MAKE | BUILD | HACK | CREATE

HackSpace

TECHNOLOGY IN YOUR HANDS

hsmag.cc | November 2022 | Issue #60

BUILD THE

INTERNET OF THINGS

Homemade projects to connect your life

Issue #60 £6
Nov. 2022

PINHOLE CAMERA

Inside the DSLR of the olden days

SHADOW CLOCK

Master time with a Raspberry Pi Pico

FLAT-PACK ROCKETRY

Build flying machines the IKEA way

TILES PLANTS KEYBOARDS LEDS

Welcome to HackSpace magazine

The Internet of Things is, in theory, full of promise. It can mean our devices talking to each other and creating a seamless user experience – your alarm clock and coffee maker working together, your heating and car's navigation system making sure your house is warm when you arrive home, your washing-machine and solar panels making sure you clean your clothes when there's free electricity. However, somehow it never quite ends up like this. At least, not with off-the-shelf equipment.

Never fear, though. We are makers, and if we can't buy what we want, we'll make it. In this issue, we're taking a look at some of the best, most useful, and most innovative DIY smart home gadgets and gizmos. They're here for you to take inspiration from to build your very own high-tech haven. After all, it's 2022. We may not have flying cars yet but we can, at least, make our appliances talk to each other.

In this issue, we're taking a look at some of the best, most useful, and **most innovative DIY smart home gadgets and gizmos**

BEN EVERARD
Editor ben.everard@raspberrypi.com

PAGE 42
FREE PICO W
WHEN YOU SUBSCRIBE

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

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, Ty Logan

Photography
Brian O'Halloran

Cover design
Sam Alder

CONTRIBUTORS

Jo Hinchliffe, Marc de Vinck, Andrew Lewis, Mike Bedford, Phil King, Rosie Hattersley, PJ Evans

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

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

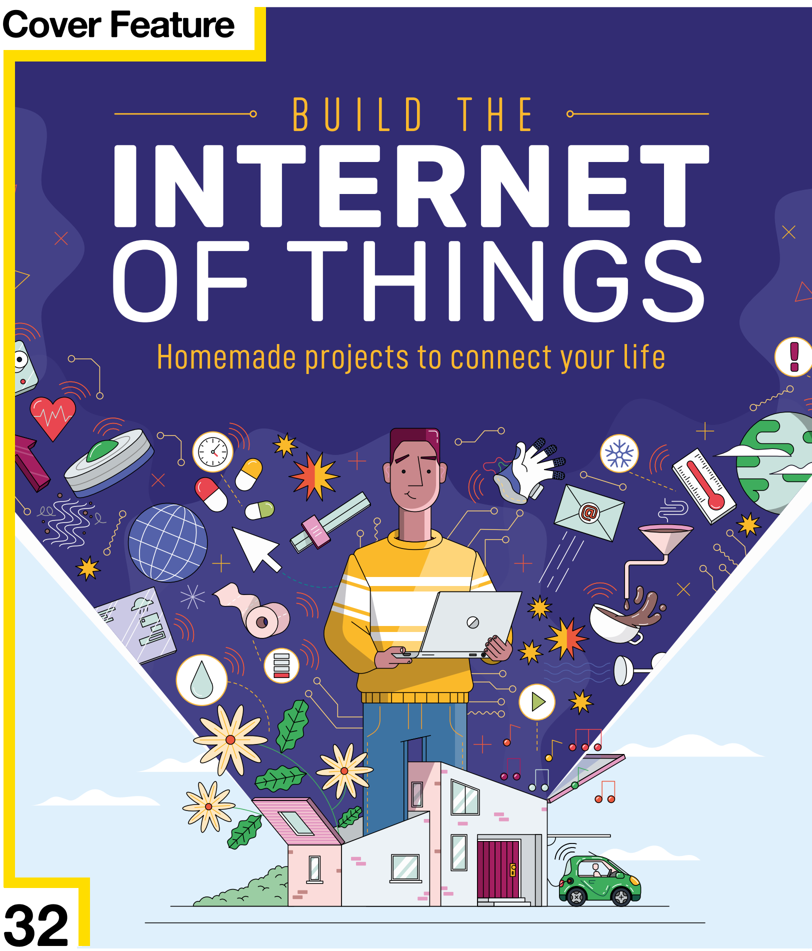
HackSpace magazine is published by Raspberry Pi Ltd, Maurice Wilkes Building, St. John's Innovation Park, Cowley Road, Cambridge, CB4 0DS The publisher, editor, and contributors accept no responsibility in respect of any omissions or errors relating to goods, products or services referred to or advertised.

ISSN: 2515-5148.

Contents

102

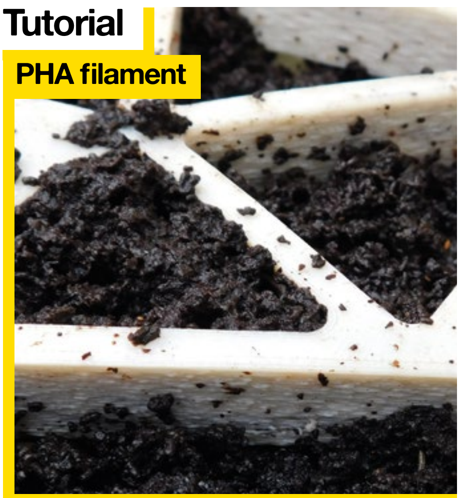
Cover Feature



32

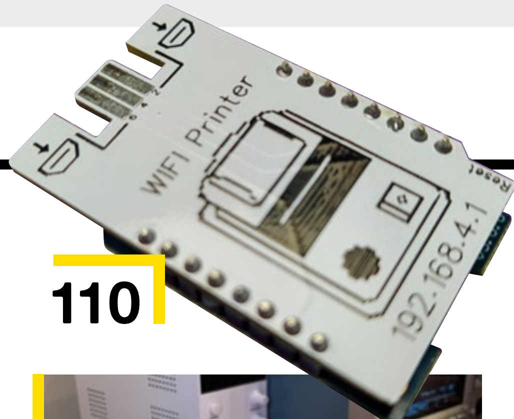
Tutorial

PHA filament



72 Next year our tomatoes will be grown in composted filament

94



Review

Bittle

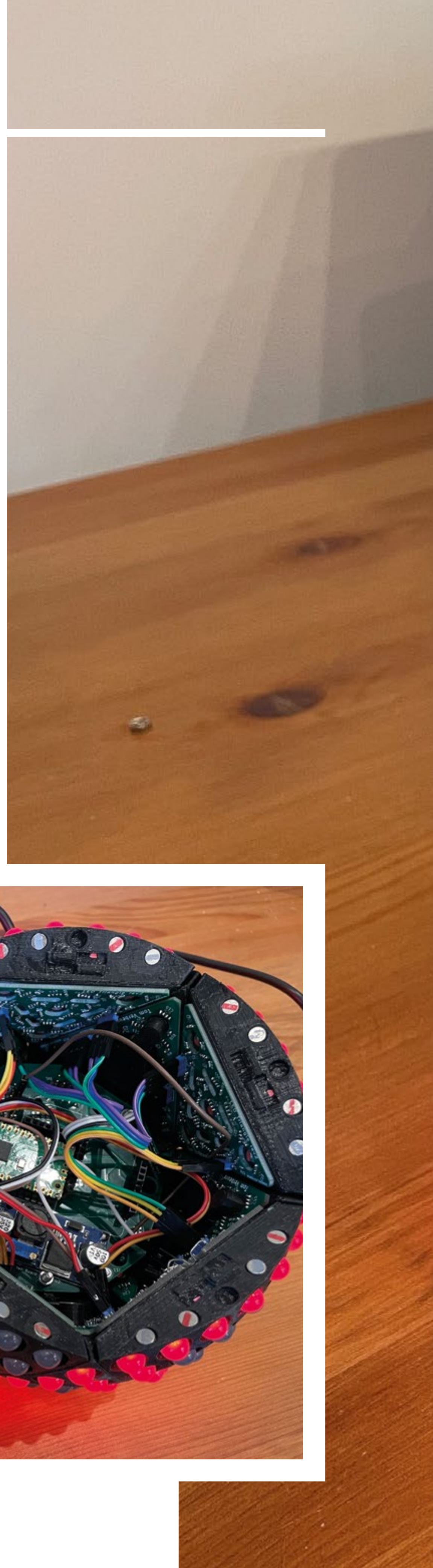
Interview

Debra Ansell

67 FORGE

101 FIELD TEST

HackSpace magazine is published monthly by Raspberry Pi Ltd, Maurice Wilkes Building, St. John's Innovation Park, Cowley Road, Cambridge, CB4 0DS, United Kingdom. Publishers Service Associates, 2406 Reach Road, Williamsport, PA, 17701, is the mailing agent for copies distributed in the US. Application to mail at Periodicals prices is pending at Williamsport, PA. Postmaster please send address changes to HackSpace magazine c/o Publishers Service Associates, 2406 Reach Road, Williamsport, PA, 17701.

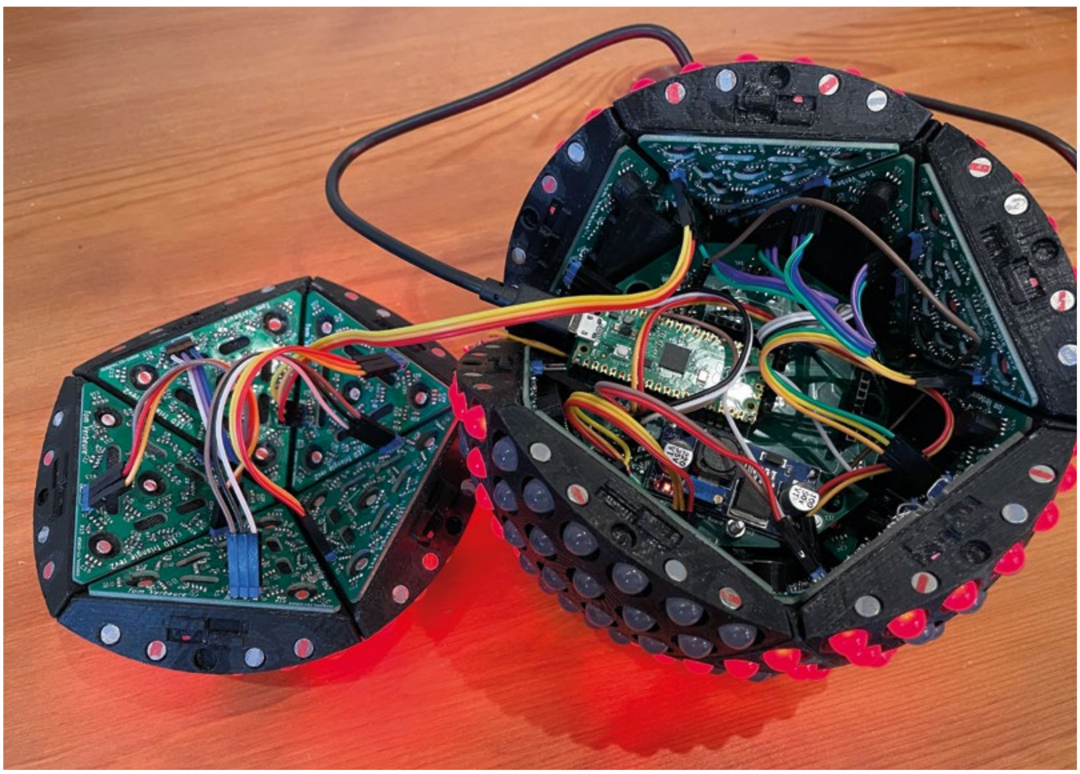


LED Sphere

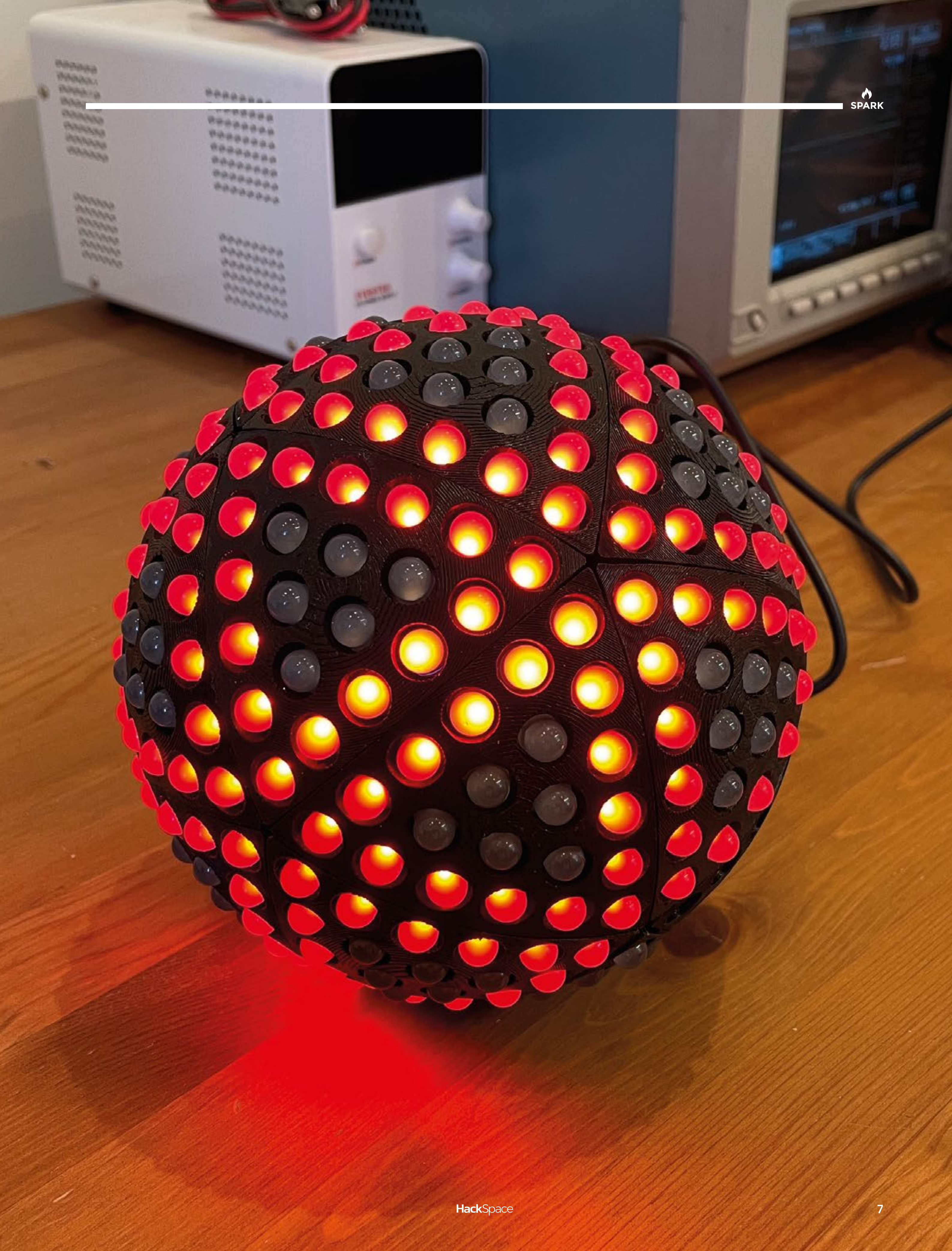
By Tom Verbeure

github.com/tomverbeure

In our interview with Debra Ansell, on page 50, we briefly discuss some of the challenges of making an LED sphere. Debra solved this by arranging surface-mount LEDs on a flat plane, then diffusing the light through curved 3D-printed parts. Tom Verbeure has gone down a different route with this creation, using larger through-hole components placed into a 20-segment, 3D-printed sphere. There's always more than one way to do it, and no one way is better than any other, as this build shows.



Right Tom's build is hollow in the middle, and is battery-powered

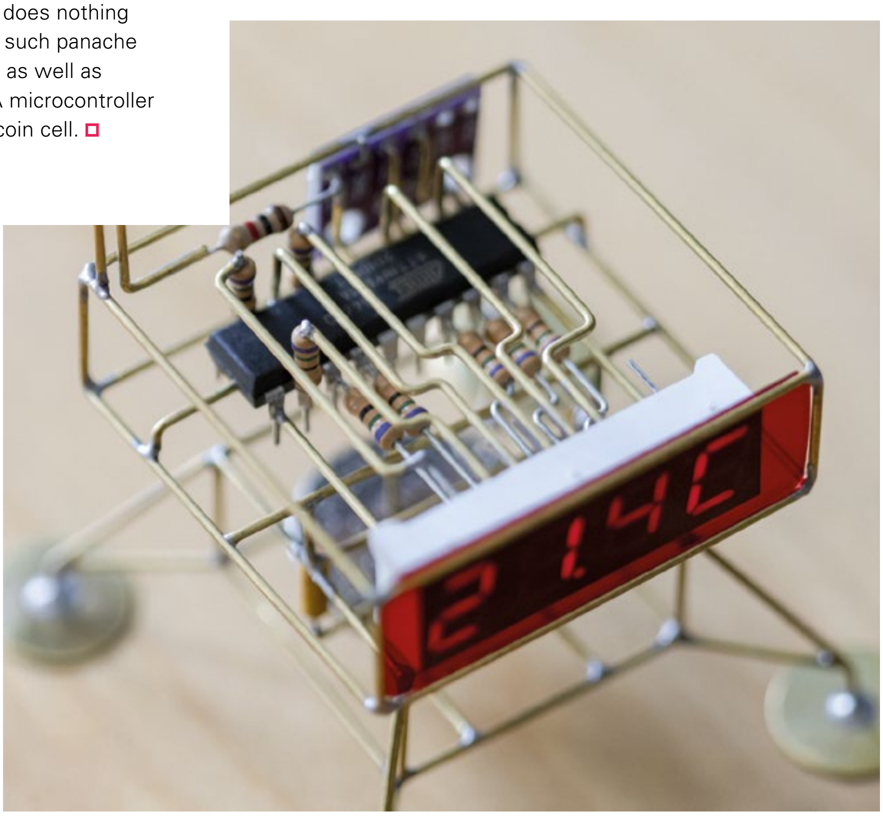


Free-form circuit structure

By Mohit Bhoite bhoite.com

Individually cut brass rods must be a pain to work with. Unlike copper sandwiched between layers of PCB substrate, copper wires aren't obliged to stay where you put them; just a single wobble, a single component out of place makes the whole thing look wrong.

Knowing that, and how hard it is to produce a free-form circuit sculpture, makes us appreciate Mohit Bhoite's work all the more. This tiny model lunar lander does nothing but display the temperature, but it does it with such panache that we've got to include it here. Components, as well as function, are minimal: he's used an ATtiny861A microcontroller and a BME280 sensor, powered by a CR2032 coin cell.



Right Since building this sculpture, Mohit has made a version with a binary, rather than a seven-segment, display

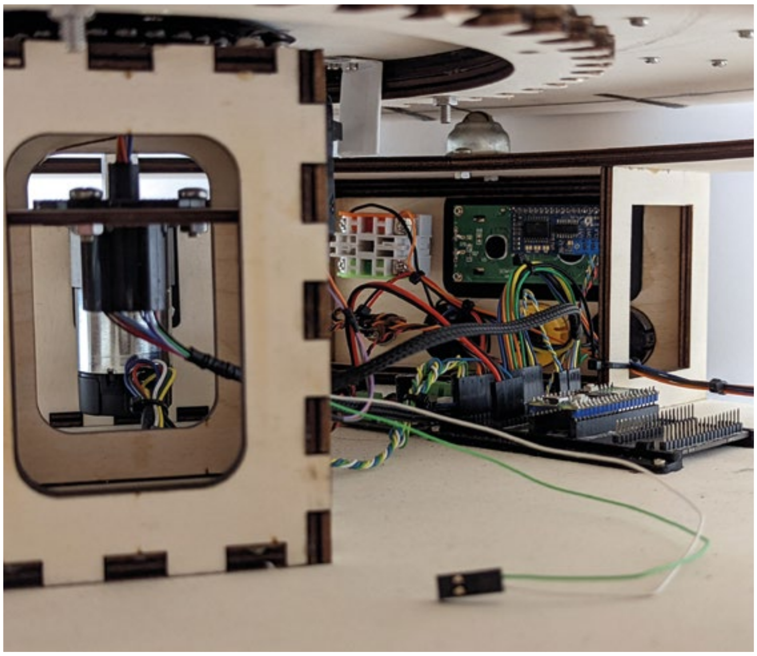

12.5C

Digital zoetrope

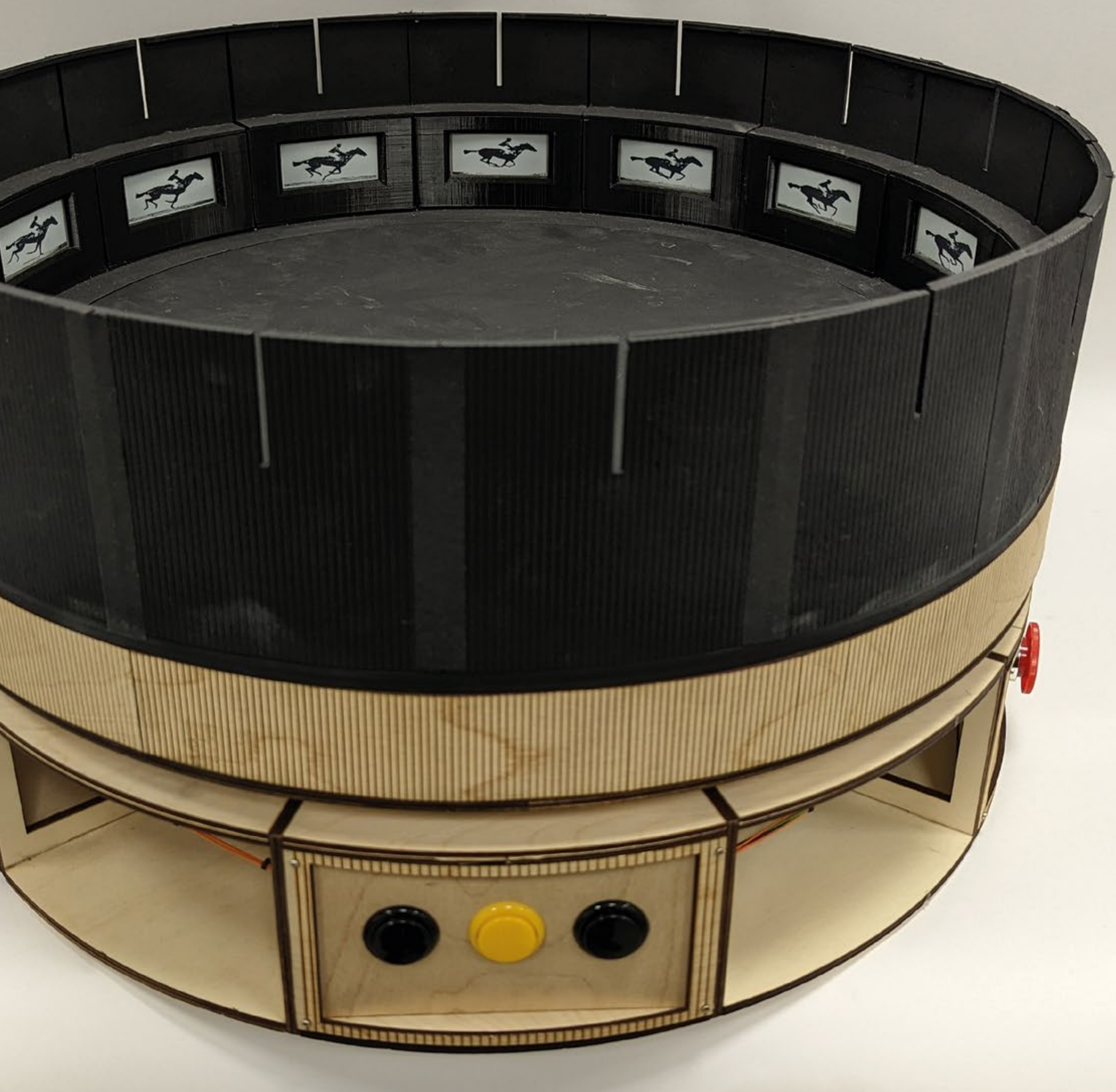
By **Brian Corteil** twitter.com/CannonFodder

Before we invented television, humans were grasping at moving images for a while. One forerunner to the idiot box, which uses persistence of vision and a series of slightly different images to create the illusion of movement, is the zoetrope.

