices a second. Some situations call for a far higher frequency, but 100Hz is about right for our motor. Imagine each pulse is on for half its time, off for half the time. That's a 50% duty cycle. 50% power is delivered on average and we get half(ish) motor speed. MicroPython can handle the detail and make this effortless.

Let's build it:
The motors usually have two wires already attached, and **Figure 5** shows where they solder to. If your motor has very short wires, solder on a little extra to give you flexibility in the final assembly. The connections to the motor can be very fragile. Don't be afraid to use more tape to strengthen things!

The remaining three connections run from the driver to the Pico. The '+' on the board takes in power. We'll connect this to VSYS on the Pico. VSYS is a connection to the Pico's on-board regulator that supplies the processor. The '-' connects to any Pico GND (they're all the same), which completes the circuit, and we have main motor power.

Finally, IN1 on the driver board connects to Pico pin GP16. We'll send out PWM signals on GP16 to be translated into more powerful pulses at the motor. MX1508 boards are not super-consistent, and yours may have different markings.

CONSTRUCTION 3: SOFTWARE

Part 1 – setup
There's a little bit of preparation to get Pico ready. There is excellent documentation, it doesn't take long, and you'll get to keep some powerful tools. The road map goes:

1. Put MicroPython on the Pico by downloading a UF2 file and dropping it onto the Pico. Follow the instructions here: **hsmag.cc/WhatIsMicroPython**

Left
Vibration motors are a great way of adding feedback

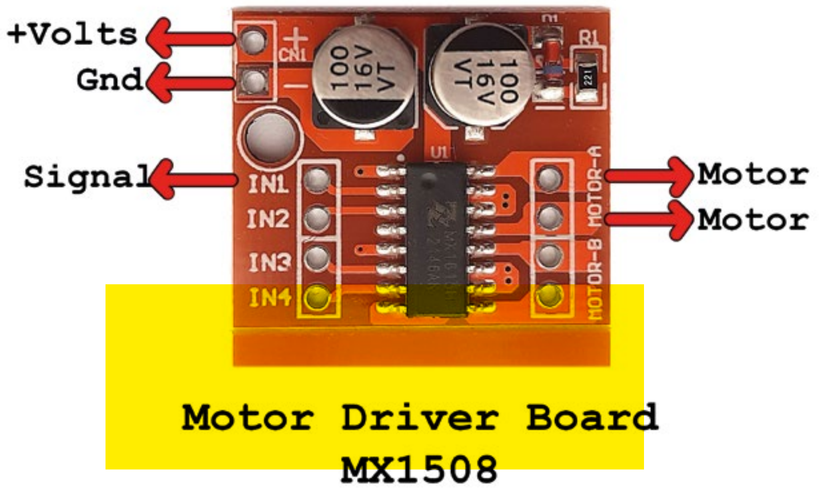
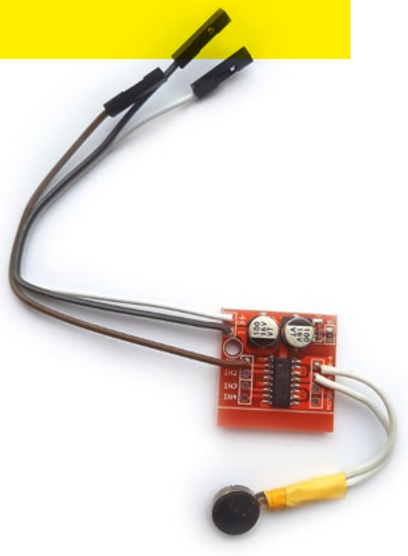


Figure 5
We only need to use a few of the driver board's connections



Above ↑
The motor driver board ready to be plugged into Pico

2. Install Thonny. A pleasingly lightweight but powerful tool with everything you need to program your Pico using MicroPython. Follow the instructions here: **thonny.org**

Part 2 – the code
The code is 13 lines of MicroPython. I suggest typing it in, but if you prefer, download it from **hsmag.cc/SnailEye**. To type it, open Thonny. Choose File > New. Type the code from the listing into the main window. The indentation matters. Choose spaces or tabs to indent, but don't mix them up. Capitalisation counts too.

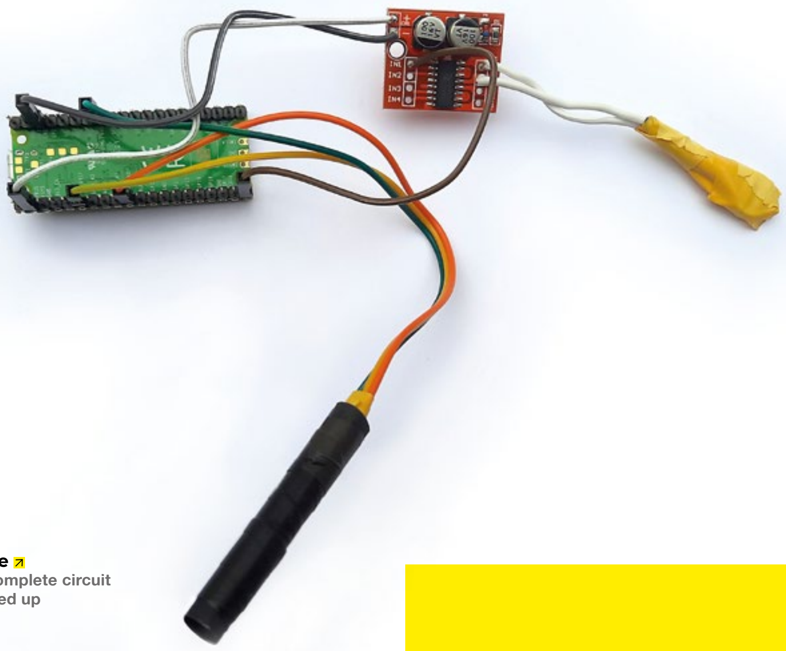
Plug the Pico in via USB and take a look at the bottom right corner of Thonny. It will say something like 'MicroPython (Raspberry Pi Pico)'. Clicking on this will reveal options to run the code locally (on your computer) or on the Pico. Set it to run MicroPython on the Pico. A housekeeping note: the 'Stop' icon near the top is your handy reset button that can fix many complaints!

Either typed or downloaded, the file needs to be saved onto the Pico with **main.py** as the file name. Choose File > Save as and you'll get options to save locally or to the Pico. We'll need a copy on the Pico and the name **main.py** means it will auto-run on power-up.

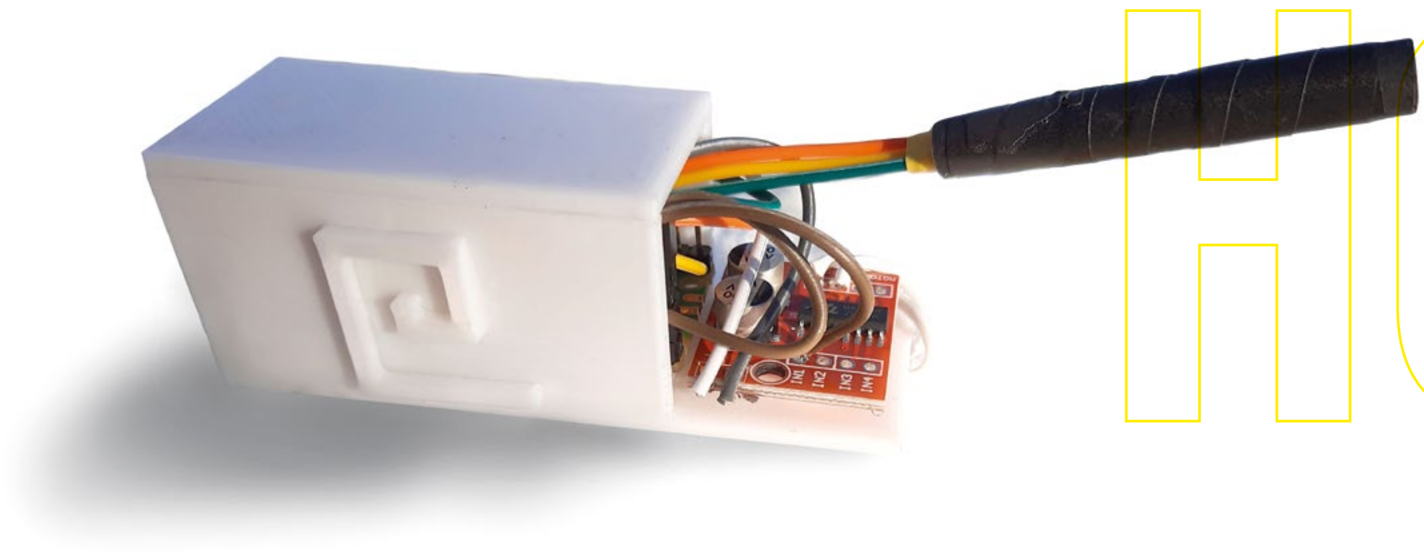
We're ready to test:
Clicking the 'Run' icon or choosing 'Run' from the menu starts the code. If all's well, your snail eye will be unimpressed and silent when looking at something bright, but buzz in approval when it sees candidates for safe, shady places. If you have a USB battery, plug the Pico into that instead of your computer for the full untethered experience. That's when we need the auto-run. Try exploring around (carefully!). Real snails bump their eyes into things all the time.

Here is the code:

```
import machine #this includes facilities
for Pico hardware
import utime   #this makes the delay
function available to us

analogInputPin = machine.ADC(26) #Get the
input pin ready
pwmOutputPin = machine.PWM(machine.
Pin(16)) #Ready the PWM pin
pwmOutputPin.freq(100) # Set PWM
frequency to 100Hz

#This block of code will loop forever...
while True: #The colon and the indenting
define the block
    x = analogInputPin.read_u16() #Read
from the eye
    y = (x*x*x*x)/12.5e12 #or how about
(x*x)/2.5e4
    y = min(65535, y) #These two lines
clip y so it cant
    y = max(0, y)     #be out of the
range 0 to 65535
```



Above ↗
The complete circuit all wired up



Above
Everything fits neatly into a 3D-printed 'shell'

```
    pwmOutputPin.duty_u16(int(y)) #Set
the PWM duty cycle
    print("PWM: ",y) #Report back to
Thonny if its connected
    utime.sleep(0.2) #Slow the loop down
for the motors sake
```

A little look at the code:
The gist is simple. We read the voltage over the LDR via pin 16. This shows up scaled between 0 and 65535 (there is a good reason!). More dark makes a bigger number, so we want more buzz. For that we set a bigger duty cycle. This is also scaled between 0 and 65535.

Line 9 is a little maths to get the right kind of feel. It makes the response non-linear and is roughly equivalent to a single neuron. I've chosen to take the signal and raise it to the power of four, making the light and dark reactions feel more 'separated'. After raising x to the power of four, it needs scaling back into the 0 to 65535 range again, thus we divide by 12.5e12. If you haven't seen that format, it means 12.5 with the decimal point moved twelve places to the right. You could type in 12500000000000, but you're more likely to make a mistake. Try changing the maths and see what happens. There's a suggestion in the comments to get you started. If you don't feel confident with the maths, definitely try experimenting with it!

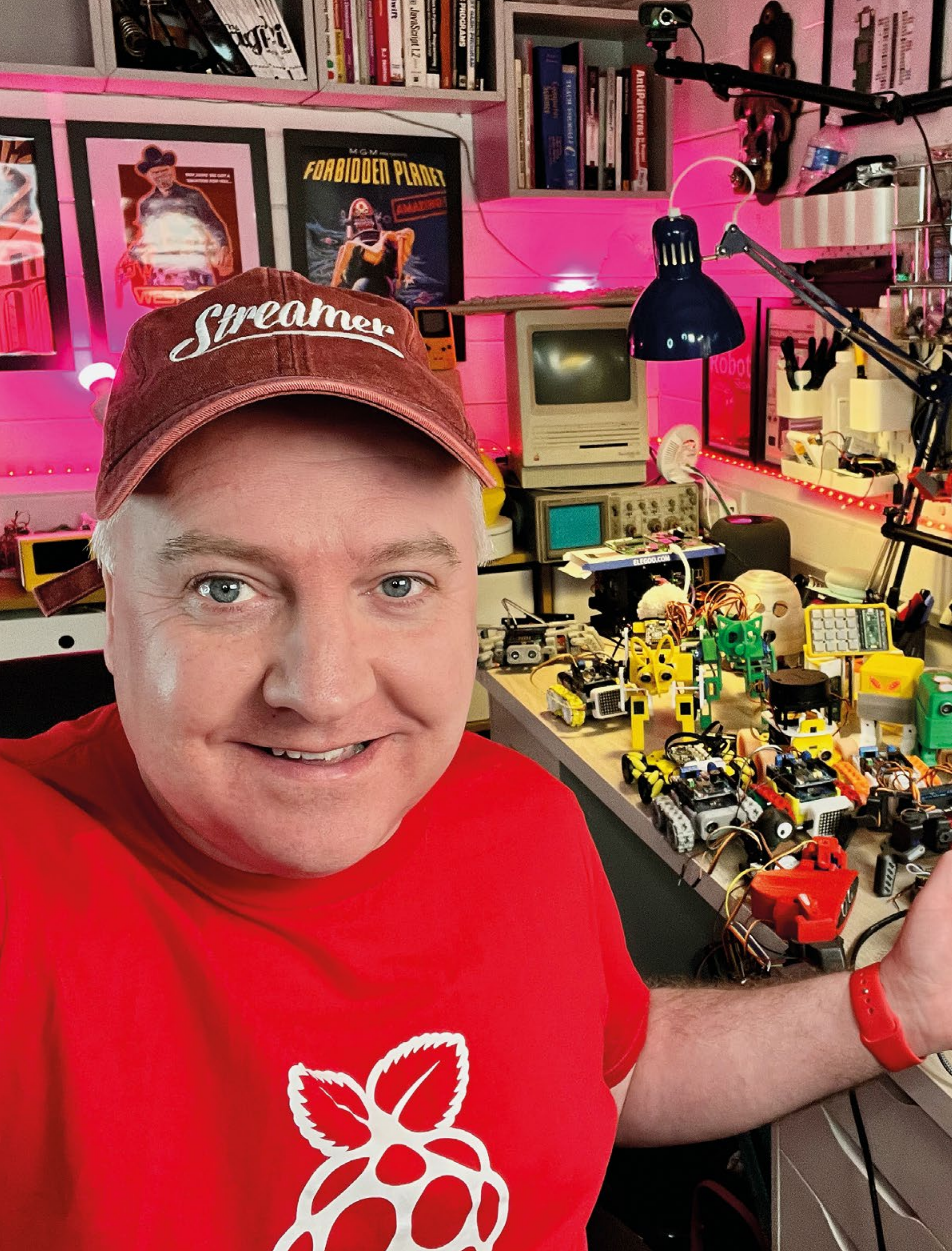
"YOU COULD TRY EXTENDING THE EYE"

WHERE NEXT?

Some kind of enclosure is a good idea. Perhaps you could 3D-print something, or use my file if you like. It reminds me of a piece Matisse did about a snail. Just not for the right reasons. I think he may have taken a bit more care with his.

You can get all the code and bits you need from **hsmag.cc/SnailEye**

You could try extending the eye to include a non-directional sensor. Another thought is that snails have a behaviour called 'rising up'. If they've been in the dark for a while then find themselves in the light, they stretch up their heads and have a good look around. Coding that would be an interesting challenge.


Streamer
FORBIDDEN PLANET
ELEGOO.COM

HackSpace magazine meets…

Kevin McAleer

Teacher, maker, builder, programmer, 3D printer, and all-round Renaissance man

Kevin McAleer is a great example of someone who wanted to do something, and so he went ahead and did it. His secret? Start small, fix your mistakes, and do it again. Starting with an obsession with modular printed robots, then MicroPython, then livestreaming, he has built up **kevsrobots.com** into a great resource for anyone who wants to get started with robotics, the intersection of mechanical engineering, electronic engineering, and programming. Since then, he's also built up a course to get beginners working with MicroPython, and even more impressively, he built a working PKE meter to detect psychokinetic energy in your neighbourhood with only a Raspberry Pi Pico and a 3D printer.

We spoke to Kevin, cooed over his lovely collection of 3D-printed robots, and made plans for an automated world. →

Left ←
Kevin builds and broadcasts in a geek paradise he's built at the bottom of his garden

HackSpace Right, first question: who are you, and what do you do?

Kevin McAleer I originally got into livestreaming because I was obsessed with SMARS robots, these little Arduino-based robots. I didn't design these – there's a guy in Switzerland, I think, who designed them. But I did start trying to amass the information about these robots, because there are lots of add-on modules that people have made. And so I created **SMARSfan.com**, with the idea of bringing together all the different information, the code, the modules, and the parts you would need into one single place.

It was a bit of fun for me designing a web page, and other geeky stuff behind the scenes, like building it in Markdown using something called Jekyll, so it automatically builds itself from Markdown files.

Then I set up hosting for it on Raspberry Pis – actually it's this little cluster here that I've got looking like a Cray supercomputer.

So that was the original website that I created. And it was great for tutorials, but I thought it would be really good if I had some video content on there.

I wasn't very comfortable being myself in front of the camera, so I thought I'd probably do a screenshot of me typing some code in, designing something in Fusion 360, and then do some kind of robot voiceover, because I didn't want to be a personality behind the thing. But the more I looked into this, the more I realised that I don't like editing – it takes too long, and I'd rather just record everything in one go and if things go wrong, things go wrong. So livestreaming was kind of a cop out to begin with, just because it was easy to do. And then I got hooked on it, and I do it every week now.

I think there are two ways you can be driven: you can either be disciplined, or you can be motivated. And for me, the Sunday livestream show is discipline; I go live every single Sunday at seven o'clock because I'm disciplined to do so. Through the week, I might think of an idea that I think would be really good, really topical or something that I've always wanted to make.

Take this for example: it's something I always wanted to have, which is a *Back to the Future* Time Machine. So I made one. This is a whole bunch of LEDs with loads and loads of wiring. By using some of what they call shift registers, you can just get each one of these rows of pins down to like a single pin and just shift them across, so it's quite simple to get this up and running. I had a bunch of twelve of these little four-segment displays. And I 3D-printed three little boxes, and then the little pieces that go on the sides... I always wanted to have one and I thought, oh, I can make one now.

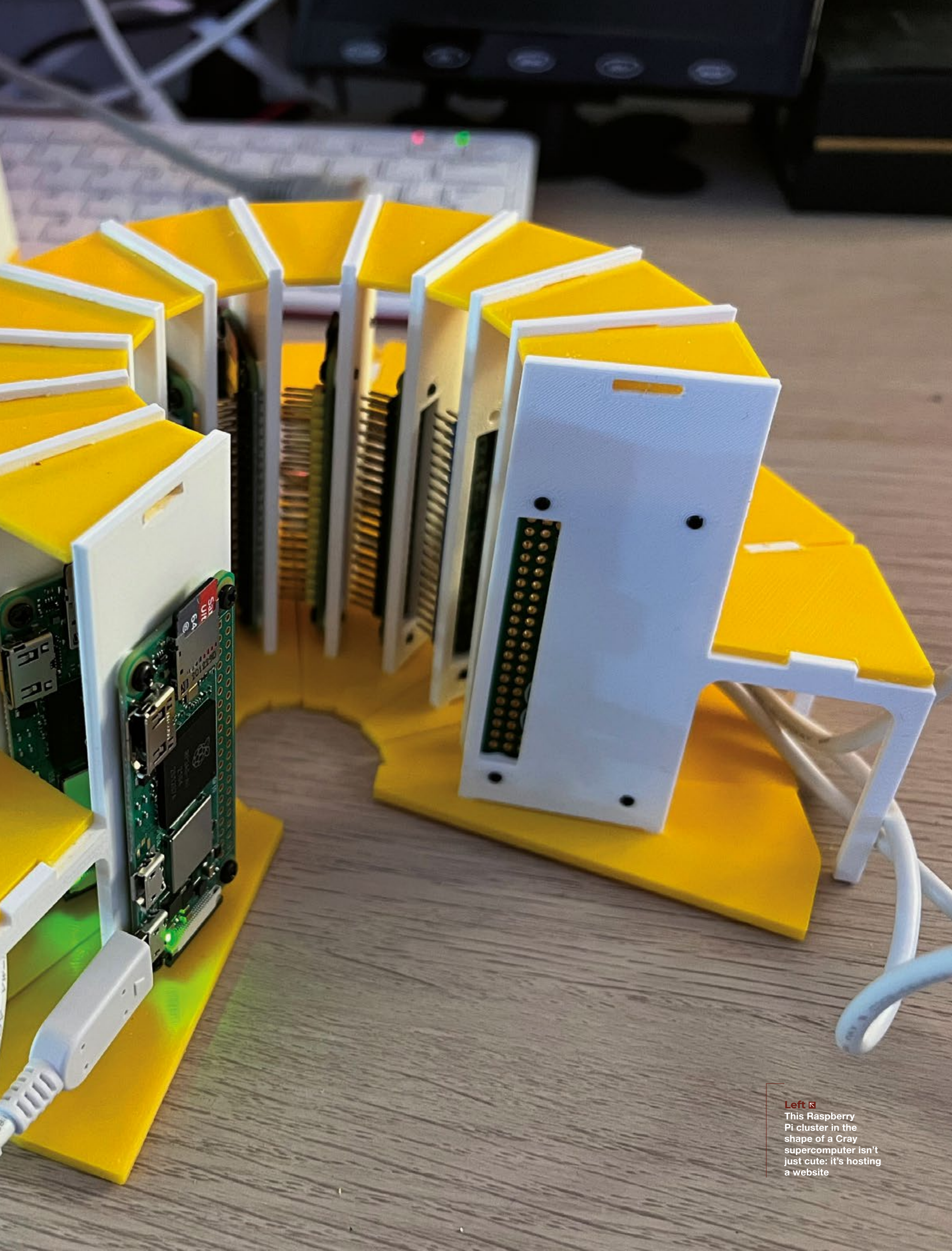
I used to work in education, for about eleven years. I wasn't a teacher, I worked in IT support. I used to manage all the computers and networks and things. But I really enjoy being in the educational environment.

In my day job about 20 years ago, when I started contracting, I could either go down the management route or the developer route, or continue in the sort of technical support route. And just for financial reasons, I went down the management route doing project management. And that meant that my creative outlets were sort of being held back somewhat. So when I discovered 3D printing, when I discovered Python as a language, certain things began to unlock.

It's like a Venn diagram of three intersecting circles. You've got mechanical engineering, you've got electrical engineering, and you've got software engineering – and they all meet in the middle at robotics.

One of the things I wasn't very good at was the mechanical engineering side. And 3D printing and 3D design, using Fusion 360 has just really helped me unlock that problem. So therefore, I was able to make more and more robots. And each robot that I make has something unique about it that's different than the ones previous to it – each one is a step up in capability.

More recently, I've set up this **KevsRobots.com** website, which, again, →



Left
This Raspberry Pi cluster in the shape of a Cray supercomputer isn't just cute: it's hosting a website

Above Strangers from across the internet have controlled this Billy Bass remotely thanks to Pico W

is hosted just behind me on four Raspberry Pis. And I've been putting more effort into supporting materials as well as during the show; a lot of people kept saying, 'Have you written this up anywhere?' And I really don't: I do a project per week and then move on. So, I do the show on a Sunday; Monday, I have a day off, I'll just relax and not think about anything related to that. Tuesday, I'll be then looking through my list of ideas, like a Trello board full of possible project ideas, things I could make. Some of them are robots. Some of them are pure software. Some of them are fun things, like the Billy Bass fish.

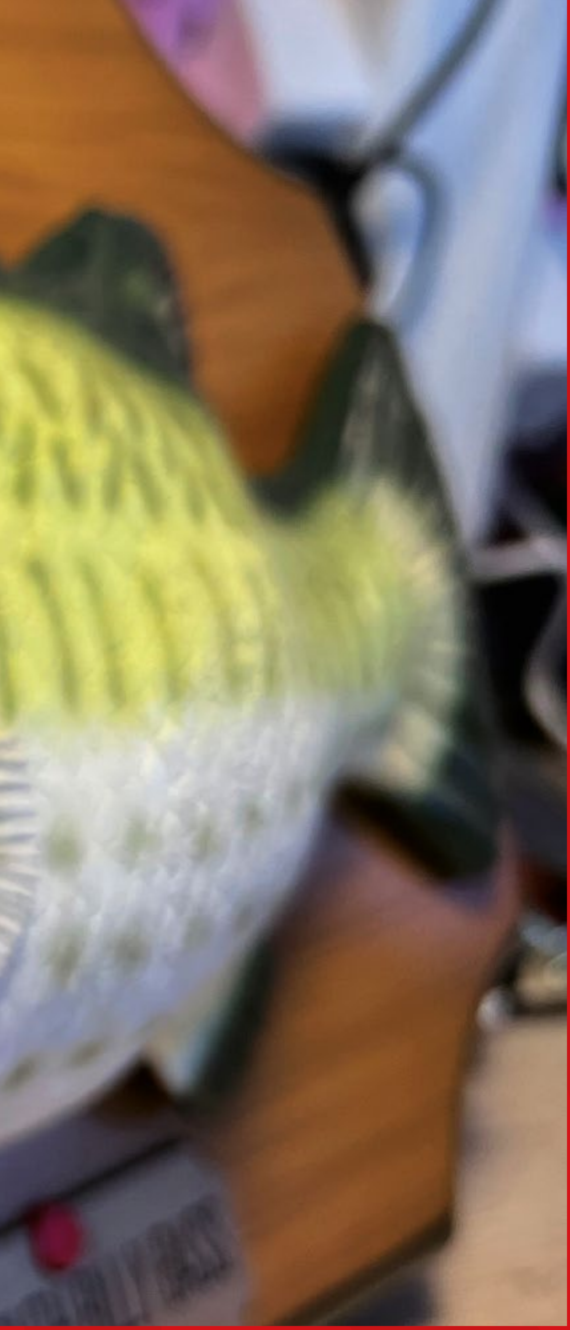
And I'll pick one of them and think, you know, that's the one I want to work on for Sunday's show. And then, so Wednesday I'll be working up more details. I might be trying out some code or physically designing something. Same with Thursday. Friday, I'll be getting close to thinking about finishing this now. What's this going to look like? How far have I got from the original idea to where I am now? Do I have any interesting insights or developments in that? Do I need to split this into two shows or what? And then Saturday, I'll be finalising the code, making sure everything works. And then Sunday will be preparing the slides for the show. Just going through [it, so] we'll have some set of slides to talk to. And then it's showtime. Rinse and repeat.

HS You mentioned teachable moments. Does that mean you're building toward one final robot that will have all these capabilities you've covered?

KM Absolutely. One of the robots I first started is the InMoov robot. It's a full-size human-robot designed by a French guy, Gael Langevin. I started 3D-printing one when I got my first 3D printer. And I realised quite quickly that I'd bitten off more than I could chew – I was way out of my depth on the mechanical side.

Code-wise, this was all built around like the Arduino Mega, and the software that they use on top like a little PC stuck in the back to drive it all. And it's all in Java – it was a bit of a mess really, because the guy who designed it was a sculptor; it's not really a software engineering project.

To me, it kind of shows, and I wanted to be able to make this thing but make my own, you know, tweak the design, using someone else's files. It's a good starting point, but it's not where I want to go with



it. And I didn't like the way that it was implemented. In order to build a full-size humanoid robot that can walk around and do stuff, there's so much you need to learn.

So where do we need to start? Let's start right at the basics. There's a reason why my robots are all small, like this, because they're quick to print. And they're quick to learn. Because if you build one that's like a full humanoid size thing, you could be printing out for a year, before you get to try it out.

So the learning curve we're creating with small robots is a lot quicker, a lot easier to manage. And as well as documenting my own journey through what I've done through videos and written things, a lot of people have found it really useful themselves. So, I think one of my natural gifts is that I'm able to explain complicated things in a simple way. Not everyone can do that.

HS Before you answered it yourself, I was going to ask if you were a teacher in a past life, because a lot of your builds read like lesson plans.