Prolific maker Brian Corteil has made this beautiful digital zoetrope, which uses 20 Pimoroni Badger E Ink® screens to display the images, a Raspberry Pi 4, and a beautiful body fabricated from laser-cut 3 mm and 5 mm plywood.



Right Because it's digital, users can update the images with their own choice of animation

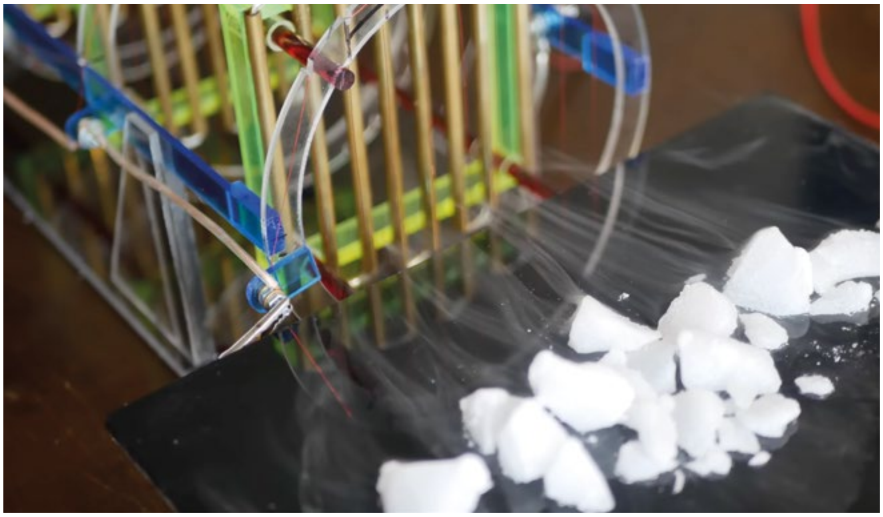


Ion Thruster

By Jay Bowles youtube.com/c/PlasmaChannel

Ion thrusters are fascinating things, producing tiny amounts of thrust that are pretty much useless on Earth (with our inconvenient gravity). They're far more useful in space however, as they're smaller than rocket engines, and they don't need atmospheric oxygen like a jet engine.

Ion thrusters are old technology, having been used since the 1950s, and now Jay Bowles, of the fascinating YouTube channel Plasma Channel, has demonstrated a way to make it better: multiple stages. By varying the input voltages and spacing between stages, Jay's been able to experiment and produce a claimed exhaust velocity of 2.3 m per second.



Right It still amazes us that one person working alone can create space-grade technology

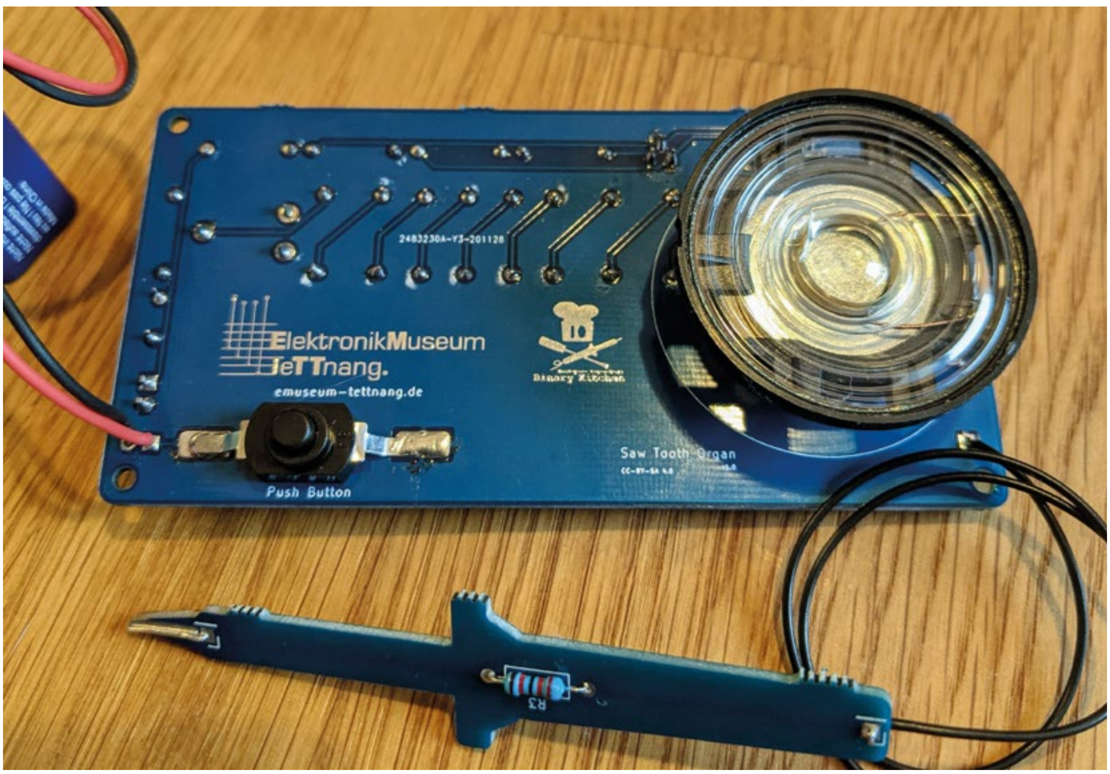
Sawtooth organ

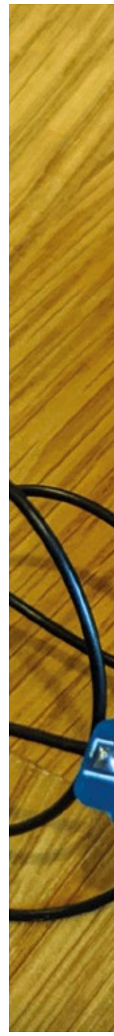
By **Blinkyparts**

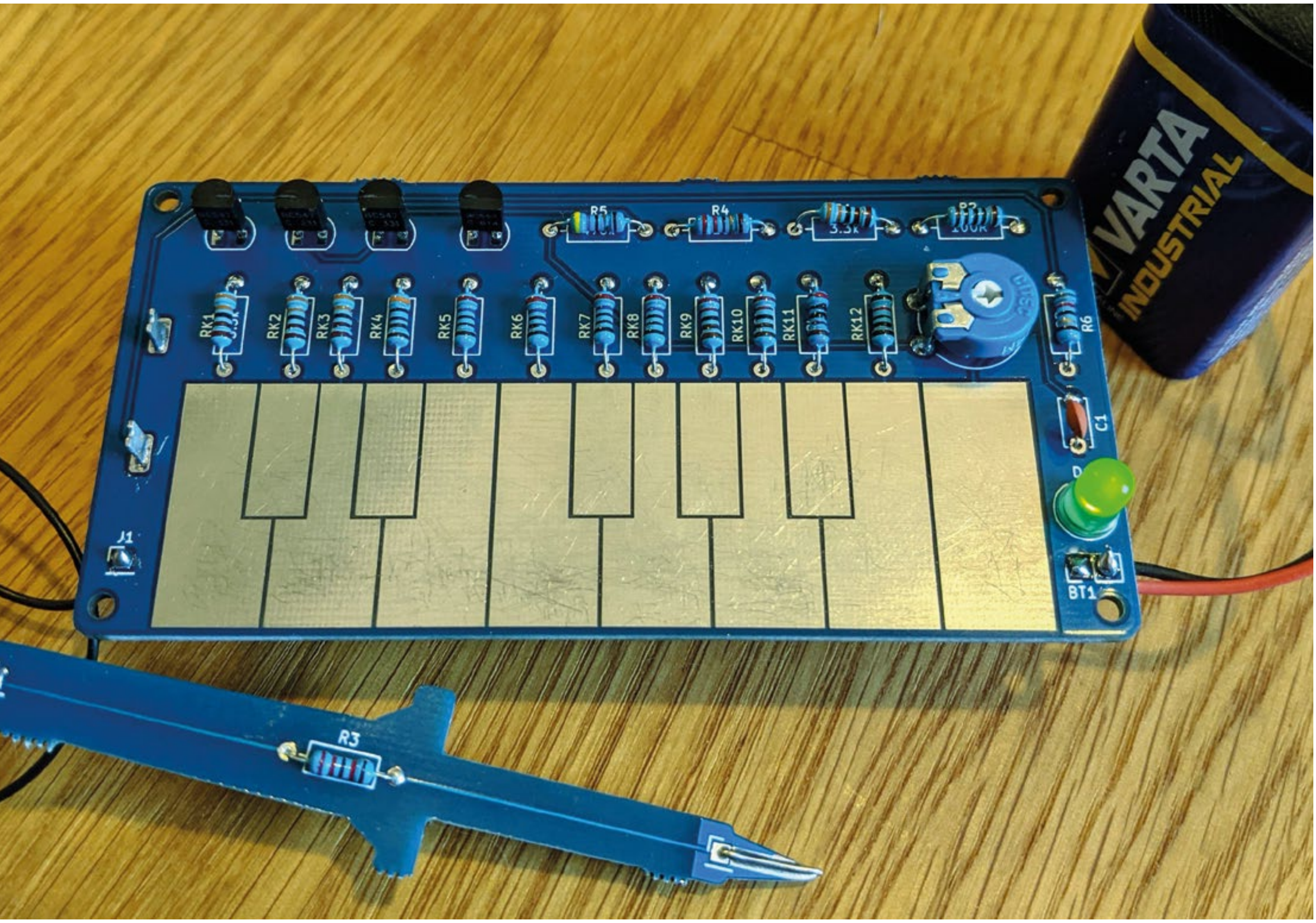
hsmag.cc/SawtoothOrgan

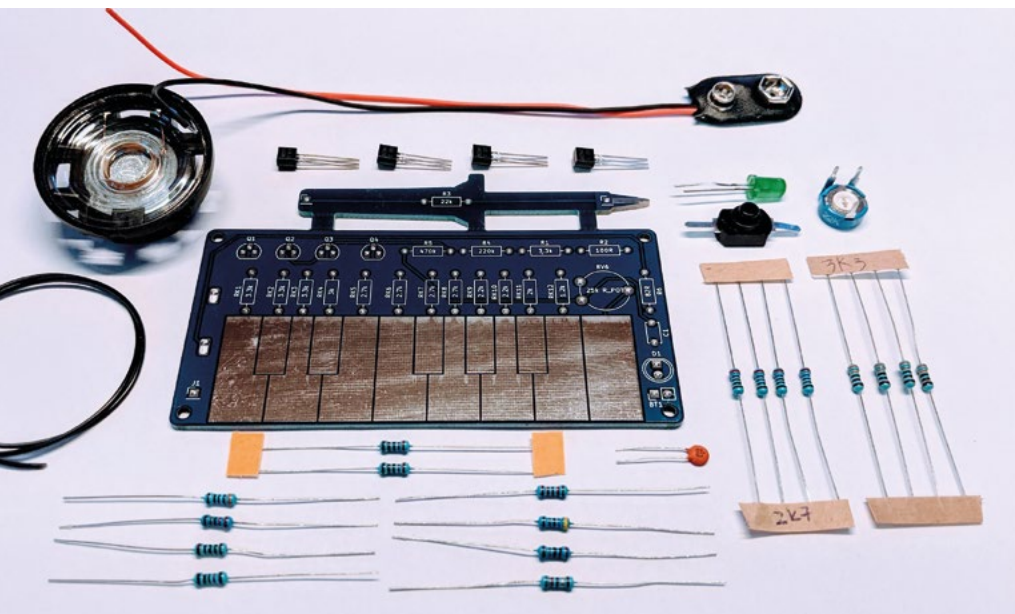
Electronic music is a fascinating hobby. You learn an awful lot, you spend a fortune on components and, in the end, you get a small box that makes a wobbly noise that's barely different from all the other circuits you've spent hours putting together.

Jokes aside, the learning curve for homemade analogue synths can be incredibly intimidating. So this circuit, which is priced at $9.99 on Tindie, is an inviting breath of fresh air. It's a sawtooth organ, which uses just a handful of components to make notes when the user presses a stylus onto a keyboard. That's it, and the simplicity means that anyone with a soldering iron will be able to put it together and bash out simple tunes.

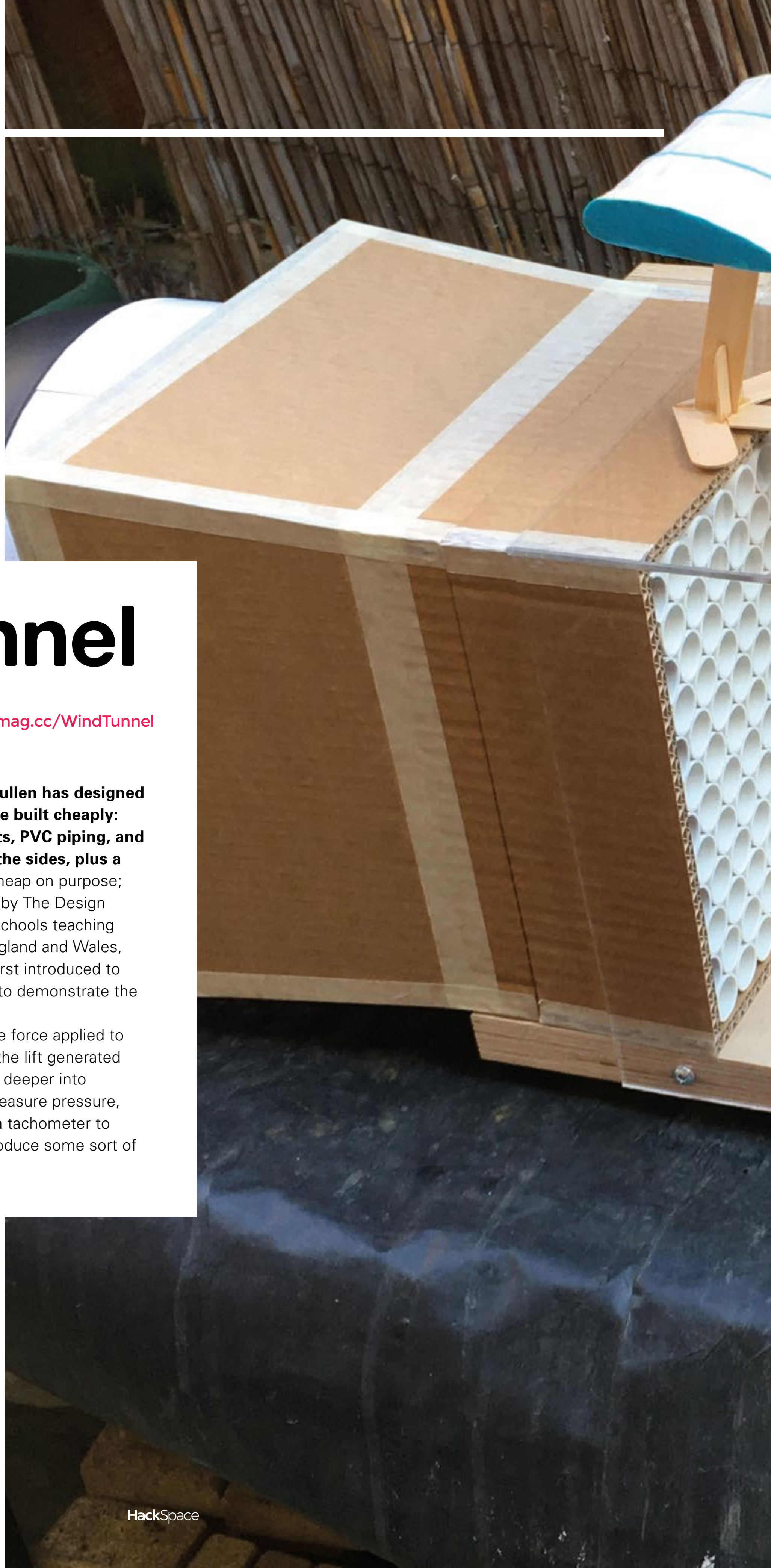








Above Similar devices were used by David Bowie and The White Stripes



Wind Tunnel

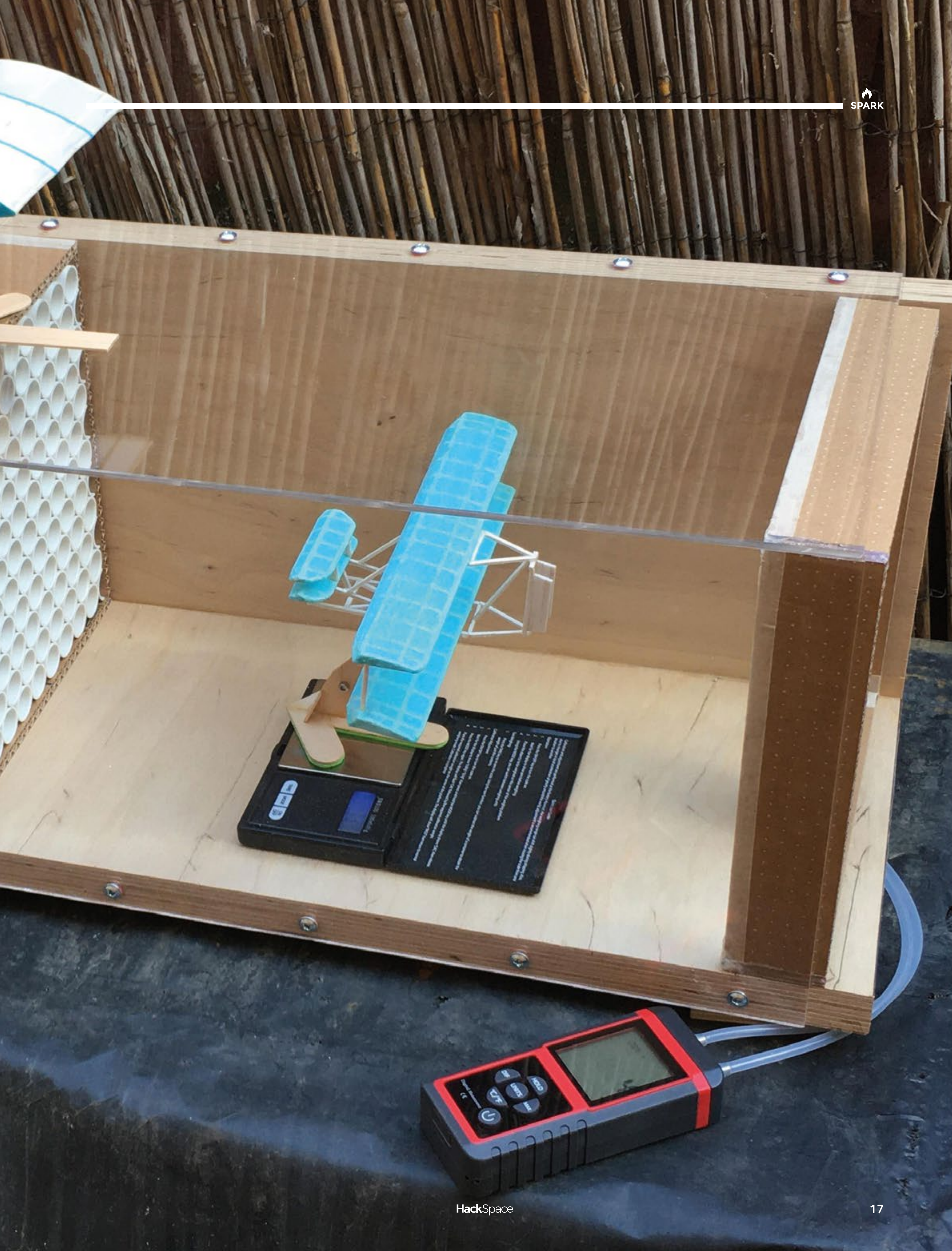
By Jude Pullen

hsmag.cc/WindTunnel

Designer extraordinaire Jude Pullen has designed this desktop wind tunnel to be built cheaply: it's made of card, wood offcuts, PVC piping, and some clear acrylic sheets for the sides, plus a set of digital scales. And it's cheap on purpose; this project was commissioned by The Design And Technology Association, which supports schools teaching design and technology. This subject has, in England and Wales, had its funding cut by over 70% since it was first introduced to the curriculum, so teachers need cheap ways to demonstrate the principles of design.

This is a minimal setup; all it measures is the force applied to the scales, from which the user can compare the lift generated by different wing designs. If you wanted to go deeper into engineering, you could add a manometer to measure pressure, an anemometer to measure wind speed, and a tachometer to measure fan speed. There's also scope to introduce some sort of vapour to visualise the airflow. □

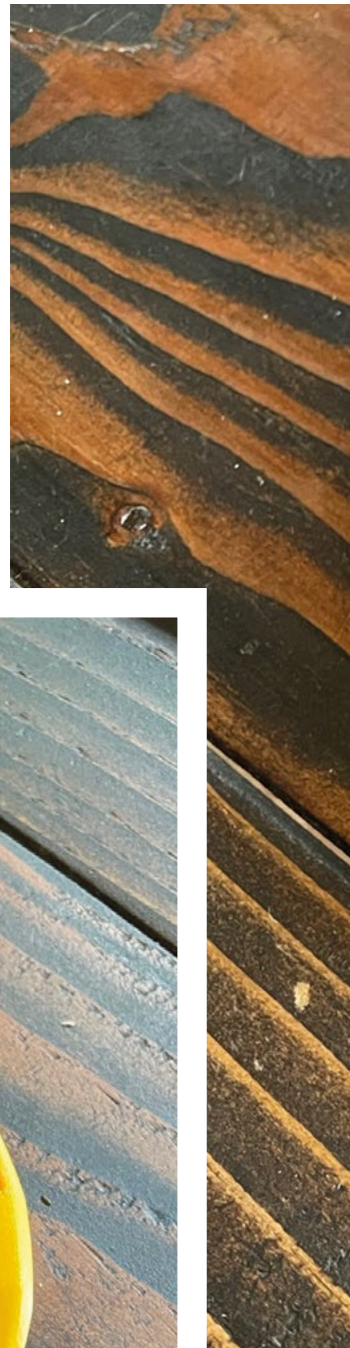
Right
Build your own aerospace engineering equipment for just £20



REGULAR

Objet 3d’art

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



If this build looks familiar, it's because its maker, Dan Bostian, has based it on his previous project, the Banana Macropad. That device comprised eight mechanical keyboard buttons and a wired USB connection, housed in the shape of the internet's favourite fruit, the banana. The Banana Split is all that and more: instead of a wired connection, this little keyboard uses Bluetooth, and it's split into two for more ergonomic banana usage. Yes, this device is silly, but it's bright yellow and we like it a lot.

hsmag.cc/Banana

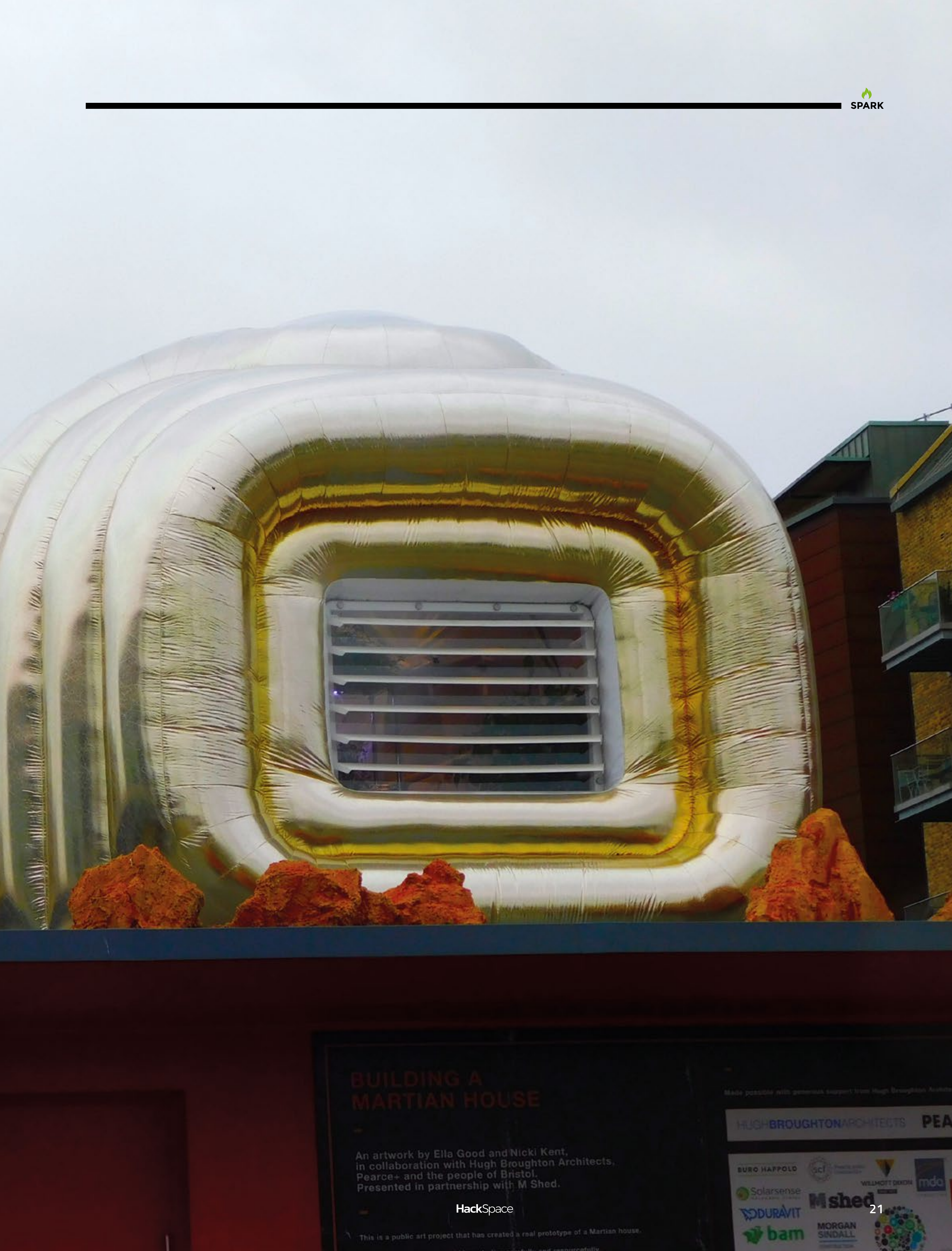
Green ideas for the red planet

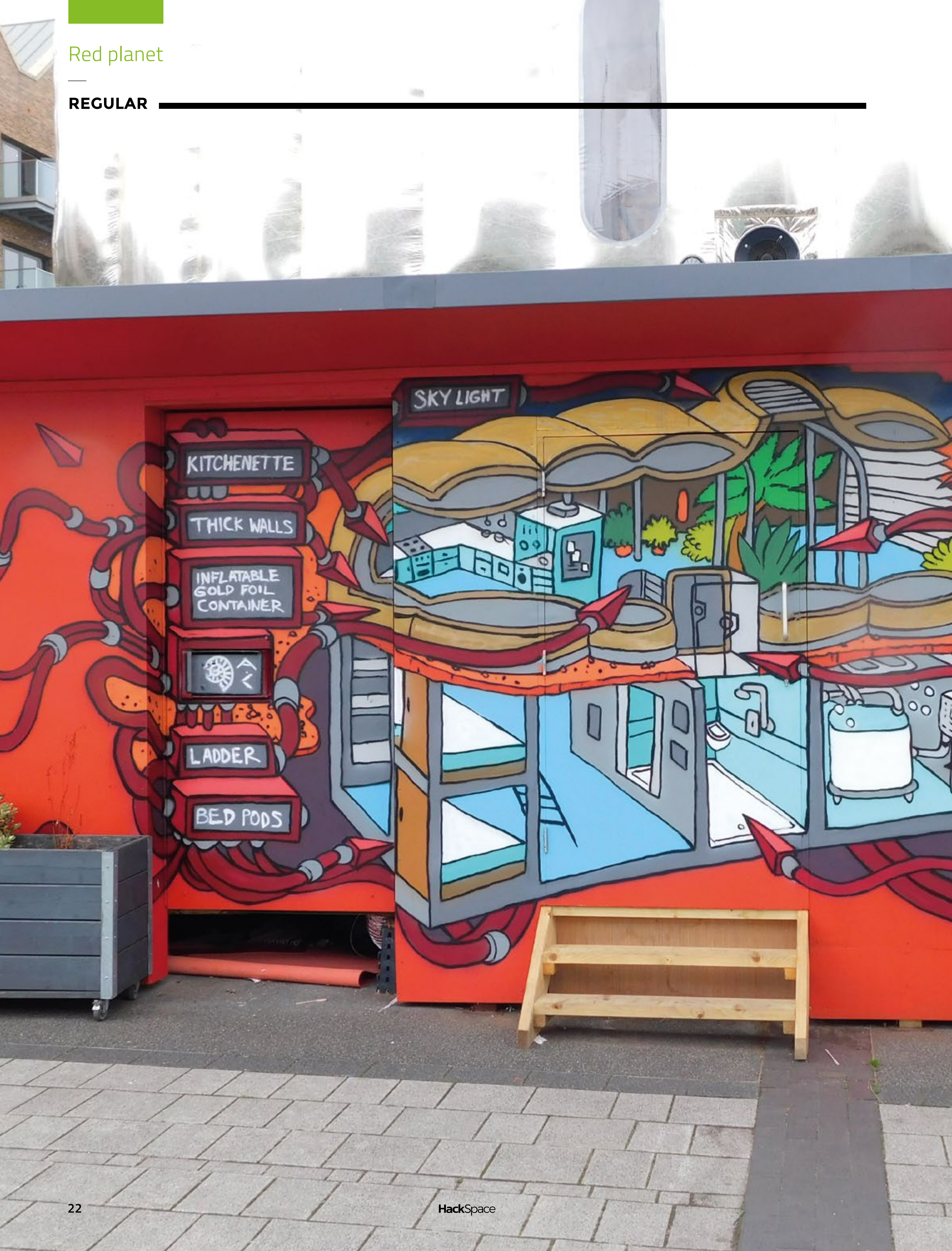
How would we live on Mars, and what can this teach us about sustainable life on Earth?

Sure, making stuff on Earth can be challenging, but what if you didn't have access to next-day delivery on kit and materials? What if you just didn't have access to a huge range of items at the click of a button? That's exactly what you'd have to do if you were a colonist living on Mars. It also happens that the same things – working with limited resources, thinking carefully about material choices, and reusing waste streams wherever possible – are also critical to sustainability here on our very own space rock, planet Earth.

Building A Martian House is a project by two artists – Ella Good and Nicki Kent – with collaboration from a large team of engineers, scientists, architects, and others, that aims to explore the challenges of living on Mars, while simultaneously exploring ways to not destroy Earth.

Right Is this what our first extra-terrestrial colony will look like?


BUILDING A
MARTIAN HOUSE
An artwork by Ella Good and Nicki Kent,
in collaboration with Hugh Broughton Architects,
Pearce+ and the people of Bristol.
Presented in partnership with M Shed.
This is a public art project that has created a real prototype of a Martian house.
HUGHBROUGHTONARCHITECTS
BURO HAPPOLD
Solarsense
DURAVIT
bam
M shed
MORGAN SINDALL
WILLMOTT DIXON
mdo


SKY LIGHT
KITCHENETTE
THICK WALLS
INFLATABLE GOLD FOIL CONTAINER
LADDER
BED PODS

The building itself looks a bit like an inflated silver-foil tent – basically, it is. The reflective foil is designed to protect the inhabitants. On Mars, the foil would be packed with soil (which can potentially be hardened with bacteria), but here on Earth – under our protective blanket of atmosphere – it's been inflated with air to allow it to be taken down and re-used elsewhere once this exhibit finishes. The two floors of the house are both above ground here, but on Mars, the upper floor would be at ground level, and the lower floor would be underground.

> "While much of the space is – out of necessity – very utilitarian, **there are some comforts**"

While much of the space is – out of necessity – very utilitarian, there are some comforts. The window on the upper floor is to let the inhabitants take stock of their surroundings. However, the limited protection from solar storms would mean that sometimes inhabitants would have to retreat downstairs for protection. Part of the project is thinking about how to balance the harsh environment with some little touches that, while not strictly necessary, help make life a little more pleasant.

This idea of balancing the emotional and social needs of humans with the constraints of off-Earth life is a recurring theme in the Martian House project. Should humans ever make it to Mars, this may well →

Left Could you live with just these essentials?


No parking in the hatched area
Up to £70 penalty charge
PLANTS FOR WELL BEING
CELEBRATING MARS
MEALTIME

Left On the right-hand side, you can just see the prow of The Matthew, a re-creation of the first boat (according to Bristolians) to reach the Americas. 525 years later, humanity is looking to a new frontier

prove to be a significant challenge as it's hard to analyse scientifically. There's no unit for relaxation, or equation to balance the joy of a sunset versus the pleasure of a hot bath. Push a person too far, and they'll crack, but if they require too much, then the mission may never get off the ground.

How much would you give for music? Would pre-recorded tunes be enough for you, or do you yearn to play or hear music live? How hard is it to turn scrap material into a working guitar? Using just 6g of his fictional four-monthly 20g allowance for guitar strings, Julian Parsons built a working electric guitar from scrap that could conceivably be found in a Martian colony (at least, for a price). The fretboard is made from wood inlaid with wire, the tuners are nuts and bolts, and the pickups are made from a broken electric motor.

Building instruments from cast-off materials is a process that's probably about as old as music, and there's no reason to think that it'll stop once we escape the confines of this green-and-blue orb. However, what form will Martian music take? Science fiction has led us to believe that electronic tunes will rule the airwaves of other planets, while Julian makes the case for a guitar. The reality will probably be dictated by the tastes and skills of the first settlers – will they be plucking strings or noodling with synthesizers?

Humans are fundamentally social animals, and one of the hardest parts of living on Mars might be the people you leave behind. Assuming there's not some mass exodus, the chances are that at least some of your loved ones will stay behind, if not permanently, at least until the colony is set up.

" Science fiction has led us to believe that electronic tunes **will rule the airwaves of other planets** "

While you can email, a handwritten note is a much more personal item, yet it's unlikely to be realistic to send paper backwards and forwards. Liz Lister explored the possibility of creating digital files from handwritten notes and then plotting them with pen remotely. Although this test didn't include sending it from space, it did create a more intimate connection than a UTF-8-encoded email.

You can read more about Julian's musical adventures, Liz's plotter tests, and other experiments around Martian life on the Martian House blog at **hsmag.cc/MartianHouse**.

While the Martian House was designed in consultation with scientists, engineers, architects, and people with a vast range of skills, the main benefits from the project – at least for us – aren't prototypes or specific bits of technology, but questions and frame of reference for asking them. In a way, it's perhaps a bit more like a live version of science fiction than part of the space program. □

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**

PIXELS WITH PURPOSE

Thanks for the interview with Alpenglow's Carrie Sundra; I was aware that citizens in Washington DC can't vote for president, but I had no idea that US citizens in the overseas territories can't, and I like to think I'm pretty clued up when it comes to politics. Thanks to her for highlighting this.

David
Boston

Ben says: One of the many things we like about Alpenglow is that it has come out of one person's desire to make things, and if other people like them too, then fair enough. There's no market research team that would have come up with a light-up capacitive touch swear-word-emblazoned unicorn, but Carrie brought it into the world, and the world likes it. Also, voting is important.

VORON

Voron reminds me of 3D printing before it got taken over by the buzzwords and hype of the MakerBot era. No, not every home is going to have one; no, a 3D printer cannot just magic you up any object you desire; it's not like the replicators in *Star Trek* or *Red Dwarf*.

What 3D printing is, or at least what it should be, is open, accessible, and accurate without being prohibitively expensive. I think the team behind Voron gets this, and I'm glad to see them getting some credit.

Brian
Kent

Ben says: **We agree, which is why we put Voron on the cover last issue. We'll be honest with you, it was a monumental task to print out all the components, but once we got it all put together, we can see why Voron is making waves: it's a leap forward in quality for the price. Of course, that's the financial price; the time it takes to print the parts should be considered as part of the total cost of ownership, but when you enjoy the process, it's all part of the fun.**

AQUAPONICS

I intend to follow your tutorial on aquaponics when I've cleared out an appropriate space in the garage. The variety of fresh salad leaves in the shops has taken a hit since Covid/Brexit/your supply shock of choice, and a fun, cheap, maker-friendly way of growing my own through the winter will be a great way to keep me fed. I don't know what I'm going to do with the fish, though…

Robert
Manchester

Ben says: **Five years ago, when we first discovered aquaponics, we were surprised by the addition of living animals to a traditional hydroponics setup (that's the key difference between the two). On a large scale, that makes sense, as with a decent amount of biomass in the system, you can grow fish large enough to eat. On a smaller scale, that's out of the question. Either way, we'd recommend the kitchen; you're effectively providing an environment for the fish, so it seems wrong to shut them away in the garage.**

CROWDFUNDING NOW

Raven CNC

A CNC that fits in your workshop

From $2729 | **hsmag.cc/Raven** | **Delivery:** May 2023

CNC routers are great bits of kit – they can cut out parts from wood with an accuracy that (we, at least) can only dream of. Simply load up your wood, press a button, and drink your coffee while your robot assistant carves out your work. Then, when it's finished, remove the work and take credit for it. Perfect.

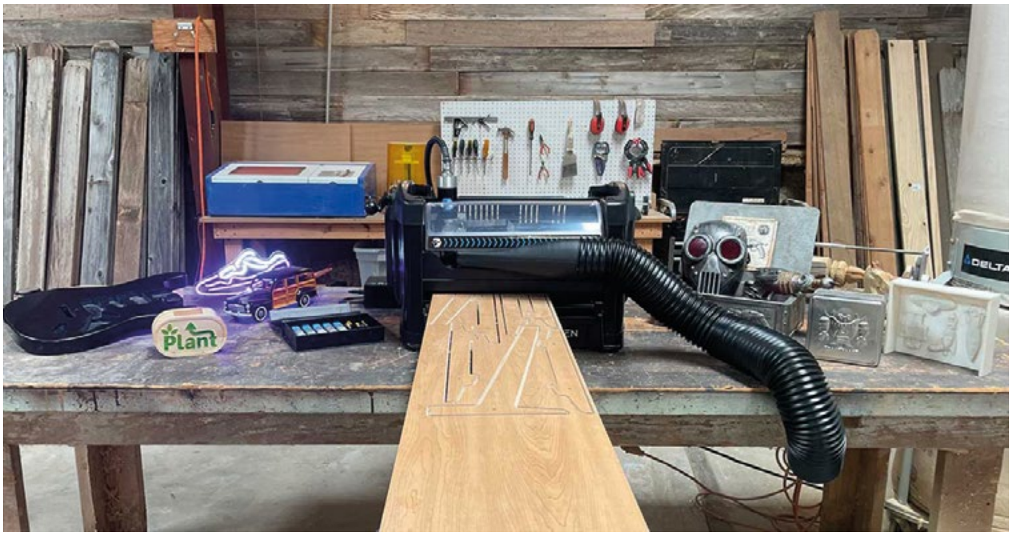
However, it's not quite perfect because CNCs come with one big caveat – they're huge. Unless you have a workshop like a YouTube wood influencer, you're probably going to struggle to fit in a CNC large enough to do serious woodwork (or if you do, it might end up being the entirety of the workshop).

Raven is an interesting solution to this problem. It's a 'feed-through' CNC, meaning that the cutting head moves across and the wood it's cutting moves through it, a bit like paper moves through a printer. While this does need a lot of space when it's working, it's a fairly small machine to tuck away when you're finished with it. The main downside of this approach is that you're limited to wood 39 cm wide (though it can be up to 365 cm long).

We've not tested this out, so don't know if it works as well as claimed, but we really want it to because it seems like a great solution to a very common problem.

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.



Above While 39 cm might seem narrow, you can fit a lot in

HackSpace
TECHNOLOGY IN YOUR HANDS

LENS

HACK | MAKE | BUILD | CREATE

Uncover the technology that's powering the future

AMAZING IOT PROJECTS

Be inspired by these wide-ranging projects to create your own **Internet of Things** devices, whether useful or just for fun

BY PHIL KING

The Internet of Things (IoT) enables devices to 'talk to each other' by exchanging data over networks. Commercially available IoT devices include smart fridges, central heating thermostats, fitness trackers, and video doorbells. But, with a single-board computer or microcontroller, you can create your own DIY IoT device, as the makers of the wide range projects covered in this feature have done.

One of the most common uses of IoT is in home automation, such as for automating the drawing of blinds and curtains. Garage doors, fans, central heating zones, and pet feeders can all be controlled remotely from afar, whether by human interaction on the web or automated from sensor readings or timings.

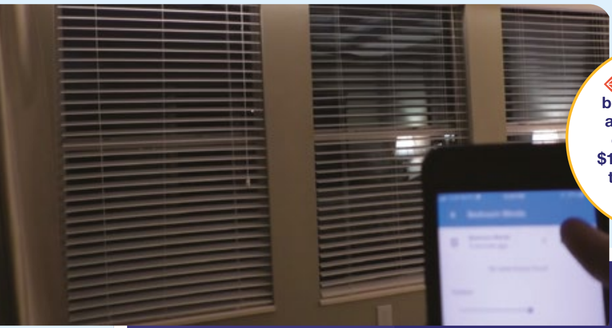
To make your daily household tasks easier, homemade smart appliances include a smart fridge that monitors its contents, along with a smarter robot vacuum cleaner, washing machine, coffee maker, mailbox, and even a bin that tells you when to empty it.

Some IoT projects involve the visualisation of data from the web, such as the classic magic mirror that tells you the weather and news as you check your hair in the morning. Others supply live railway information, the weather forecast, and even bin collection alerts. Or you can play internet radio stations from around the world by spinning a physical globe.

IoT devices like a smart pill dispenser and pothole detector have a genuine use case and social benefits, while others are made just for fun, such as a jelly bean-pooping rainbow unicorn! Whatever you want to make, with IoT, your imagination can run free. →

HOME AUTOMATION

Automate your home with the aid of **IoT** and you'll never need to draw the blinds again



Making your blind motorised and smart only costs around $10, and you can then control it wirelessly

Motorised MQTT Blind

MQTT is a popular lightweight network protocol used in many IoT projects, enabling the sending of machine-to-machine messages using a broker and one or more client devices. For this project, The Hook Up YouTube channel demonstrates how to use it to control a DIY motorised blind via WiFi, from Alexa, Home Assistant, or SmartThings.

A standard Venetian blind was motorised with the addition of a stepper motor – a previous attempt using a servo was abandoned because it whined continually while adjusting its position. A cheap unipolar motor was converted to bipolar to provide the necessary torque to tilt the blind slats to the full extent. A 3D-printed adapter was made to fit the motor to the blind's tilt rod.

A NodeMCU microcontroller receives MQTT messages and controls the stepper motor. Code and STL files for various size adapters and a mount are in the GitHub repo. Alternatively, you could try automating your curtains: **hsmag.cc/mqttcurtains**.

→ hsmag.cc/MQTTBlind

Smart Window Fan

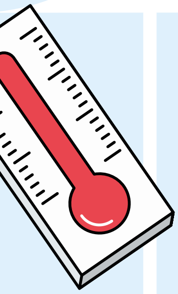
Keep your home cool during summer with this smart window fan which turns on automatically whenever the outdoor temperature is lower than that inside. Rather than using an exterior temperature probe, this project receives the latest data from **Weather.com**, which maker Ishmael Vargas says is accurate enough. Interior temperature is monitored with a DHT22 sensor connected to a Raspberry Pi Zero W running a Python script.

To keep things simple, the fan is connected to a smart WiFi power plug rather than a relay switch. A dashboard running on Raspberry Pi using Pygame can be accessed from a smartphone with the VNC Viewer app, enabling you to view info and set the target temperature.

For a more ambitious project, you could even tint your window automatically according to the light level, and control it via an iOS app: **hsmag.cc/smartwindow**.

A Raspberry Pi Zero is wired to a DHT22 sensor to monitor the room temperature; the external temperature is sourced from the web

→ hsmag.cc/WindowFan



ThermOS

Smart thermostats are becoming ever more popular, enabling you to control your central heating system remotely, but commercial off-the-shelf models can be expensive and have limitations. A DIY option like Joe Truncale's ThermOS offers more flexibility for less outlay.