KM So I didn't work as a teacher, but I was very much mindful that when I was working in the school, they had these little whiteboards put up there. They're only A4 size, and every classroom had one. And it's because they needed to have a plenary, like a learning objective for the particular lesson. And it's one of those things that stuck with me that they've invested so much in these little whiteboards, specific for the learning points. And I thought, that's a really great way of understanding it yourself.

One of the ways that you can master something is to teach other people. One of the things that's helped me learn, and retain what I've learned, is by reflecting that back and saying, 'This is what I've learned, and this is why I found it difficult', because other people will relate to that.

HS If hardware was the missing part of the Venn diagram, does that mean that you're already familiar with coding?

KM I've done software development for a very long time in many languages, but never commercially, never professionally. It's always been a hobby. And so, when I work with companies like Pimoroni, or a host of other companies, I guess Kitronik is one we're working with recently. You know, you see their code and you think, 'Oh, that's interesting why they've done that'.

I worked with a company doing call centre software, which on the face of it is a really boring topic. But during the lunchtimes, I would hang out with the software developers and we would talk about stuff. In fact, one of the things that made me double down on to the livestream is that we had a regular lunch and learn. So every lunchtime, somebody would bring something – could be anything. There's a girl there who did one on Norse mythology. So you could learn all kinds of stuff in half an hour. And yeah, I brought my robots and put them all on the table and talked about how everything worked. And they're all captivated. So yeah, some of those software developers have really encouraged me to sort of, you know, get my teeth into this more.

HS Why do you use Python or MicroPython in most of your builds?

KM Great question! I've learned a lot of programming languages, right from university where we learnt COBOL, Assembly language, all kinds of things like that. And they're all very esoteric; you have to know special code words. There's all kinds of curly braces, or semicolons, and conventions. And it doesn't lend itself to very easy code. In fact, I was looking through some really old BASIC programming books, and even that's quite difficult to follow.

So it doesn't lend itself to being understood by other people. If you've written something years ago, and then you come back to it, you think, what was I writing here? What does all this mean? So when I discovered Python as a language, it reads like pseudo code – it reads like the kind of code you'd write before you then write it in real code somewhere else. In fact, it reads like English.

And the conventions that it uses, like the spacing, the indentation, the fact it hasn't got all those curly braces and things, that means that it's clearer and cleaner.

And I've only learned Python probably about five years ago. Obviously, if you know one language, you can transfer your knowledge to other languages, and that does make it easier. But with Python, I thought, I just get this, this is for me; it's how I think programming languages should be.

Then there's the fact that it's interpreted rather than compiled. One of the good things about this is that it's instant to run. So any device that you can write code →

for, it will just run. There's no compile times, no waiting around. With a compiled language like C or C++, you might wait 20 minutes for something to compile... that's not great if you're trying to develop something quickly. So, for me, Python kind of gets out of the way as a programming language, which makes it so much easier to dive into.

And the other great thing with Python is that because it's 30 years old or something, it's surprisingly mature as a language. I was really surprised when I read up on it. And because of that maturity, there are a number of people that have embraced it and used it. In data science, for example, it's the go-to language.

I know Microsoft Excel is going to have a Python plug-in very soon. And very soon, they're gonna get rid of all the Visual Basic stuff, because it's terrible. Python is just where you want to be. So there's a reason why it's so popular. Between it and JavaScript, I think they're the two most common and popular languages.

And there is something called PyScript, which is probably about to revolutionise browsers and get rid of a lot of JavaScript. I know it's very well-used, but Python is far better.

And MicroPython? That's just like the chef's kiss. I started out using the Arduino for robotics and stuff, but I very quickly hit the ceiling with Arduino. There are limits to how much you can stuff into one of the chips, how much code you can run at a time, the limitations you hit pretty quickly.

So, I was looking around for something better, and I discovered the ESP8266 and ESP32 chips. And I was like, 'Oh, these are nice'. And then I discovered MicroPython. And I was like, 'this, this is genius, you can run interpreted code in Python, on an embedded device'. It blew my mind when I discovered that you could do this. And this is before the Raspberry Pi Pico came out.

That cycle of trying something out, developing it further. The fact you can do that in real time, without having to compile stuff and wait for it to upload and all that... it just works. For me, that was just one of those little moments when the light bulbs go on. So when the Pico came out, I was like, this couldn't be better: it's an embedded device; it's from Raspberry Pi; and it runs MicroPython. So it's not a big surprise to me that this has become one of the most popular platforms out there, particularly in education as well, because it's cheap enough that you can break it and not worry about the cost.

HS What do you find more fun? Building or coding?

KM Building is more fun. The joy that you get when you write code and the code works: imagine that multiplied when you make a physical thing. It's just the best feeling in the world. When you make a robot for the first time, and you've 3D-printed it...when you get that moving round and interacting with its environment. It's like when you run a piece of code for the first time, and it works and does what you expect, you get this kind of endorphin adrenaline rush of just like, 'wow, it works'. So that gets magnified when it's a physical thing and it's really interacting with the world. So doing something like hacking a Billy Bass fish, when that actually works, and you can see people, you know, running your code over the internet, interacting with it, and it's moving and opening its mouth, you get double the joy.

"The joy that you get when you write code and the code works: imagine that multiplied when you make a physical thing"

You also get frustration as well; I think that's part of the thing with building real objects. One of the things that I didn't appreciate early on when I was making 3D-printable things was how many iterations you need to go to. For me, it's about three iterations before I hit something I'm satisfied with, before I'm happy with how everything fits and works together. So yeah, it can take a little while. For example, this radar robot that I made: this one took quite a few goals at getting all the different parts just right, and it uses the same base as a few of my other robots as well, so I got to recycle and reuse some of my ideas. So, for me, that was an unexpected frustration, that sometimes you can wait hours for something to print out, and then you put it together and it doesn't work. I find myself thinking 'if only I'd put a hole there, or just allowed a little more tolerance'.

But then the beauty of 3D print is that you don't always have to go back to the drawing board, but you have to go back and redo it, tweak it, and then print it again. That's just part of the design process now that I factor in. That's the joy of 3D printing: you fix a job, come back in a few hours, and there's a new part waiting for you.

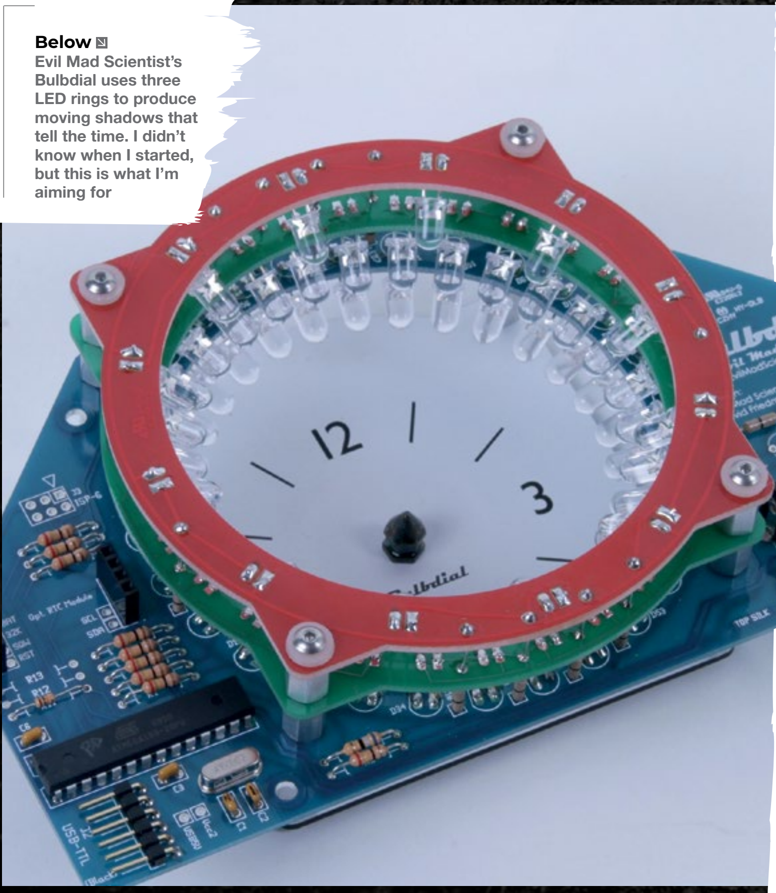
I think it helps when you've got a project in mind; you can learn so much more if you have to physically make something that works. So, rather than just pure coding or pure design, when you make a whole thing, you have to make compromises. If you ask any maker about something that they've made, they'll tell you that it's not finished – there's always something else that they want to do, they feel a bit guilty that if you look behind the scenes, there's a bit of a botched job or a broken bit. But I quite like that, like on this *Ghostbusters* thing I made. I was showing the missus and she said 'yeah, it'll be nice when you get rid of all the wires sticking out'. No, no, no! That's part of the aesthetic! It's not supposed to be like a finished product. You look at things in a whole different way. And I think it just gives you a more positive outlook, because rather than looking at the past and thinking about what you should have done better, you're looking towards the future thinking about what you're going to build next. □

Above ↗
The world's most over-engineered Christmas tree bauble: Cheerlights in a 3D printed tribute to the game *Portal*

IN THE WORKSHOP: An indoor sundial

By Andrew Gregory

Yes, yes, we've got light bulbs these days. But where's the fun in that?



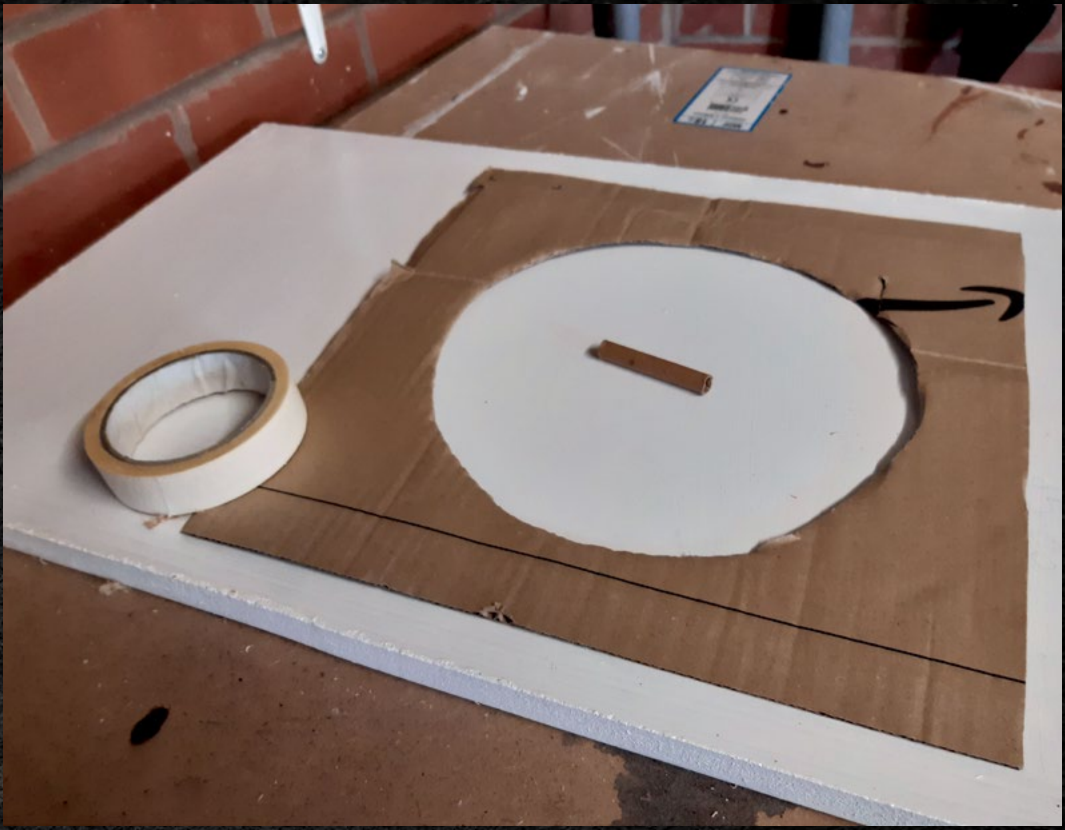
Below
Evil Mad Scientist's Bulbdial uses three LED rings to produce moving shadows that tell the time. I didn't know when I started, but this is what I'm aiming for

The bright glare of a phone screen is really bad for your circadian rhythms, which is why it's terrible that so many people use their phones to tell the time at night.

What led to this project was a desire to know what time it is when I wake up in the middle of the night, without confusing my monkey brain with bright blue light and without having to fumble around and knock things over looking for my glasses so I can see a clock.

It hit me: instead of telling the time with light, why not use shadows instead? A light moving around a stick once every 60 minutes will create a shadow that moves in a circle every 60 minutes. You'd be able to tell the time using the position of the shadow. Brilliant!

My first prototype was pretty ramshackle. I taped LEDs to the hands of a clock mechanism. I could either power the LEDs with wires (which would get horribly tangled before long), use very small batteries, or put the whole thing into an inductive coil to power wireless LEDs. I went for the cheapest option, using coin cell batteries, which were too heavy for the clock mechanism; also the LEDs got in the



Left
It makes sense to prototype with cheap stuff. I'm using scrap MDF, the inside of a roll of dog poo bags, and some Amazon packaging before I break out the Indian rosewood and maple

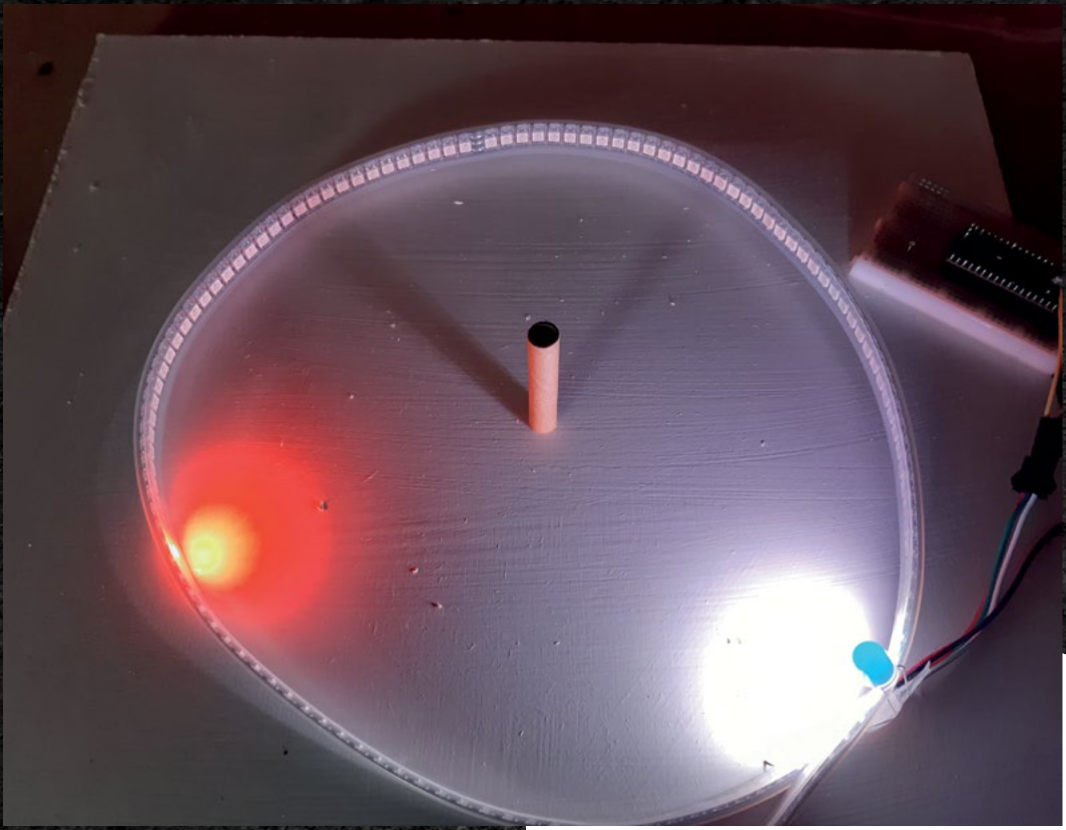
Below
It's not pretty, and it doesn't work. Still, it pointed us in the right direction

Someone had already come up with the same idea that I had

way of the clock hands. It didn't work. This was disappointing.

However! While looking around for alternative designs, I discovered that someone had already come up with the same idea that I had. David Friedman of Ironic Sans (**ironicsans.com**) envisaged his light sources running around on tracks – a little like an orrery. The intriguing thing about this, though, is that he also had a crack at putting it into practice… and ended up with something that looks nothing like his original concept. The Bulbdial, developed at Evil Mad Scientist Laboratories (**hsmag.cc/Bulbdial**), uses a ring of LEDs surrounding a central gnomon. Instead of moving a single light source, via a motor, different LEDs light up in sequence to create the movement. Oh, and there are three rings of LEDs in different colours, to denote hours, minutes, and seconds. Evil Mad Scientist used through-hole LEDs, but whenever we see a ring of shiny things like that, we think of NeoPixels.

Problem 1: to connect the Pico to the Raspberry Pi 4 I was using to program it, I needed a micro USB cable. "Great, I've got a few of those around the house", I naively thought. I was wrong. It turns out that there are cables and there are *cables*. Some, such as the ones you use to charge your phones, only transmit power. Others, such as the ones you use to get information from a computer to a development board, transmit data as well as power. The internet will tell →



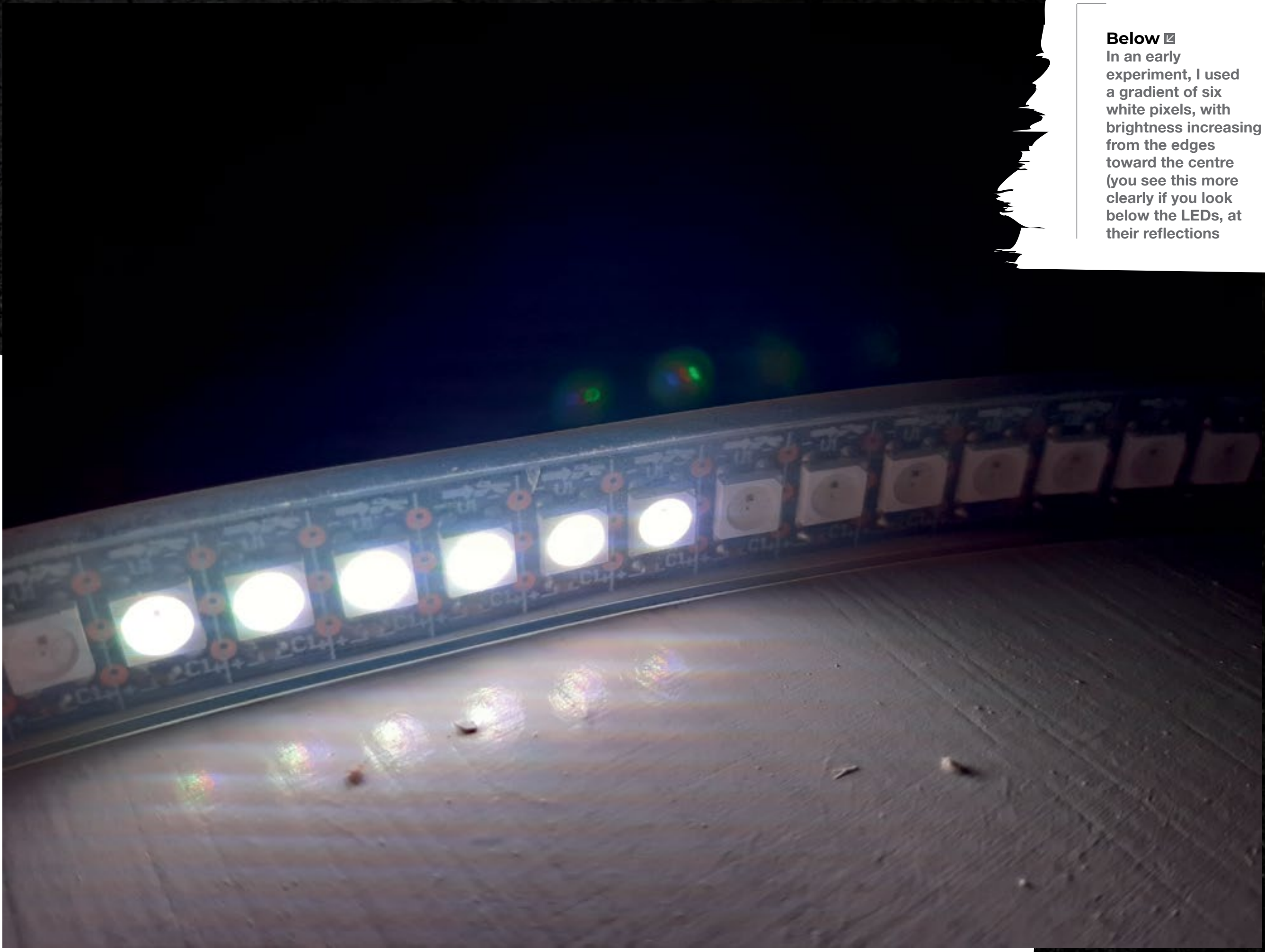
These three problems could only be solved by trial and error

Above
The hour-hand says 10.30; the minute hand says five past the hour. It's not perfect, but it proves that shadows cast by NeoPixels are a workable way of telling the time

Right
There's light leaking out of this all over the place. It would not be a restful object for the sleeping room

you that there's a way of telling between the two – a look in the micro USB end should show more indentations to expose more copper in a data cable compared with a power-only cable – but this method of identification didn't work for me. Perhaps the manufacturer of mine was canny enough to put fake markings on to make it look like a data cable, I don't know; but it doesn't matter, because I bought a handful for this and future products from Pimoroni, purveyors of known quality goods. This took about a day to diagnose, which is only just shorter than the length of time it took for the new, trusted cables to arrive.

Problem 2: Looking at the Pico with the logo the right way up, the pin in the top-left is in position 1; it's even helpfully numbered 1 on the device's silkscreen. Unhelpfully, this is not GPIO 1: it's GPIO 0. I spent far too long stuck on this. I only realised that this was holding me back when I encountered problem 2a, which is that when you copy someone else's code, even when you're only trying to run something that's been done thousands of times before on a standard bit of kit like a NeoPixel strip, you can't just run it as-is and expect it to work. At the very least, you have to make sure pin numbers line up. When you import code that you're hoping to learn from, even basic errors like this will trip you up.



Below In an early experiment, I used a gradient of six white pixels, with brightness increasing from the edges toward the centre (you see this more clearly if you look below the LEDs, at their reflections

Problem 3: We just used the phrase "a standard bit of kit like a NeoPixel strip". The hubris of that assumption came back to trip us up, as it turns out that not all addressable RGB LED strips are the same.

We have a strip in front of us that's wired, from top to bottom, 5V, Data In, Ground. We have another strip that's not labelled, but by slow trial and error we determined is wired Ground, Data In, 5V. Unfortunately, we used the unlabelled version first and had to work out which wires were which by trial and error. To make matters worse, NeoPixels are not always wired in this order, but many people don't realise this, so you can find yourself following a tutorial online that confidently gives you the wrong information. If you're in any doubt at all, the lower the pixel density, the more room on the strip there is for the manufacturer to write important details about which electrons should go where.

Sadly these three problems could only be solved by trial and error. A more fun version of that was messing with the code. Once you've got a known script working, you can break things as much as you want, then reinstall fresh code and go again. I used the pi-pico-neopixel library by Blaž Rolih, aka blaz-r. (Apparently he built on the open-source code of one Ben Everard, HackSpace editor, so cheers Ben!)

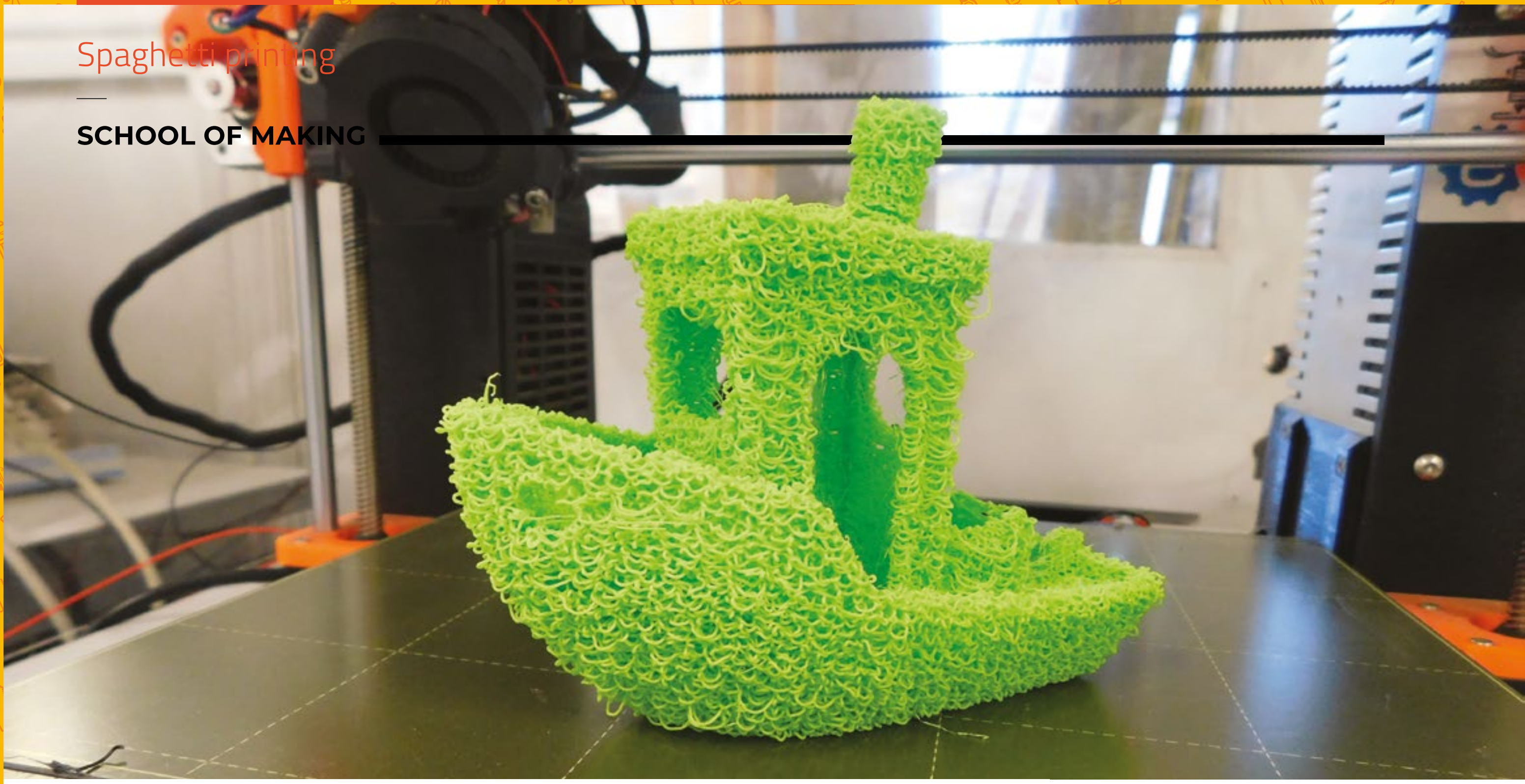
I'm not going to pretend to know what we're doing with code, but using blaz-r's examples I changed values of the `sleep()` function, to make 120 pixels change every 30 seconds, creating an animation that lasted for a minute; the colour and brightness, and I misguidedly created a gradient, thinking that it would make a larger shadow that could function as a minute hand. All this does is make the shadow weaker at the edges. Now, I just need to get some sleep to see what this looks like if I wake up in darkness. □

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



Spaghetti printing

When 3D printing goes wrong in the right way

Ben Everard

@ben_everard

Ben's house is slowly being taken over by 3D printers. He plans to solve this by printing an extension, once he gets enough printers.