Needing a new smart thermostat to replace the six old thermostats controlling his complex and ageing home heating system, Joe opted to build his own solution powered by a Raspberry Pi 4. With Apple HomeKit integration, he can control his system's six heat zones for different rooms from his iPhone.

The Raspberry Pi is the brains of the operation, programmed in Python and making use of the HAP-Python HomeKit framework. It's connected to six relays to turn each heating zone on or off individually. DS18B20 temperature sensors, one for each room, are connected to the system using the existing thermostat wiring in the walls.

→ hsmag.cc/ThermOS

Housed in a plastic case, the homemade ThermOS smart thermostat cost around $155 to build

Hands-Free Garage Door

The ultimate in convenience is a hands-free garage door that opens at the press of a button, so you can drive straight in – assuming your garage isn't full of junk like ours. When spilt coffee caused Andrew Pena's remote controller to fail, he looked into replacing it with a homemade version. As a newbie maker, he says he learned a lot during the seven months creating this IoT project.

Based around a Particle Spark Core board equipped with a high-power relay shield, it makes use of the IFTTT (If This Then That) service to open the garage door when an icon is pressed on a phone. To check the door's position, the original plan was to use a magnetic reed switch, but he eventually opted to use an old phone camera for visual confirmation.

Alternatively, you could use a Pico W to create a sensor to notify you of your garage door's position, as Jeff Geerling did, using ESPHome and Home Assistant: **hsmag.cc/PicoWGarage**.

→ hsmag.cc/GarageDoor

Housed in a 3D-printed case, the device operates the LiftMaster door opener when triggered on a phone

“

WHEN ACTIVATED – BY A TIMER OR FROM A PHONE DASHBOARD – IT MAKES A BEEPING NOISE TO ATTRACT THE PET

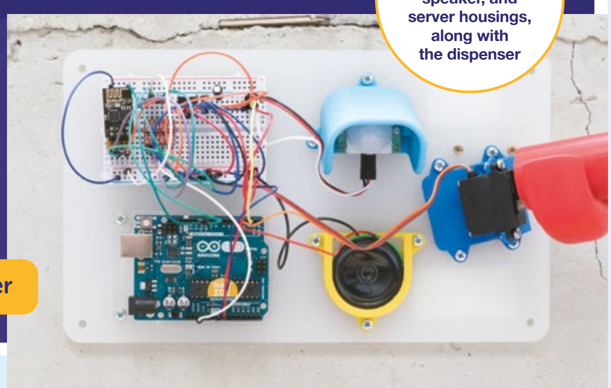
”

Pet Feeder

As well as using smart home automation to control blinds, curtains, and other devices, why not extend it to your pets? This IoT food dispenser enables you to feed your dog or cat the smart way. When activated – by a timer or from a phone dashboard – it makes a beeping noise to attract the pet, which is detected by a PIR motion sensor; a servo is then triggered to rotate the dispenser (a Coke bottle with a 3D-printed top) to drop the food into the bowl.

Devised by the **circuit.io** team, the project runs on an Arduino Uno equipped with an ESP WiFi module to connect to the internet. The phone controller was created using Freeboard (**freeboard.io**), which makes it easy to design interactive IoT dashboards.

Other IoT pet projects include a cat-weighing litter tray (**hsmag.cc/LitterTray**) and GPS tracking vest (**hsmag.cc/IoTKitteh**). →



3D-printed parts were made for the sensor, speaker, and server housings, along with the dispenser

→ hsmag.cc/IoTPetFeeder

SMART APPLIANCES

Make dumb household appliances smart with the addition of **IoT** connectivity

Power is supplied by a portable phone charger; the list of items is read out on a Bluetooth speaker

Smart Fridge

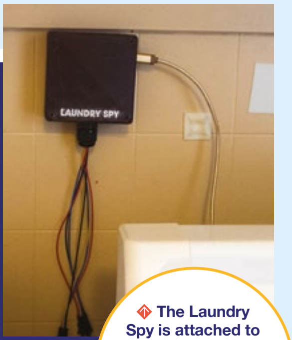
Sure, you can buy commercial smart fridges these days, but where's the fun in that? This project enables you to turn any standard refrigerator into a smart IoT one with a camera view and machine learning to identify the contents.

A Raspberry Pi is placed inside the fridge and equipped with a Camera Module pointing at the interior, and a USB LED to provide light when the door is closed. The AWS Rekognition API is used to identify items and create a list of the fridge contents. With the use of a text-to-speech API, the list can also be read out as audio via a Bluetooth speaker. Maybe you could even add Alexa voice control?

Alternatively, you could build an IoT fridge/freezer monitor to send data to a web dashboard, with alerts, as the aptly named Rick Kuhlman did: **hsmag.cc/FridgeMonitor**.

→ hsmag.cc/SmartFridge

Laundry Spy



The Laundry Spy is attached to accelerometers on the rear of a washer and dryer, to sense when the cycle is finished

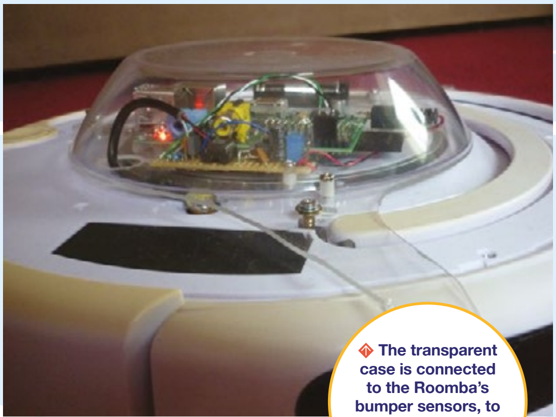
Want to make your washing machine smarter and send you a notification when the laundry cycle is finished? It's fairly easy to do with a microcontroller and a sensor. For his Laundry Spy, Andrew Dupont used an ESP8266-based NodeMCU board, housed inside a small 3D-printed case, connected to two tiny accelerometers (also in cases) – each being mounted at the rear of his washing machine and dryer respectively, to sense the vibrations.

For IoT communication, Andrew used the MQTT protocol to send messages from the NodeMCU to a Raspberry Pi running a Node-RED home automation program. This, in turn, sends a notification to his phone via the Pushover service.

An alternative approach is to sense the washing machine's LEDs using a photoresistor – as done by 'bitluni', who also added a servo to press the start button: **hsmag.cc/SmartWash**.

→ hsmag.cc/LaundrySpy

> “IT'S FAIRLY EASY TO DO WITH A MICROCONTROLLER AND A SENSOR”



The transparent case is connected to the Roomba's bumper sensors, to detect if it catches the underside of furniture

PiRoomba

Robotic vacuum cleaners can navigate their way around a room, but some models aren't all that smart, taking a semi-random route that's likely to cover any given spot multiple times. Peter Wallis decided to modify his Roomba 530 to make it smarter.

To this end, he mounted a Raspberry Pi on the top and connected it to the Roomba's serial port to communicate. The Raspberry Pi is protected by a clear plastic bubble that's also connected to the Roomba's bumper sensors to detect if it catches the underside of furniture. To control the robot, Peter wrote a Java package to implement the Roomba Open Interface, making use of the RXTX library. In addition, he created a simple GUI with sensor levels and a virtual joystick to steer the Roomba from a remote computer on the network. Perhaps you could go one better by adding a camera for a vacuum's-eye view?

→ hsmag.cc/PiRoomba

K-Fee

Boiling the kettle to pour your own hot drink is so last century. Created by four French engineering students, K-Fee is a smart connected coffee machine that lets you use a web interface to choose your strength and size of coffee and how many sugars you want in it – or select an existing recipe. There's also the option to make your drink at a specific time and date.

3D-printed components include a funnel each for coffee and sugar, with distance sensors in the lids to check the levels, and stepper motors attached to screws and discs to dispense the contents. A proximity sensor checks that a cup is in place. Sensors and motors are connected to an ESP32 board, which communicates via MQTT with a Raspberry Pi 3 that hosts the web server.

An alternative IoT hot drinks project, the Teasmade 2.0 is a modified vintage tea-making alarm clock that takes orders via Google Calendar: **hsmag.cc/Teasmade2**.

→ hsmag.cc/KFee

The K-Fee comprises numerous 3D-printed parts, including funnels to hold the coffee and sugar

Smart Waste Bin

Most bins are a bit rubbish compared to this one created by a team of students in Thailand with the aim of preventing public bins overflowing. To supervise the state of the trash placed in it, the smart bin incorporates several sensors linked to an Arduino board.

Crucially, an ultrasonic distance sensor in the lid detects how full the bin is. It also includes sensors for flames (in case something catches fire), temperature/humidity, and moisture (both indented for use in a composting bin), along with a microswitch to detect the bin opening.

To communicate the readings over a long range, a Sigfox gateway is used with a LoRa antenna. The MQTT messages are read by a Raspberry Pi 3 which functions as an application server using a Node-RED back-end, MariaDB database, and web application to show all the data.

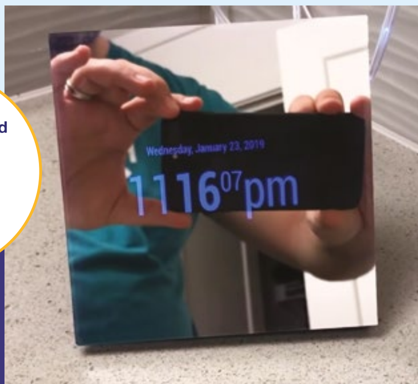
A similar project is this IoT mailbox enabling you to check your mail remotely: **hsmag.cc/IoTMailbox**. →

→ hsmag.cc/SmartBin

The components are fitted into a 3D-printed case mounted near the top of the bin

DATA VISUALISATION

IoT projects to bring data from the web into the physical world for your viewing or listening pleasure



You could build a smaller magic mirror, such as this one with a 3D-printed case by GeekToolkit

Magic Mirror

Building a magic mirror is one of the most satisfying and achievable IoT projects for new makers. The basic build comprises a display, such as an old monitor or TV, and a sheet of special two-way mirror acrylic mounted together in a frame. Your magic mirror doesn't have to be full-size either: you could make a mini version using a small screen, such as this one: **hsmag.cc/MiniMirror**.

Magic mirrors are often powered by a Raspberry Pi, in which case you can use the ready-made MagicMirror² software (linked below) by Michael Teeuw, which offers all sorts of modules to show you the latest weather, sports, news, and much more. You can also embed video feeds, such as from YouTube, and connect your mirror to your own calendar. You can even link it up to Alexa, Google Home, and others to issue voice commands. The possibilities for customisation are almost endless.

→ magicmirror.builders

Weather Tide Clock

A popular use of IoT is to hook up some sensors and send weather data to a web dashboard. One example is to build a Raspberry Pi weather station with a Weather HAT and external sensors connected to the Adafruit IO service (see your author's guide in The MagPi issue 119: **magpi.cc/119**).

Fiona Hopkins took a different approach with her Raspberry Pi-based Weather Tide Clock, which offers a unique physical representation of data from the Dark Sky API. Constructed largely from laser-cut parts, including numerous tiny gears, it features a large wheel with a pointer to show the general weather conditions, while coloured LEDs indicate temperature and impending rainfall.

Another row of LEDs on the bottom displays the 24-hour forecast. Above that, a moving bar with two pointers shows when the two daily low tides will occur, with a chime sound played for low and high tides.

The Weather Tide Clock offers a physical representation of weather and tide data from the web

→ hsmag.cc/TideClock

Live CTA Railway Map

Mini train departure boards – such as this one: **hsmag.cc/TrainBoard** – are a popular IoT project, showing arrivals and departures from a station. Jordan von Mulert has gone one step further by creating a complete map of Chicago's 'L' elevated railway, lit up with LEDs to show the positions of trains at stations in real time.

Mounted on a wooden board, the map features 191 LEDs – one for each station – wired individually to pins on eight 24-channel PWM LED driver boards controlled by a Raspberry Pi Zero W. A Python script pulls data from the CTA API every seven seconds; when it reads an 'approaching station' flag for a train, it lights the LED for the corresponding station on the map, which also features a zoomed-in view of 'The Loop', a busy 1.79-mile long circuit on the railway.

Rather than soldering the huge number of connections required on the rear of the map, Jordan opted to use wire wrapping, which he says is a lot quicker and super-secure.

→ hsmag.cc/LiveCTAMap

Pulling live data from the CTA railway API, the map shows the positions of trains by lighting up station LEDs

RadioGlobe

Spin the globe and position the reticule to find a nearby radio station and play it

You may have come across the excellent **radio.garden** website where you can listen to radio stations around the world. Accomplished maker Jude Pullen – who we interviewed about his Good Air Canary IoT project in HackSpace #58 (**hsmag.cc/issue58**) – was inspired to make a RadioGlobe that offers similar functionality in a physical device made with 3D-printed parts.

The award-winning project enables the user to spin the globe and move an arm to centre the reticule on a location; when a nearby radio station is found, the Raspberry Pi-based device plays it. Two rotary encoders are used to read the reticule's position, to determine the correct latitude and longitude. Station info is shown on an LCD screen in the base, with a speaker and rotary knob to control the volume.

→ hsmag.cc/RadioGlobe

"Iris displays sinuous shapes of various colours to signal the arrival of messages"

Iris

Smartphone notifications are handy, but can mean we're perpetually glued to our phones to check them all. It's an issue that Davide D'Alessandro is seeking to address with his light-up desktop device, Iris, which displays sinuous shapes of various colours to signal the arrival of messages. The idea is that these won't negatively affect your concentration as much as phone notifications.

Based around an Arduino Uno, the device is made from 3D-printed components and features two RGB LEDs along with single-colour ones, with their light diffused by a polypropylene window. Favourite contacts can be set and assigned a particular colour, with a rotary knob used to select and play back messages.

Another notable light-up IoT notifier project is the BinDayCator (**hsmag.cc/BinDayCator**) created by Darren Tarbard to indicate when it's bin collection day, based on data pulled from his local council's website. →

→ hsmag.cc/IrisNotifier

WEIRD AND WONDERFUL

With a bit of imagination, you can create all kinds of **IoT** projects – for practical use or just fun

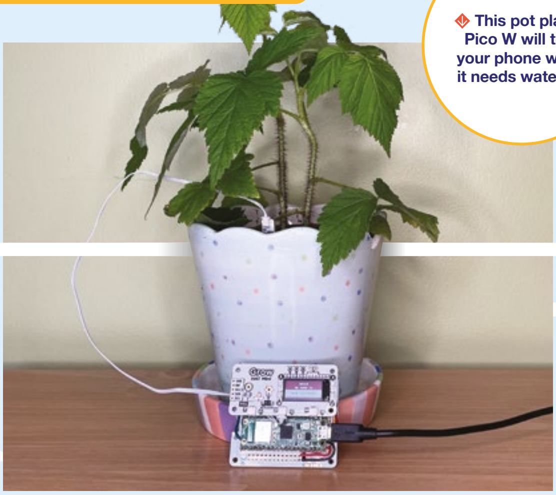
Texting Pot Plant

Many people talk to their plants to encourage their growth, but how about a plant that sends you text messages? That's the concept behind Sandeep Mistry's project, which makes use of the new Pico W microcontroller with wireless connectivity.

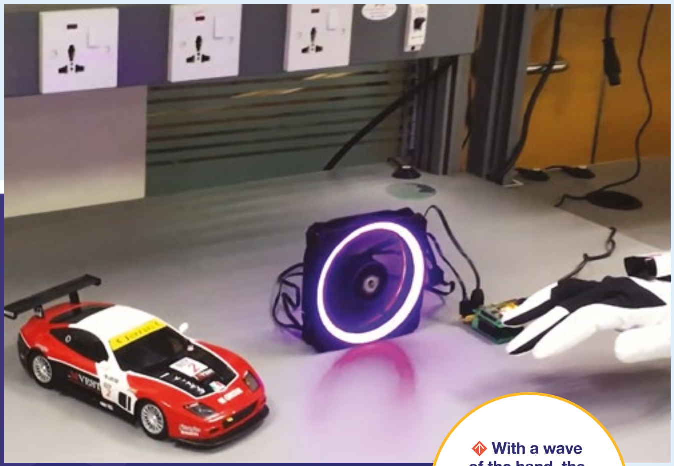
The Pico W is combined with a Pimoroni Grow Kit to monitor the moisture level of the soil. If it's too dry, a text notification is sent – using the Twilio API with a free account – to the owner's phone to remind them to water it. For extra personality, Sandeep suggests adding random messages, along with a light sensor to detect sunrise/sunset and say 'good morning/night'.

Alternatively, the PyPortal IoT Planter (**hsmag.cc/PyPortalPlanter**) sends moisture and temperature level data to an Adafruit IO web dashboard, as well as showing it on the pot's mini LCD.

→ hsmag.cc/TextingPlant



This pot plant's Pico W will text your phone when it needs watering



With a wave of the hand, the Smart AI Glove can operate IoT-connected devices

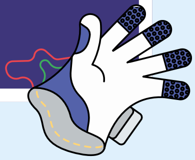
Smart AI Glove

You don't need to be a superhero, or a royal, to make things happen with a mere wave of your hand. Based around a Raspberry Pi Zero W, Ashok Fair's Smart AI Glove enables you to use hand gestures to control other IoT devices, such as a light, fan, and remote-control car.

Attached to a golf glove, a SmartEdge Agile IoT device works in conjunction with Brainium, a cloud-based tool for performing machine learning tasks. The Brainium app installed on Raspberry Pi sends MQTT messages based on data from a Rapid IoT Kit.

Another smart glove project, the Arduino-based Nero (**hsmag.cc/NeroGlove**) enables you to control inaccessible devices remotely from anywhere on Earth.

→ hsmag.cc/SmartAIGlove



> A RIDABLE ELECTRIC UNICORN THAT 'POOPS' JELLY BEANS WHEN YOU LIFT ITS TAIL

Estefannie takes a ride on the rainbow unicorn, which is also a nifty electric vehicle

Pooping Unicorn

While many IoTs have serious, practical uses, this one is purely for fun. Created for the Kids Invent Stuff YouTube channel at the request of an eight-year-old child, it's a ridable electric unicorn that 'poops' jelly beans when you lift its tail.

Wanting to give it an IoT upgrade, the team enlisted the help of fellow YouTuber Estefannie. With the use of a Raspberry Pi W connected to a relay switch, it will now trigger whenever **@mythicalpoops** is mentioned on Twitter, at which point its tail lifts and jelly beans shoot out!

While you're cleaning up the poop, you might even want to make use of an IoT toilet roll holder (**hsmag.cc/IoTToiletRoll**) to monitor your sheet usage.

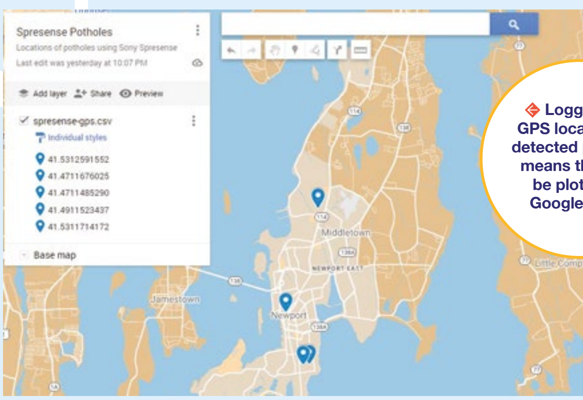
→ hsmag.cc/PoopingUnicorn

Pothole Detector

Potholes in the road are the bane of motorists the world over – in the US alone, they cost drivers $3bn in vehicle damage every year. Many aren't fixed because the relevant highways authority doesn't know about their existence. That's where Justin Lutz's Pothole Detector comes in.

A Sony Spresense board is equipped with a camera to film the road, while an Edge Impulse machine learning model is used to detect potholes. When one is found, an MQTT message is sent to the AWS IoT Core service. Since the Spresense has built-in GPS, the exact location can be logged and plotted on a map.

→ hsmag.cc/PotholeDetector



Logging the GPS locations of detected potholes means they can be plotted in Google Maps

When it's medication time, the patient presses a button on the touchscreen to dispense the pill into a cup

Smart Pill Dispenser

Remembering which medication to take and when can be confusing for patients. This Smart Pill Dispenser addresses the problem by enabling healthcare professionals to use a SmartPill web app to dispense pills and control the dosage remotely.

The dispensing device itself is built from 3D-printed parts, while the main electronics brain is an Arduino Mega. The latter is connected to servos, a real-time clock, and an IR barrier sensor to detect the pills. A Microchip AVR-IoT board is used to send and receive MQTT messages to and from the AWS Cloud.

Another, rather 'tongue in cheek', medical project is the IoT Pregnancy Test (**hsmag.cc/IoTPregnancy**), which instantly tweets the result! □

→ hsmag.cc/SmartPill

SUBSCRIBE on app stores

From £2.29

Available on the App Store

GET IT ON Google Play

Buy now: hsmag.cc/subscribe

Free Pico with print subscription only

FEATURE

HOW I MADE

A DIVE WATCH

Building some customised wrist bling

By Jo Hinchliffe

Lurking around watch communities there are lots of examples – you'll find people 'watch modding', where they customise a watch, or 'watchmaking', where they assemble watches from parts. I prefer the term 'watch assembling' as I think watchmaking describes the process of creating gears and pivots to make the watch movements.

After reading a few different accounts, I decided I wanted to assemble a dive-style watch, which basically means it's pretty large, waterproof, and has a rotating marked bevel. A popular starting point is to pick a movement: the mechanics that sit inside the watch, making it tick. I didn't want to go in at the cheapest level with a battery-powered quartz movement, but rather a mid-range mechanical/automatic movement. These movements can be wound, but they also self-wind when worn on a wrist as you

Right →
The completed custom-built dive watch

move around. The classic movements for this are the Japanese NH35 or NH36. These movements are commonly found in watches that cost between £200 and £500 in brands such as Seiko and Invicta. However, the movement itself can be bought for £30 to £40 online. Before you go and grab one, though, there are a few variants to consider. You can get the NH35 with a single date dial, which just shows the date number, or the NH36 with a dual date dial, which has the day of the week *and* the date number. You can see both movements suffixed with the letter 'A', but this can be ignored as they are all the same. The other option to consider is the colour of the dials; you can get black text on a white background or white text on a black background. An NH35 or NH36 should come with a dummy stem and crown inserted into the movement. This will be replaced with another stem which will be cut to length and threaded into the crown that will be supplied with your case – the stem is the shaft that sets the time and date and also winds the movement. Sometimes the movements come with an uncut stem, and sometimes a case is supplied with an uncut stem – either option is OK, but you definitely need one, so check your product listings carefully (**Figure 1**).

Having settled on an NH35 movement, I set about case shopping. You need to search for watch-cases listed as compatible with your movement – there are plenty available for the NH35 movement. I wanted a watch-case that had two particular features. I've always liked watch-cases where the crown and stem are at the 4 o'clock position rather than the conventional 3 o'clock position. In terms of the NH35 movement, this is totally cosmetic, and it can be rotated to whatever crown position; however, you need to ensure your 'dial' – the face of the watch – is compatible with a 4 o'clock crown. I've never owned a watch with an exhibition case back – where the back of the watch-case has a crystal window so you can see the watch's movement. This attractive element was my second feature of choice. I bought a budget case for £34 that ticked both boxes.