When we were testing out active foaming PLA (a filament that swells up as it comes out of the nozzle), we found an interesting failure mode. This filament can print in layers thicker than the nozzle is wide, but if you push it too far, you get a spaghetti-like effect that just about holds together enough to make something resembling the desired object, while still looking like a tangle of spaghetti. Other people have compared it to dehydrated noodles, or a loofah.

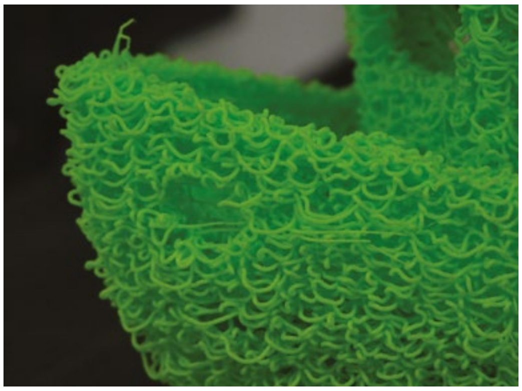
There are a few tricks to make this work, and honestly, we've no idea how well this will work on other printers – it might be very reliant on the exact model we've used to test this out (a Prusa i3 MK3S), or it might work on a wider range, or it might work on almost any model.

First of all, you need to make sure you're using the right filament. We tested this out with colorFabb's LW-PLA. There are other brands of LW-PLA, but not many work. Crucially, the filament we used is active foaming, which means it includes an ingredient that foams up when it's heated. Some other LW-PLA filaments are pre-foamed, and these may not give the desired effect.

We did this using PrusaSlicer, though it should be possible to set this up similarly in other slicers.

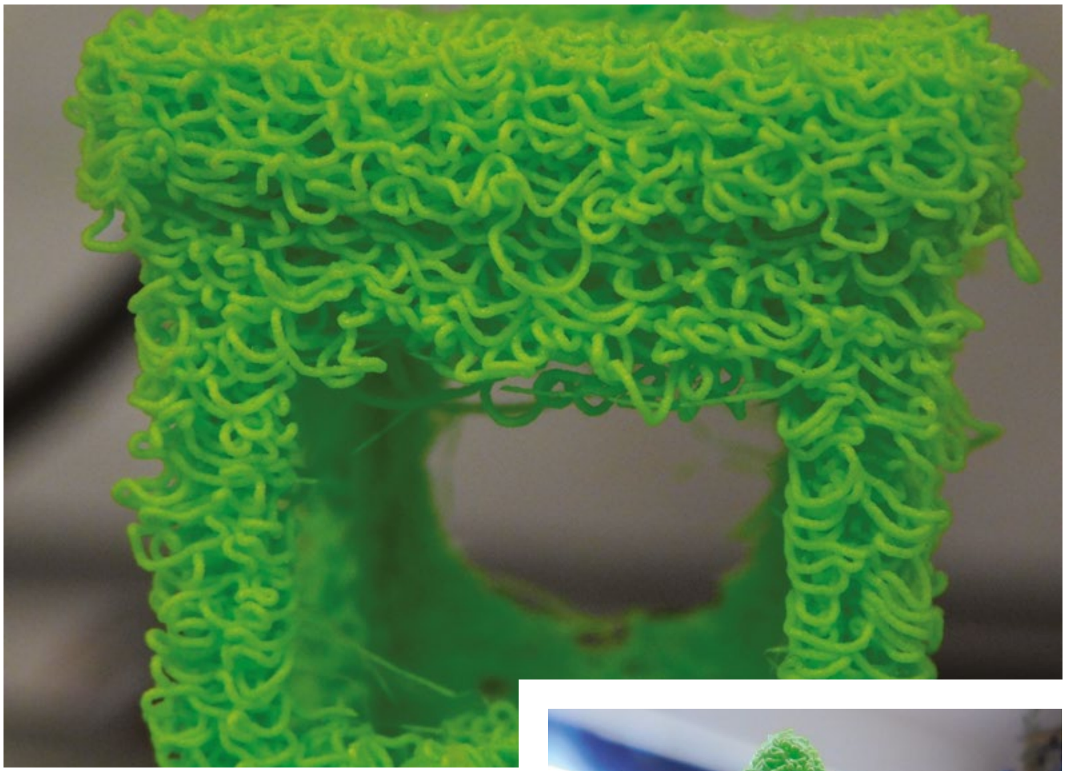
Firstly, you have to set the nozzle diameter. The slicer is intelligent enough to know that you can't print a 0.8 mm layer with a 0.4 mm nozzle, so in the Printer Settings tab > Extruder 1, enter 1 in the nozzle diameter tab. You can now go to the Print Settings tab > Layers and Perimeters and enter 0.8 in the layer height box.

If you try and slice now, the extruder will calculate how much filament it has to extrude through a 1 mm nozzle to create a 0.8 mm layer. This is quite a lot and far more than we need. The critical thing now is to change the Extrusion Multiplier setting in the Filament Settings tab > Filament. Changing this number has quite a big impact on the final result. We used 0.2



Above
This double-sized Benchy is one of the most unusual we've made

Right
Noodles, loofah, sponge. Everyone has a different idea of what this looks like



Left Honestly, we're amazed it managed this overhang

Below The textured finish seemed ideal for Christmas trees

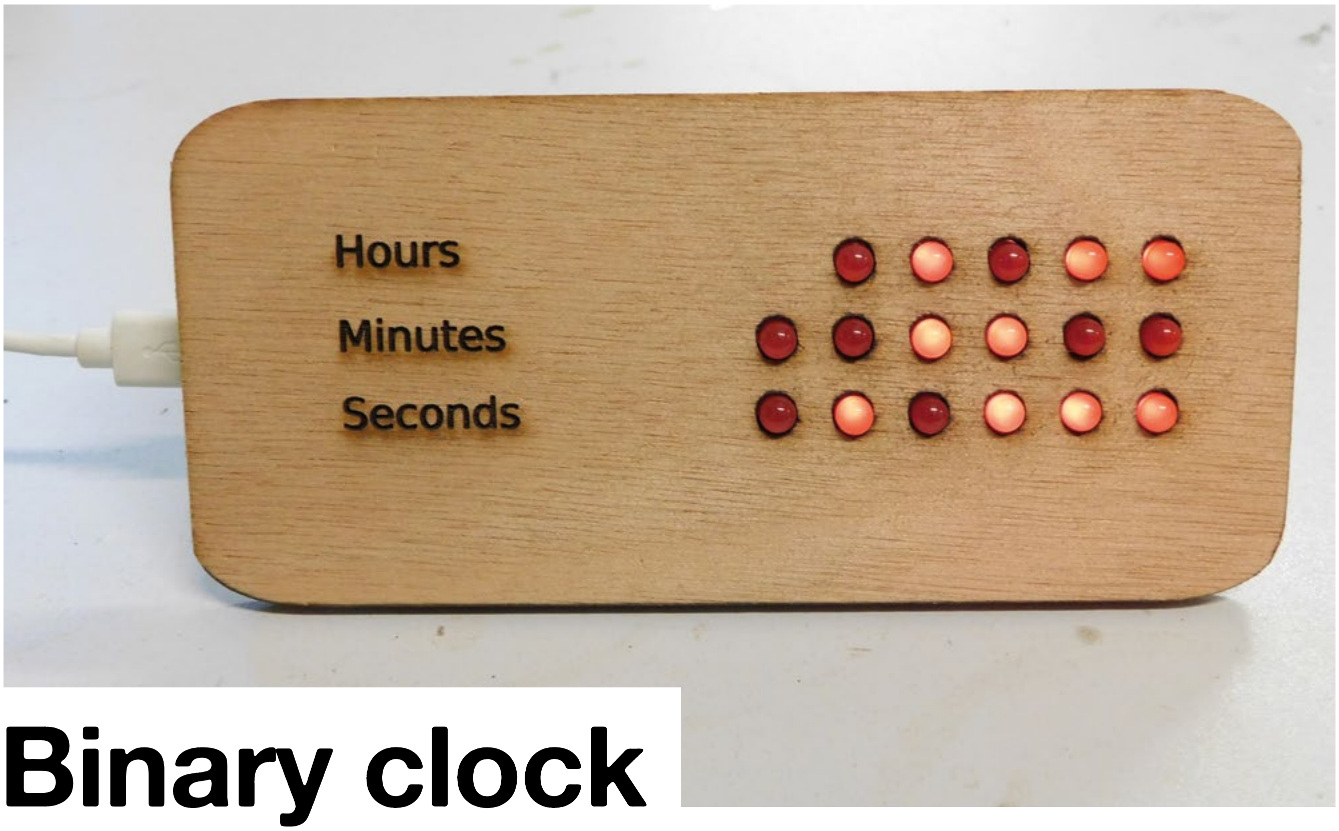
to create quite a loose structure, but you could try setting it a little higher for a more dense structure. The final thing is the temperature. We want the PLA to come out very runny, with foaming to add some chaotic twists and turns, so we set the temperature to 260. That's all there is to it. Now, you just need to grab a model and slice it.

To be honest, we expected overhangs and bridges to fail completely, but it can handle surprisingly steep overhangs and long bridges. It doesn't cope particularly well with low levels of infill – we'd recommend starting with about 40%, and adjusting it depending on how your model turns out. Don't worry, you're not wasting much plastic, as each print uses just 20% of the plastic it should. You also won't be wasting much time, as print times are far faster than can be achieved with traditional prints.

WHY?

We'll be honest – there's probably no reason to do this, other than if you like the look. However, that hasn't stopped us searching for a reason.

The porous nature of these prints could be useful for something. We've started an experiment trying to use them as a substrate for growing moss on, in the hope that this could create a living, 3D-printed Christmas tree design, but it's too early to see how this is going. Perhaps there's some other use for this. We don't know, but if you've got any ideas, we'd love to hear them. □



Binary clock

Keep track of time, and learn to count in 1s and 0s

Ben Everard

@ben_everard

Ben's house is slowly being taken over by 3D printers. He plans to solve this by printing an extension, once he gets enough printers.

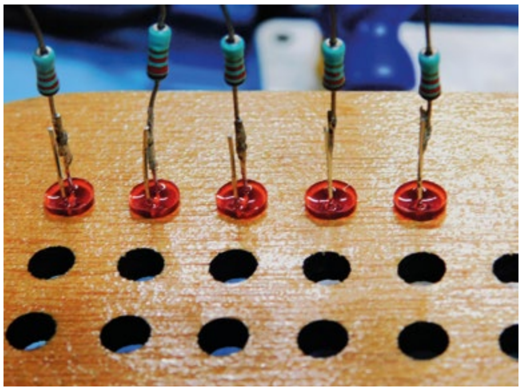
Binary is the key concept in how computers operate. If you drill down far enough on any operation, transmission, or storage, you'll find 1s and 0s, on or off, positive or negative. This ability to build up to complex behaviours from a simple distinction between, and manipulation of two states, is what computing is all about.

Yet, despite this being absolutely fundamental to the way computers work, it can be a bit alien to us humans. We're brought up in a world of ten digits, not just two, so it can be hard to get an intuitive sense of binary, especially if you don't use it very much. Our solution is to make a clock that displays time in binary.

We're going to use MicroPython for this because it's quick and easy to get started wirth, and has access to Pico's timekeeping features. While MicroPython is fairly standard across several boards, this particular project requires having an on-board real-time clock (RTC), so won't work with all boards. We're going to be using a Pico W, but it might work on others that have both wireless networking and RTC hardware. Along the way, we'll learn how to connect Pico W to the internet to get the time, and how to use Pico's Real Time Counter to keep track of the time in hours, minutes, and seconds. Let's take a look at this first.

There's a protocol for getting time on a network, and it's unimaginatively named Network Time Protocol (NTP). This actually does a few things around synchronising time over a network with various degrees of lag, but we don't need to worry about being a few milliseconds out, so we're just going to grab the current time.

We're also not going to worry about daylight savings time. We're doing all this in MicroPython, so we're going to take the simplest solution – twice a



year, you'll connect to your clock and adjust the offset manually. If you want, you could add a toggle switch that flips back and forwards to adjust for daylight savings.

We suspected that setting the time on an RTC via NTP might be a popular thing to do, so we wrapped up the example code for NTP in a simple library that does just this – you can get it from **hsmag.cc/SimpleNTP**.

You'll need to copy the **simple_ntp.py** file over to your device – you can do this by opening it in Thonny, then selecting 'Save as' and MicroPython device. Remember, it must have the same name on the device.

We'll shortly look at what's in that file, but for now, we'll use it to grab the time and set the RTC. You can do this with the following code:

```
import network
import socket
import time
import struct
from machine import Pin
import simple_ntp

led = Pin("LED", Pin.OUT)

ssid = 'your_ssid'
password = 'your_password'


wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect(ssid, password)

max_wait = 10
while max_wait > 0:
    if wlan.status() < 0 or wlan.status() >= 3:
        break
    max_wait -= 1
    print('waiting for connection...')
    time.sleep(1)
```

```
if wlan.status() != 3:
    raise RuntimeError('network connection
failed')
else:
    print('connected')
    status = wlan.ifconfig()
    print( 'ip = ' + status[0] )

led.on()
simple_ntp.set_time()
print(time.localtime())
led.off()
```

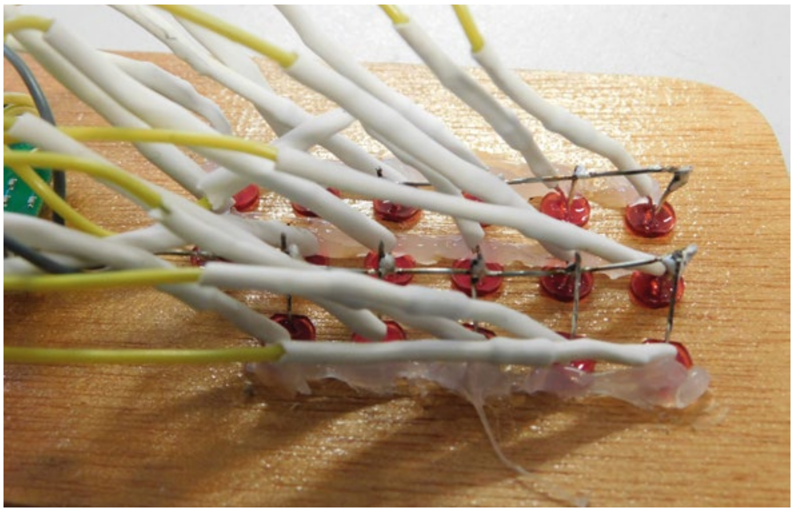
Obviously, you'll need to put in your network's SSID and password, but other than that, it should just run. At the end of it, it'll print the result of `time.localtime()`. Firstly, we should note that this is a misleading name as it won't print your local time, it'll print the time in GMT. Secondly, it won't actually print the time in a usual format, it'll print it as a list. The items are: [year, month, monthday, hour, minute, second, weekday, yearday]. All of these should be set correctly. →

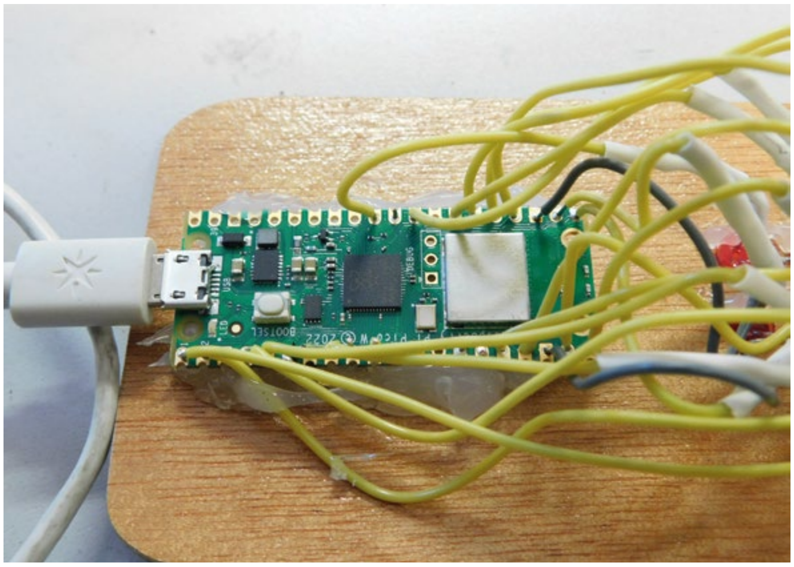
DIMMING

Our program turns the LEDs all the way on or all the way off. This is fine for us, and we can control the brightness by picking the right resistor. However, you can control the brightness of LEDs by flicking them on and off really fast. This is known as pulse width modulation (PWM). Pico W does support PWM but is limited to just 16 pins, so we can't use it in this project. We could create a PIO program that took in details of all the pins that should be on, and do a global PWM across all of them, but this is more than we need.

Left
Soldering resistors in-line is an easy way to build circuits without a circuit board

Below
The ground wires connect across all the LEDs





Above Pico W is a great choice for this because it's cheap, has a lot of GPIOs, has WiFi for time, and has an RTC

This uses `simple_ntp.set_time()` to set Pico W's RTC to the correct time. We won't go through the whole of this, but let's look at a few key bits. The function is defined with:

```
def set_time(offset=0, delta=2208988800,
host="pool.ntp.org"):
```

There are three optional parameters here. The first is `offset`, which is a number in seconds that you want to set your RTC's time to – this is useful if you want to set it to a particular time zone. The second is `delta`, which you can probably ignore most of the time, but it's the difference between the way time is encoded in NTP. The final parameter is the NTP server. Again, `pool.ntp.org` is probably a good choice for the vast majority of cases. This isn't a specific server but a domain that will resolve to an NTP server close to your location. By being close to you, it should (in theory at least) have a low latency and, therefore, you'll get a more accurate time. However, for our purposes, the difference between high- and low-latency servers should be irrelevant.

Now we've got our time, we will need to create the hardware to display it. Here, it's up to you what you want to do. We obviously need a lot of LEDs, but how you mount them depends on what you have and what look you want to create. You could create a bare-bones look on perfboard. You could design a PCB for it. You could get some wood and drill holes to mount the LEDs in. Or, you could 3D-print a front panel for it. We opted to laser-cut ours because we wanted a wooden front panel, but any of these options should be reasonably straightforward.

Our laser-cut panel has holes for 5 mm LEDs in a line, so you can push the LEDs in place and secure them with a drop of hot glue.

The circuit is quite simple because the output is in binary, and our GPIOs work in binary. Each GPIO connects to a current-limiting resistor, then to the positive leg of an LED (usually the longer leg), and the negative leg of the resistor connects to ground. There are more LEDs than ground connections available, so you'll need to connect them all together.

The resistors need to be at least 220 Ω, but you might want to make them higher to limit the brightness of the LEDs. This will depend a lot on the particular LEDs you have, and how bright you want them. We found that around 1 kΩ usually works well, but it's obviously worth experimenting to see what you like.

> "We've used the classic 5 mm red LEDs that give a retro feel, **but 3 mm LEDs will give you a more compact design**"

You can use any LEDs that work with 3.3 V. They can be any colour and any form factor. We've used the classic 5 mm red LEDs that give a retro feel, but 3 mm LEDs will give you a more compact design. You could use surface-mount LEDs if you're making it on a PCB. You could even get creative and use 3 V flex LED 'noodles' to build a more abstract-looking clock.

We need five pins to display the hours (in 24-hour format) and six each for minutes and seconds. That means 17 LEDs in total on the GPIO pins 0 to 16.

We found it easiest to first solder the LEDs and resistors together, then put the LEDs in their holes, and then solder the connectors to Pico W's GPIO pins and ground. One thing you need to be aware of is making sure that there's no risk of short circuits if anything gets bent. We did this by covering the LED positive leg and resistor in heat shrink. That way, all the ground wires are exposed, but it doesn't matter if anything shorts between them. It would perhaps be a bit more secure if the grounds were also protected – it's up to you how you set it up. If you don't have heat

NEW TO MICROPYTHON?

If you've not used MicroPython before, you have an exciting time ahead of you! It's a great way of programming Pico and Pico W with very little setup. It's powerful, and user-friendly. There's a quick guide to getting started at **hsmag.cc/DocMicroPyth**. For a more in-depth guide, take a look at our book *Get Started with MicroPython on Raspberry Pi Pico*, which you can download for free or buy in print at **hsmag.cc/mpbook.**

shrink, you could also cover it with electrical tape, but given how much there is to do, it might be easier to just get some heat shrink.

Now we've done the hardware, it's time to turn our attention back to the software. The main thing we need to do is convert the time as numbers into the LEDs that we want to turn on and off. We've done this in a method called **`display_num`**. This method takes two arguments: the number you want to display, and a list of pins on which to display it.

Converting a number to binary isn't too hard. We've used **`enumerate(pins)`**. This is a useful Python method when you want to loop through an iterable but still want a counter – as you can see, it returns both.

Each loop, we want to know if the number remaining is greater than or equal to the current digit. If it is, we light that digit up and take the value of that digit away from the number, if it's not, we turn that LED off and move to the next number.

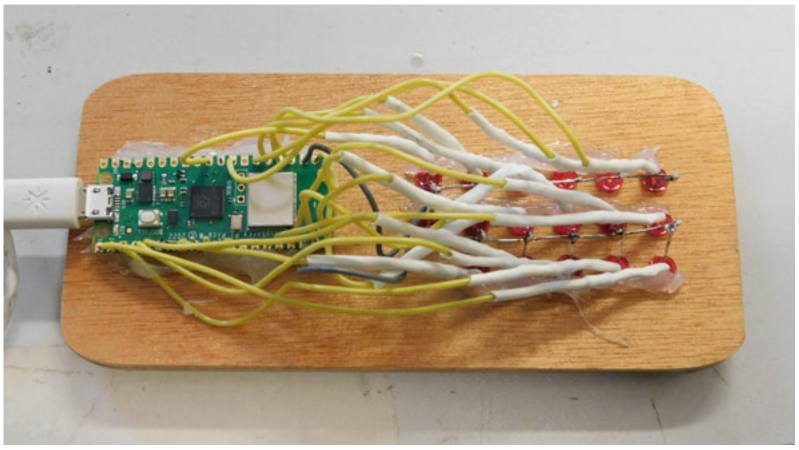
Since Pico W's RTC can keep track of time, we can just grab the time when the computer boots up and let Pico W take care of the rest.

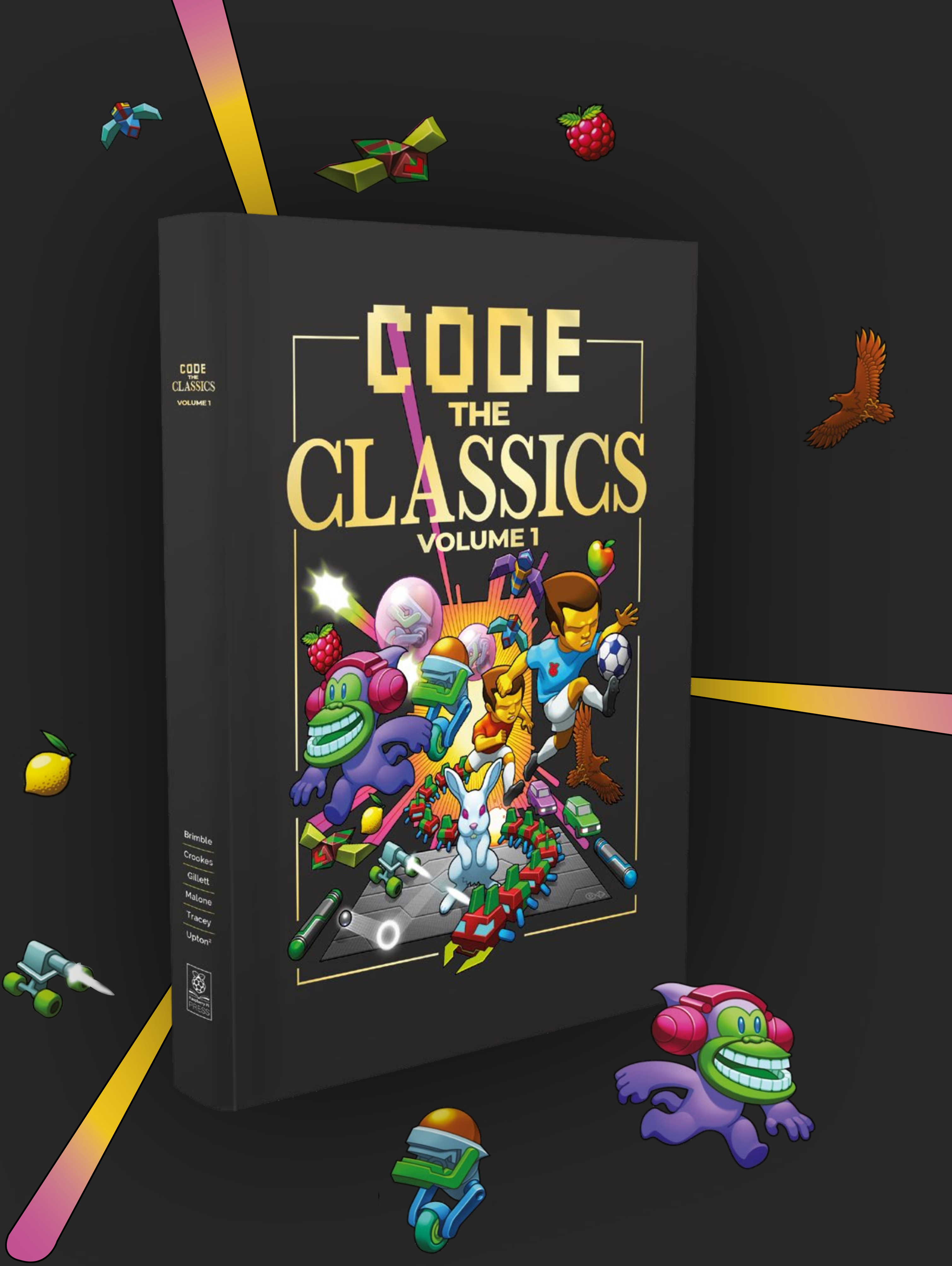
Here's our final code for the clock:

```
import network
import socket
import time
import struct
from machine import Pin
import simple_ntp

led = Pin("LED", Pin.OUT)

ssid = 'your_ssid'
password = 'your_password'

hour_pins =  [Pin(4,Pin.OUT), Pin(3,Pin.OUT),
Pin(2,Pin.OUT), Pin(1,Pin.OUT), Pin(0,Pin.OUT)]
minute_pins = [Pin(17,Pin.OUT),Pin(16,Pin.
OUT),Pin(15,Pin.OUT),Pin(14,Pin.OUT),Pin(13,Pin.
OUT),Pin(12,Pin.OUT)]
second_pins = [Pin(26,Pin.OUT), Pin(22,Pin.
OUT),Pin(21,Pin.OUT), Pin(20,Pin.OUT),Pin(19,Pin.
OUT),Pin(18,Pin.OUT)]

wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect(ssid, password)

max_wait = 10
while max_wait > 0:
    if wlan.status() < 0 or wlan.status() >= 3:
        break
    max_wait -= 1
    print('waiting for connection...')
    time.sleep(1)

if wlan.status() != 3:
    raise RuntimeError('network connection
failed')
else:
    print('connected')
    status = wlan.ifconfig()
    print( 'ip = ' + status[0] )

led.on()
simple_ntp.set_time()
print(time.localtime())
led.off()

def display_num(number, pins):
    for counter, pin in enumerate(pins):
        if number >= pow(2, len(pins)-
(counter+1)):
            number = number - pow(2, len(pins)-
(counter+1))
            pin.value(1)
        else:
            pin.value(0)

while True:
    display_num(time.localtime()[3], hour_pins)
    display_num(time.localtime()[4], minute_pins)
    display_num(time.localtime()[5], second_pins)
    time.sleep(0.1)
```

This is a very bare-bones clock, but you can extend this basic timepiece in any way you like. You could add an alarm, a date function, a time control, or anything else you'd like to keep track of. □

Below
We attached Pico W with hot glue, but you could also use screws in the mounting holes




CODE THE CLASSICS
VOLUME 1
Brimble
Crookes
Gillett
Malone
Tracey
Upton²
Raspberry Pi PRESS
CODE
THE
CLASSICS
VOLUME 1

Part 02

Pico W IoT with Anvil: Weather dashboard

Send temperature and humidity sensor readings to display in an Anvil web app

MAKER

Phil King

Long-time contributor to *The MagPi*, Phil is a freelance writer and editor with a focus on technology.

@philkingeditor

Anvil's Pico W IoT toolkit enables you to connect easily, and securely, to web apps you create. Using the Anvil UF2 firmware image, you can code programs on Pico W in MicroPython, with a few extra lines to connect to Anvil, then create a web app on the Anvil site which can send and receive data to and from your Pico W program.

Last issue, we made an RGB LED mood lamp whose colour was controllable by moving sliders in an Anvil web app. That involved sending data to Pico W; this time we'll explore how to send data from Pico W to a web app dashboard to show temperature and relative humidity readings from a DHT11 sensor.

01 Install the firmware

If you already have a Pico W with the Anvil firmware installed and connected to your wireless network, you can skip the first two steps of this guide.

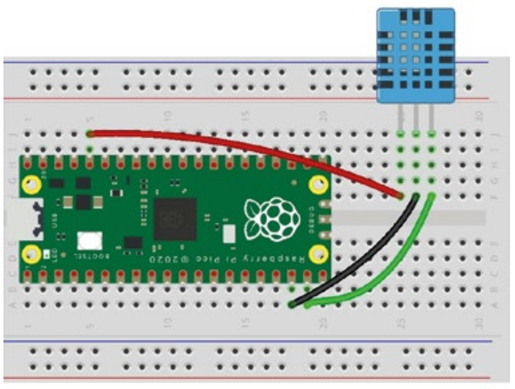


Figure 1 The wiring diagram for the circuit using a DHT11 temperature and humidity sensor

To be able to link your Pico W to the Anvil framework, you'll need to use Anvil's special firmware file. Go to **magpi.cc/anviluf2** and download the latest 'complete' UF2 file. (If you want to avoid overwriting any existing files on Pico W, use the 'firmware-only' version.)

As usual, hold Pico W's BOOTSEL button while connecting it to a computer via USB, then drag the UF2 file to the mounted 'RPI-RP2' volume. Once it's copied across, Pico W will automatically reboot and reappear as a volume called 'Pico W'.

02 Connect to WiFi

With Pico W connected to your computer, open up the Thonny IDE and make sure the Python interpreter (shown at the bottom right) is set to 'MicroPython (Raspberry Pi Pico)'. The **main.py** file on Pico W will run automatically, so you will need to stop it by pressing the Stop icon. Open the **boot.py** file and enter your wireless router's SSID (name) and password at the top.

```
WIFI_SSID = "<put your network name here>"
WIFI_PASSWORD = "<put your WiFi password here>"
```

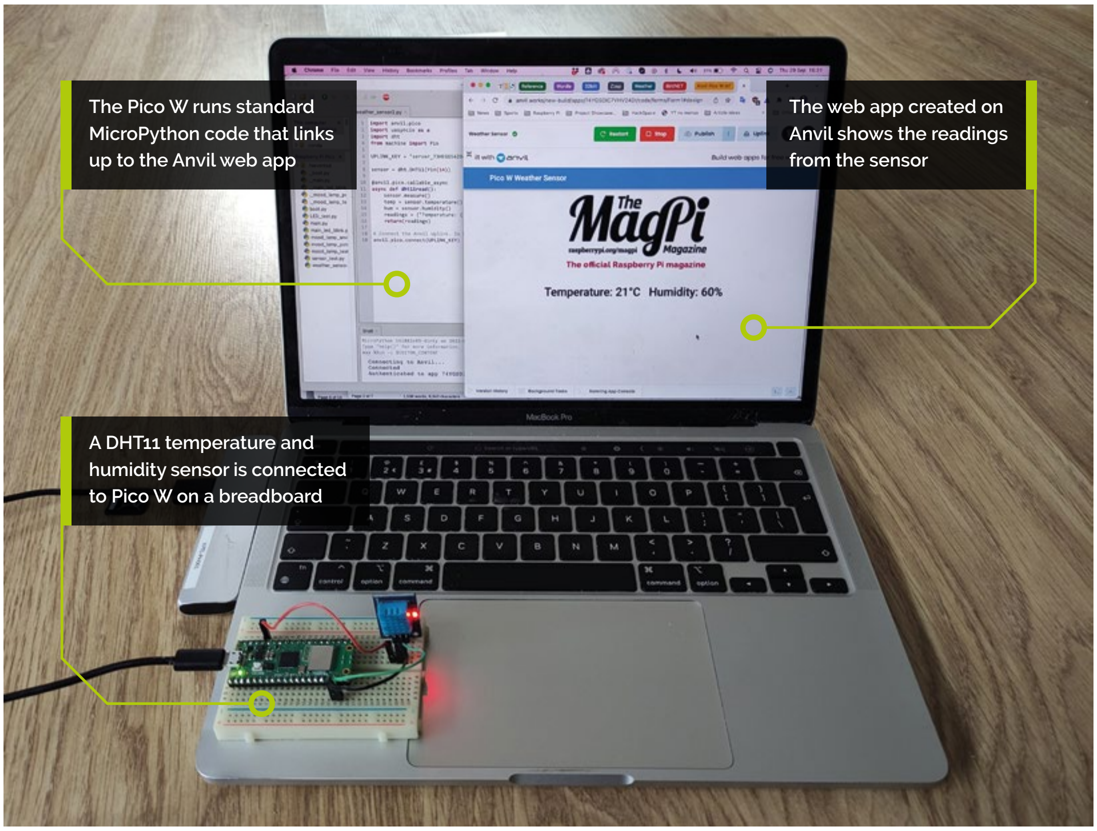
Now when Pico W reboots, it'll automatically connect to your wireless network.

03 Wire up the circuit

Place your Pico W on one end of a breadboard, as shown in the **Figure 1** diagram. The DHT11 sensor combines a digital thermistor with a capacitive humidity sensor and outputs digital readings, so it's easy to use. With the power turned off, connect the DHT11 sensor to Pico W

You'll Need

- Raspberry Pi Pico W
- Anvil account (free tier) **anvil.works**
- DHT11 sensor **magpi.cc/dht11**
- 3 × male-male jumper wires


The Pico W runs standard MicroPython code that links up to the Anvil web app
The web app created on Anvil shows the readings from the sensor
A DHT11 temperature and humidity sensor is connected to Pico W on a breadboard
Temperature: 21°C Humidity: 60%

THE MAGPI

This tutorial is from in The MagPi, the official Raspberry Pi magazine. Each issue includes a huge variety of projects, tutorials, tips and tricks to help you get the most out of your Raspberry Pi. Find out more at **magpi.cc**

as in the diagram: power is supplied to its VCC by Pico W's 3V3 pin, GND is wired to GND, while the sensor's DOUT (digital out) pin is connected to GPIO 14 (you could use any GPIO pin).

04 Test the sensor

To make sure everything is connected correctly, we'll run a simple program on Pico W. Connect it to your computer via micro-USB, then open the Thonny IDE. Make sure the Python interpreter (shown at the bottom right) is set to 'MicroPython (Raspberry Pi Pico)'.

Open a new file and add the code from the **sensor_test.py** listing. At the top, we import the `dht` module for reading our DHT11 sensor, and the `Pin` class from the machine library so we can read the sensor from a GPIO pin. We create a variable, called `sensor`, and set it to take a reading from the sensor connected to GPIO 14.

```
sensor = dht.DHT11(Pin(14))
```

The `sensor.measure()` command is used to take a sensor reading. We then create variables for

"The DHT11 sensor combines a digital thermistor with a capacitive humidity sensor and outputs digital readings"

temperature and humidity and set them to the sensor values we just read. Since the DHT11's accuracy is 1°C and 1% humidity, there's no need to round the values.

```
temp = sensor.temperature()
hum = sensor.humidity()
```

To show the values in a more easily readable format, we turn them into a string to display:

```
readings = ("Temperature: {}°C   Humidity: {}% ".format(temp, hum))
print(readings)
```

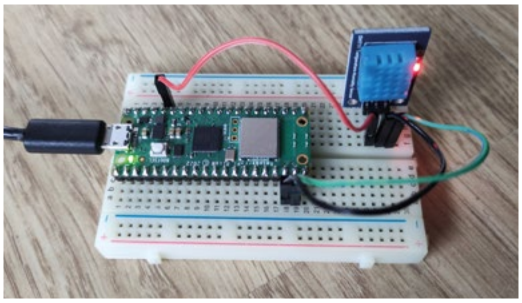
Run the code and you should see the temperature and humidity readings from the sensor in the Shell pane of Thonny. →

Top Tip

DHT11 pins

The pin order of different DHT11 sensors varies, so double-check the labels on the board when wiring it to Pico W.

The DHT11 sensor measures both temperature and relative humidity



05 Write the code

Now let's write the MicroPython code for our Anvil-connected weather sensor, as in the **weather_sensor.py** listing. The top line, `import anvil.pico`, enables Pico W to connect to Anvil's servers. The second, `import uasyncio as a`, sets up an asynchronous scheduler for running concurrent functions. We import the `dht` module and `Pin` class from machine, as in our sensor test code.

Further down is a line starting `UPLINK_KEY =`. This is where you will later need to paste the uplink key for the web app you create on Anvil so your Pico W program can link to it. At the bottom of the program is a line to connect using the key.

As in our test code, we create a `sensor` variable to take a reading from the DHT11 sensor connected to GPIO 14.

Top Tip

Publish your app

You can make your web app public by clicking the Publish button and sharing the private link there, or switch to a public link.

06 Callable function

This time, we'll create a function that will be called from the Anvil web app every few seconds. At the top, we add a 'decorator', starting with `@`, and also add `async` at the start of the line defining the function:

```
@anvil.pico.callable_async
async def dht11read():
```

This lets our connected Anvil web app know that this function is available to call from the web app. It's similar to our test function, but we add the line `return(readings)` to send the sensor data from Pico W to the web app.

07 Design the web app

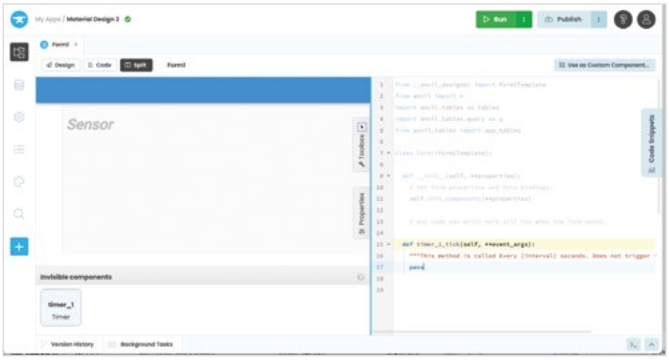
If you haven't done so already, point your web browser to **anvil.works** and sign up for a free account. Create a new blank app and select a theme – we chose the legacy Material Design.

From the right-hand Toolbox panel, drag a 'Display label – text' component over to the main panel of the app. This will be used to show our DHT11 sensor temperature and humidity readings. In the Properties panel, change its name to 'Sensor' – it will change itself to 'self.Sensor'. You can also set the font and size of the text in Properties.

08 Add a timer

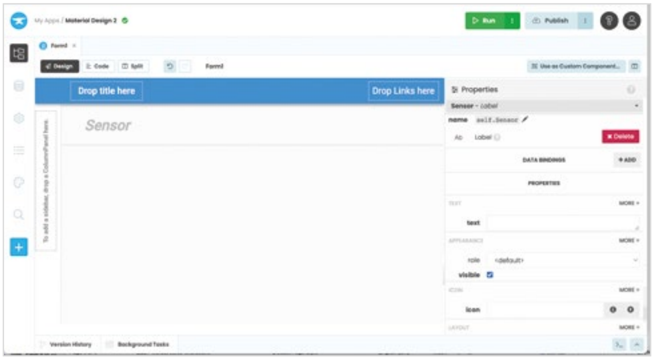
While we could add a button to the web app to tell Pico W to run the function to send the sensor data, this wouldn't be very user-friendly. Instead, we'll add a timer function to do it. In the Toolbox panel, click 'See more components' and then drag a Timer component below the Sensor label field. This is an invisible component, so won't show on the finished dashboard. In the Properties panel, change the timer's Interval value to 2 (seconds).

> "The Timer is an invisible component, so won't show on the finished dashboard"

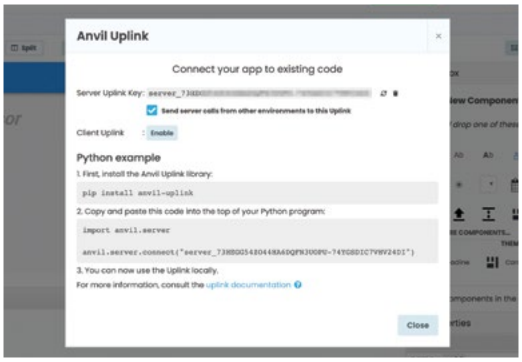


```
from ._anvil_designer import Form1Template
from anvil import *
import anvil.tables as tables
import anvil.tables.query as q
from anvil.tables import app_tables

class Form1(Form1Template):

  def __init__(self, **properties):
    # Set Form properties and Data Bindings.
    self.init_components(**properties)

    # Any code you write here will run when the form opens.

  def timer_1_tick(self, **event_args):
    """This method is called Every [interval] seconds. Does not trigger
    pass
```

You need to add a couple of lines to the code for the invisible timer element



Changing the name of the text field we'll use to show the readings



Anvil Uplink

Connect your app to existing code

Server Uplink Key: server_7JHB[redacted]

✓ Send server calls from other environments to this Uplink

Client Uplink : Enable

Python example

1. First, install the Anvil Uplink library:

```
pip install anvil-uplink
```

2. Copy and paste this code into the top of your Python program:

```
import anvil.server

anvil.server.connect("server_7JHBGG54Z044RA6DQFN3UOPU-74YGSDIC7VWV24DI")
```

3. You can now use the Uplink locally.

For more information, consult the uplink documentation

▲ Enable Server Uplink for your web app and then copy the key to paste into your Pico W program

Double-click the Timer element to bring up its code in a split-screen view. Before `pass`, add the following two lines (make sure they're indented):

```
data = anvil.server.call_s("dht11read")
self.Sensor.text = reading
```

The first line calls the `dht11read` function in our Pico W code, causing it to take a sensor reading. The second line sets the text of our 'self.Sensor' web app field to the reading string created by the function, to show temperature and humidity.

09 Enable server uplink

Now to link our web app to our Pico W program. Click the '+' button at the bottom left and select Uplink, then click Enable Server Uplink. This will generate a Server Uplink Key for the app which you should paste into line 6 of your MicroPython code on Pico W so it can connect to it.

10 Run both apps

First, run your Pico W program in Thonny. You'll see messages in the Shell pane to show it connecting to Anvil. Once it has, run your Anvil web app. You should now see the temperature and humidity readings appear in its text field. To make your web app look a little more interesting, you can add a title and maybe a graphic (by pasting an image URL into the Source field). ◻

Top Tip

Autorun code

To make your Pico W code run automatically upon bootup, save it as **main.py**. If you already have a **main.py** file, you may want to save that under another name first, to keep it.

sensor_test.py

DOWNLOAD THE FULL CODE: magpi.cc/github

▶ Language: **MicroPython**

```
import dht
from machine import Pin

# Set GPIO pin for sensor
sensor = dht.DHT11(Pin(14))

# Take sensor readings
sensor.measure()
temp = sensor.temperature()
hum = sensor.humidity()

# Turn readings into a string to display
readings = ("Temperature: {}°C   Humidity: {}%
".format(temp, hum))
print(readings)
```

weather_sensor.py

▶ Language: **MicroPython**

```
001.  import anvil.pico
002.  import uasyncio as a
003.  import dht
004.  from machine import Pin
005.
006.  UPLINK_KEY = "<put your Uplink key here>"
007.
008.  # Set GPIO pin for sensor
009.  sensor = dht.DHT11(Pin(14))
010.
011.  # Callable function for sensor readings
012.  @anvil.pico.callable_async
013.  async def dht11read():
014.      sensor.measure()
015.      temp = sensor.temperature()
016.      hum = sensor.humidity()
017.      readings = ("Temperature: {}°C   Humidity: {}%
      ".format(temp, hum))
018.      return(readings)
019.
020.  # Connect the Anvil Uplink. In MicroPython, this call will
      block forever.
021.  anvil.pico.connect(UPLINK_KEY)
```

Learn to solder

Get started with this essential maker skill

Ben Everard

@ben_everard

Ben's house is slowly being taken over by 3D printers. He plans to solve this by printing an extension, once he gets enough printers.

Having looked at a lot of 'learn to' solder tutorials recently, we actually think that there's a bit of a problem with them. It's not that they have incorrect information (though some do), it's that they're so keen to tell you how quick and easy soldering is, they miss out some of the key information you need to know and, in doing so, actually make it much harder for you to get consistent results. Here's our no holds barred guide to how to actually solder.

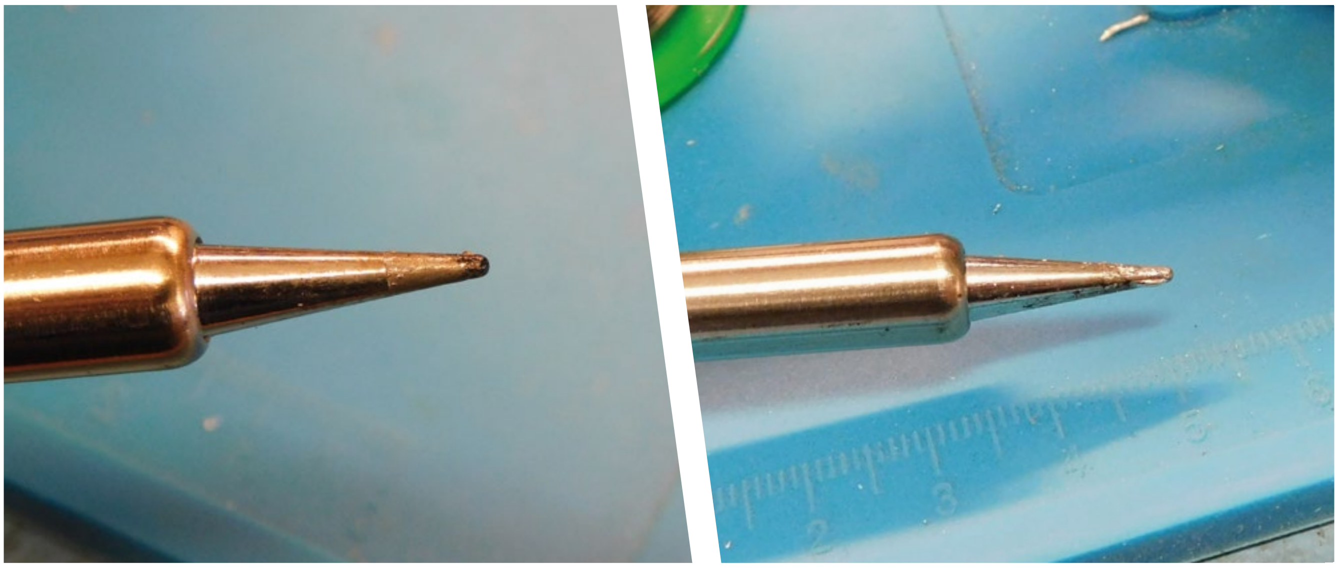
First, before you do anything else, inspect your iron. The soldering iron is the key to good soldering. The better condition your iron is in, the better you'll be able to solder and, when it comes to soldering, cleanliness is the key. As you solder, you'll build up some black gunk on your iron and this doesn't conduct heat very well and makes it hard to solder. You might find that there's a load of gunk on it before you even start. Perhaps the most important skill in soldering is keeping your soldering tip shiny.

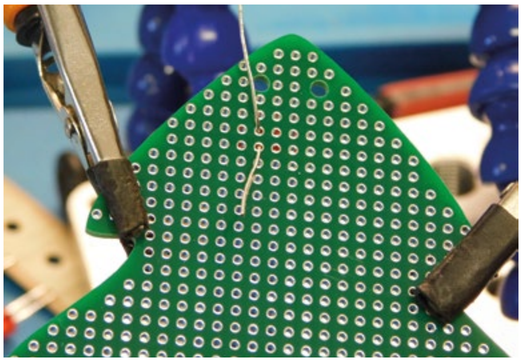
You'll need something to keep your iron clean, and there are three options for this:

- **Wet and flick**. This technique is very much a bodge. Put some solder on the iron and 'flick' it or tap it. The flux in the solder will dissolve the black gunk a bit and, when you flick it, the solder will dislodge it and will come off. Well, some of it will. There are a few problems with this; firstly, it's simply not very effective at cleaning the iron. You'll probably get a bit of the gunk off, but not much. Secondly, you're flicking molten metal about. Sooner or later you're going to end up hurting yourself or somebody else. Thirdly, your soldering iron has a delicate heating element that doesn't like being flicked and you could break it. We'd strongly discourage this method.

- **Damp sponge.** Many soldering irons come with a damp sponge to clean them. Soak them in water, then squeeze out at much as you can, and then you can wipe a cold or hot iron over them to clean it. This is probably the most common method of cleaning your iron. However, we'd actually recommend against this method. There are a few problems with it, but our two main

Below The tip, oxidised and then cleaned with a drop of solder and a few pokes in a brass sponge. It took us a bit longer to set up the photo than we'd like, so it's just starting to tarnish in the clean photo





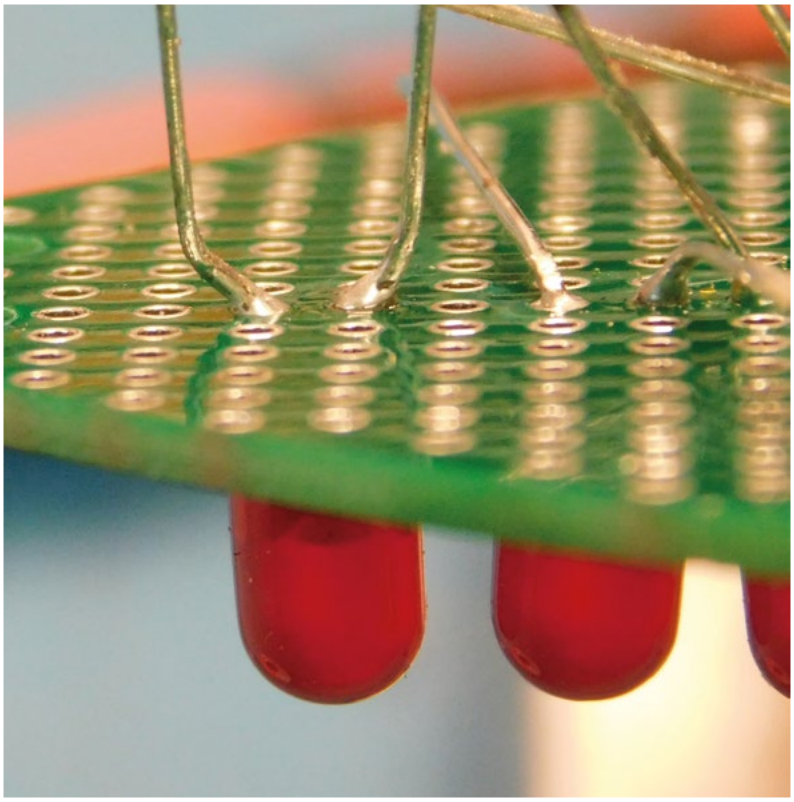
Above A through-hole LED, with wires bent, to hold it in place while soldering

issues are that it's too good and that it cools down the tip. The first might not seem like a problem, but you don't actually want your tip completely clean, or rather, you don't want to wipe all the liquid off the tip. You want to remove the black gunk while leaving a small amount of solder on the tip, as this will prevent the tip from oxidizing, but the sponge will remove everything. Cooling the iron down then means you have to wait for it to heat back up before soldering again. Some irons will tell you when they're back up to temperature, but many cheaper ones don't. If you're an experienced solderer, you can tell when an iron is at temperature from the way the solder behaves, but if you're not, it's hard to know when to start soldering.

- **Brass sponge.** We don't know why these are called sponges, but they are. They're tangles of brass shavings. Poke your soldering iron in a few times and they will dislodge the black gunk, while still leaving a thin layer of solder coating your iron. You can get them for a couple of pounds online, and they're probably the single thing that will help your soldering the most. This, in our view, is the best way of keeping your tips clean.

PICKING YOUR TIP

There's a vast range of different tip shapes available, but almost all soldering irons come with conical ones, so that's the shape we'll be using. The same basic technique works with all shapes, though. Different shapes fit better into different joints and the better the tip fits in, the more effectively it can transfer heat.



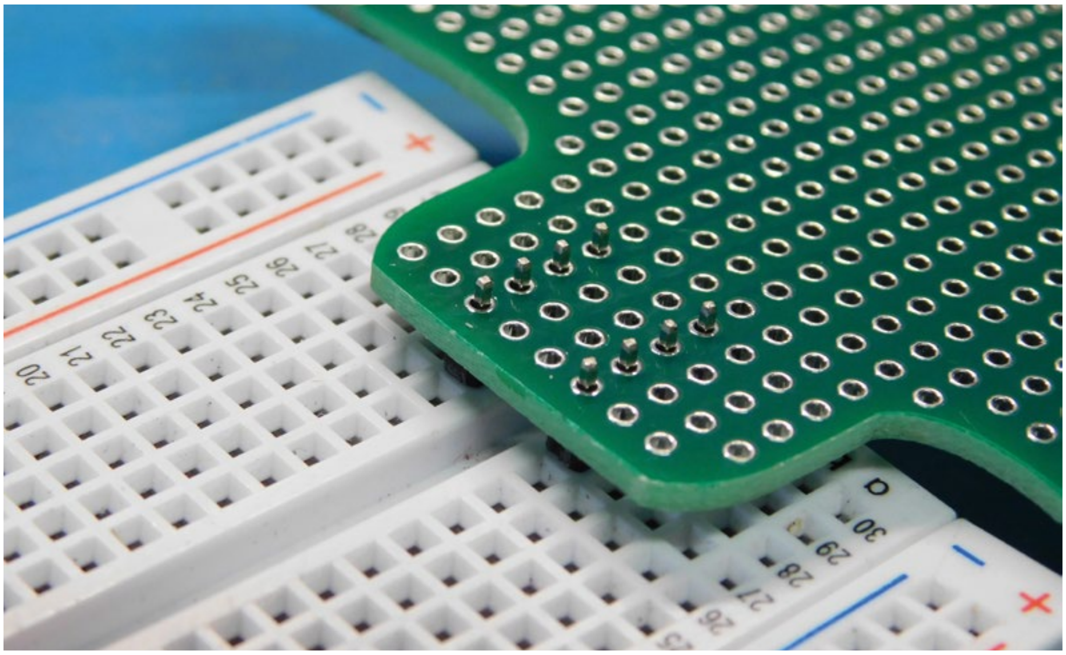
Above Some joints with about the right amount of solder on them

"**Leaded is easier to solder with, and lead-free is** better for the environment, and possibly your health"

PICK YOUR WEAPON

There are many types of solder available, so first, let's narrow it down and say we're only going to talk about flux-cored electronics solder. Don't get solder designed for jewellery making or plumbing – it will only lead to problems. We're also going to be talking about solder wire. It's also possible to get solder paste for electronics, but this really serves a different purpose.