"YOU CAN SEE THE WATCH'S MOVEMENT"

When researching your case, you will find that it mentions the diameter needed for the dial or watch-face. Many NH35-compatible dive-style watch-cases will →

Figure 1 ↑
An NH35 movement with a dummy stem and crown; an oversize stem was also supplied with the movement

Above ◈
The dive-style watch-case with the exhibition case back and a stick-on bezel cover

QUICK TIP

Using a dial with numbers or hour markers makes it easier to align the hands correctly.

require a 28.5 mm diameter dial. When choosing a dial, you need to make sure it has the correct aperture for the type of day and date dials you have on your movement. On the rear of the dial, there are some tiny pins that slide into holes in the band around the movement, and these pins need to be compatible with your case setup. So, for our build, we needed to check that the dial was compatible with a 4 o'clock crown position case. This is important as the date aperture still needs to be at the 3 o'clock position when the dial is fitted. Although there are thousands of dials out there in a wide variety of designs, I decided I wanted to go with

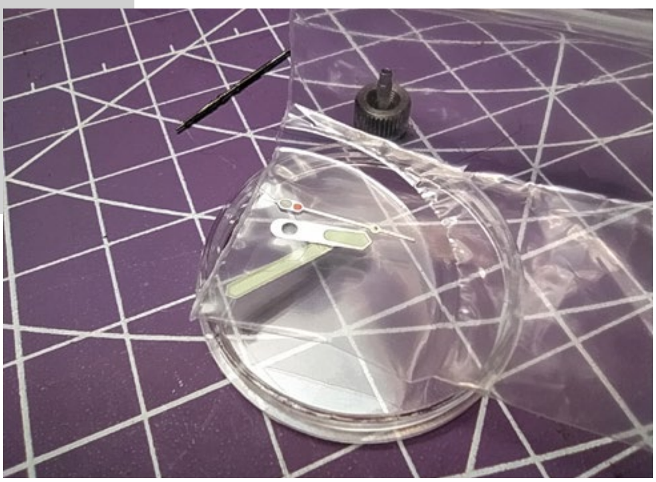


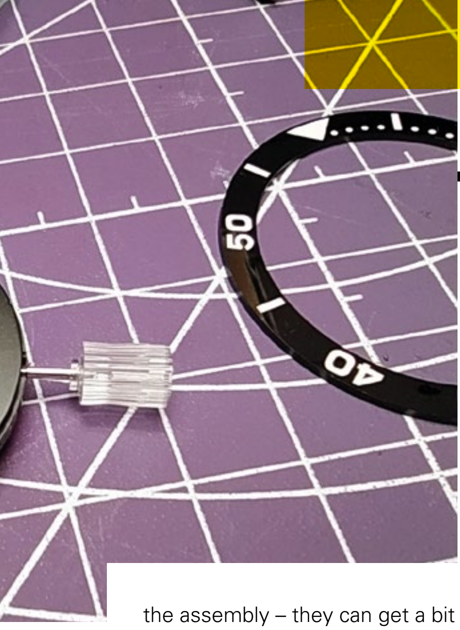
Figure 2 →
A set of NH35-compatible hands chosen from a massive range of those available. They arrive well packed inside a solid case

a plain black unmarked dial, which I love. However, it actually made things a little trickier when fitting the hands.

Hand choice again needs to be compatible with the movement. The NH35 movement needs an hour, a minute, and a second hand, and the tiny tubes/holes in the hands are very accurately sized to be fitted onto the NH35. I went with some chunky-looking hands, as I guessed they might be slightly easier to handle (**Figure 2**).

For the assembly of the watch, I've tried to keep extra tools to a minimum. In issue 31 (**hsmag.cc/issue31**), I made a bench block from a rubber hockey puck, which I occasionally used to place the watch assembly on as a no-scratch, non-slip platform. You can buy small leather stuffed blocks specifically for handling watch assemblies, but you could get away with a clean patch of a rubber cutting mat. I also don't currently own a movement holder, which is a sort of vice that gently holds the movement. This would make the job much easier and safer – it is on my list of things to make (or buy). A cheaper approach is to always place the movement onto the back cover of the case. Be extremely careful, if you need to place the movement face down, that the centre hand shafts or indeed the hands can't come into contact with a surface. The tools I did buy, in addition to those mentioned earlier, were cheap hand setting tools. These rods have plastic end caps, and each end has a hole. The holes and end cap diameters vary – you can find from the choice of six, one that will push the target hand onto the target shaft without touching the other smaller shafts.

So to begin, I placed the brand new movement on top of the case back on top of my rubber bench block, and the first job was to fit the dial. It's important to work cleanly, so I wore vinyl gloves throughout



the assembly – they can get a bit sweaty, so a lot of watch modders just wear 'finger cots', which are individual rubber gloves for fingers. A fingerprint on the dial can be removed with a lint-free cloth, like a lens wipe cloth, or you could buy some 'Rodico', which is a watchmaker's cleaning clay that you dab onto parts to remove oil, finger marks, and dust particles.

Removing our dial from the packaging, we inspected and then aligned the pins on the back with the correct holes in the movement so that the dummy crown and stem are at 4 o'clock and the date aperture complication is correctly positioned at 3 o'clock (**Figure 3**).

Next up is fitting the hands. Have a good look at the centre shafts using a loupe or other magnifier. You should see three different sizes: the widest diameter at the base receives the hour-hand, the middle shaft receives the minute hand, and the tiny, thinnest shaft will receive the second hand (**Figure 4**). Be very careful with these shafts – if they bend, it is near impossible to make the movement work correctly. Looking at the hands themselves, you'll see the underside of each hand actually has a short hollow tube; these tubes are press-fitted onto the corresponding shaft. Find the hand fitting tool that has a hole large enough to clear the two smaller upper shafts but will press onto the thickest shaft at the bottom. Don't press down with great force on these shafts – the forces involved are very small, particularly when it comes to the second hand. Before fitting the hour-hand, you need to get the date dial to the position where it has just clicked over to a new date. Pull out the dummy stem as far as it will go and wind the stem forward until the date begins to transition; as soon as the date finally slips into the correct position, stop turning the stem. When the watch is assembled, the date should begin to transition after 11pm and be fully transitioned at midnight. This means that if we now fit the hour-hand pointing to the 12 o'clock position, the date change and hour-hand will be correctly co-ordinated.

Use a pair of tweezers to place the hour-hand so that the tube is slightly over the bottom shaft. Gently push and move the hand fitting tool to manipulate the hand onto the shaft. You want to get it to catch onto the shaft in the first instance and then check that everything is looking OK before pushing it further on. Never rock the tool too far from vertical, certainly never more than 10 degrees off axis, as this is liable to damage the shaft. Slowly work the hour-hand down onto the shaft so that the flat section of the hand is flush with the top of the shaft. If you leave it too high, it will interfere with the fitting of the minute hand. Keep checking the hour-hand with a loupe as you fit it, looking →

"I WENT WITH SOME CHUNKY-LOOKING HANDS"

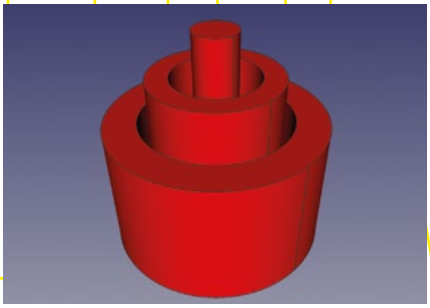


Figure 4 ↑
A representation of the tiny posts that you press the watch hands onto. The thinnest post for the second hand is less than a millimetre in diameter

QUICK TIP

It is preferable to fit the hands correctly first time; however, you can carefully remove them using either a set of hand lifting tools, or indeed placing tweezer tips either side of the shaft and gently pushing the hand off.

Right ←
Some budget watchmaker's tools; some hand setters, a dust blower, tweezers, and a bench block

Above ↑
Using the correct hand fitting tool and tweezers, you need to exercise extreme care in handling and fitting the hands

both at the shaft fitment and also checking the hour-hand is horizontal and parallel to the dial face when observed from the side.

Once you are happy with the hour-hand, use the stem to wind it around the dial twice to observe if the hour and the date change align correctly. Then, accurately wind the hour-hand to the 6 o'clock position. Repeat the process to fit the minute hand onto the second shaft, again with the minute hand pointing precisely at the 12 o'clock position. Check that the minute hand is flush with the top of the shaft and parallel to the dial and not drooping down or rising upwards – this is critical as the hands obviously need to clear each other as they rotate when the watch is running.

> **"SLOWLY ROTATE THE HANDS USING THE STEM"**

Once you are happy with the minute hand, slowly rotate the hands using the stem, and triple-check that they clear each other when they pass over each other in numerous positions. To fit the second hand, it's not too critical where the other hands are in relation to it. A common approach I used was to set the hour and minute hand to the 12 o'clock position, then fit the second hand over the top of them. The second hand is by far the hardest to fit – you need to be extremely careful to not bend the second hand shaft, which is a fraction of a millimetre thick. It can even be quite hard to see the second hand shaft using a loupe, as any shadow can suddenly render it invisible. So again, you need to find the hand fitting tool, probably with the smallest hole, which will again press onto the hand but not the tiny shaft. A good practice tip I found was to place the second hand on a hard surface and then rest the fitting tool on it and see how little pressure you need to cause the second hand to become horizontal on its tiny fitting tube. You can then practice gently rotating the second hand slightly to train your hand and brain with the necessary forces. Using the loupe, place the second hand so that the tube is resting on the top of the shaft. Before you press anything down, triple-check that it is in the correct position, as it's easy for this to end up incorrectly in the gap between the second hand shaft and the minute hand. Once you are happy, use a little force to bring the second hand horizontally over the shaft and gently push it down into position. Check it from all sides – adjust if needed. Once you are satisfied that it's correctly installed – horizontal relative to the dial and hands move freely – you can then give the stem a couple of winds and the assembly a gentle shake to get the movement running. Let it run for a while, checking that the second hand doesn't catch anything when it passes the hour and minute hands.

Below ↓
Using a small pin tool to release the dummy stem from the movement

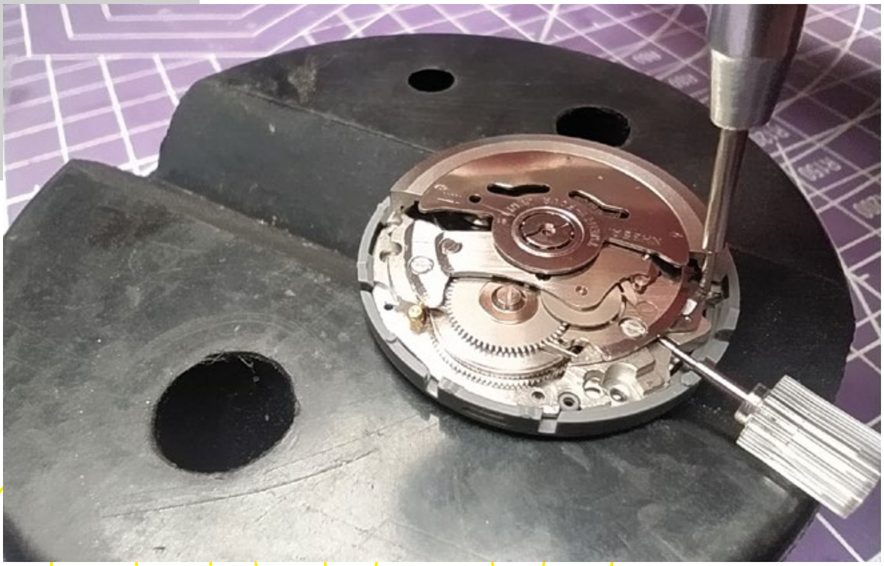


Figure 5 ↑
Using a set of callipers to measure the amount of stem that needs to be removed to make it fit the case correctly

It would be best practice to let the watch movement stop before continuing again but, if you are like me, you may be impatient. The next step is to remove the dummy stem from the movement so that you can insert the movement into the case. You need to find a way to support the movement upside down without damaging the hands or centre shafts. I managed to do this using the v-cut out on my bench block, but this

QUICK TIP

Almost all watch movements have instructions online, or sometimes they are marked with where to press to release a stem.

Above ↑
The exhibition back case allows you to inspect the movement through the back of the watch

is where a DIY or a commercial watch movement holder would be useful. Press the release bar to release the dummy stem and withdraw it.

Taking care to align the stem position with the hole in the case, I placed the movement into the case. You might need to use a cocktail stick to gently press the edge of the movement to encourage it into position – obviously, limit this to the fitment ring around the movement; don't press down on any moving parts!

The stem of an NH35 movement watch is threaded, meaning it threads into the crown. Due to different case sizes, the stems are supplied oversized and need cutting to length. In my build, this was slightly more complex as I had a threaded sprung crown. This is a common feature in some dive watches that makes the crown and stem assembly more waterproof. The crown itself is internally threaded to screw down onto a threaded shaft protruding from the case. Inside the crown is a rubber gasket which, when tightened down, creates a robust waterproof seal. This means that the stem has to compress and elongate – it does this via a sprung piston inside the crown, into which the threaded stem is inserted. I threaded the stem into the crown and fitted the oversized stem and crown into the watch, ensuring the stem was fully engaged and sat in the winding position. I then carefully compressed the spring to push the crown closer to the case – when it wouldn't press in any further, the gap between the base of the crown and the edge of the case, where the crown should be touching, was the amount I needed to cut off the stem to bring it to the correct size. I measured this gap *many* times and concluded that I needed to remove 6.3 mm to bring the stem to the correct length. To achieve the cut, I removed the stem and took the stem of the crown, and then I measured the complete stem, then took away 6.3 mm from that value and locked my callipers into place at the required length (**Figure 5**). I then placed the inner tip of the stem against one jaw of the calliper and then rested the other end of the stem on the upper surface of the other jaw. I then placed some side cutters around the stem, with the flat side of the cutter flush to the calliper jaw. Checking everything twice, I performed the cut. Refitting the freshly cut stem to the crown, I test-fitted it once more, this time screwing down the crown to meet the case. Everything was in the right place with no gaps, so I disassembled the stem once more to place a tiny drop of superglue on the thread inside the crown to act as a thread locker.

I could now relax! All the challenging and potentially watch-damaging activity was over. I simply fitted the back of the case, stuck the bezel surround onto the double-sided bezel tape, and then fitted the strap. Watch-cases have different width straps – you measure between the lugs to discern the strap width you need. Both my watch-case and my watch strap arrived with telescopic pins, which you can compress to fit, and they expand into the lug holes. The ones with my strap arrived and were a bolt-action type, where they have a small handle that allows you to easily fit the pins without any tools. However, if you've made it this far into a watch assembly project, you'll fit the pins with ease!

I'm pleased with the watch and very proud to wear it. Be warned, though, it's addictive! I'm already planning and accumulating parts for another build!

OPEN AND SHUT CASE

Opening and closing watch-cases is a basic task in watch assembling or repair. There are a couple of ways that case backs attach and are removed. A common case back is the press-fit type, where you need to use a thin tool to prise up the cover. Often there is a small lipped part that allows you to place a flat tool underneath. A common tool for this is a 'spudger', but you can also use a thin-tipped, flat screwdriver or a thin guitar plectrum. The other case back type is a screw down back. These often have a collection of small square grooves around them, and you can use a case opening tool which has adjustable tips. You can move the position of the tips to match three of the grooves in your case back and then use this as a wrench to open and close the case. It's a really good idea to cover the case back with masking tape to minimise the risk of you scratching it.

HackSpace magazine meets...

Debra Ansell

Why do art at all? Because we can!

You may have seen Debra Ansell's sound-reactive LED embroidered party dress. Or her internet-connected, intelligent edge-lit acrylic light paintings. Or you may have recreated one of her builds yourself – following the instructions she generously puts up on her site **geekmomprojects.com**. Alternatively, you may recognise the name because almost everyone we speak to nowadays cites her as an influence on their work. We spent an hour with her talking about everything from manufacturing, creativity, and how you get from a physics PhD program to teaching kids electronics. →

POWER TO THE PIXELS

Above The LED bracelets are Debra's latest creation; they incorporate elements of electronics, laser cutting and 3D printing

HackSpace magazine **Morning Debra! Let's kick off with what is simultaneously the easiest and yet also the hardest question of all: who are you?**

Debra Ansell That's existential! I'm a maker. I was a stay-at-home mom, with a tech background, who's always loved science and learning and creating. I've always been very creative, but not at all artistic. I can't draw freehand; I really have no eye for colour or anything. But when I first saw that you could do something digitally, it was a revelation to me – I can get exactly what's in my head, into the real world, without the limitations of my lack of artistic ability and dexterity. And if it's digital, you can edit it and revise it infinitely, until you get what you want. Digital making really opened doors for me.

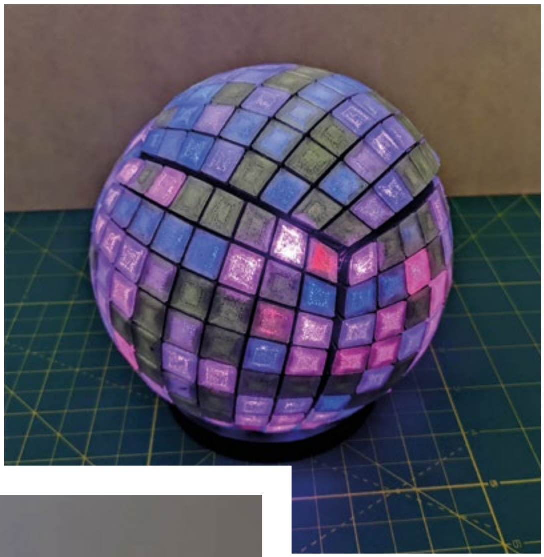
I love science. I love pure science. I don't necessarily love the nature of how to practice science – you pick a very small piece of a very big puzzle and drill down on it. And it's absolutely essential that you do that. But I find the big picture interesting, and the smaller practice less so. I still like to keep up with where things are going – string theory, Higgs boson – but I realised that if I was going to stay in academia, I was going to, you know, pick an integral and add a decimal place every few years or something of that nature, which did not appeal to me. So I ended up leaving the physics PhD programme I was working on. I had learned to code (mostly in Fortran) to analyse my data. And I started working as a software engineer. I liked the interactive, iterative nature of software. But, you know, I was working for an internet startup, and it was not at all family-friendly. Software isn't to begin with, but the internet startup was 25-year-old guys, and I came to the office pregnant and they had no idea what to do with me. So I retired, you know, to stay home with my kids, but I still love science. And, of course, kids are natural scientists.

That was my outlet for a long time – I was always the mom who volunteered during the science experiments at the after-school activities, things like that. And then my middle son discovered LEGO MINDSTORMS... from there, I discovered Arduino.

I'd never heard of it at the time, but I've looked into it. And coincidentally, I was booked in to do a workshop that was using a LilyPad and conductive thread: a wearables workshop, which is my sweet spot. I got lucky – it's fun, it's affordable, it doesn't take up too much space, and it progressed from there. I just kept seeing really cool projects online that people created and detailed for you to reproduce.

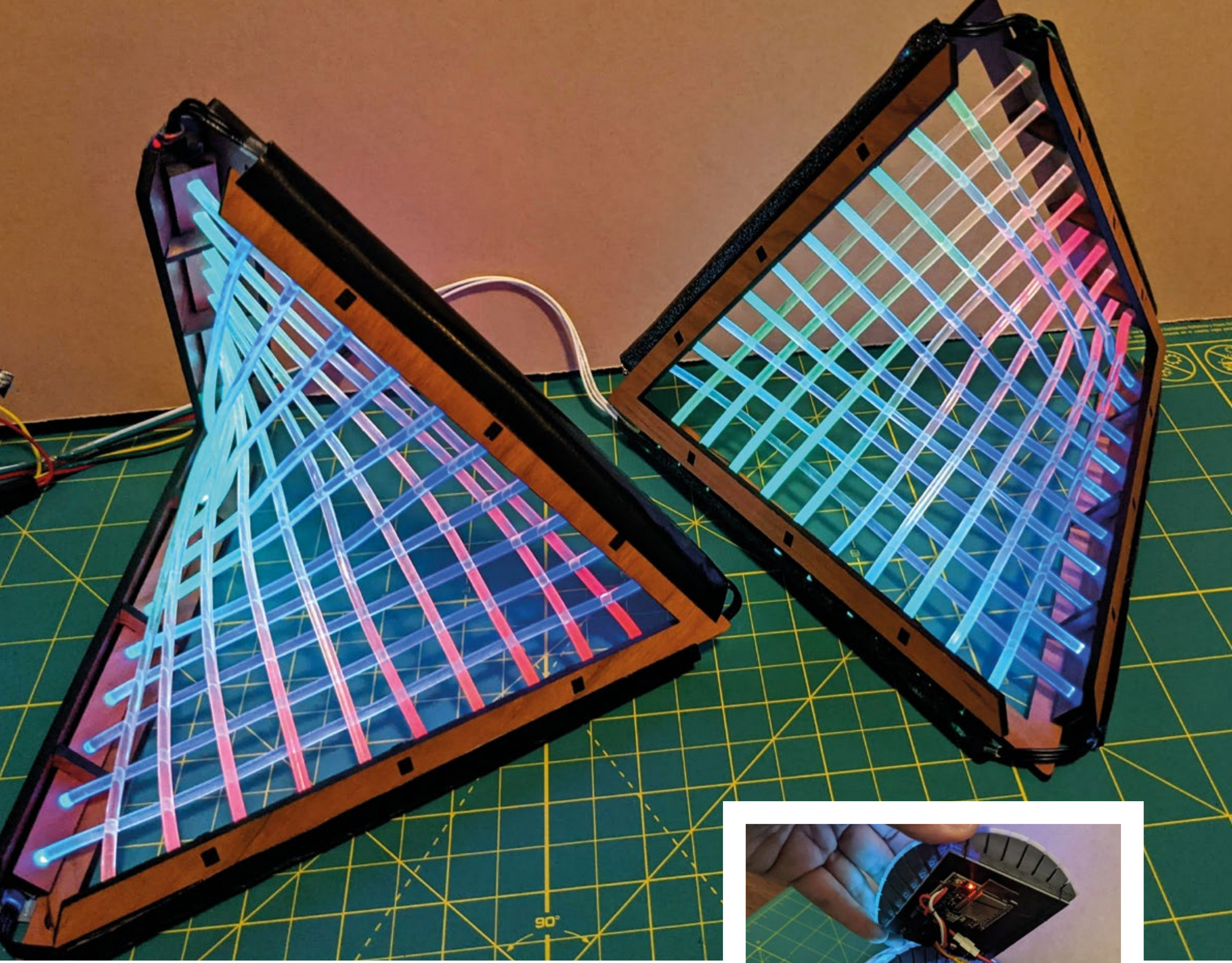
I had a strong physics background. I had taken electronics classes, but I'd never had hands-on experience of electronics before that. I mostly took observations of astronomical radio data and analysed it; I wasn't really in there with a wrench and a screwdriver.