The first distinction to make is leaded or lead-free. Basically, it comes down to the fact that leaded is easier to solder with, and lead-free is better for the environment, and possibly your health. If you're soldering commercially, there may be legal restrictions on what you can use. Many hobbyists use leaded solder and don't have any health problems. However, this author has a young family →

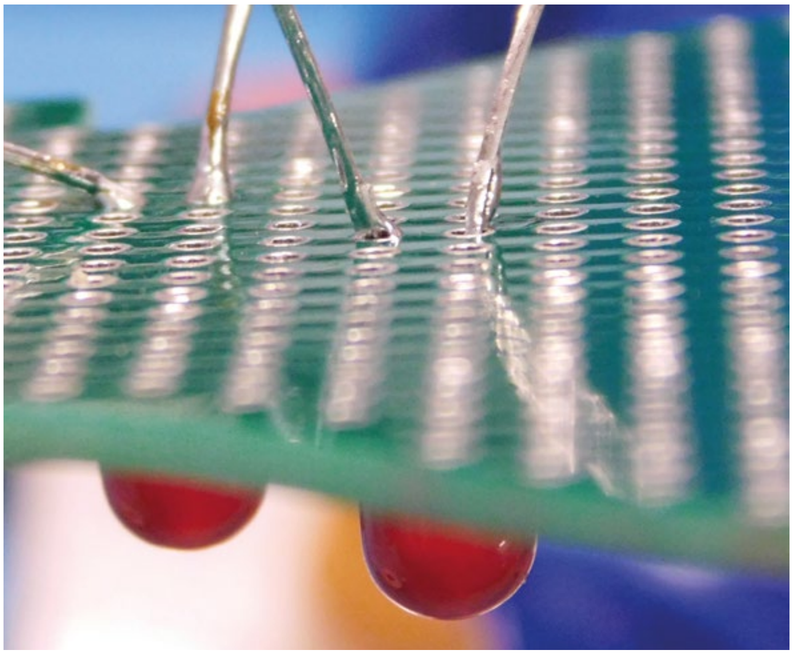


Right You can use a breadboard to hold the headers in the right position while you solder them on. You might have to support the other end of the PCB if it won't stay flat

Below On the right-most two joints, you can see there's not enough solder. On the furthest right joint, it's not even flowed onto the PCB

FLUX

In this article, we've looked at using flux-cored solder. You can also get additional flux in pens and paste. You shouldn't need it for the type of soldering we're doing here, but equally, it won't do any harm if you do use it.



and lead is a particular risk for children. He uses lead-free exclusively, because it's not that hard to solder with, and it's not worth the risk. You'll have to make up your own mind.

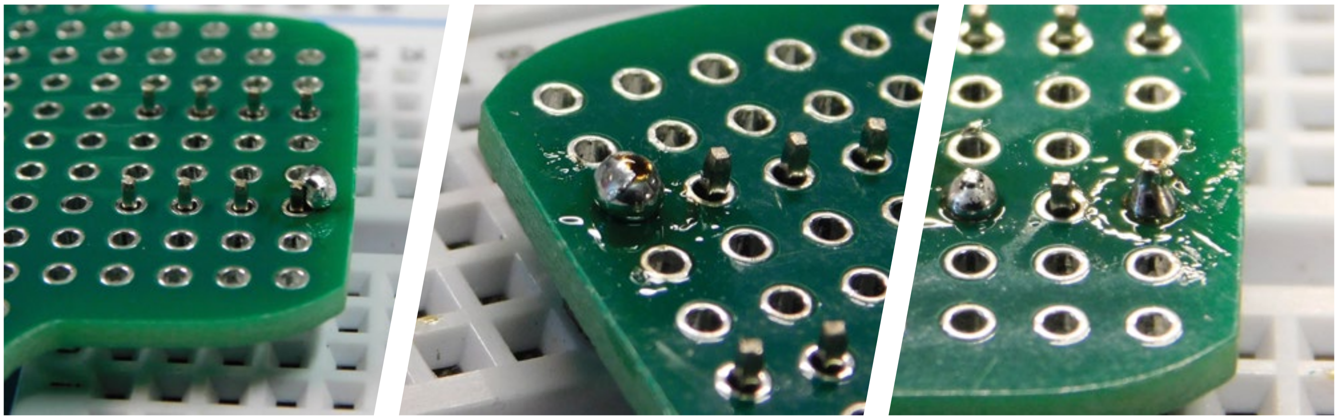
If you chose leaded solder, there are basically two types commonly available: 60/40 and 63/37. This refers to the percentage of tin and lead, respectively. We would strongly recommend using 63/37. If you have some 60/40 already, then you might as well use it, but 63/37 is much less likely to cause problems.

If you chose lead-free solder, then there's all manner of different types available, but we'd split them into two categories – those with silver and those without. (The other ingredients are mostly tin and copper, but there might be some others as well). Solder with silver flows much better than solder without but, unsurprisingly, it's more expensive. The ingredients are usually given by their chemical symbols, so the ones to look out for are Sn (tin), Cu (copper), and Ag (silver).

Solder wire also comes in different diameters. Basically, thinner wire (around 0.5 mm) is good for smaller surface-mount components, while thicker wire (1 mm plus) is good for through-hole. That said, you can use either wire for either and it's not a big deal. We use 0.5 mm wire for everything.

FEEL THE HEAT

Now we know how to clean our iron and have picked our solder, it's time to get to work.



The first thing we need to know is how hot to set our iron. Some irons don't have a temperature gauge, in which case, don't worry about it, just switch them on and wait for them to heat up. A lot of people set their irons to very different temperatures, and in truth, you can develop a technique that works for a very hot or very cold iron, but let's look at the starting points. Leaded solder – 290°C / 550°F, lead-free solder 330°C / 575°F. You can increase these temperatures if you like, but we would recommend staying under 400°C/750°F. Once you go above this, your tips will start to deteriorate quickly.

"Solder with silver flows much better than solder without but, **unsurprisingly, it's more expensive**"

We're going to look at through-hole soldering in this tutorial; that is, soldering where the component has a lead or pin that you poke through a hole in the PCB to solder it. This is the traditional place to learn, and many beginner kits use components like this. The alternative is surface-mount soldering, where components sit on top of the PCB and attach to pads. This requires a slightly different technique that we'll look at in a future article.

First, poke the legs through the hole in the PCB. If they're long legs, you can then bend them to hold the component in place. If it's something like a chip, you can use a bit of Blu Tack on the component side to keep the bits in place. If it's something like header pins on a dev board, you can pop the pins in a breadboard and then put the dev board on top of them, and this will help keep everything aligned. However you've got everything lined up, it's time to start soldering.

Inspect your soldering tip. Is it clean? Does it need a clean? Usually, the best option is to melt a bit of solder on the tip and poke it in the brass sponge a few times. It should come out shiny and clean. If it doesn't, take a look at the 'Tip refreshing' box before continuing.

Once you've got a clean tip, melt just a little bit of solder on the tip – not a lot. Let's pause here to talk a bit about what we're trying to do, and how we can help it. Soldering is all about heat transfer, and →

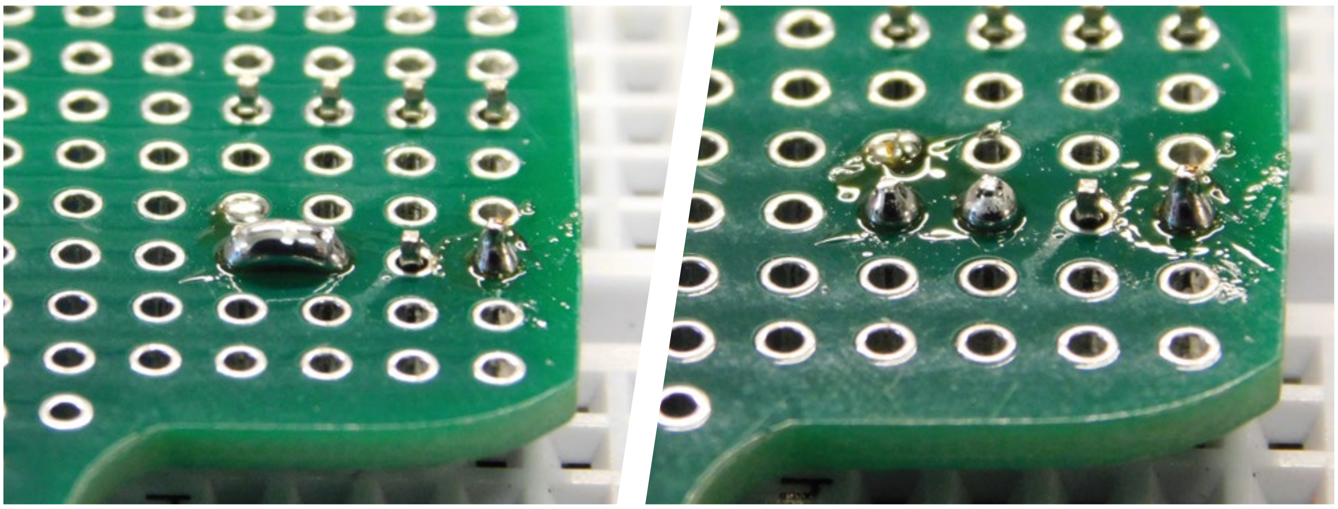
TIP REFRESHING

If you can't get your soldering iron clean using a quick scrub with a brass sponge, then it's time to get more serious. The first thing you can try is a tin of tip refresher. This is a pot of tin mixed with some cleaning chemicals. Put the tip in the pot for a few seconds, then give it a run through the brass sponge. You might have to do this a few times.

Soldering iron tips are usually made of copper because it is an excellent conductor of heat. However, copper dissolves in molten tin, so these tips have to be coated to protect this core. Typically, soldering iron tips are coated in iron. If you scratch through this iron coating, the tip will dissolve surprisingly quickly.

If you find that you can't get the tip clean with tip refresher and a brass sponge, you can try more drastic action, but bear in mind that you don't want to damage the iron coating. Some soldering iron manufacturers recommend using a fine emery paper. Some sell brass brushes attached to motors for cleaning (which you could probably replicate with a brass wheel and drill or rotary tool). Obviously there are risks to these methods, but if you'd have to dispose of a tip because you can't get it clean, you might as well try.

Above
The solder hasn't flowed properly on this pin. We fixed it by heating up the pin again and adding a drop more solder. This left the joint with too much solder. It would have been fine to leave it like this as too much solder isn't really much of a problem, but we used a solder sucker to get rid of the excess



Above Too much solder can cause pins to join together. We split them apart by running a soldering iron between the two pins. There's a slight excess of solder here, but not enough to cause any problems

managing this effectively. To solder well, you need to heat up three things – the solder, and the two things you're joining. However, to get the solder to work effectively, the two surfaces you're trying to join have to be clean. Most of the time, the flux in the core of your solder will be perfectly capable of doing this, but we need to make sure that it doesn't burn off before it has a chance to do its job.

SAFETY

There are a few simple rules to keep yourself safe when soldering:

- Wash your hands. If you're using leaded solder, this is hopefully obvious, but even if you're using lead-free solder, you should follow basic hygiene. Flux contains stuff you don't want to ingest.
- Don't breathe the smoke. Actually, this isn't just for soldering, it's just a rule for life – all smoke is pretty bad for you. When preparing to solder, think about your physical setup. If you've put the thing you're soldering on a table, you might end up with your head directly over it, with the smoke going straight in your face. If possible, raise the workpiece up. If you can work closer to eye level, with your head to the side, the smoke will naturally miss your face. You can simply get in the habit of blowing on the joint as you solder. Ideally, though, you should use some form of fume extractor, especially if you find yourself soldering a lot.

We once asked a makerspace manager why they insisted children wear goggles when soldering – apparently it's not unheard of for young people to absent-mindedly scratch their faces, or eyes, while holding hot irons. Bear that in mind if working with younger makers.

We add a little bit of solder to the soldering iron tip first because this aids the heat transfer. If you just put a bare soldering iron against a piece of metal, only a tiny bit of the soldering iron will touch the metal, and therefore only a small amount of heat can transfer. However, if you add a tiny bit of solder to the iron first, then this solder will envelop the thing that we are soldering and allow much more heat to transfer. However, the flux that was in this solder will have burned off before it gets to the joint, so we need to add a bit more solder once we're at the joint, and the flux in this will clean the joint and let everything stick. However, we don't want to end up with too much solder in the joint; so, at the start, we're looking to add the smallest amount possible to help the heat transfer work. This means that you're looking to have a bulge of solder on the side of the soldering iron, but only a slight bulge.

Now we need to use that slight bulge of solder on the soldering iron to heat up the two parts we're joining. That means that both of them should be in contact with the solder on the iron if possible.

People often talk about getting a feel for soldering, and it's this part that they're usually referring to. With experience, you'll learn about the angle and position you need to get your soldering iron in to get heat into the joint. Unfortunately, it's quite dependent on the specific soldering iron tip shape (even slight differences in a cone tip can have an effect), and the type of soldering you're doing, so it's something you just have to experiment a bit with yourself to find what works for you.

Now it's time to add solder to the joint. This is probably the bit where most people go wrong. If you just poke solder in blindly, you'll probably end up with a big blob of solder on one of the parts to be soldered. This typically happens if everything hasn't got hot enough.

If everything is right, you can heat the whole joint up for a second or two, then put the solder against one of the components that you're soldering and it'll just flow into the joint and everything will be fine. Another option is to touch the solder wire to the small amount of solder already in the joint from the iron, and use the wire to move the puddle through the joint in an almost dragging motion. As you pull it across the joint, you'll be forcing it into contact with more of the component in the joint and this will help the heat transfer across.

A critical part of this is to stop adding more solder when there is enough in there. The final part of soldering is known as the 'soak'. This is where you leave the soldering iron in the joint for a few seconds after you've removed the solder wire. This lets the solder in the joint flow. This 'flow' can be quite quick. You'll often see the solder very suddenly disperse into the joint.

Soldering is temperature-critical. You need to get everything hot enough that the solder flows, but not so hot that you damage components or the PCB itself. Most of the things you'll be soldering aren't particularly temperature-sensitive, so we'd urge you to not worry too much about adding heat. Far more problems are caused by joints being too cold than too hot. That said, if you find yourself holding the iron on the joint for more than ten or so seconds and things aren't working, it might be a good idea to take the iron away from the joint, give it a few seconds to cool down, and then try again.

At this point you can take the soldering iron away, but be very careful not to move the components for several seconds while they cool down. Then it's time to inspect the joint to see what it's like.

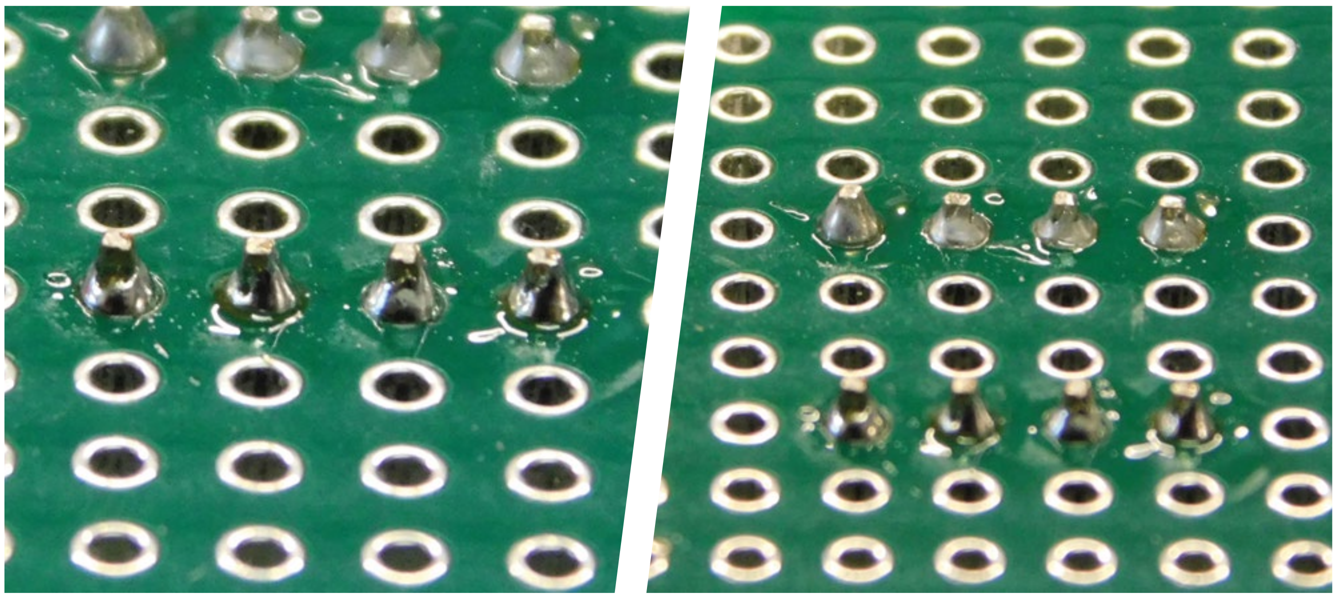
If the solder has flowed properly, it should go all the way through the hole and be visible from both sides. It should also flow up the lead of the component slightly. Where solder has stuck to a surface properly, it creeps up it. Where it hasn't, it tends to ball.

"**You need to get everything hot enough that the solder flows,** but not so hot that you damage components"

That's all there is to it. Soldering isn't especially difficult, but does require you to look closely at what's happening in the joint, and perhaps make adjustments to your technique if it's not quite working.

Our final words on the topic will be our top tip for effective soldering: make sure your tip is clean! □

Below
Solder that has just tin and copper (left bottom) doesn't flow quite as well as solder which is tin, copper, and silver (right top), but it's fine for most purposes and is a lot cheaper



Control your mouse by playing the trombone

Play the game Trombone Champ (or control your mouse) with an authentic trombone action

Rob Miles

@robmiles

Rob Miles has been playing with hardware and software since almost before there was hardware and software. You can find out more about his so-called life at **robmiles.com** and follow him on Twitter at **@robmiles.**

Trombone Champ is an excellent music game (hsmag.cc/TromboneChamp). Figure 1 shows the game in action. You toot your way through tunes old and new, trying to get a perfect pitch from your trusty trombone. It is one of those rare games where playing badly is at least as amusing as playing well. The game is controlled using the mouse. But what if you want some real trombone action? In this article, we are going to build a Raspberry Pi Pico-powered trombone controller programmed with CircuitPython that uses a laser rangefinder and an air pressure sensor to get an authentic trombone slide action from a couple of pieces of drainpipe.

TROMBONE ANATOMY

The pitch of a note played by a trombone is controlled by changing the length of the pipe producing the sound. The player slides a U-shaped section of the instrument back and forth to set the pitch. A trombone player hits the right notes by learning the position for each note and the distance between them. The trombone controller needs to determine the position of the slider part of the 'trombone' and use this to send mouse movement information to the game. It also has an additional input so that the game can be told when the player is blowing the trombone.

Figure 3, overleaf, shows a finished controller. The handle on the left-hand side contains a sensor

Figure 1 The game is available from the Steam store for Windows PCs. It runs well even on a low-performance machine

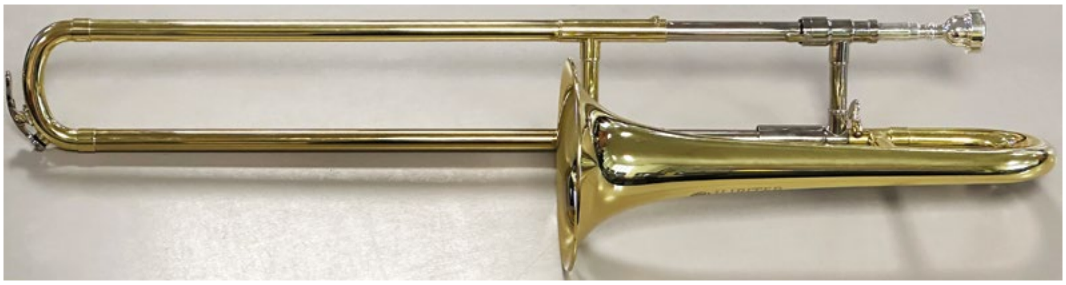


Figure 2
This is a Soprano-sized trombone. It works in the same way as a larger one but is around the same physical size as a trumpet. The slider is on the left

which measures the distance to the target on the handle on the right. The target is connected to a piece of pipe which slides over one connected to the handle. This means that the balance of the controller will change during play, just like a real trombone. It also means that the handle will fall off the end of the trombone if you move it too far, which is just like a real trombone too.

DISTANCE MEASURING

Initial distance measuring tests were performed using an HC-SR04 ultrasonic sensor. This worked, but it could only generate around 20 readings per second, and the reading rate slowed down as the target was moved further away from the sensor. The speed of sound, 343 metres per second, limits the rate at which readings can be produced using an ultrasonic sensor. So instead, the project uses

> "Using CircuitPython running in a Pico, **the author was able to get 100 distance readings per second from this sensor**"

a VL53L4CD Time of Flight (TOF) sensor which measures the time taken for reflected pulses of laser light to arrive. Because it uses light pulses, it can take readings very quickly. Using CircuitPython running in a Pico, the author was able to get 100 distance readings per second from this sensor.

SENSOR SOFTWARE

The **TromboneController.py** program and all the libraries that you need in the GitHub repository for this project can be found here: **hsmag.cc/TromboneControl**. You can use Thonny (**thonny.org**) or Mu (**hsmag.cc/MuEditor**) on your computer to open the program and download it into your Pico. The program uses a VL53L4CD sensor to measure distance. The **TromboneController.py** program exposes the sensor in the form of an object. When the program starts, it creates an instance of a `DistanceSensor` object that connects to the sensor.

```
i2c_sda = board.GP0
i2c_scl = board.GP1

i2c = busio.I2C(i2c_scl, i2c_sda)
distance = DistanceSensor_vl53l4cd(i2c)
distance.start()
```

The statements above create a distance sensor connected to a VL53L4CD sensor. The distance sensor object exposes a method called `start`, which is used to start the sensor running. →

BEST-LAID PLANS

The initial plan was to put the distance sensor inside the inner pipe of the sensor and put a reflector on the end of the outer sliding pipe. This would prevent external interference with the sensor signals and keep the hardware tidy. It turned out that the sensor was a perfect fit inside the pipe (which the author should have seen as deeply suspicious – things were going too well), and so a box was made for the sensor and the Raspberry Pi Pico. The only problem with this design is that it didn't work, perhaps due to internal reflections in the pipe causing the sensor to get confused. The final design puts the distance sensor on top of the sensor box and puts a target on a handle fitted to the outer pipe.

YOU'LL NEED

- **A Raspberry Pi Pico**
- **A VL53L4CD Time of Flight sensor**
- **A BMP280 temperature and pressure sensor** or two push-buttons that will fit in 12 mm diameter holes
- **45 cm of 32 mm diameter waste pipe**
- **45 cm of 40 mm diameter waste pipe**
- **A micro USB cable** to link the Pico to the host
- **Connecting wire.** The author used coloured wire wrap (search for '30 AWG wire wrap'), which needed a wire wrap tool (search for 'wire wrap tool')
- **A case and target.** There is a 3D-printable set which fits around the tubing and works well
- **PG9 cable gland** for the mouthpiece connector
- **8 mm clear vinyl tubing** for the mouthpiece. You'll need a length of around 50 mm for each mouthpiece
- **Screws.** You'll need some screws size M2 4 mm in length to fix the environmental sensor and Pico to the case (search for "laptop screws")

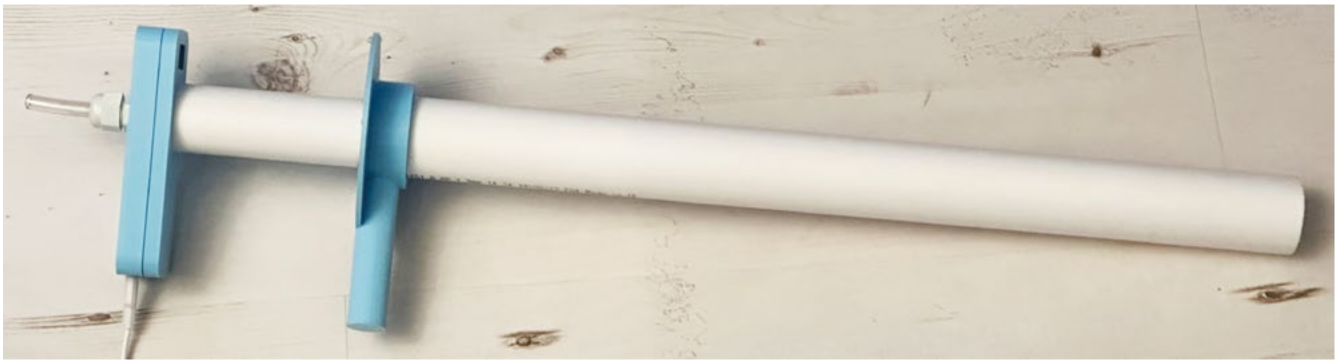


Figure 3 This is the breath input version. It uses a cable gland and some 9 mm tubing so that you can take the mouthpiece off and clean it. There is also a button-triggered version

The program can read distance values from it by calling a method called **`blockingWaitForValue`** provided by the sensor object:

```
print(distance.blockingWaitForValue())
```

The above statement would print out the current distance reading on the console. The distance is given in centimetres. The method **`blockingWaitForValue`** is named to indicate that method will cause the program to pause while the reading is obtained from the sensor.

ROLLING WITH THE AVERAGES

The readings from the distance sensor are a bit noisy. Each reading is a few millimetres different from the previous one. If the program uses the readings as received from the sensor, this would make the mouse jitter up and down, making precise positioning difficult. This problem can be addressed by averaging the noisy values. We work out the average of a series of values by adding them together and dividing by the number of values.

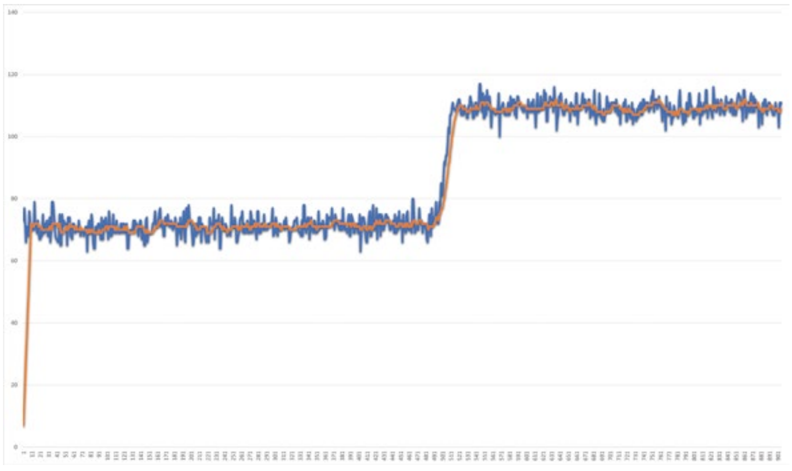
We could add five distance values together, divide the result by five and (assuming that the noise in the signal is randomly distributed) we will get an average which is closer to the actual value. However, this would reduce the rate at which we can update the mouse position. If we average five values to get a less noisy reading, the program would have to wait for all five to arrive before it could send a new mouse position. This would mean we would swap a jittery mouse for a sluggish one.

The solution is to use a 'rolling' average. The program keeps a list of recent values – each time it gets a new value, it discards the oldest, subtracting it from the average total, and stores the newest, adding it to the average total. This generates an updated average with each successive reading. You can see the code below:

```
def blockingWaitForAverage(self, timeout=1):
    distance = self.blockingWaitForValue()

    self.valueTotal = self.valueTotal + distance
    self.valueTotal = self.valueTotal - self.
rollingValues[self.valuePos]
    self.rollingValues[self.valuePos] = distance
    self.valuePos = self.valuePos + 1
    if self.valuePos == self.
rollingAverageLength:
        self.valuePos = 0
    return int(self.valueTotal/self.
rollingAverageLength)
```

The method **`blockingWaitForAverage`** from the **`DistanceSensor`** class updates the rolling average for a distance sensor. The number of rolling average values used is set by the **`rollingAverageLength`** property of the class. The **`rollingValues`** property holds the readings being averaged, and the **`valuePos`** property holds the current position in the value array.

Figure 4 The graph was produced by printing values to the Python console and then creating a comma separated value (CSV) file which was imported into a spreadsheet



GETTING HYSTERICAL

The rolling average technique settles the mouse and removes a lot of jitter. However, a problem with this technique is that it adds ‘hysteresis’ to the signal, as the average value takes a while to catch up in the face of changes. Hysteresis in a system is a delay between an input signal and a corresponding change in output. You see it in electrical circuits involving coils, and in lots of real-world situations. For example, if you turn the steering wheel while driving, it takes a little while for the effect to emerge in the form of a change of direction of your car. The hysteresis in the steering control is one reason why learning to drive is so tricky.

Figure 4 shows hysteresis in action. The blue trace shows raw sensor values, and the red trace shows the rolling average. You can see that the average starts at zero and then rises to the blue level when the program starts running. Then, in the middle of the graph, the distance is changed by moving the target, and the red average value lags the raw value as the rolling average catches up. Too much hysteresis in our controller will make it hard for the player to accurately hit notes. After some experimentation, it was found that an average size of five readings gave a stable mouse cursor without too much lag.

" Too much hysteresis in our controller will make it hard **for the player to accurately hit notes** "

MOUSE DRIVER

The values from the distance sensor are used to move the mouse up and down. CircuitPython provides a library which allows a program to behave as a USB mouse. A program can send X and Y movements and press and release mouse buttons.

```
import usb_hid
from adafruit_hid.mouse import Mouse
mouse = Mouse(usb_hid.devices)
```

The code above creates a mouse object which can be given commands to move the mouse.

```
speed = 0.4
dist = distance.blockingWaitForAverage()
change = old_dist-dist
mouse_dist = int(change*speed)
if mouse_dist != 0:
    mouse.move(y=mouse_dist)
old_dist = dist
```

The code above uses values from the distance sensor to move the mouse up and down. It calculates the change in distance, multiplies it by a speed value, and then moves the mouse the specified distance up or down. If the controller has not been moved, the mouse is not moved. The trombone controller moves the mouse up or down, and so it only sends a value for Y. If required, it could send a value for X and move the mouse left and right as well. The speed variable controls the responsiveness of the mouse to movements of the trombone. You may have to adjust this value so that it works correctly on your PC.

TAKING A BREATH

The Trombone Champ game reads the left mouse button to detect when the player is blowing into →

QUICK TIP

The last time the author used rolling averages like this was in a motion-compensated weighing machine built to weigh fish in a trawler when at sea.

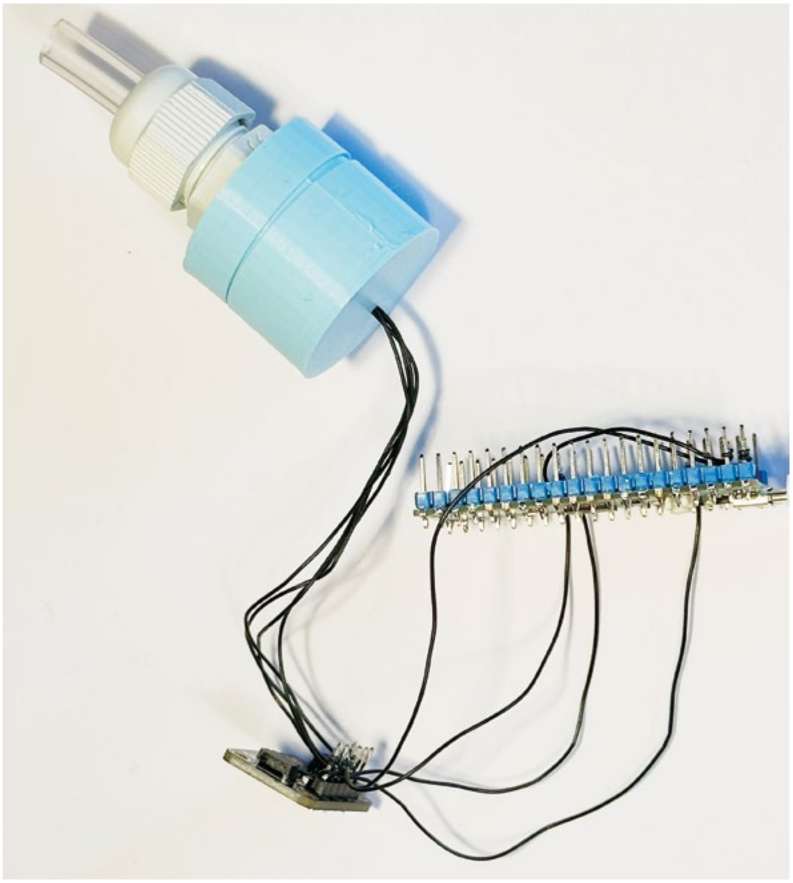


Figure 5 The pressure sensor is connected after the distance sensor on the same I2C connection

QUICK TIP

When using the controller to play the game, you can align the mouse position with the trombone position by moving the target all the way out at the very start of the tune, pushing the Y position of the mouse to the upper limit of the screen, and then pulling the target back in to get the high notes.

the trombone. Trombone players create the sound by blowing into a mouthpiece. The trombone controller detects the breathing action by using an air pressure sensor and sends clicks the left mouse button when the pressure goes up.

Figure 5 shows the hardware. The BMP280 environmental sensor is in the small container which has the mouthpiece connected to it. When the player blows into the mouthpiece, this causes a detectable change in the pressure inside the container, which is used to trigger the left mouse button press.

```
import adafruit_bmp280
class PressureSensor_BMP280(PlaySensor):

    def __init__(self, i2c):
        super(PressureSensor_BMP280, self).__
init__()
        self.bmp280 = adafruit_bmp280.Adafruit_
BMP280_I2C(i2c, address=0x76)
        self.pressure_detect = 5
        self.base_pressure = self.bmp280.pressure

    def play_pressed(self):
        pressure_change = self.bmp280.pressure -
self.base_pressure
        return pressure_change > self.pressure_
detect
```

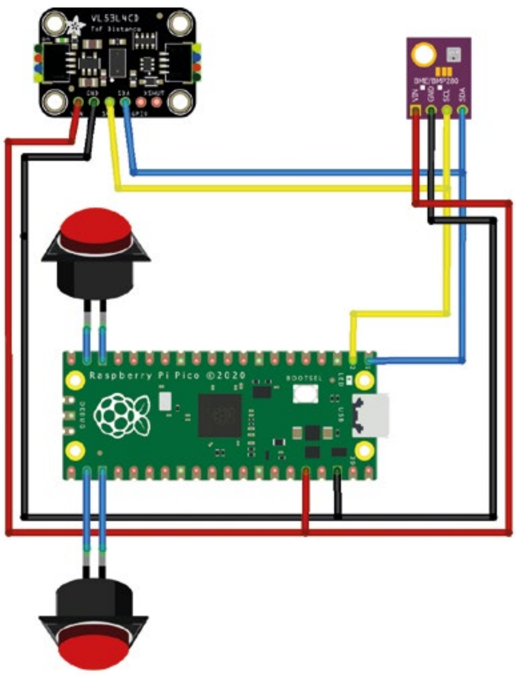


Figure 6 A second button was added but is not presently used by the software. Here, we see the complete circuit diagram for the controller with both the buttons and the BME280 sensor connected. If you don't want to use breath control, leave out the BMP280 sensor

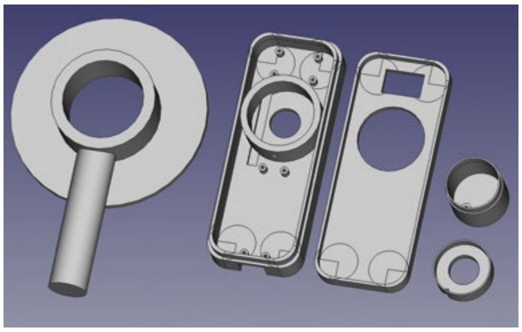


Figure 7 There is a different controller box design for the two-button version

The code above implements the pressure sensor as a class. The `__init__` method is passed an I2C interface which is used to create a BMP280 sensor connection. The `__init__` method records the current air pressure and also sets the `pressure_detect` value to 5. The `play_pressed` method returns `True` if the current pressure reading is above the `pressure_detect` threshold.

```
if play.play_pressed():
    if mouse_down == False:
        mouse.press(Mouse.LEFT_BUTTON)
        print("toot")
        mouse_down = True
else:
    if mouse_down == True:
        mouse.release(Mouse.LEFT_BUTTON)
        print("untoot")
        mouse_down = False
```

The code above uses the result of `play_pressed` to control the mouse. It uses a flag called `mouse_down` so that mouse events are only sent when the pressure state changes. It also prints messages that show the status of the breath sensor.

DRIBBLE TROUBLE

If you look carefully at the trombone picture in **Figure 2** at the start of this article, you will notice a little finger-operated valve at the end of the trombone 'D' on the left of the picture. This is called the 'spit valve' – it is a very important part of the instrument. It turns out that blowing into something doesn't just produce a lot of air, it also produces a lot of moisture. Trombone players use the valve to drain their instrument every now and then. Unfortunately for our trombone controller, it doesn't have a valve to let out liquid. The problem in the controller is not

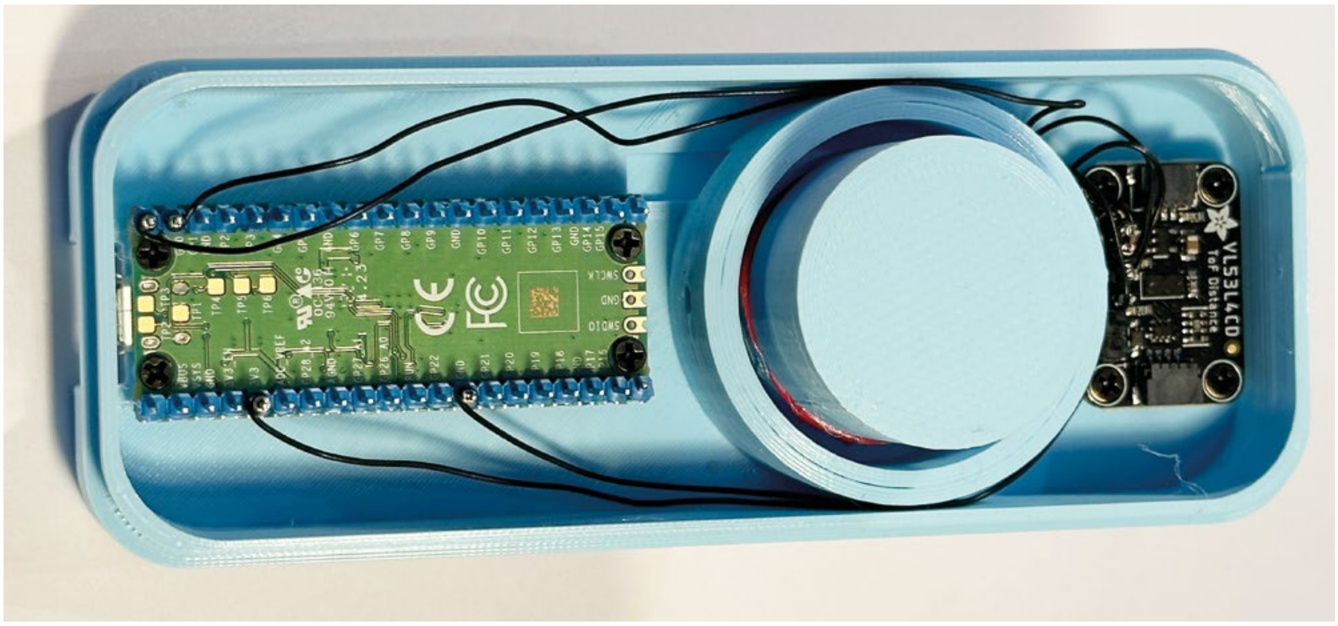


Figure 8
The pressure sensor fits inside the pipe

"It turns out that blowing into something doesn't just produce a lot of air, **it also produces a lot of moisture**"

as great as for a trombone player, in that a player is just blowing to produce a pressure change in a sealed box, not blowing air through the controller, but the sensor can still get sufficiently moist for this to become a problem. A future version of the design could contain a service of baffles and a valve to get rid of any stray dribble, but to keep things simple, it was decided to implement a button-operated version of the controller where a push-button replaces the breath sensor. The author also wondered whether passing round a device and taking turns to blow into it was really a good idea in these virus-ridden times.

```
class ButtonPlaySensor(PlaySensor):

    def __init__(self, pin):
        super(ButtonPlaySensor, self).__init__()
        self.red_button = DigitalInOut(board.
GP14)
        self.red_button.switch_to_
input(pull=Pull.UP)

    def play_pressed(self):
        return self.red_button.value == False
```

The driver class for the push-button play sensor is shown above. It contains a `play_pressed` method, which means that a `ButtonPlaySensor` instance can be interchanged with a `PressureSensor_BMP280` instance without needing to change anything else in the code. The `play_pressed` function returns true if the value of the button has become false because the switch in the button has connected the input pin to ground.

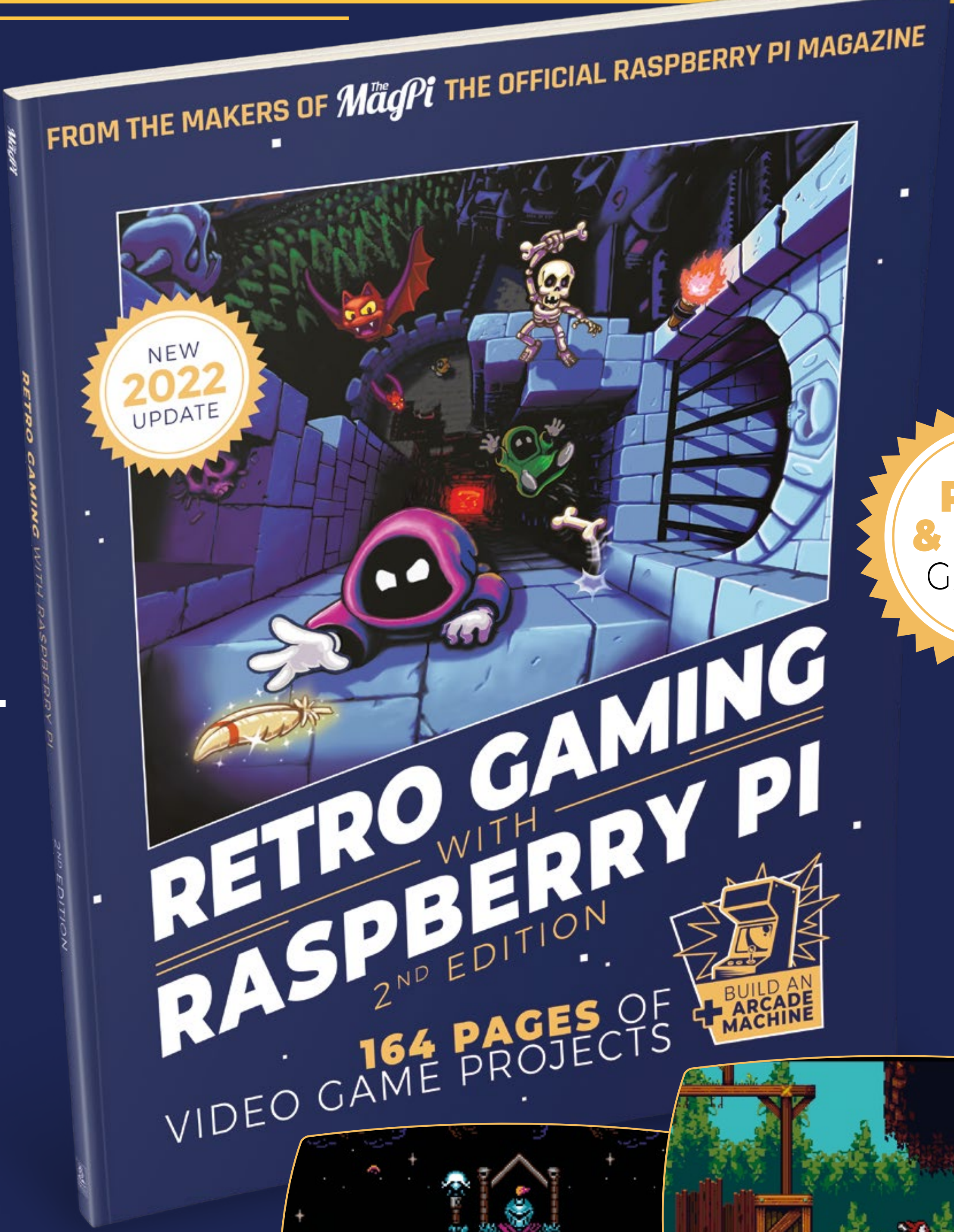
CONTROLLER CONSTRUCTION

Figure 7 shows the 3D-printable components that make up a controller. They are available for download from the GitHub repository for the project: **hsmag.cc/TromboneControlGH**. The small drainpipe is a friction fit into the base of the controller, and the large drainpipe is a friction fit into the target.

Figure 8 shows the Pico controller inside the controller handle. The distance sensor is at the top of the handle, and the 32 mm diameter waste pipe fits over it.

TROMBONE POWER

The trombone controller works well, although it is very hard work using the breath interface to control the game. It gives you a lot of respect for people who play the real instrument. The push-button version is much easier to use, and not prone to dribble trouble. The author is very proud of getting a 'B' rating on one of the tracks in the game using the controller. It certainly adds a lot to the gameplay. □


FROM THE MAKERS OF The MagPi THE OFFICIAL RASPBERRY PI MAGAZINE
NEW 2022 UPDATE
RETRO GAMING WITH RASPBERRY PI
2ND EDITION
164 PAGES OF VIDEO GAME PROJECTS
+ BUILD AN ARCADE MACHINE

PLAY & CODE GAMES!

Needle felting: Learn to sculpt 3D shapes with wool (adding a little LED bling)

Piercing wool fibres with needles to form aesthetically appealing shapes is an easy, inexpensive, and highly therapeutic craft

Nicola King

@holtonhandmade

Nicola King is a freelance writer and sub-editor. Crafting and making stuff generally helps to keep her grounded and semi-sane…just ask her long-suffering family.

The increasingly popular craft of needle felting, also known as dry felting, is the art of transforming wool fibres into, usually, 2D or 3D shapes, with the aid of some barbed needles, and this author is happy to announce that needle felting is her new obsession. She'd read that it was an addictive hobby, but little did she realise just how captivated she'd become. At the end of her first project, she was already looking for potential needle felting accessories that her family could gift her for Christmas… we're just pre-warning you that if you try this craft once, you'll likely be back for more. It's a hobby that only requires a few materials to get started, and these are easily purchased online or from a craft store. It's also inexpensive, and very uncomplicated to get to grips with in terms of skill level. As well as that, it's extremely therapeutic… there's just something about stabbing wool endlessly with needles that is so very calming!

So, in this tutorial, we are going to needle-felt a 3D shape. And to give it that HackSpace twist that we know you all crave, we are also going to add a simple circuit so that we have some LED lights adorning our newly felted make. Light-emitting diodes undoubtedly look good on anything.

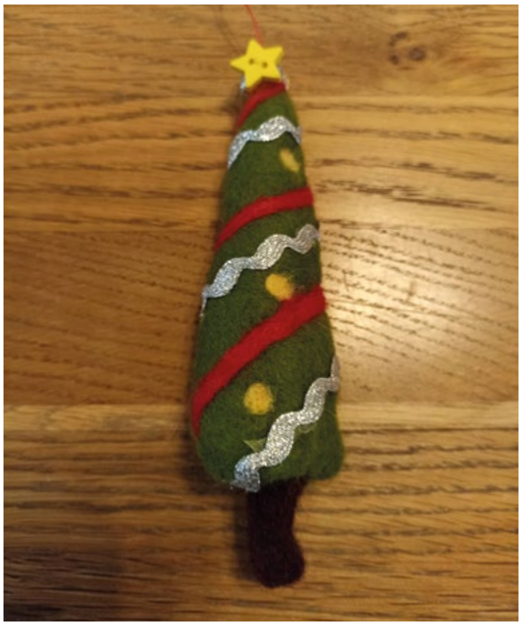


Figure 1 First attempt, and we're pretty pleased. A Christmas decoration that we hot-glued some trimmings to. It's surprisingly easy to felt the trunk to the tree, just keep stabbing!

FELTING FUNDAMENTALS

The process of needle felting involves the repetitive stabbing, or piercing, of wool fibres – this means that the needles, and specifically the barbs or notches on the needles, grab the top layer of fibre and then tangle or knit this layer with inner layers of fibres. Usually the needle notches are facing downwards, so they don't pull the fibres out as the needle exits the wool. Therefore, the wool, which progresses to a solid felt, becomes stronger after the fibres are compressed and tangled, and forms more solid shapes. The more you stab it, the more the fibres slowly matt together.

Figure 1 shows this author's very first stab (!) at needle-felting a 3D shape, a Christmas decoration, and it turned out pretty well. After some time piercing away, the fibres eventually started to condense, and the piece began to take its conical shape, with the texture and density firming up as felting progressed.

There are many types of wool and needles that can be used for needle felting (see 'Types of felting

TYPES OF FELTING WOOL/NEEDLES

WOOL FIBRES

There is a daunting array of wools available to felt with, some more fine than others, some more coarse, and newbies may be slightly bamboozled by the choice, so here is a very quick guide to a few of the options:

- **Batts** – this is wool that has been washed, teased out, and carded into batts, which is like a sheet of wool. Great for needle felting
- **Raw/unwashed wool** – straight off the sheep but, as it's not been washed, you'd need to wash your hands regularly or wash it yourself before use
- **Cleaned locks** – wool that's been washed, but not combed, so it's great for adding effects to a finished piece
- **Tops/roving** – washed and combed wool, and the fibres are all going in one direction. Perfect for felting and easy to find. Natural wool roving is what we've used in our tutorial project, and it comes in many different colours, so there is lots of choice

Different fibres and wools will create different looks – just experiment!

NEEDLES

Again, a huge range to choose from. One thing to note is that felting needles are very delicate and can break easily if you don't hold them and stab them at a 90-degree angle, which minimises the chances of breakage. Needles come in different gauges, with the gauge referring to the diameter of the needle – the higher the number, the finer the gauge. Usually you would start a project with a lower gauge to do the bulk work and the shaping of your piece, and then move onto a finer needle for the surface and detail element.

Felting needles tend to fall within the 32–42 gauge bracket and, from what we've learnt so far, a 38-gauge is a great all-rounder. A few examples of varieties of needle include:

- **Triangle-shaped** – three-sided blade with downward barbs on all three sides
- **Twisted** – spiral blade with notches that point downwards and twist around the needle
- **Reverse** – barbs on these point the other way, upwards, so fluffy finishes can be created

You can also buy needles in a pen form, which makes them easier to hold (**hsmag.cc/PenFeltingTool**).

wool/needles' box for more information). You can buy needle felting kits (from an array of craft kit purveyors) which come with everything you need to get you started; we purchased a couple of these to practise with. One came with a plaster, which we found hilarious and alarming in equal measure – needle felting obviously involves working with sharp needles but, as long as you are careful, a plaster shouldn't be required (at least, we've not needed one yet).

Do be aware that some kits come with polystyrene shapes to mould your felted shape around. Adding polystyrene to the mix undoubtedly speeds things up enormously and saves on wool fibre usage. However, be conscious of the fact that polystyrene is slow to degrade and not particularly environmentally friendly, so we'd recommend that you avoid kits that include it.

Finally, before we get started, it's worth noting that your piece of needle felting is unlikely to actually resemble what you are aiming to make until you are around three quarters of the way into your project. Be patient, as it will come together in the end. After quite a lot of stabbing and piercing, this author was not at all confident that she possessed any skill whatsoever in the needle felting department but, after some persistent needle action, her faith was restored and she could see it finally taking shape. The moral of this tale being: keep at it!

Above
Our completed and illuminated toadstool in its natural habitat. (Seasonal elf styling is entirely optional)

FUNGI FELTING

So, we have chosen to create a needle-felted toadstool because, frankly, who doesn't want one of those in their life? You can, of course, create absolutely any shape you like with any colours you wish to use and, if you want to add some luminosity as we're doing, how about creating a felted car, tree, or house? With a little practice, you'll soon see how intuitive this craft is, and you'll be felting all sorts of shapes in no time. →

TAKE CARE

Needle felting is definitely *not* a craft for young children due to the sharpness of the tools involved, (but fine for anyone aged ten or above), and you should keep the needles away from your pets too. Mind those fingers, and remember to store your needles safely out of harm's way when not in use.

YOU'LL NEED

- **Wool fibres** (e.g. carded wool, wool batts, merino wool, tops, or roving)
- **Felting needles**
- **Needle holder** (optional)
- **Thick foam pad** (hacker's economical alternative: a car washing sponge)
- **Finger guards** (optional)
- **Cookie cutters** (optional)
- **Hot glue gun and sticks** (optional)
- **Embellishments** (optional)
- **Sewing needle** (for creating the circuit with thread)
- **Conductive thread** (hsmag.cc/ConductiveThread)
- **CR2032 battery holder with switch** (hsmag.cc/ SewBatteryHolder)
- **CR2032 battery**
- **2 × (or more) easy-sew white-coloured LEDs** (hsmag.cc/ LEDlights)
- **Scissors** (for tidying up stray wisps of fibre)
- **Plasters** (in unlikely case of mishap – optional)

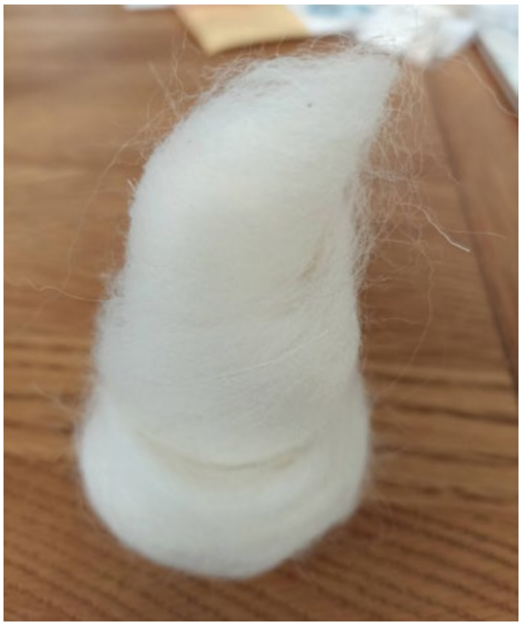


Figure 2 The stem in its raw, unfelted form, just rolled into the vague shape that we are looking for – it's about to get the needle!

STEP 1 GIVE IT SOME NEEDLE

You need to be sitting down, ideally with a flat, clear work surface in front of you – the kitchen table would be perfect. Make sure you have your foam pad in place, in order to protect your surface, and arrange all of your materials and tools so that you can easily and safely access them.

We began by creating the base or stem of our toadstool and we used some cream/off-white wool roving, pulling off a small amount and gently shaping it with our fingers into a rough stem shape (**Figure 2**). To give you an idea of size, our stem measures around 6 cm in height, and is about 4 cm in diameter. We wanted to have the option of possibly fitting the battery holder to the base of the toadstool – in the end, we fitted it under the cap closer to where we wanted our LEDs, but the choice is yours. Just make sure that you make the base wide enough if you want to attach the holder there.

Once you are happy with the rough size and shape, you can begin felting. Use your needle at a 90-degree angle to minimise the chance of needle breakage (as they are very delicate tools) and gently stab down into the fibre, using only the lower section of the needle. Note that you should not be pushing the needle all the way down into the foam pad. Keep your wrist straight when stabbing, and gently move the piece round, so that you are not stabbing in only one place. Continue this light piercing motion and you will find that the more you stab, the tighter and thicker the wool gets. As you agitate the fibres, they bond together. You want to get to a point where the stem of the toadstool feels quite firm. If you need to add more wool to shape it, just pull some off and start felting it into the shape. This process took us around half an hour, but time flew!

"**Keep your wrist straight when stabbing,** and gently move the piece round"

QUICK TIP

Remember that the shape that you end up making will be smaller than the shape you started with, and this is normal – it will be around 30 to 50% smaller. As you felt, you are condensing the fibres.