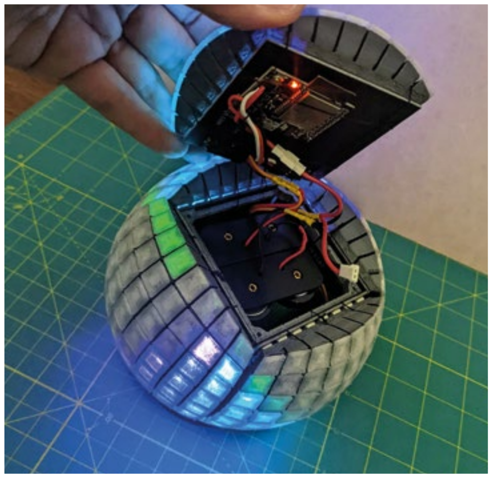
So I learned hands-on electronics through Arduino, and eventually, I [became] comfortable enough making



Above
It's hard to implement electronics on a sphere – Debra's done it by curving the diffusion, not the electronics

Below
The smaller a sphere gets, tighter the tolerances need to be. This looks pretty jolly nice to us





Above ↑
Debra's aesthetic is getting light to fill spaces in interesting ways

projects that I'd seen other people make, and I started to be able to improvise. It was a great creative outlet. There's a lot you can do with this that nobody's done before. So let's see what happens.

I love working with kids. I volunteer teaching kids to code. My job is to keep them from doing anything unsafe, but otherwise, let them go. And they're appropriately enthusiastic about really cool things that might not resonate with other people. I was so happy when I had my Code Club meet right after the Mars rover landed and we were talking about the code in the parachute. And the kids thought this was so cool [that NASA had encoded a message into the parachute]; then I walked around all afternoon talking to grown-ups – the reaction was not the same at all. It's really fun to deal with people who share your enthusiasm for whatever it is you're excited about.

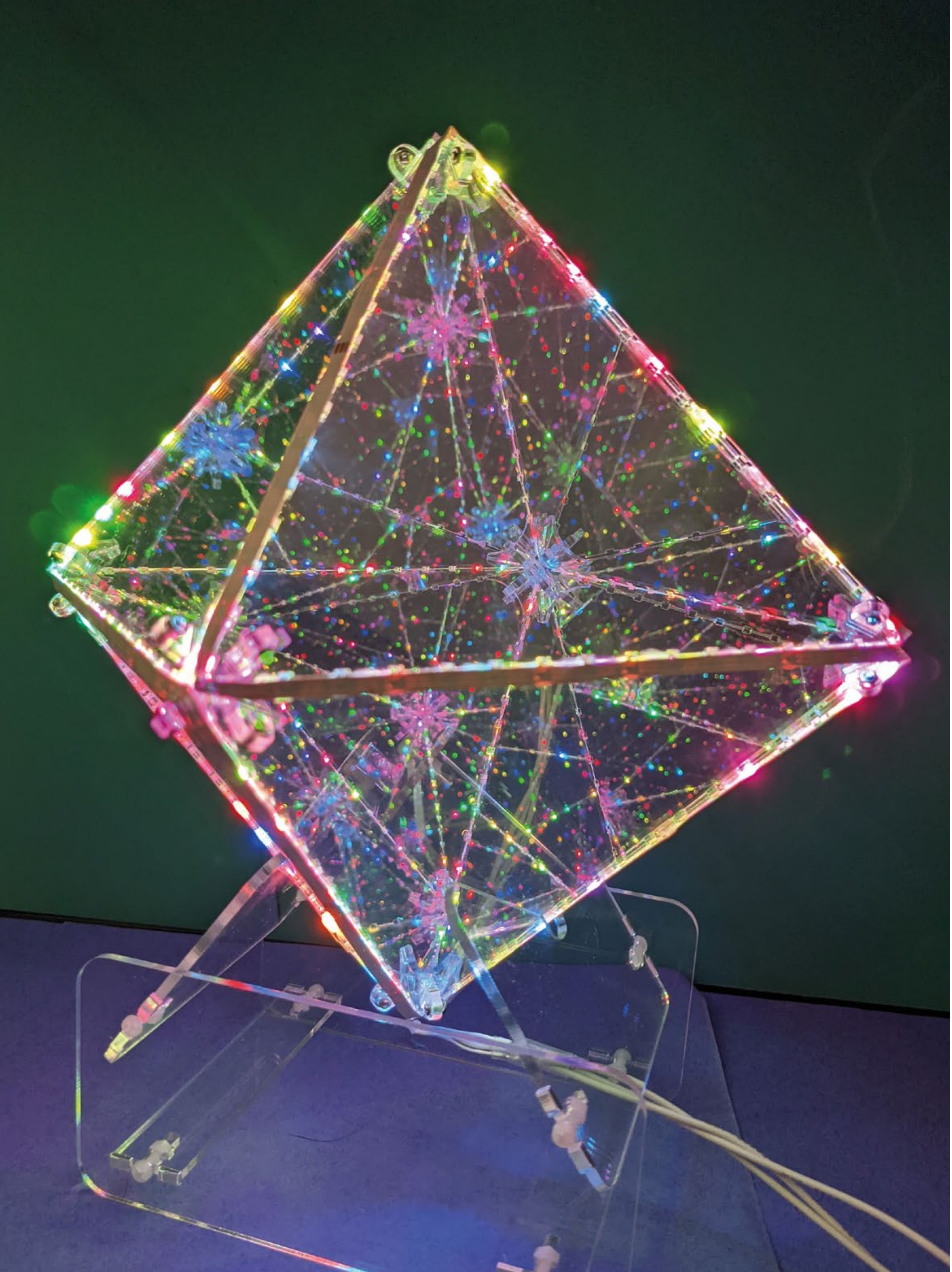
HS Do you think there's something inherently childlike about makers?

DA Oh, absolutely. There is an absolute sheer unfiltered joy in reaching vision and making it work. And there's definitely an appreciation for silliness. I think you don't have to take it too seriously. And in fact, because tech is such an innately serious subject matter, to kind of subvert it and do interesting things with it is appreciated. Like I love Jiří Praus and Mohit Bhoite's work with circuit sculptures. Take an inherently practical medium, and make it beautiful, but still functional.

There's no doubt that it's art, because why else would you do it?

Above ↑
The lid of the sphere is held in place with magnets

HS You've touched on it already, but you go to a lot of effort to document your builds – why? →



Left Debra's been working with Jason Coon and Ben Hencke to create these animated LED pendants

DA That's the ongoing challenge for me. I don't love documentation. But in the beginning, I benefited so much from what everybody else would put out there. I want to make anything that I do that I think somebody else could make or learn from available. So I do try, because it's fun to create something. But it's even more fun to share that creation and watch other people experience that joy. I really try to not only make a project but, most of the time, I don't consider it done until I can make it buildable by somebody else.

And that's a challenge: it's often harder to make a project accessible than it is to make the project in the first place. I find it particularly satisfying when I can post a tutorial, and other people can experience the joy of making something really, really cool. A good example is the LED sphere that I'm currently working on.

I am so happy with this project. The joy of this orb is that it's far more reproducible than [any comparable project] that I've seen, because people have made LED spheres and they are a ton of work. Electronics go much more easily into a plane, and to make a sphere, you want to put them on a curve – there's the challenge right there. The breakthrough realisation for this orb was that you can have all your LEDs in a plane using standard LED matrices – you can just curve the light. I mean, I spend my whole time trying to pipe light into different places… my whole aesthetic has been to take the light and don't just make it a pinpoint; fill space with it instead, do something more interesting. It's nice to make things that are just fun to look at for no other purpose than just to look at them.

> "It's often harder to make a project accessible than it is to make the project in the first place"

HS When you say that the project isn't finished until it's in reproducible form, that really sounds like a scientist talking. I wonder if that's why your name keeps cropping up when I talk to other makers.

DA Well, there's a bit of coincidence in whom you've reached out to in a way – Jason Coon, Carrie Sundra, and I are all in a maker group – Jason and I developed a project together. Carrie has a business, and I admire her tremendously for it. So many makers are creative, and everybody wants to start a business, including me, and I did. But it's a very different animal, and the overlaps between making and running a business are not at all obvious: the skills that make you good at one are not the skills that make you good at the other. I do not love the business aspects of selling your creations, and I admire Carrie tremendously. I think she's very good at it.

I've decided from my experience with trying to run a business that I'm much happier leaving the business aspects to others. I'd be Steve Wozniak than Steve Jobs.

HS Another maker we've had in HackSpace mag, Odd Jayy, also mentioned you as an inspiration.

DA The funny thing about Jayy is that he and I share a kind of obsessive tendency to lock ourselves in a room with our creations. We met years ago, and he just wanted to build robots, cool robots that shared his aesthetic. And it wasn't for any other purpose than he liked and needed it obsessively. And it turns out, it's a really cool thing that others want to do.

I feel a certain familiarity with that story. I've been obsessively doing the →

things I like for a very long time. And after twelve years or so, I've gotten good enough that I can make them good. And people like them; it's not that I've tried to make projects that people are interested in, but I have obsessively focused on this one niche thing that, fortunately, turns out to be interesting to people – LED wearables in particular. I also think that the projects I build lend themselves very well to social media, because they're blinking and bright and colourful. So my work gets shared a bit on Twitter.

I do like to think that I'm putting novel ideas out there that inspire others, like LED string art. I hadn't seen that before. The internet craves novelty, and bright, blinky pictures, and I'm in the sweet spot, and it gets me widely shared.

HS Are you still using Arduinos? Or have you moved on to Raspberry Pis or Adafruit devices?

DA I hop around from project to project. I do have a lot of Raspberry Pis – I think I have one of everything. And I love my Raspberry Pi 400 because it's super-cheap and easy to take for tech demos; you can just plug it in and take it, and I take it places where I'm gonna be working with kids. When Raspberry Pi first started, I hadn't seen the idea that you make an accessible platform that was designed to bring education to places where it wasn't easily accessible. And it was brilliant. Really, really brilliant.

And so I've used Raspberry Pis a lot in my projects. As I've gotten more prolific, I've gone cheaper and simpler. I don't need an operating system, so even a Raspberry Pi Zero is overkill for a lot of what I do.

My favourite Raspberry Pi creation is a jacket with programmable LEDs – I put in a Raspberry Pi-based web server, so you can actually code your own patterns on the fly, as long as you have access to a device with a web browser. It has a drag-and-drop coding interface to create LED patterns. You can actually code your own LED patterns; you can be totally separate over there that run on my jacket while I'm wearing it.

It's fantastic. Everything I do is because this tech is so accessible. Any organisation that strives to make it more accessible is wonderful, especially to communities that don't get to see and play with tech in the way that more privileged communities do.

I volunteer for a Los Angeles-based non-profit that brings science into underserved elementary schools, because in Los Angeles, the public elementary schools don't have a science curriculum until middle school [when kids are twelve years old].

Now, most schools try to fill that gap by raising funds; the poor communities can't. So this programme brings a science curriculum to underserved schools. I managed to catch up with them and said, "Hey, I know you know a bit about coding; I like working with kids; maybe I can help with that aspect of your work". And right at that time, I think micro:bit had just come out. It wasn't big in the States, but it was this wonderful platform, and inexpensive – that was the best thing. The interface was good, but the price point made it a no-brainer. So I go into these classes; my own kids are very lucky, they've had iPads and whatever they want – they're digital natives. But you'd go into these classrooms and see these kids get so excited about the tech and coding. And I'd say, you know, well, even if you don't have a micro:bit, you can go home and use the interface in your browser. And they'd say I don't have a computer at home.

I'm a big believer in the saying that talent is equally distributed, but opportunity is not. Some of them are amazing, and if they don't get the opportunity to access these tools, you lose

Left
The diffusers (the dragon scale-like parts) are 3D printed onto a thin mesh, which is reinforced with leather for strength and flexibility

so much. [Messing about with electronics] is engaging and fun, but it's good for so much more.

I'm hoping that years from now that this will translate into something. I just want them to know that if they do manage to make it to a place where they could take a programming class, they'll say, "Oh, yeah, I can program; I've done that." Not, "Oh, programming is hard and weird." Programming is mysterious if you haven't tried it, but once you do it, it's really nothing special. It's laying out a series of instructions, which anybody can do.

You asked what platform I was using. I'm a big fan of anything that makes tech accessible. There's a reason I built my LED purses around micro:bit. But it started as a kids' project.

I wanted to make codable wearables that would maybe appeal to girls who are not so interested in robotics and traditional tech projects. There's no reason coding should be restricted to robots and video games and circuit boards – you can code your clothing. And so, I was trying to provide a platform and accessible again. I want to get back to accessible and inspired platforms for people to experience tech, that might not be what they expect and might reach people that weren't drawn to other uses of tech.

I'm drawn to controllers that make things accessible and easy. It's why I've got very into CircuitPython, because it has libraries that make it easy to do complex things with relatively little code, which lowers the barrier to people getting into it. And yeah, I'm drawn to inexpensive, of course. That's because at this point, I've accumulated so much. It gets expensive. [I like] inexpensive, accessible controllers, especially ones I think that people can get their hands on and use and enjoy and learn from.

HS What are you working on at the moment?

DA I have a lot of ideas. My brain's like a bubbling cauldron of soup with all these random things in it, and things keep popping up. I'll take an idea, look at it, and then push it back down into the soup. If it keeps resurfacing, I know it's generally a good idea. And I'll eventually tackle it.

I've been wanting to do a spherical LED project for a while, and when the idea of the flat PCB with a spherical shell clicked in, that took it into the realm of the practical.

And I'm working on a collaboration with Jason Coon and Ben Hencke [creator of the Pixelblaze LED controller] on an LED wearable. That's interesting, because we're now in the realm of producing ten of something – we've moved on from ideas, and we're stepping into production. My contribution to the design was the battery holder, which fastens with a magnet; it's also a switch that turns it on.

They're selling, which is wonderful. And, as I'm making the latest batch of battery holders, I'm thinking, what do we do if this scales? It's a really interesting problem. I'm thrilled with this design. I love it. And I'm happy to make them. I'm happy to make one. I'm happy to make ten. I'm happy to make the 78 I just made – so it's doable, but it's relatively labour-intensive.

> "You asked what platform I was using. I'm a big fan of anything that makes tech accessible"

I don't want to be so arrogant to think that it's going to be wildly successful, but it's stupid not to prepare for potential success. So how do I scale this? So that's a kind of novel project for me.

I'm incredibly lucky. I get to come up with these projects and write them up. I get to design for the summer camp that we run for the elementary school kids, and that's fun, too. I'm doing workshops for other people, too. I have a workshop proposal in for Supercon. I've got a lot of ideas for fabric-based wearables that I'd like to try to execute. I'm super-excited about the potential for these LED cuff bracelets... I have a problem with prioritising, but there's certainly no shortage of things that I'd like to do.

Above ↑ You don't have to be interested in robots to get into programming

IMPROVISER'S TOOLBOX

TILES

Tiles aren't just to stop your bathroom getting mouldy, they're a statement in their own right, says **Rosie Hattersley**

Rosie Hattersley

@RosieHattersley

Rosie Hattersley writes tech, craft, and life hacks and tweets **@RosieHattersley**.

Ever since we've been throwing up walls, we've been coming up with ways to cover them up or enhance them. Paint, plaster, wallpaper, and pictures all have their place, but tiles quite often outshine – and outlast – them all. Tiles can, in theory, be made to cover almost anything. For proof, look no further than the Sagrada Família church or Parc Güell in Barcelona, both dreamt up by Antoni Gaudí. Tiles and mosaics are also favourites of Spanish 'starchitect' Santiago Calatrava, whose soaring concrete structures are given beauty by their pristine white, light-reflective glazed coverings.

Tiled structures in ancient Egypt, Samarkand in Uzbekistan, Mexico, and Morocco all have distinctive designs that mark out their origins. The prevalence of tiled buildings across China, North Africa, and the Middle East points to the spread of this artisanal craft along the lines of the Silk Route, while European settlers subsequently brought their colourful decorative designs to Mexico and South America.

As with other decorative arts, tile design reflects the culture and interests of its makers. For example, Moorish designs, particularly those from Morocco, tend to feature complex geometric designs and there is still, just about, a tradition of hand-building beautiful zellige concrete tiles by hand. The skill involved can be appreciated in this YouTube video: **hsmag.cc/ZelligeTiles**. Relatedly, having been under Moorish rule for several centuries, southern Spain and Portugal have their own rich designs of ceramics known as azulejo.

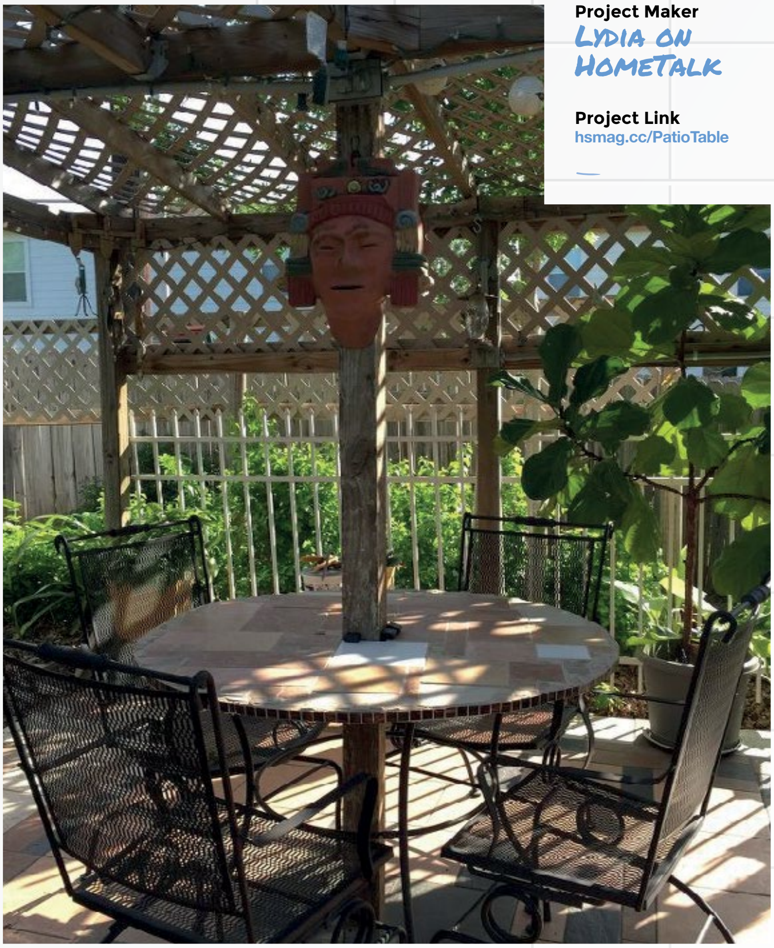
Western versions of decorative tiles, meanwhile, tend to focus on figurative designs, whether religious – tiles were extensively used as striking, but fairly affordable, wallcoverings in churches – or secular. Nature and animals are common motifs in tiles from the Arts and Crafts/Art Nouveau period, neatly coinciding with the introduction of bathrooms in domestic settings; now, of course, the most popular rooms you're likely to want to tile.

As you might imagine, most of the alternative uses for tiles play to its decorative strengths: prettying up an otherwise plain table, wall, or garden walkway. However, one clever hack that caught our eye was this by the by reference to using an unglazed ceramic tile or two in a conventional oven, as an alternative to a pizza stone: **hsmag.cc/PizzaTiles**. If you've so far resisted the vogue for outdoor pizza ovens, this is a cheaper-than-chips option you might feel like trying.

PATIO TABLE

A tile-top coffee table or side table is the sort of achievable project that you can apply to nearly any style of furniture, updating its look while making the surface more durable and resistant to spills and scorch marks. If you really want to go to town, you could repurpose some old scaffold boards or pallet wood for the frame, as per this stylish, but pricey, example: **hsmag.cc/TiledTable**.

However, this new patio table from old, featured on DIY site Hometalk, is arguably a more sustainable option. The table was gifted to the couple who upcycled it. They promptly cut the tabletop in half, and removed two triangular pieces at the centre to accommodate a parasol post. The table was then spliced back together with extra supports underneath. Next, the couple set about arranging assorted leftover tiles, from redoing their patio, to fit the available space on their new tabletop. Once the tiles had been set in place, grout was added and finish applied to protect the patio table from showers. The design was completed with a line of small tiles around the edge of the table. In all, the project cost the couple $20. →



Project Maker
LYDIA ON HOMETALK

Project Link
hsmag.cc/PatioTable

Above
Tiles don't only look good, they're a strong and sustainable material

TEXTURED CERAMIC TILES

Basic tile design can be as simple or as detailed as you choose, largely depending on whether you want multiple colours and whether you're introducing additional materials such as marble chips for a terrazzo look, using special glazes, or carving out the clay to create a relief design, as per the impressive handmade tiles showcased in this production video by Stow: hsmag.cc/HandmadeTiles.

For a more DIY approach to crafting individual tiles, look no further than the detailed video and accompanying Instructable by John Whitmarsh, who takes inspiration from the textures he notices around him, and sometimes takes impressions to use as moulds. For this project, he created ceramic clay moulds cast from old roadside guardrail posts in Sausalito, California. "I've always loved the texturing of old utility poles. No two look alike," he says. "Some are cracked and weathered from the sun and rain; some are charred; some have creosote and chemicals leeching out." He then takes a vertical block of clay, slices through it horizontally with cheese wire to create individual tiles, and presses each one on to the underside of the mould he's made. John then trims off any excess clay around the edges to ensure he'll end up with uniform-sized tiles. Having allowed them to dry slightly, he fires them and gives them a coating of melted wax to bring out the texture. Finally, he assembles each tile to form a unique textured tile wall: **hsmag.cc/CeramicWallArt**.

Project Maker
John Whitmarsh

Project Link
hsmag.cc/TexturedTiles

Right
Like the wood grain, but hate wood? Then here's the tile for you!

LOST AND FOUND: PANEL 38120

Discontinued industrial tiles form the basis of this photomosaic by ceramic artist Rita João of Pedrita Studio. Together with her studio co-founder Pedro Ferreira, she produced the Lost & Found photomosaic series of tile portraits which began with, "a small, black and white photograph found, by chance, in the street, with the inscription 'my grandfather' scribbled on the back." The artists paired the concept of a discarded or lost object being rediscovered with a technique they developed known as Grão. This system sees them build up collage-like portraits from unwanted tiles based on the principle of the pixel or photographic grain. This example, from their 2018 Lost & Found series, is on display at Portugal's National Tile Museum in Lisbon.

Project Maker
RITA JOÃO

Project Link
hsmag.cc/PortugueseTiles

Left Pixels don't have to be digital

REPURPOSED ELECTRONICS TILE SPLASHBACK

Maker miglarsh was on the lookout for a way to add interest to otherwise unadorned areas of the kitchen he'd recently refurbished. "I knew how many TVs, CD/DVD players, radios, computers, [and] anything with a circuit board wound up in the trash/landfill," he says of his decision to use salvaged electronics to create wall tiles. Having sourced the parts for free, he set about disassembling everything into constituent parts, organising it all by size so he could plan a rough pattern when he began to create the wall tiles. A palm sander and screwdrivers were used to dismantle and sand down parts, ready for arranging in a range of paraffin wax-primed formers where they were set in place.