COOKIE CUTTER CREATIONS

If you are new to this craft, using a cookie cutter as a shape guide is a great idea. Whatever you have in the kitchen drawer, cookie cutter-wise, can be utilised to get you going on your felting journey.

Choose your cutter, take some wool roving, or whatever you are using to felt with, and fill the cutter with your fibre. Using the instructions we've already covered, begin felting. When you remove the shape from the cutter, you'll need to finesse the edges and do some shaping work to refine the shape. Use your needles to create definition around corners for example, and use scissors to trim away stray fibres.

If you have small star or heart shapes, you could string some up to make bunting or, for individual shapes, maybe add a loop for hanging with some thread, per our Christmas tree decoration. You can also attach your needle-felted shapes to brooch backs, hair accessories or, if you use less roving and create a flatter shape, you can make appliqués that can be attached to other items, such as bags.

STEP 2 CAP IN HAND

Next, we created the cap of the toadstool. Using our red-coloured fibre, we created a vaguely round shape, placed it on the foam pad, and started to work the fibre with our needle. Take your time: this is a walk, not a sprint, so don't panic that you're spending too long on it, just give it as much time as it needs.

As we wanted to fix the battery holder under the cap, and to achieve that authentic toadstool cap shape, we created a bowl-type form, so that the underside had plenty of room to hide the battery holder (see **Figure 3**). Again, keep working on the shape until it feels firm and cap-shaped. Add more fibre around the sides to form that brim if you need to – sculpt it with your needles until you are satisfied.

When you are content with the form of the cap of the toadstool, you can then add those final touches to make it look more convincing – some little white spots. Using the cream fibre, we rolled some tiny balls, (you don't need very much at all!), and gently felted them into the top of the cap. Use a finer needle, if you have one, as this is more delicate work, and remember to keep fingers safe as this surface work can be very fiddly.

STEP 3 CIRCUIT-MAKER

So far, so good – if nothing else, you've learnt how to needle-felt a 3D shape. Next, we added a little radiance to our creation. The first step was to attach the battery holder, and we decided to sew it underneath the cap. As the holder will only be a short distance from the sewable LEDs, we only needed a short length of conductive thread, so we first cut a 15 cm length, threaded our needle and knotted the end. We then attached the thread with a stitch into the base of the cap. Next, we sewed into one of the positive holes on the holder a couple of times (**Figure 4**), then pushed the needle up to one of the white spots on top of the toadstool. Then, we took one of the LEDs and sewed through the positive sewable side of the LED, again, a couple of times. We then took the needle down to the underside of the cap and unthreaded the needle, letting the short length of conductive thread drop down.

Next, we took another 15 cm length and did the same again on the negative side of the holder, sewing through the negative hole, taking the thread up to the same LED, but this time sewing through the hole on the negative side of the LED. It's then just a question of doing the same thing for the other LED, as we are using just two of them in parallel, using the two negative and two positive holes on the battery holder. (Note that you could use more LEDs if you wished and set them up in a series, linking one to the next with thread, negative to positive.)

Finally, we inserted the battery into the battery holder. Our holder has an on/off switch, so we switched it to 'on' and the two LEDs lit up beautifully. A simple circuit that just adds an extra touch. If your LEDs don't light up, check for an easily rectifiable short-circuit – for example, the ends of the conductive threads could be touching/crossing each other, or they could be touching the battery.

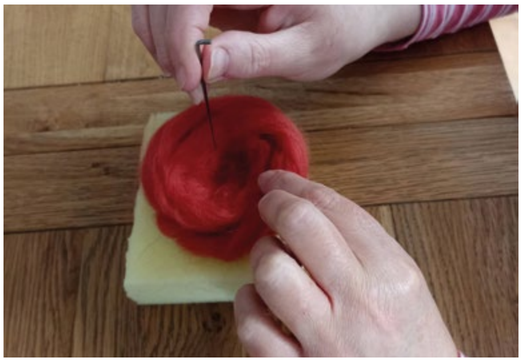


Figure 3
Stabbing the cap into shape. Use whatever colours you want or have to hand. You can even mix colours by laying them on top of each other while felting, or by manually mixing the fibres before you start the felting process

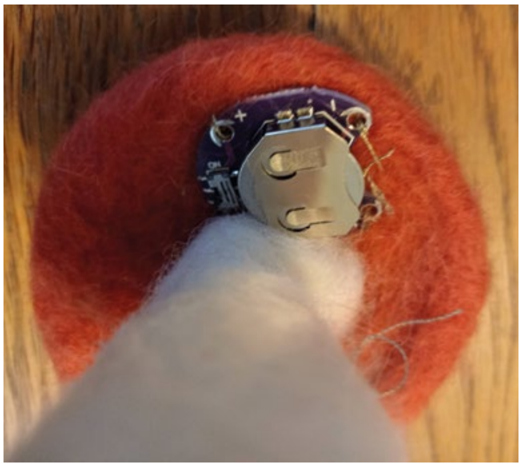


Figure 4
Our concealed battery holder sewn into place. Use a reasonably fine sewing needle to do this, as it needs to fit through the holes on either side of the holder when the holder is being sewn in

COMPULSIVE CRAFTING

Needle felting is a very easy-to-learn textile craft. It's portable, rewarding, and a great way of making personalised gifts for others. Compared to something like knitting or other fibre arts, you can actually produce a finished item very quickly. The sound of the needle crunching into the wool is one of the very pleasing aspects of the craft, and it's extremely tactile too. As with every skill, the more you practise, the more you will learn and, as you progress, you can incorporate things like wire into your work to help shape it and create some amazing pieces with real character. If you are looking for a way to escape the mundane, needle felting is totally absorbing. □

QUICK TIP

Start with less wool than you think you will need – it is much easier to add more, and meld it into your creation, than it is to take some away if you've used too much to start with.

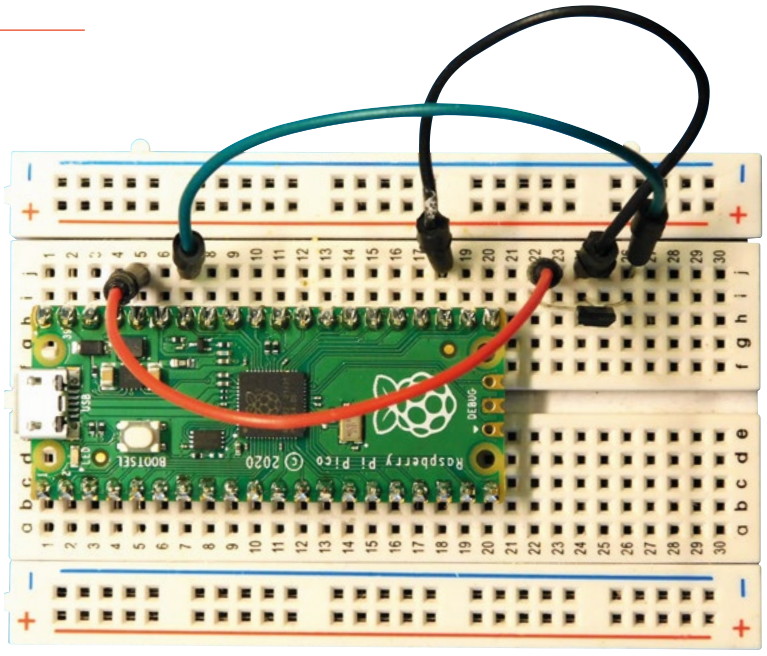
I WANT TO KNOW MORE!

Here are a few information/product sources to get you going:

- There are numerous instructive books on the subject of needle felting, which you can either purchase or borrow from your local library. *Needle Felting for Beginners* (2020) by Dace and Balchin, or *Making Simple Needle Felts* (2018) by Stern are both educational options.
- If you want to add some LED bling to your needle-felted project, Light Stitches (**lightstitches.co.uk**) specialises in supplying electronic textile kits and components, and sells some needle felting e-textile kits too. It also has conductive thread, battery holders, LED lights, and more, which are useful for all sorts of HackSpace-influenced projects. Also, The Felt Box (**thefeltbox.uk**) offers a plethora of needle-felting goodness, including monthly surprise boxes if you want a treat!
- There is a huge range of inexpensive kits available to buy, and they make great gifts for someone who you think will enjoy some repetitive fibre stabbing, maybe as a stress-buster! Try this one for a start-off: **hsmag.cc/UltimateKit**.
- Of course, YouTube is brimming with helpful 'this is how you needle felt' videos, and they are well worth a watch for tips, guidance, and general inspirational ideas. Here's a useful link to help steer your felting techniques in the right direction: **hsmag.cc/AvoidMistakes**. These sheep baubles (**hsmag.cc/SheepBaubles**) are adorable, too – ewe have to give them a go!

Magnetic input

Press buttons without ever touching them



Ben Everard

@ben_everard

Ben's house is slowly being taken over by 3D printers. He plans to solve this by printing an extension, once he gets enough printers.

Magnets are almost magical. They apply a force across space. You can use them to move things without touching them. You can also use them to interact with your electronics projects. In this article, we'll look at a couple of electronics components that are affected by magnets and learn how to use them in our projects. We'll use MicroPython for the examples, but they should be trivial to port to other languages.

The reed switch is a type of switch that is controlled by magnets. It works in almost exactly the same way as a momentary push-button, except that instead of a finger pressing a switch, there are two contacts that are pulled together if there's a magnet nearby.

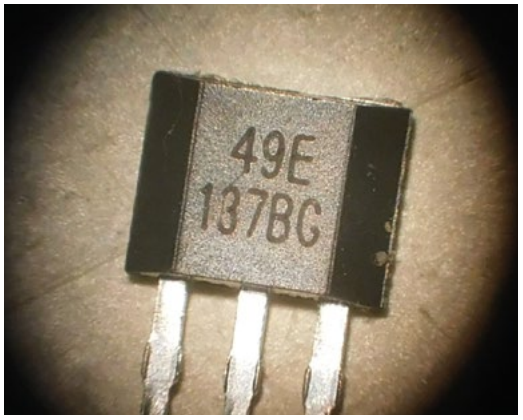
The wiring, then, is exactly the same as for a regular button. You need to connect one end to a GPIO pin and the other end to ground. The end connected to the GPIO pin needs to be 'pulled up' to 3 V by a high-value resistor, but Pico has internal resistors that do just this if we configure them to, so we don't need to add one externally.

Reading one is exactly like reading a regular button.

```
from machine import Pin
import time

reed_switch = Pin(0, Pin.IN, Pin.PULL_UP)

while True:
    if reed_switch.value() == 0:
        print("magnet detected")
    time.sleep(1)
```

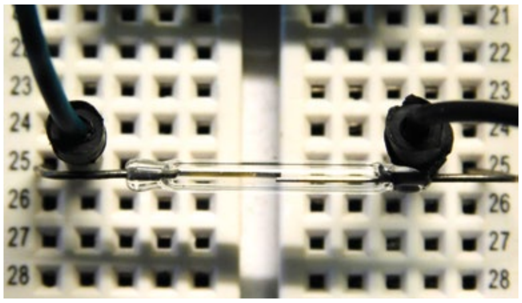
Because reed switches are identical to regular buttons (at least from an electronic point of view), you can use them interchangeably. For example, if you need to detect if a door is closed, you could use a microswitch, or you could use a reed switch and magnet. The wiring and code for this wouldn't change, so you can simply pick the best one for your particular physical setup.

One slight drawback of reed switches is that they can be sensitive to vibration since this can jiggle the contacts and cause button presses when there isn't really one. You might be able to solve this in software with some debouncing.



Above We took this photo through our soldering microscope

Above Wiring a Hall effect sensor is really easy



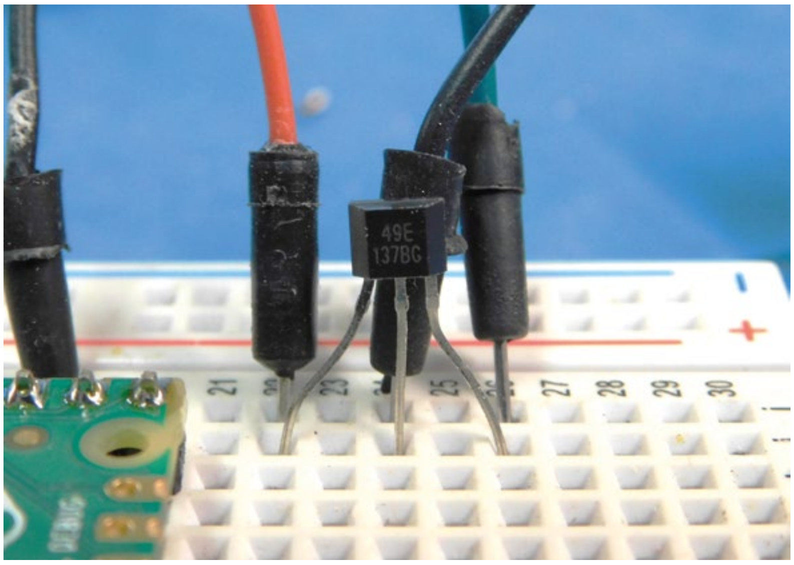
Above You can see the two contacts in the reed switch

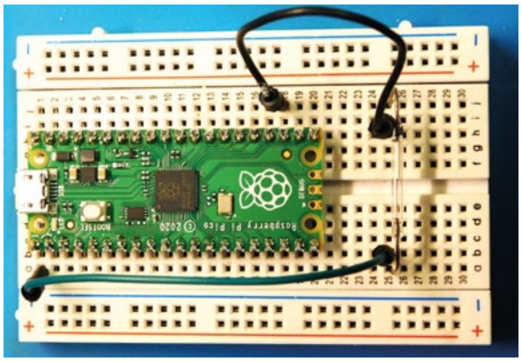
ANALOGUE INPUT

Hall effect sensors are a little different from reed switches. They come in many forms, but the ones we'll look at are 49E. They come in packages with three legs that look a bit like a transistor; one positive, one ground, and one output. There are other Hall effect sensors available, and most should work in a similar way. While reed switches just detect any magnetic field, Hall effect sensors can tell the direction of a magnetic field. Some are only sensitive to one direction (unipolar) while others – including the 49E we're testing with – are sensitive to both. The 49E has an analogue output that varies in relation to the strength of the magnetic field. You can also get digital Hall effect sensors with a simple high or low voltage output, depending on the strength of the field (note that the threshold is usually configurable).

To connect it up, connect the positive leg to 3 V, the ground to ground, and the output leg to GPIO 28 on Pico. You can then read the input with the following code:

```
from machine import ADC, Pin
import time

HALL_THRESHOLD = 0.2

hall_effect = ADC(Pin(28))

hall_midpoint = 0

for i in range(10):
    hall_midpoint += hall_effect.read_u16()

hall_midpoint /=10

while True:
    if hall_effect.read_u16() > hall_midpoint * 
(1+HALL_THRESHOLD):
        print("Hall effect sensor high")
    if hall_effect.read_u16() < hall_midpoint * 
(1-HALL_THRESHOLD):
        print("Hall effect sensor low")
    time.sleep(1)
```





Above The 49E sensor looks a lot like a transistor, but is sensitive to magnetism

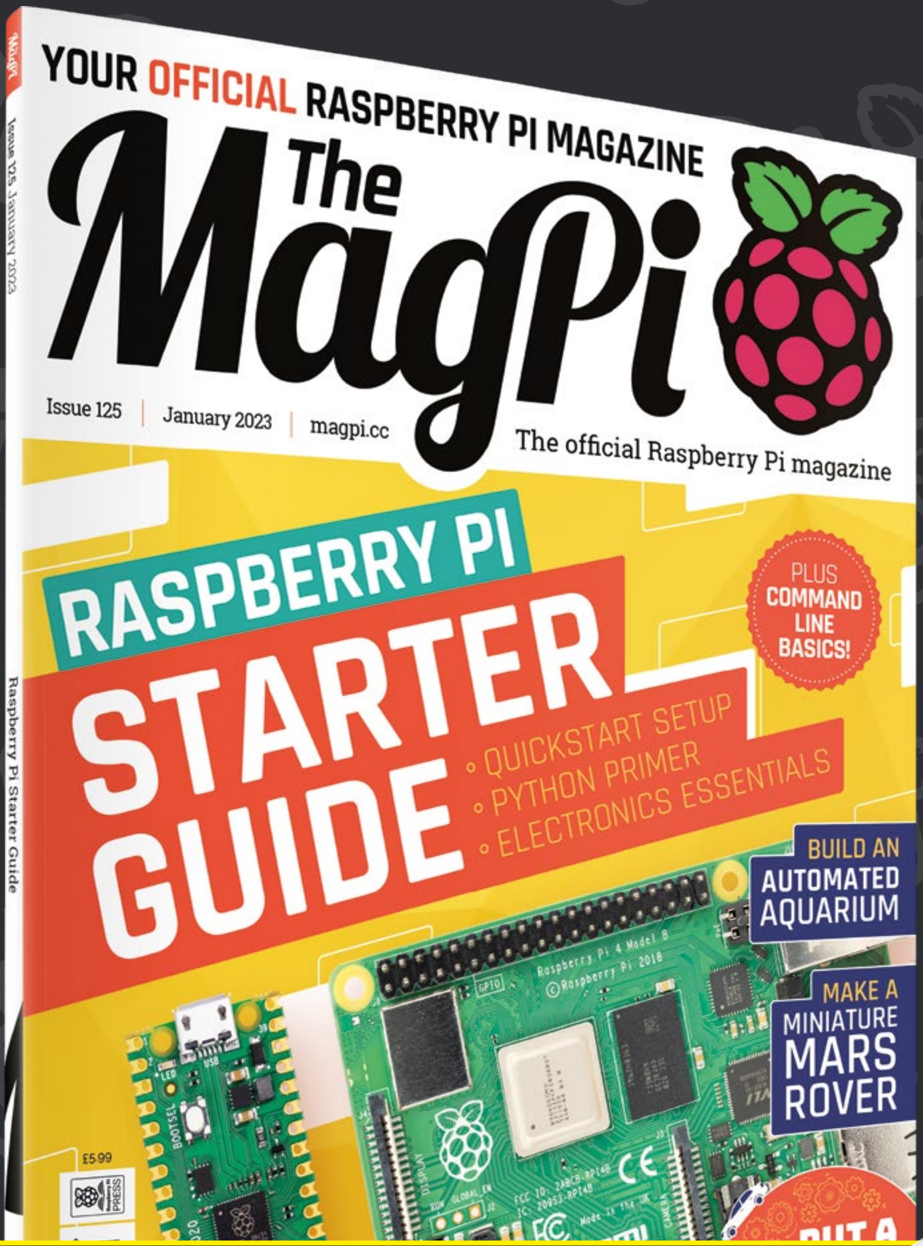
Left Reed switches are wired like any other switch

Changing the threshold will make it trigger at a smaller or higher distance to the magnet. However, if you get it too low, you'll be at risk of phantom triggers from the random jitter in the readings. Reed switches and Hall effect sensors can be useful in a range of projects. They can create novel user interactions where people move magnetic objects, detect if something is open or closed, they can let you input data into a completely waterproofed build, and more. What's more, they're really easy to use.

GENERATING **ELECTRICITY**

Obviously, all circuits are sensitive to magnetism because it can interfere with the electrical currents in the circuit, even generating currents where there wasn't previously any current. Unfortunately, it can be a bit hard to work with induced current like this, at least from digital circuits. Unless they're moving really fast, the sorts of small magnets you're likely to use in hobby projects are unlikely to create any significant voltages in your project.

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


YOUR OFFICIAL RASPBERRY PI MAGAZINE
The MagPi
Issue 125 | January 2023 | magpi.cc
The official Raspberry Pi magazine
RASPBERRY PI STARTER GUIDE
QUICKSTART SETUP
PYTHON PRIMER
ELECTRONICS ESSENTIALS
PLUS COMMAND LINE BASICS!
BUILD AN AUTOMATED AQUARIUM
MAKE A MINIATURE MARS ROVER
£5.99

SUBSCRIBE FOR JUST £10!

- **THREE!** issues of The MagPi
- **FREE!** Raspberry Pi Pico W
- **FREE!** delivery to your door

+ FREE RASPBERRY PI PICO W*

Three issues and free Pico W for £10 is a UK-only offer. Free Pico W is included with a 12-month subscription in USA, Europe and Rest of World. Not included with renewals. Offer subject to change or withdrawal at any time.

magpi.cc/subscribe

HackSpace
TECHNOLOGY IN YOUR HANDS

FIELD TEST

HACK | MAKE | BUILD | CREATE

Hacker gear poked, prodded, taken apart, and investigated

ONLY THE BEST

Robots that can help you expand your business

Affordable and useful robotics for manufacturing

By Marc de Vinck @devinck

Robots have been around for a long time and can be found everywhere. They have infiltrated our homes, in the form of oversized hockey pucks randomly patrolling empty rooms for dust bunnies. There are even drones that can drive around, or even fly, looking for nefarious visitors when you aren't home. And there are bots that can keep an eye on your pet, and dispense some treats when you feel guilty for working late.

And although some of those bots have practical jobs, many of the affordable bots, especially robotic arms, are more akin to toys than reliable and useful tools. Many robots are fun to play with, but don't really do too much other than demos. That's not to say that $100 robotic arms are bad. They can be inspirational and educational, and that's great! But I was looking for something more practical and useful in manufacturing and, historically, those robotic arms were expensive. However, I have found that in the past few years, the market for useful and capable robotic arms has come down a lot in price and advanced in their capabilities.

In my own business, I use a modified Rotrics DexArm to pick and place components when manufacturing custom LEDs. The robotic arm only cost me about $900, and it has been well worth the money. In fact, I'm adding a few more as we speak! Sure, I'd love to get my hands on a DOBOT Nova or CR16, or if money was no object, a KUKA, but the simple and affordable DexArm has saved me countless hours of manufacturing, and at a very affordable price.

In this Best of Breed, we'll be looking at a collection of robotic arms that are all under $1000, and can be useful in small-scale manufacturing. These aren't toys, although they are fun! They may not replace an employee, but they can improve safety and accuracy. And in my case, eliminate repetitive tasks, which allows me to do things that are a bit more interesting.

Rotrics DexArm vs Hiwonder JetMax

ROTRICS $799 | rotrics.com/products/dexarm

HIWONDER $789 | hiwonder.hk

As mentioned in the intro, I use the DexArm by Rotrics regularly. It's also the inspiration for this Best of Breed roundup. I was a little sceptical at first, but after watching countless videos and reading everything I could, I took the plunge and ordered the deluxe kit. It took some time to figure everything out, but I got the hang of it and, more importantly, fully understood its capabilities and its limitations.

The DexArm is capable of drawing, 3D printing, laser etching, and pick and place with a vacuum pump or soft gripper. I use a modified gripper for my daily tasks, and it has worked out great. The working area is about 210 mm × 297 mm, which is plenty for my application. It can carry a 500 g payload, and is compatible with Mac or PC.

I found that the included software was good, but if you really want to get the most out of the bot, you must dive into the manual and learn all the specific G-code the bot understands. I was able to fully customise mine, and with amazing repeatability. I really do enjoy pressing a button and watching it repeat a task a few thousand times without any complaints or, more importantly, errors!

Left This author's daily driver

Below left Jetson gets mobile

The Hiwonder JetMax is powered by the Nvidia Jetson Nano, a well-known processor in the AI community. This makes JetMax a great platform for artificial intelligence-based robotic systems, including deep learning and model training. You can manually control the bot via a PC, mobile app, mouse, or even a wireless game controller. It has a high-definition camera for motion and gesture recognition, and is easily expandable thanks to the Jetson Nano at its core. It's all open-source and can be programmed via Python or ROS. There are three different kits available, but I'd recommend the developer kit since it includes so many extras like multiple grippers, penholder, a handful of additional sensors, and materials. I have a few complex tasks I need to perform, and this bot has made the short list for my next purchase. Head to the website to learn more about this capable little bot. →

VERDICT

Rotrics DexArm

A very capable bot.

10/10

Hiwonder JetMax

Interesting vision capabilities with great brains.

10/10

myCobot 280

MYCOBOT ◆ $799 | elephantrobotics.com

It's got a Raspberry Pi inside so, at first glance, that makes this little bot very desirable for many of our readers. It also has some interesting specs, including six DoF (degrees of freedom), 250 g payload, and a range of 280 mm. And, if you are familiar with Python, then this might make a good starting point for learning about 6-axis robotic arms. Just keep in mind, from what I see in demos posted online, it does have some repeatability and accuracy issues compared to other, less complicated, 3- and 4-axis robotic arms.

If you just need to pick up something small and lightweight, it could work, especially if you include some homing functions prior to any routines. A nice feature is the ability to flash it with alternative firmware and get this bot working with ROS (Robot Operating System), which opens the robot to advanced simulations and programming. And for many, this is a great way to learn about 6-DoF arms without jumping into the typically $10,000+ price point, and that's why we have included this bot in the roundup.

Left ◆
Controlled squiggle and squirm

VERDICT

myCobot 280

This bot packs a Raspberry Pi.

8/10

EDUEXO BY AUXIVO

AUXIVO ◆ $295 | auxivo.com/eduexo-20

It's time to mix things up a bit and look at making you a bit more robotic! The EduExo Robotic Exoskeleton Kit is an interesting intro into educational robotics, regardless of age or skill level. Everything you need to build an exoskeleton is included in the kit, along with a comprehensive handbook to get you up and running fast. You also will learn about the hardware, how it works, how to design a basic controller, and how to connect the exoskeleton to a video game. It looks like a unique educational experience. Auxivo also makes more expensive exoskeletons with advanced features and capabilities.

myPalletizer 260 M5Stack

MYPALLETIZER ◆ **$881** | **elephantrobotics.com**

The myPalletizer 260 M5Stack is a compact 4-axis robotic arm that offers a wide range of features and capabilities, making it a good choice for industrial automation applications. It features four degrees of freedom and advanced motion control, based on an integrated dual-core ESP32. That's not a lot of power compared to a modern processor, but these types of robotic arms don't need that much to function. And an ESP is capable of advanced processes like object, facial, and image recognition which can be implemented in your code.

The myPalletizer has a 2-inch screen, 20 I/O, three customisable buttons, a secondary LED matrix on the arm itself, and it can handle a payload of up to 250 g with a working radius of 260 mm. Its max speed is 120º/s and it can work via ROS, which is important for anyone wanting to use robots in commercial applications. And, just like all the other robots in this roundup, there are many possible use cases for implementing this bot in small-scale manufacturing. →

Above ◆
The sucker can hold flat objects …

Left ◆
… or you can add a hand for more complex control

VERDICT

myPalletizer 260 M5Stack

A Raspberry Pi-based robot is hard to pass up!

9/10

BEST OF BREED

mechArm Pi

MECHARM $799 (Base model) | elephantrobotics.com

Above
One of the more compact options

The mechArm Pi is a Raspberry Pi-based 6-axis robotic arm that has been designed for makers, tinkerers, designers, and anyone who just loves technology. It's compact, requiring little space to use, and comes preassembled and ready to use right out of the box. Its modular design allows for easy customisation by adding different end effectors, like specialised sensors or grippers, depending on your needs.

And, since it's based on the Raspberry Pi, many of our readers will feel right at home when using it. The creators claim accuracy up to 0.2 mm and high-speed movements at up to 10 m/s² acceleration and 25°/sec angular velocity, which is plenty enough for simple manufacturing processes. You can also use ROS for controlling and visualising the movements, along with simple drag-and-teach inputs or block programming. It looks like a good introduction to robotics.

VERDICT

mechArm Pi

Another Raspberry Pi-based bot for instant familiarity.

8/10

DOBOT CR16

DOBOT $35,000 | dobot.nu

Yes, this robotic arm is $35,000, and to be honest, that's not expensive when it comes to robotic arms with these capabilities. The DOBOT CR16 is a collaborative robotic arm that features high precision, repeatability, and accuracy, with its 6-axis arm capable of handling loads up to 16 kg. It has a built-in camera for precise pick and place moves, and customisable end effectors. It looks very easy to set up and use, which is just another reason I'd like to add the CR16, or maybe a smaller CR5, to my manufacturing process.