At first, he planned to use the green circuit boards and discard the electrical components, but the range of colours convinced him to use more of the unwanted electronics in his design. " I found a treasure of colours and textures among the things attached to the boards," he explains. Having previously designed an epoxy counter-top, miglarsh combined his knowledge of epoxy with the traditional idea of tiles on the splashback. Referring to the "little gems" he found inside common electronics, which contain a lot of recyclable plastic that his tiles have kept out of landfills, miglarsh declares it a win-win! □

Project Maker
MIGLARSH

Project Link
hsmag.cc/ElectronicsSplashback

Left Tiles don't have to be ceramic – with a bit of epoxy, they can be made out of whatever you like

IN THE WORKSHOP: The Fairy Lantern

By Andrew Lewis

Recreate a classic stage illusion in small-scale using a Pimoroni Pico LiPo and an SPI screen

You'll need

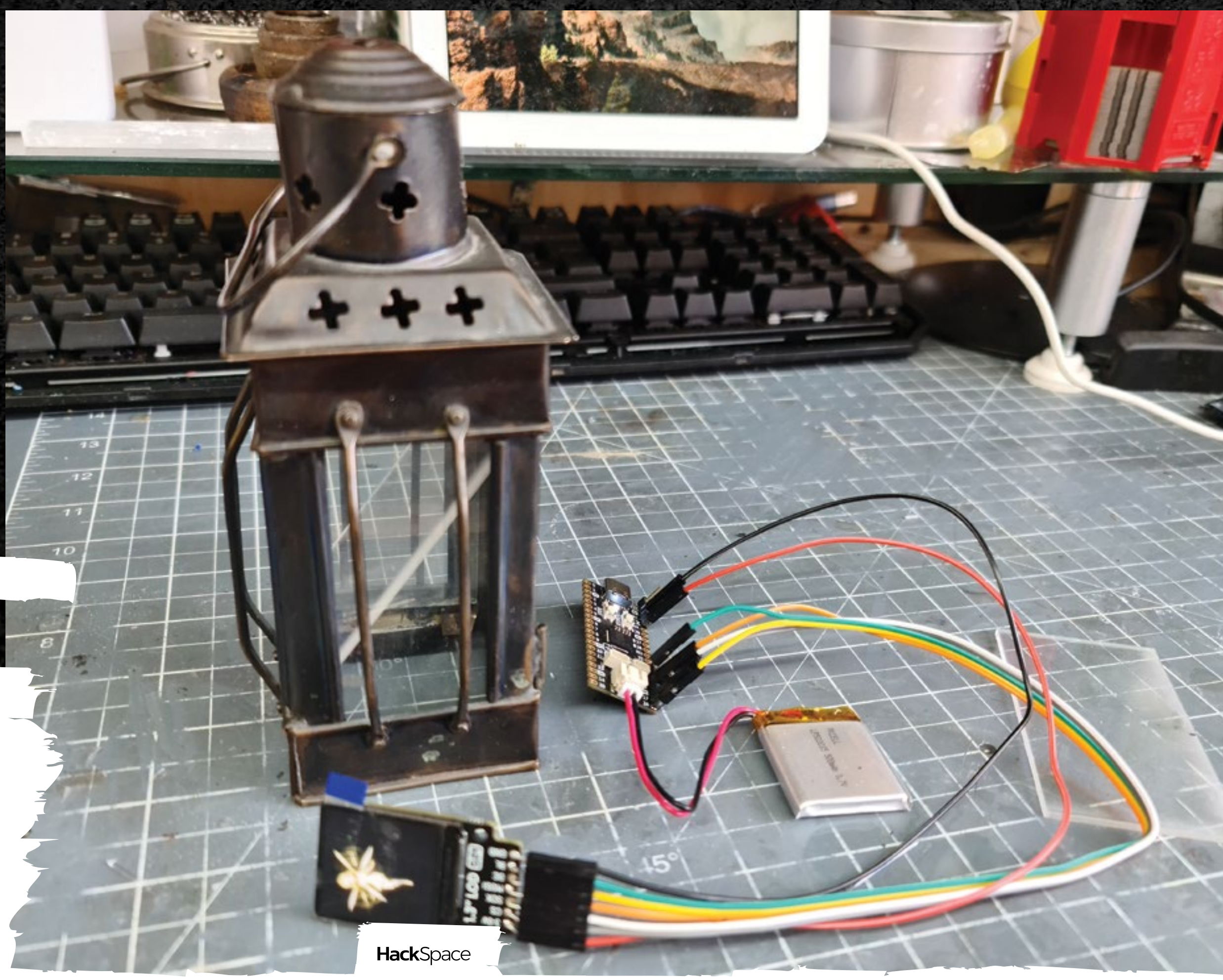
Pimoroni Pico LiPo
hsmag.cc/PicoLiPo

500 mA LiPo Battery
hsmag.cc/BattPack

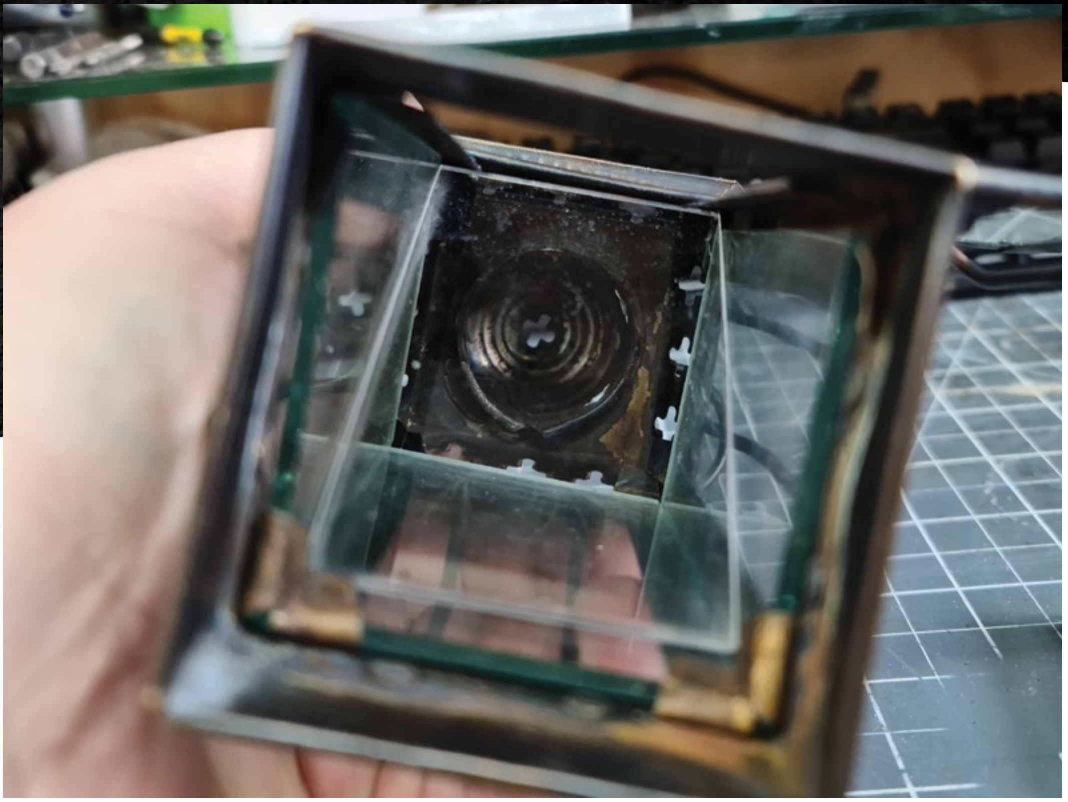
1.3″ Square SPI LCD screen
hsmag.cc/ColourSquareLCD

Lantern or box with glass or acrylic sides

Sheet of acrylic the correct size to fit inside the lantern



Right You don't need a huge list of parts to make a big visual impact. With these few parts, you can recreate one of the most famous stage effects of the 19th century



Left
The clear acrylic you use for the screen reflector should be as close-fitting to the lantern as possible, and free from scratches. You don't need to be at exactly 45 degrees for the illusion to work, so you have some leeway when it comes to finding the best position to reflect the screen

The Pepper's ghost illusion wowed the theatre-going public in the 1860s, spawning a slew of ghost-themed plays. With this project, you can create your own version of this popular illusion to capture a fairy inside a lantern, using the power of an RP2040 board, LCD screen, a suitable GIF image, and transparent acrylic.

The Pepper's ghost illusion is actually much older than you might expect, and was first described in the 1500s. Although it's not a true hologram, the eerie effect that it creates often gets called as such, and is occasionally seen as an advertising gimmick at trade shows and conferences. The illusion uses an artfully positioned pane of glass or plastic to reflect a concealed object or screen in such a way that the image appears to be floating, semi-transparent, in mid-air. The technique is also the basis of how an autocue works: since the illusion is only visible from a certain position, it's possible for a public speaker to look directly at the text of an announcement without the text being visible to those people behind the transparent screen.

IT'LL WORK IF YOU BELIEVE IN IT

To make your own version of the Pepper's ghost illusion, you'll mount an acrylic sheet at 45 degrees to a concealed screen inside a lantern. You'll want the lantern to have glass sides, so that the semi-transparent effect of the illusion is apparent to the viewer. If you try to project the illusion against a plain background, the power of the effect is lost and you might as well just put the screen up there instead.

Before you fit the hardware into the lantern, you'll need to solder the pieces together, choose a GIF image you like, size it to match the size of your screen, and then convert it into a format that the Pico LiPo can process fast enough to display as an animation. After choosing a GIF, the easiest way to resize and manipulate it is with an online service like **ezgif.com**.

Crop or resize your GIF to 240×240, and choose an appropriate compression level. Save the GIF to your computer. You can also apply different playback and effects for your animation, like reverse or ping-pong.

ALL AT C++

Converting the GIF into a useful format for the Pico LiPo is easy, thanks to the work of Larry Bank. Larry has produced some extremely useful code that lets us shortcut a lot of the issues around playing back a GIF and connecting to the screen. So, download the image_to_c application from **hsmag.cc/ImageToC**, and the code from **hsmag.cc/SPILCDCode**. The image_to_c application lets you automatically reformat the GIF file in a way that's more easily handled by the bb_spi_lcd code, which takes the reformatted GIF and displays it on an SPI LCD screen. Assuming that you're using Windows, there's a →

Quick Tip

It's easy to run out of memory quickly when you're dealing with animated graphics. The Pico LiPo has a generous allowance, but keep this in mind when choosing a GIF.

Right ◈
It's worth creating a non-reflective black lining for the base of your lantern from felt or foam. The black base with a hole cut through to expose the screen will cut down any unwanted light or reflections from the concealed electronics

Below ◈
Solder the screen to the Pico LiPo, as shown in this diagram. You don't need to connect the BL pin. It's used to turn the backlight off and on, but it isn't used in this project

C++

Capturing a fairy in a lantern is a difficult task, and so is displaying a GIF image on an LCD. You'll be using C++ on your Pico LiPo to process the animation fast enough. Don't panic, while C++ is more complicated to set up than CircuitPython or MicroPython, it's easier than you think to get started. There are some excellent guides about how to get started with Pico and C++ at **hsmag.cc/PicoC++** if you're using Windows, or **hsmag.cc/SDKC++** if you're using Linux. The rest of this article will assume that you have installed C++, the Pico SDK, and the example files.

precompiled **image_to_c32.exe** in the **dist** folder of the image_to_c repository. If you read through the notes for the image_to_c app, you'll see that the application outputs the modified GIF to stdout. That means that if you have a file named **fairy.gif** on Windows and you want to create a properly formatted header file for C++, you'll use the command `image_to_c32.exe fairy.gif >> fairy.h`.

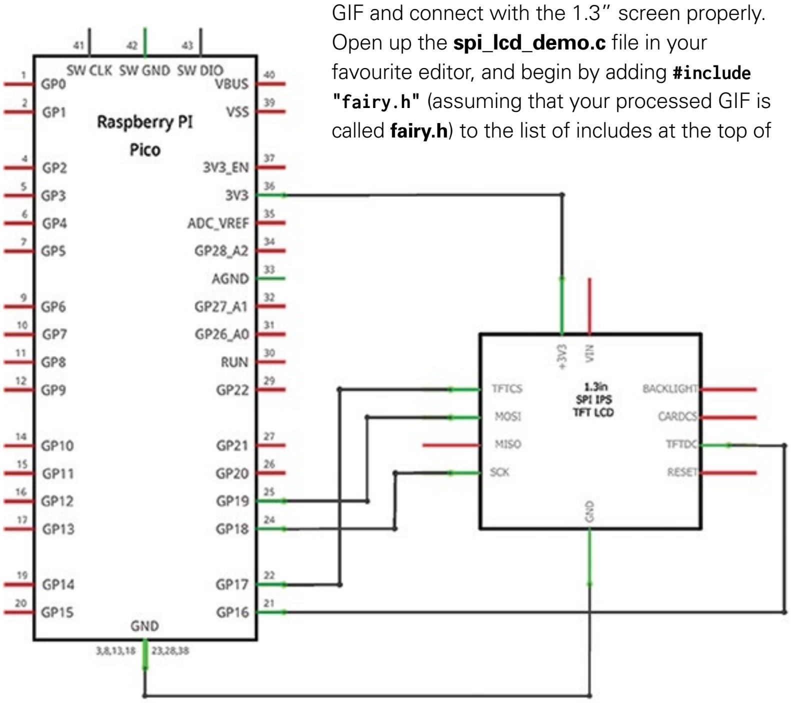
You'll need to make a few changes to the example provided with bb_spi_lcd so that it will load the correct GIF and connect with the 1.3″ screen properly. Open up the **spi_lcd_demo.c** file in your favourite editor, and begin by adding `#include "fairy.h"` (assuming that your processed GIF is called **fairy.h**) to the list of includes at the top of the code. Next, change the value of `#define DISPLAY_WIDTH` and `#define DISPLAY_HEIGHT 240`, to match the resolution of the screen.

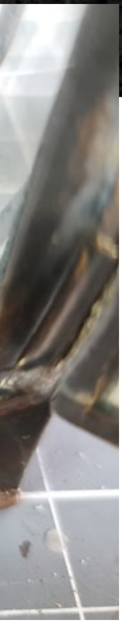
Jump down near the bottom of the code and find the `setup()` function. You need to change the call to the `spilcdInit` function so that it uses the correct LCD driver settings. The correct driver is LCD_ST7789_240, so the full line should read:

```
spilcdInit(&lcd, LCD_ST7789_240, FLAGS_NONE,
CLOCK_SPEED, LCD_CS, LCD_DC, LCD_RESET, LCD_
BACKLIGHT, LCD_MISO, LCD_MOSI, LCD_SCK);
```

Finally, find the `main()` and look for the `if` statement that encapsulates a `GIF_openRAM()` call. Change the call to read `GIF_openRAM(&gif, (uint8_t *)fairy, sizeof(fairy), GIFDraw)`. The name 'fairy' here refers to the name of the const defined in the **fairy.h** file you generated from the GIF file, and is normally the same as the original GIF file name.

Save the changes, and you're almost ready to compile and upload the file to the Pico. You'll need to copy the **pico_sdk_import.cmake** file from your **pico-sdk/external** folder into the folder where you've downloaded the bb_spi_lcd repository. This file helps the compiler to locate the Pico SDK on your system. If it isn't the **bb_spi_lcd** folder, then you'll probably get an error when you try to compile. As is traditional, create a folder called **build** and navigate into it. Do `cmake ..` to create your build files, and then use `make` to build the project. After a few screens of hopefully green text have passed by, you should have a file named **spi_lcd_demo.uf2** in your **build** folder. Attach your Pico LiPo to your computer via USB, and put it into bootloader mode by turning it on with the BOOT button pushed down. Copy the UF2 file to the Pico LiPo (which should appear on your system as a drive





named RPI-RP2). Restart the Pico LiPo and you should see your GIF playing on the screen.

Now that the hardware is done, you can set about fitting it all inside your lantern. The exact instructions for this will vary depending on the size and shape of the lantern that you're using, but there are a few tips that might make the process a little bit easier. Firstly, pay attention to the location of the USB socket and the power button on the Pico LiPo. You'll need access to the USB port to charge the batteries, and the power button to turn the board on and off. You should also be aware of the lights on the board and make sure that they don't interfere with the illusion.

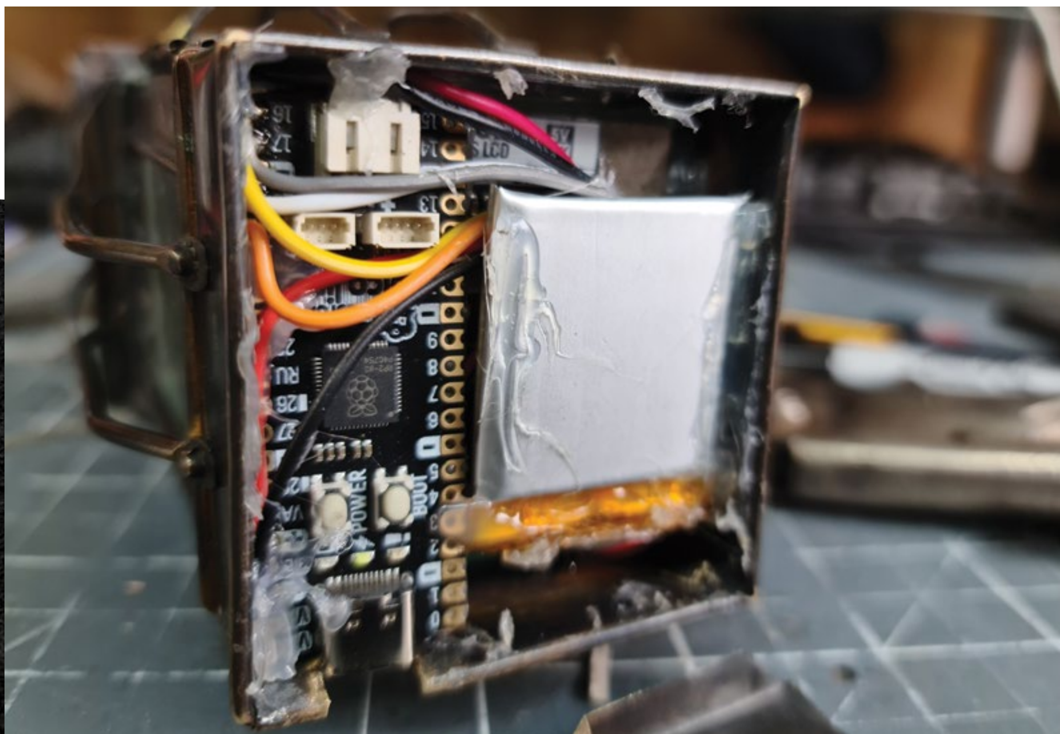
If the base of the lantern is too inaccessible, you could try fitting everything into the top of the lantern instead. As long as the screen is concealed from sight, the illusion should work fine. Keeping your cables short and soldering wires directly to the boards rather than using DuPont connectors can help to keep the footprint of the electronics small.

Chances are high that this project will get handled quite a lot, so more hot glue is probably better than less. If you're not interested in capturing a fairy, you could use the same effect to play a holo-message from the Rebel Alliance, revive a Hogwarts ghost, or visualise the great and glorious Oz in portable format. □

More hot glue is probably better than less

Quick Tip

If you got lost in all of the C++ speak, don't panic. Just copy the UF2 file provided with this project onto your Pico LiPo and it will start displaying a flapping fairy.



Above Test-fitting the screen is important to find the best position for the illusion. Expect to assemble and disassemble the parts a few times to get things right

Left Thanks to the small size of the Pico LiPo and screen, you can squeeze this project into even the most modest of tealight lanterns

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

Getting started with slicing for 3D printing

Turn your digital designs into instructions for your printer

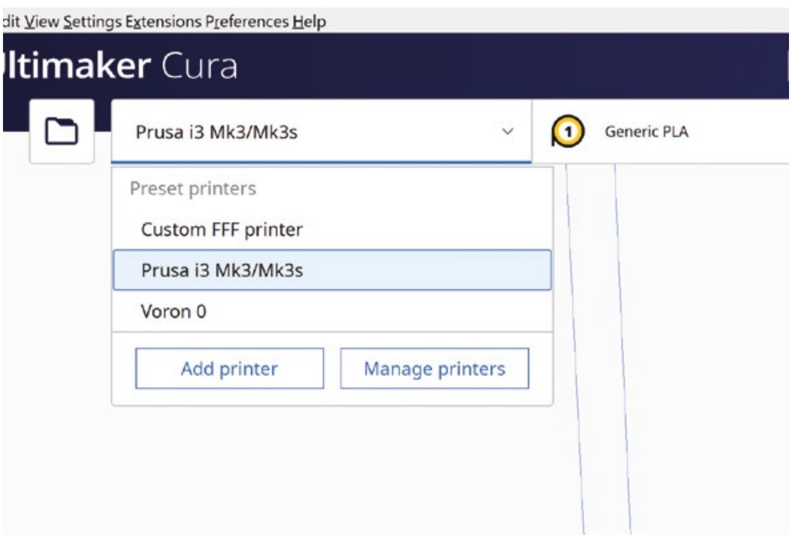
Ben Everard

@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.

Most 3D printers (like most plotters, laser cutters, and CNC mills) understand a language called G-code. This is a text-based language that describes what they should do. Each line relates to a command, such as a movement or the setting of a particular parameter. Slicing is the process of taking a 3D design and converting it into a series of G-code instructions.

It's called slicing because, usually, 3D printers print a series of layers (aka slices), one on top of the other. There's nothing inherent about the hardware of 3D printers that requires them to print like this, but doing it this way simplifies the whole process.



Above
You need to tell Cura which printer you'll be using

There are two common slicers for hot plastic 3D printers: Cura and PrusaSlicer. We'll look at how to use them shortly, but first, let's start with an overview of what we'll be trying to achieve.