Right
How much?!

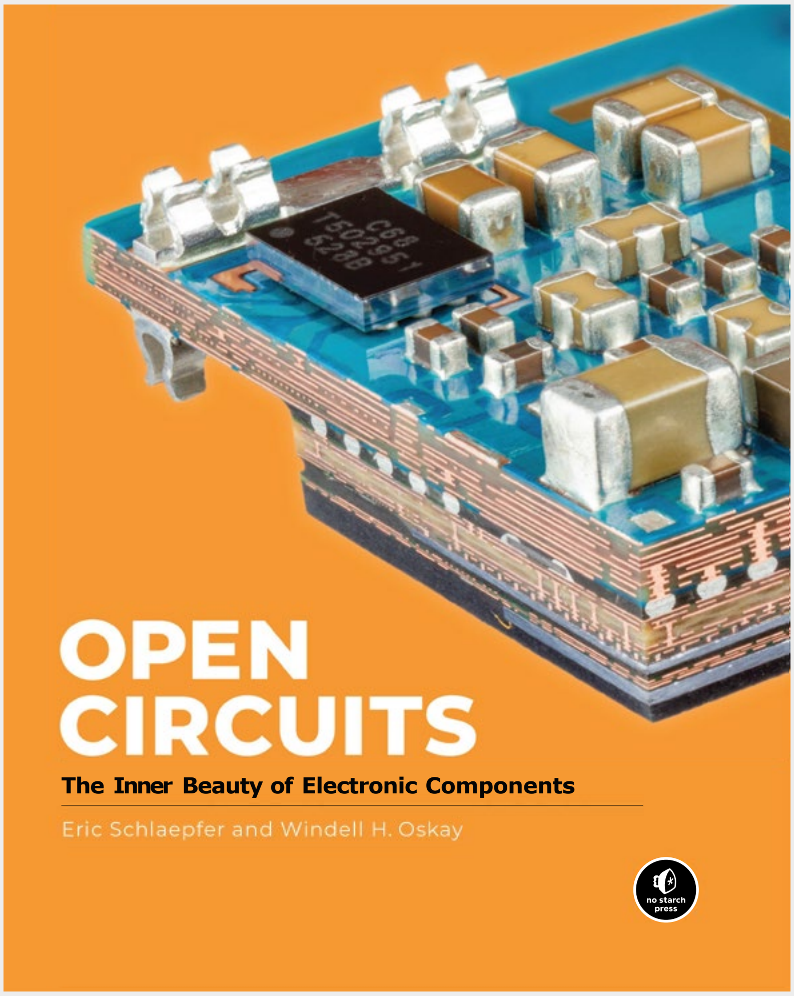
REVIEW

Open Circuits

Peek inside the hidden world of electronics

Eric Schlaepfer & Windell H. Oskay ◈ **£27.95** | **opencircuitsbook.com**

By Ben Everard @ben_everard

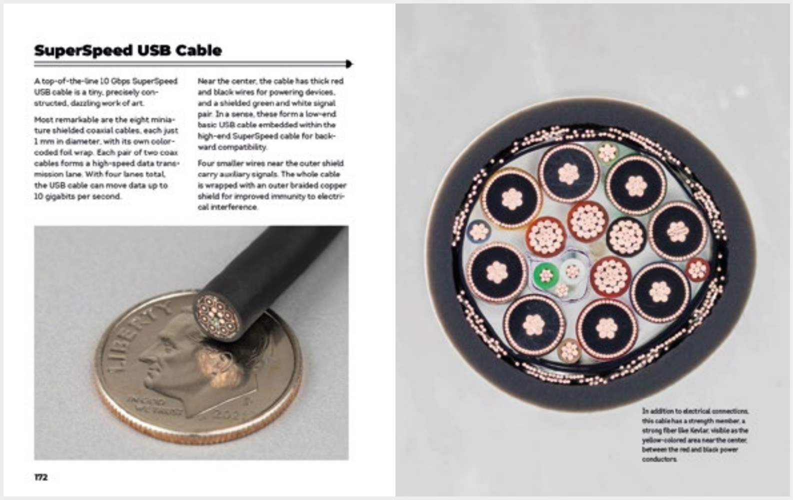

OPEN CIRCUITS
The Inner Beauty of Electronic Components
Eric Schlaepfer and Windell H. Oskay
no starch press

Though it's fallen out of favour now, there was a time when the term 'micro' was an adjective almost synonymous with electronics. It wasn't really used in its scientific meaning, but more generally. It was used to describe circuits too small to easily see what was going on, whether they were packed inside a chip or crammed into a circuit board and hidden inside a box.

From this 'micro' era of the 1980s onwards, electronics have been mostly hidden from view. Sometimes this is by design – to allow designers to make fancy-looking cases – and sometimes it's just because they're too small to make sense of. *Open Circuits* fights back against this gradual trend towards invisibility. It takes the invisible and celebrates its beauty.

Open Circuits is a book about electronics components unlike any other we've seen. The basic premise is to cut open components and use macro photography to show how they work, and show off the hidden details of the design.

The chapters go through the various different classes of component, including passives, semiconductors, and retro-tech. In each part, components are very carefully sliced in half and photographed in incredible detail. The afterword details the process they took, which included encasing in resin (for some components) and then sanding or slicing the part in half to expose what worked. A technique known as 'focus stacking' enabled Schlaepfer and Oskay to get as much of the part in focus as necessary.



SuperSpeed USB Cable

A top-of-the-line 10 Gbps SuperSpeed USB cable is a tiny, precisely constructed, dazzling work of art.

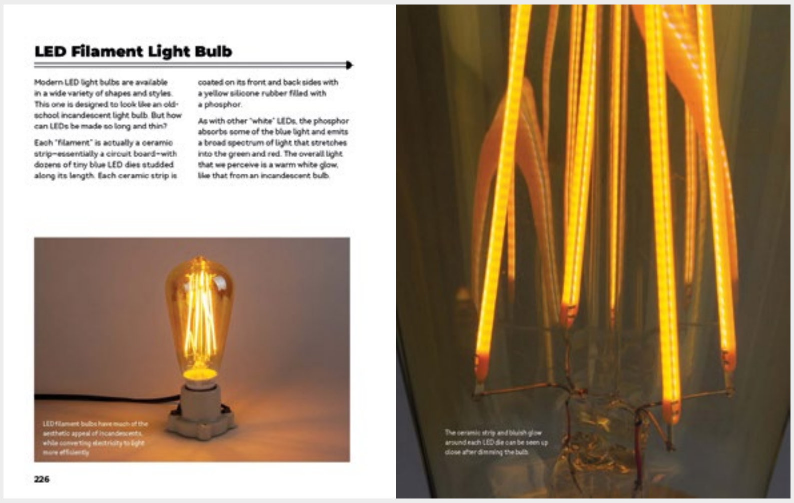
Most remarkable are the eight miniature shielded coaxial cables, each just 1 mm in diameter, with its own color-coded foil wrap. Each pair of two coax cables forms a high-speed data transmission lane. With four lanes total, the USB cable can move data up to 10 gigabits per second.

Near the center, the cable has thick red and black wires for powering devices, and a shielded green and white signal pair. In a sense, these form a low-end basic USB cable embedded within the high-end SuperSpeed cable for backward compatibility.

Four smaller wires near the outer shield carry auxiliary signals. The whole cable is wrapped with an outer braided copper shield for improved immunity to electrical interference.

172

In addition to electrical connections, this cable has a strength member, a strong fiber like Kevlar, visible as the yellow-colored area near the center, between the red and black power conductors.



LED Filament Light Bulb

Modern LED light bulbs are available in a wide variety of shapes and styles. This one is designed to look like an old-school incandescent light bulb. But how can LEDs be made so long and thin?

Each "filament" is actually a ceramic strip—essentially a circuit board—with dozens of tiny blue LED dies studded along its length. Each ceramic strip is coated on its front and back sides with a yellow silicone rubber filled with a phosphor.

As with other "white" LEDs, the phosphor absorbs some of the blue light and emits a broad spectrum of light that stretches into the green and red. The overall light that we perceive is a warm white glow, like that from an incandescent bulb.

LED filament bulbs have much of the aesthetic appeal of incandescents, while converting electricity to light more efficiently.

226

The ceramic strip and bluish glow around each LED die can be seen up close after dimming the bulb.

The authors don't stop at just individual components, they also look at composite devices and circuit boards. *Open Circuits* is, in its own destructive way, a guide to how electronics (both vintage and modern) are created. While manufacturing techniques aren't explicitly covered, you can see the laser trimming lines, the drill holes, and the epoxy blobs that all go together to create the electronics that power our everyday lives. Through these details, we can get quite a bit of information about how the parts are made. Through the range of parts looked at, it covers both the cheaper and higher-end spectrum of components.

"It's a chance to see things that **are normally hidden from view**"

It's a book full of surprising things, but perhaps the most surprising is what parts look most spectacular when pulled apart. The always-impressive Nixie tubes are interesting, but they didn't capture this reviewer's eye as much as the usually humdrum SuperSpeed USB cable or carbon film resistor.

Usually, we review things that will help you make your projects, but this isn't like that. Will you design circuits differently after having read this book? Maybe. It's possible you'll have a slightly better appreciation of the range of capacitors available. This author at least has a strong urge to build something involving a 3656 Isolation Amplifier purely because it looks just so damn beautiful. However, really, this isn't a book that's useful – it's a book that's interesting.

It's a chance to see things that are normally hidden from view, learn more about the manufacturing process, and marvel at the secret beauty of electronics. Perhaps this little peek inside the world of components will inspire you to build things in a way you wouldn't normally. Perhaps it'll be a companion to soothe your troubled soul when the bloody circuit just won't do what the bloody circuit should, and you're sure that the components are ganging up against you. Perhaps it'll sit on your coffee table or lab workbench for you to pick up and browse through in quiet moments. □

Above It's such a shame that this is usually hidden inside an unassuming grey package

VERDICT

Electronic components as you've never seen them before.

10/10

REVIEW

S1 Filament Dryer

Rejuvenate your old plastic

SUNLU ◆ **£35** | **sunlu.com**

By Jo Hinchliffe @concreted0g

If you've been 3D printing for a while, you've likely experienced the problems that can occur if your filament absorbs moisture. To a certain degree, all filaments are prone to absorbing moisture, which can create problems in your 3D prints. One of our early experiences of this was with a moist reel of cheap PLA that printed with a poor surface finish and, when left loaded in the machine, would simply snap between the reel and the machine. Another issue we often have is when printing PETG. Sometimes you can hear the slight bubbling and sizzling sound of moist PETG, and that burning off of the inherent moisture can lead to tiny bubbles and imperfections in your print.

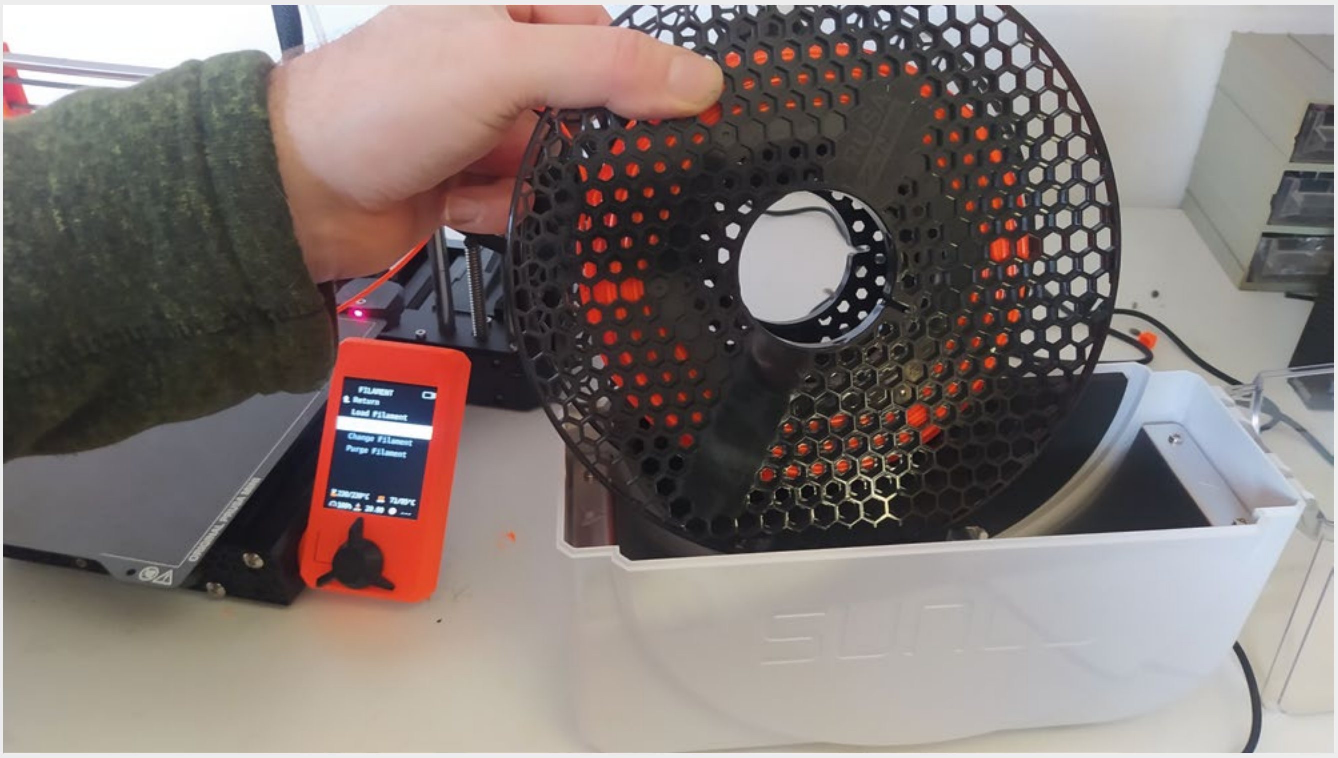
Whilst returning filament to sealed storage containing silica gel can mitigate this problem somewhat, filament dryers have become a useful and practical addition to filament printing. There are lots of DIY approaches out there, from placing filament reels in ovens at low temperatures to placing dehumidifiers in large plastic boxes – this author has been known to place a moist reel near the wood-burning stove when desperate! The SUNLU S1 Filament Dryer is a cheap and compact unit that aims to dry single reels of filament whilst not taking up much of a footprint on your workbench.

It arrives fully assembled and is well-packed in a cardboard box. It's a pretty straightforward machine to operate. The upper clear section hinges open, and you can place a standard filament reel (1 kg type, you won't fit a large 5 kg reel in here) inside placed on two metal roller beams that allow the filament reel to rotate when printing. There is a choice of two holes to feed the filament out of: one on the side at the point where the upper and lower sections meet, and one on top. This works well in practice and gives you a range of options as to where to place the device in relation to your printer.

It comes supplied with a 24 V DC power supply, and when powered up, the lower half of the unit, which is lined with a metal foil, begins to heat up to the set temperature. You can set the temperature between 35 °C and 55 °C, and you can also set the heater timer for between 1 and 24 hours. The settings are visible on the 50 mm square LCD panel on the front of the unit. The LCD is backlit and is pretty readable from the front, but tricky to read from side angles, which might be a slight issue depending on where you place the dryer. There's a cheap-looking plastic covering panel on the LCD, which is a shame as, other than that, it's quite a smart-looking unit.

To adjust the temperature, you can press either button and then use the left button to lower and the right button to raise the target temperature. To set

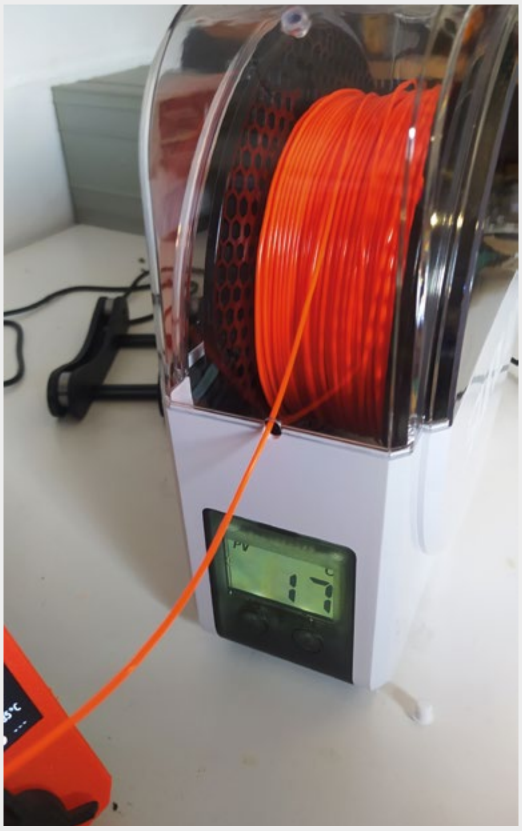
Below ◆
With its clean lines and plain colours, the SUNLU S1 Filament Dryer looks pretty smart on the worktop



the time, you long-press the left button and then adjust using the left and right buttons. It's pretty straightforward. We like the timer options as it means you can, perhaps, put the unit on for a few hours before printing, or indeed with a moist reel, you can set it up to run for a full 24 hours to try and drive out the moisture.

Speaking of driving out the moisture, one criticism of these dryers is that they don't vent and allow the moisture to truly move away from the reel. In fact, after a long session drying a very moist reel of PLA, we could see slight condensation on the inside of the unit. It's simply solved by wiping the unit out to remove the moisture, but we also see a lot of users placing silica gel packets in the reel centres so that there is a way for driven-out moisture to be captured. This seems like an excellent additional approach to take.

When printing, we've not noticed any issue with the filament feeding out of the unit – there are some small lengths (5 mm or so) of PTFE tubing that can be fitted into the filament holes in the dryer, but we haven't had any problems running the filament out of the unit without them. In our tests so far, we've certainly taken a cheap roll of PLA that was unusable back into good service, and it's cured our slight moisture issues with PETG, and has improved print quality. □



Above
Loading a reel is straightforward, but you are limited to 1 kg reels

Left
There's a choice of two holes for the filament, which allows for a range of setup positions

VERDICT

For the price, this is an excellent unit to try if you think moisture is affecting your filament and prints.

8/10

CROWDFUNDING NOW

Gliss

Swipe your way to beautiful music

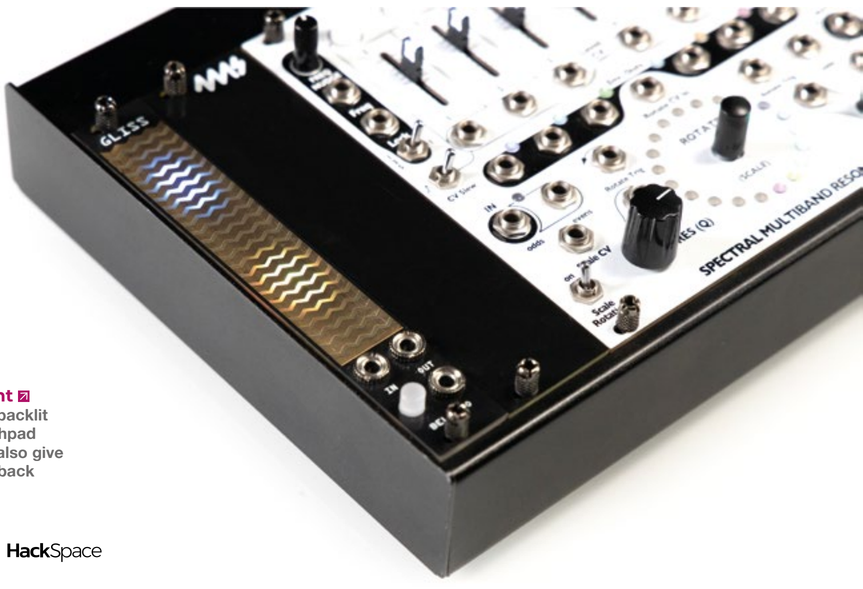
From $139 | **crowdsupply.com** | **Delivery:** May 2023

Many microcontrollers can work with capacitive touch sensing. This is where they sense the change in capacitance on an I/O pin that happens when a person touches it. It's fairly easy to create capacitive 'buttons' that are triggered when a person touches them. In principle, you can connect lots of capacitive sensors together to create touchpads that can tell where they're being touched. However, getting this to work reliably can be tricky.

Fortunately, the folks at Bela have done the hard work for you. Its Trill line of products has touch sensors in all sorts of shapes and sizes that are accessible over I2C, so you can add them to your projects easily. It's now brought that expertise to the modular synth world.

The front panel of Gliss is dominated by a touch panel that lets you draw in gestures of up to a minute that can then be looped as a low-frequency oscillator (LFO) waveform. A strip of lights behind the panel means it can be used as an output as well as an input. Of course, you do need the rest of a modular synth to make all this work.

If you like the idea of capacitive interfaces, but don't have a spare wall to dedicate to making strange-sounding music, you could start with Trill. Bela is launching a new and improved Trill line-up as part of this campaign, so take a look at the Crowd Supply page for details.

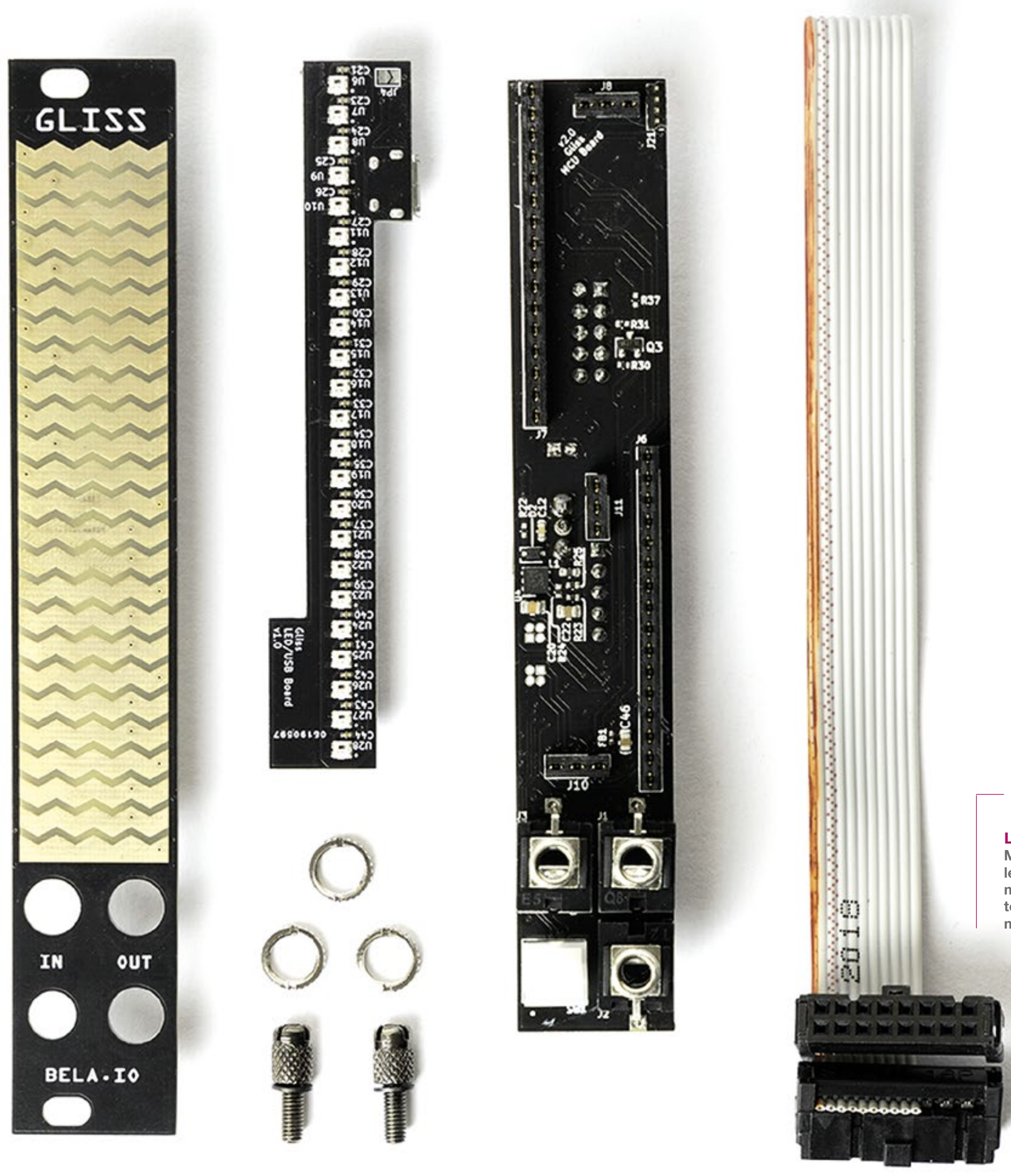


Right The backlit touchpad can also give feedback

"The front panel of Gliss is dominated by a touch panel that lets you draw in gestures"

BUYER BEWARE

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



Left Modular synths let you connect multiple devices together to make sound next

issue #64

ON SALE
16 FEBRUARY

SOLDERING IRONS

ALSO

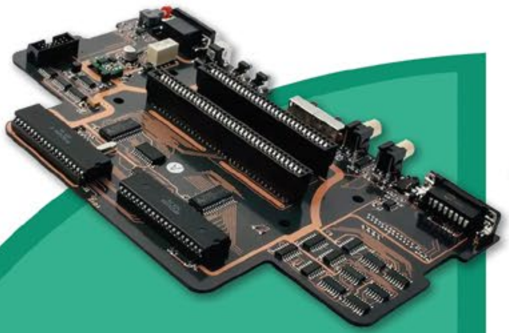
→ LEDS
→ PICO W
→ 3D PRINTING
→ WOODWORK
→ AND MUCH MORE

DON'T MISS OUT

hsmag.cc/subscribe

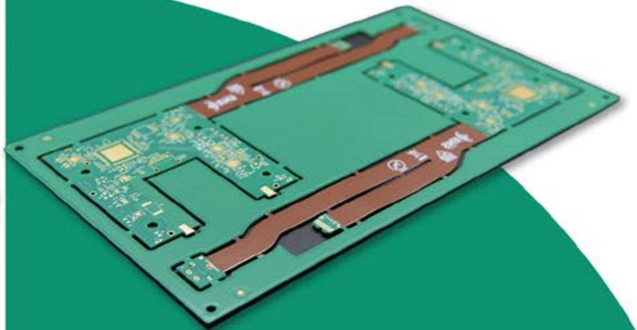
PCBWay

— Custom PCB Service —



- High-Quality PCBA with component sourcing
- Start from only $30
- Free stencil & Free shipping all over the world

Turnkey PCB Assembly



- Instant quote online & User-friendly web interface
- Fast turnaround in 24 hours
- Make your unique and high-end products the easy way

Advanced PCB Fabrication

Quality Guaranteed

3D Printing & CNC Machining

- Custom metal & plastic parts for only $25
- On-demand production and rapid prototyping in as fast as 1 day leading the digital manufacturing revolution.

More Custom Options

- Professional Flex/Rigid-Flex PCB
- New soldermask colors: Pink, Gray, Orange and Transparent
- Perfectly implement your idea with different types of PCB

More information:
www.PCBWay.com

Email:
service@pcbway.com

Phone:
(0571) 8531 7603

HiPi.io

PiKVM

Manage your servers or
workstations remotely

A **cost-effective** solution for data-centers,
IT departments or remote machines!

PiKVM HAT
for DIY and custom projects

Pre-Assembled version

● Real-time clock with rechargeable super capacitor ● OLED Display ● Bootable virtual CD-ROM & flash drive ● Serial console ● Open-source API & integration ● Open-source software

Available at the main Raspberry Pi resellers

core ELECTRONICS · PiShop.ca · RPishop.cz · Welectron. · elektorstore · PIMORONI
Pi-Shop.ch · The PiHut · PiShop.us · CloudFree · PiShop.mx

Reseller suggestions and inquiries:
wholesale@hipi.io