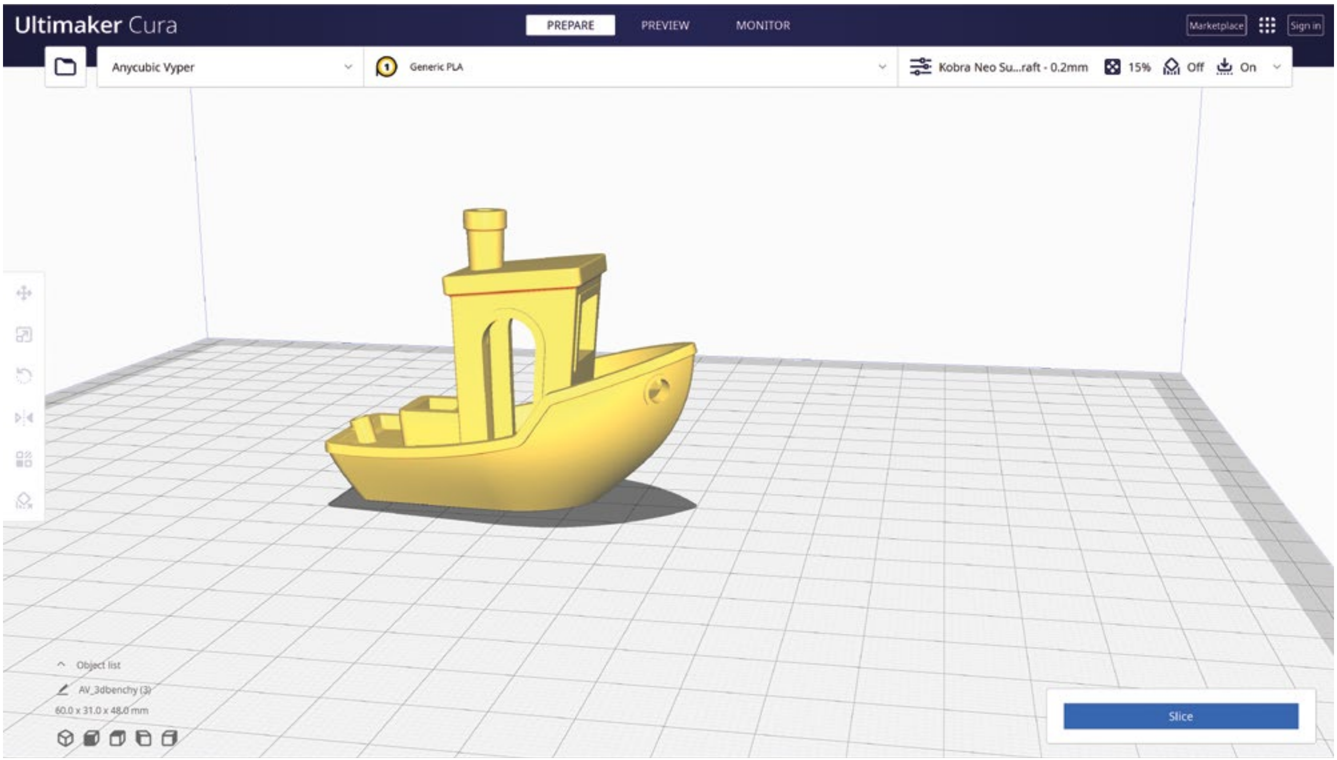
Usually, you start with a 3D design in an STL file, but some slicers can now read STEP files, so this might become a more common format in the future. Whichever format it is, you'll start with a file that describes the shape of the 3D object.

Since we're trying to get machine-specific instructions, we need to ensure that the slicer is configured for the exact model of printer we have. Some of this is fixed – for example, a printer will have an exact print area, and any attempt to print outside this will cause a problem. Some of it is more fluid – for example, there will be recommended speeds that you can print at – a well-setup printer may be able to go beyond this. All of these things should be set in the printer configuration that you'll need to load into your slicer. Generally, the 3D printer manufacturer will supply this, and you should have instructions for setting up a slicer with your printer. As you gain experience, you might want to tune it to your particular setup, but it's best to start with the manufacturer's recommendations.

Next, you must ensure you have the correct configuration for the particular filament you're using. Usually, printers will come with a recommended profile for PLA; if yours doesn't, or if you're using an unusual filament type, these are often fine to share between different printers.

There are some differences between different manufacturers of filament, and we'll look at how to tweak this for your particular filament a little later, but for basic use, the default profile should be fine.

Finally, there are some options that you might want to tweak on a per-print basis. The main one is



Above The main window of Cura lets you position the object where you'd like it

the layer height. This is the depth of each layer in the print. Smaller layers will print slower but give more detail, while thicker layers will print faster but with less detail. If you need parts to fit together, you might find that you need a thinner layer height to get the parts the exact shape needed.

Aside from the settings for the printer, you will also have some settings for the model itself. Once you import it, the first thing you need to decide is which way up to print it. Sometimes, model designers will ensure that the model automatically imports the right way up, but not always.

A few things to consider when deciding which way up to print a model:

- The print will be significantly weaker along layer lines, so if there's any force on the model, you want to ensure that it's perpendicular (or at least not parallel to) the layers

- Support material (see next step) will slow down a print and waste plastic, so it's good to minimise it

> "It is possible to cut a model up into **smaller pieces and print them separately**"

- Any flat surface that sticks on the print bed will be the smoothest flat surface on your print

- A large flat surface on the print bed will help layer adhesion

Of these points, you can mitigate against all of them except the first, so that is perhaps the most important.

If you can't satisfy your needs with any orientation, it is possible to cut a model up into smaller pieces and print them separately, but this can introduce problems where warped prints don't fit together as they should. →

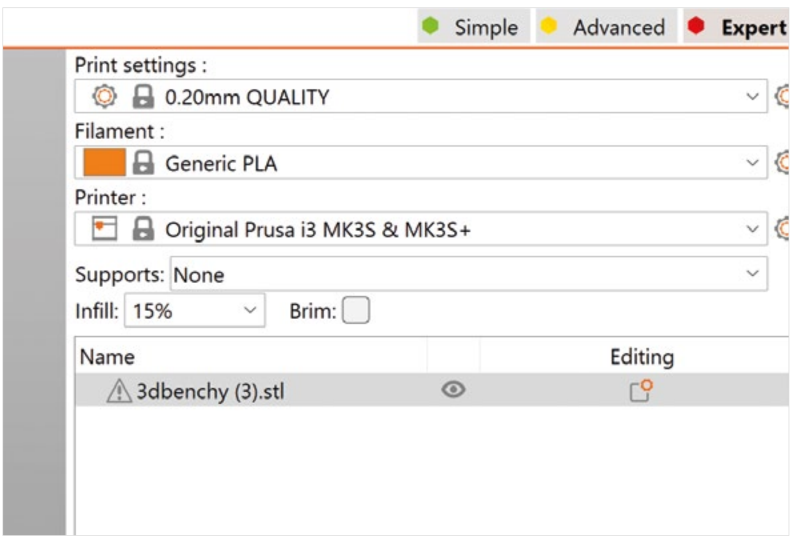
The final two things that you may want to consider are whether you need supports and whether you need a brim. Your printer should be able to handle a certain amount of overhang (where a layer above protrudes beyond the layer below) and bridging (where a top piece is supported on either side but not in the middle). However, if things go too far, you'll need some support.

Support material is a structure printed alongside your model to support overhangs and other bits that don't have model below them to print on top of. Once you've printed, you can snap or cut away the support material, leaving you with just the model you wanted.

There aren't any hard and fast rules for when you'll need support other than the fact that it's not possible to print in thin air. The safest option is to let your slicer add supports if it thinks they're necessary, but it will be very conservative, and you'll have far more supports than you need. With experience, you'll learn what your printer can handle, and sometimes it's good to try prints that you're not sure about without support, just to see if they'll print.

The risk of not having supports where you need them is either a messy print, if you're lucky, or a completely failed print.

Below
This author has a bad habit of forgetting to change the filament setting to the one in his printer



If you only have a small amount of print in contact with the print bed, you might want to add a brim, which is a bit of print one layer high around the edge of the print to help it stick down.

Let's look at how to do this for the ever-popular Benchy model in both Cura and PrusaSlicer. You can download the **3dbenchy.stl** file from **hsmag.cc/Benchy**.

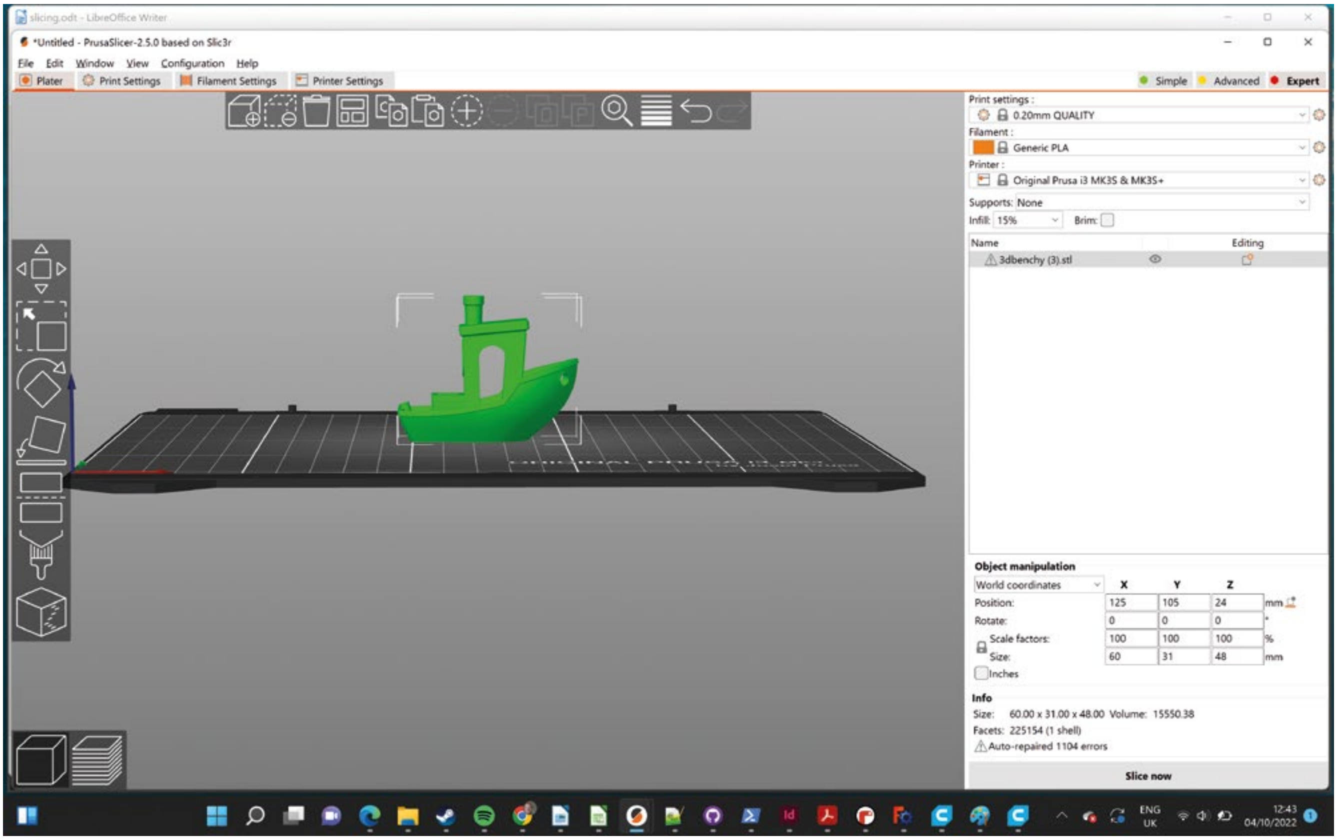
"The risk of not having supports where you need them is either **a messy print, if you're lucky, or a completely failed print**"

CURA

First, open Cura and start a new project if you don't already have an empty build plate (File > New Project). Add the Benchy model with File > Open and select **3dbenchy.stl**. This will load the model onto the build plate.

You can view the build plate from different angles by clicking the boxes in the lower left-hand corner.

If you click the model, it will get a blue outline to show that it's currently selected. The most important thing with a model is to ensure it's the right way up. Benchy should already be the right way up, but it's worth getting familiar with the controls. Hover the mouse cursor over the toolbar to see what each tool does. We want 'Rotate'. There should be three options: 'Free rotate', 'Lay flat', and 'Select face to align with build plate'. We rarely find the 'Free rotate' useful – at least for getting things the right way up; it's sometimes useful for positioning items diagonally on the build plate. 'Select face to align with build plate' is usually the best for lining things up. Click this (the square will get a black outline), and then click on the Benchy to line this up with the build plate. We want the boat the right way up, but it's worth getting familiar with this tool so you can line other models up.



PRUSASLICER

Starting with a blank project (you can go to File > New Project if there's already one loaded), import your Benchy STL with File > Import > Import STL and select your Benchy STL. This should appear on the build plate.

Benchy will appear on the build plate highlighted in green. Currently selected objects are green; unselected objects are orange. Click anywhere to deselect Benchy – it will turn orange. You can click and drag anywhere on the build plate to spin it in 3D to see how the object looks.

Click the object to highlight it green, and this will activate the toolbar on the left-hand side. You can hover your mouse cursor over each button to find out what it does. The most important of these, at least when you're getting started, is 'Place on face'. This will let you select which face you want on the build plate. If you select this tool, all the possible faces that could be placed on the build plate will be highlighted in translucent white. In the case of Benchy, it's already the right way up, so you are ready to go.

The print settings in the top-right control how the file will be sliced. Make sure that you have selected the correct layer height, filament, and printer. You can also select supports, infill, and brim. We don't need supports for Benchy, but you can select either supports everywhere or supports on the build plate only. A brim will help objects with a small footprint stick to the build plate. An infill of around 15% is fine for Benchy.

With all these set, select 'Slice now' to slice your model. This changes the interface slightly – in the bottom right-hand corner, you'll see the option to view the 3D editor or the preview. Up until now, we've been working in the 3D editor, but now we're in Preview mode. If you want to change anything, you need to switch back to the 3D editor. If you change anything that affects the slicing, the preview will go blank, which can be a bit confusing, but just select 'Slice now' again to regenerate the preview.

The orange vertical line can be used to select which layers are visible in the preview. This can be useful for taking a close look at areas where you're not sure about support material.

If you're happy with the result, press 'Export G-code' to save the G-code to a file ready to send to your 3D printer.

Above
All the key settings are listed on the right-hand side

PHA filament

Can truly compostable printer filament really work?

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.

Plastic pollution is a scourge on the planet. It's clogging our rivers, oceans, and almost every bit of land. These plastics will stay in the ecosystem for a few millennia and are almost impossible to clean up. Anyone creating things out of plastic needs to think carefully about how they are doing so responsibly.

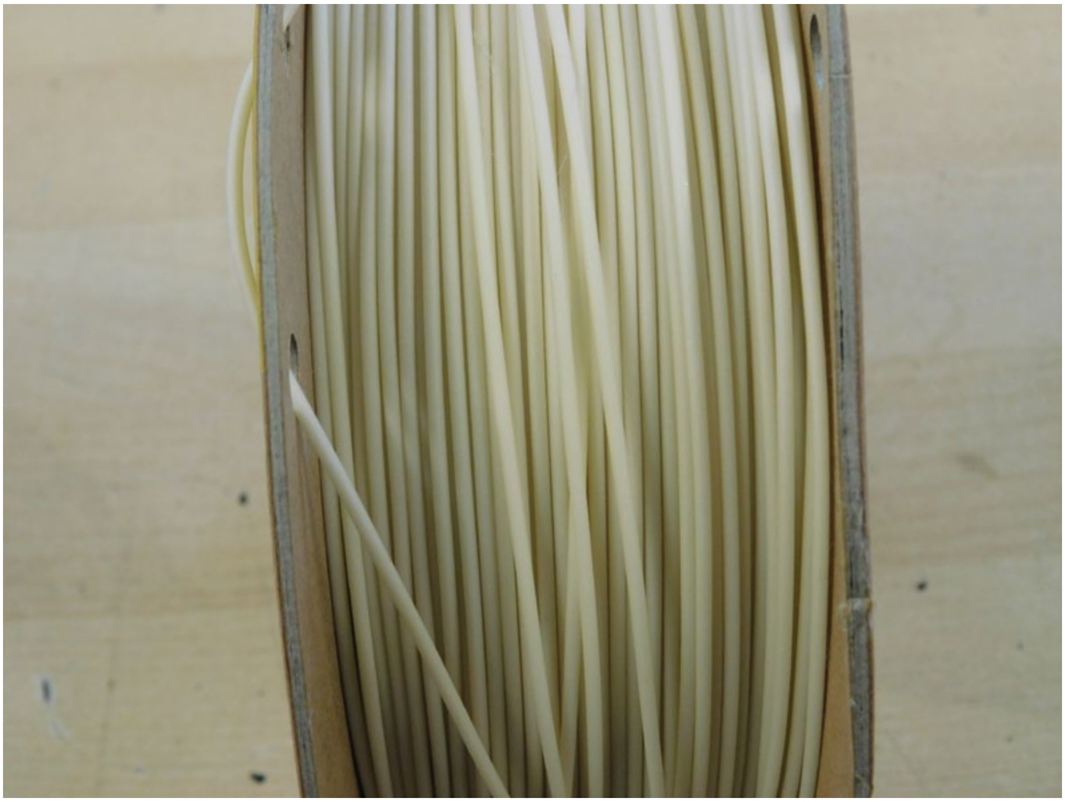
Obviously, people who 3D-print tend to use quite a bit of plastic. The most common plastic used in 3D printing – Polylactic acid (PLA) – is a bioplastic that is compostable. At least, it's theoretically compostable. In order to compost, it needs to be held over 60 °C for an extended length of time. Some compost systems do reach these temperatures, but if any PLA objects get accidentally discarded in the environment, they'll sit around for a long time.

However, there is another option – Polyhydroxyalkanoates (PHAs). Like PLA, PHA is a bioplastic, but it biodegrades much more easily. There are some filaments that are a blend of PLA and PHA; however, for this article, we're only looking at pure PHA as this is the only filament that's easily compostable. Specifically, we tested out ColorFabb's allPHA filament, but other pure PHA filaments should behave similarly.

The main problem with PHA is that it doesn't like sticking to the build plate. Part of this is because you need to have a cold build plate (to avoid crystallisation). For small parts, a brim might be enough, but for most parts, you're going to need something to help stick it down.

We didn't have a lot of luck with a glue stick, but we did find that 3DLAC spray adhesive worked well. This both adds to the cost and the environmental

Right The bugs are making their homes in the filament, but four weeks in, it's not yet decomposing



Left
PHA filament comes in various colours, but this is its natural hue

impact. At this point, we're left with a bit of an eco-conundrum. Is it better to print with PLA which isn't really compostable but is a bioplastic, or print with PHA which will compost but leaves you with more waste in the form of spray adhesive cans? Part of this comes down to the type of printing you're doing. If you do fewer larger items, then you'll use less 3DLAC per spool of filament than if you're printing lots of little things.

In perhaps the largest irony in 3D printing, a second problem with PHA is that occasionally it sticks so well that it becomes very hard to remove. In this case, ColorFabb (one of the main manufacturers of PHA filament) recommends heating your build plate to 90 °C to encourage the filament to crystallise. We found that this made it a bit easier, but it was still a chore to get it off.

PHA has good layer adhesion, and the plastic is slightly flexible, so it should absorb impacts well. The slightly malleable feel makes it more tactile than other plastics.

PHA also has good environmental credentials – it's actually compostable, and the lack of a heated build plate means significant electricity savings. However, adhesion is a challenge. Part of this may simply be that it's not a common filament, so while printers are optimised for PLA, anything that needs a slightly different printing environment is perceived as being troublesome. It might be that there are some build surfaces that are better suited to PHA but that aren't in common use yet. Whatever causes it, this adhesion challenge is a big issue for PLA – we found many of our parts warped, sometimes very significantly.

"The lack of a heated build plate means **significant electricity savings**"

Ultimately, it's hard to see PHA as a drop-in replacement for PLA for general-purpose printing at the moment. The extra cost of the filament and the need for adhesive, as well as the increased likelihood of warping, means that it's just not quite suitable, at least at the moment.

We should all be conscious of our waste streams. What happens to objects after you've printed them? What happens to support material? Perhaps you are able to recycle or compost PLA, but if you're creating something that you know is likely to end up in the biosphere, or something that's disposable and large amounts of it are going to be discarded, PHA is a responsible choice, and it might be worth putting up with the extra difficulty in printing.

Part 02

Build a robot: add sensors to the chassis

MAKER

PJ Evans

PJ is a writer, software engineer and tinkerer. His robots bring all the geeks to the yard.

twitter.com/mrpjevans

Last month, we started our build of the CamJam Robotics EduKit. Now we have a roving robot, it's time to add some smarts!

If you followed last month's tutorial, you should now have a working robot that you can control with Python. Hopefully, you've played with the code and had the little 'bot' zooming around the place. Now it's time to add some sensors, so our new pal can start to sense the world around it. With the ultrasonic and light sensors included with the CamJam EduKit #3, we can add some autonomous capabilities and make our robot a little smarter. Finally, we can look at what you can do to improve the robot even more with custom chassis and additional sensors.

01 Get sensitive

Included with your CamJam kit is a light sensor. It works by sending out infrared light (that we can't see) and detecting how much of it bounces back to a sensor. If we point it at the ground and measure the sensor's output, we can easily tell when the robot passes over a line. The key to success is high contrast, so a jet black line on a white surface is perfect. We're going to mount the sensor on the front of the robot, point down, so we can detect a line.

02 A little light wiring

To wire up the light sensor, we're going to use the breadboard (the small block with lots of holes). Holding it with the longer edge horizontal, each column of holes are connected together, with a gap in the centre. Breadboards allow us

You'll Need

- CamJam Edukit #3 - Robotics **magpi.cc/edukit3**
- Printer
- Roll of paper (optional)

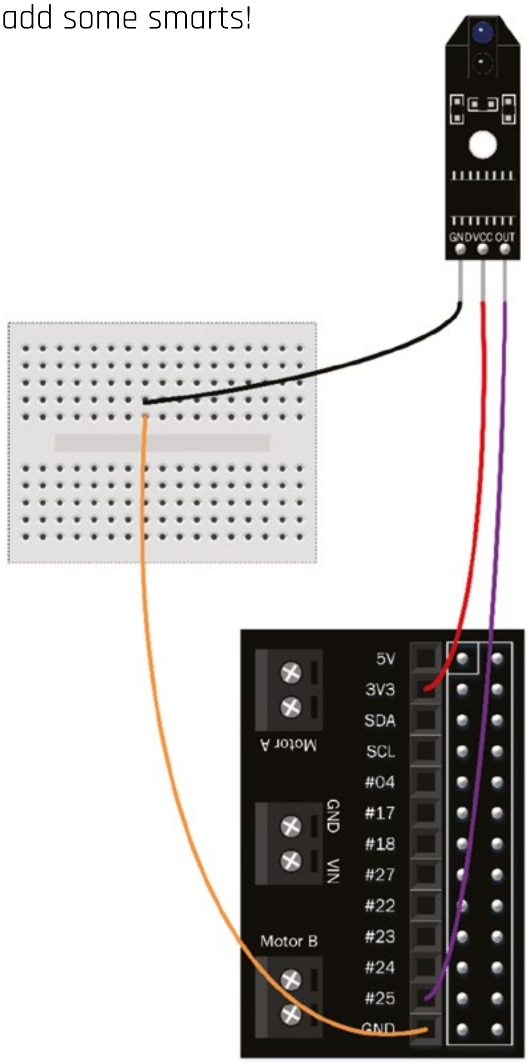
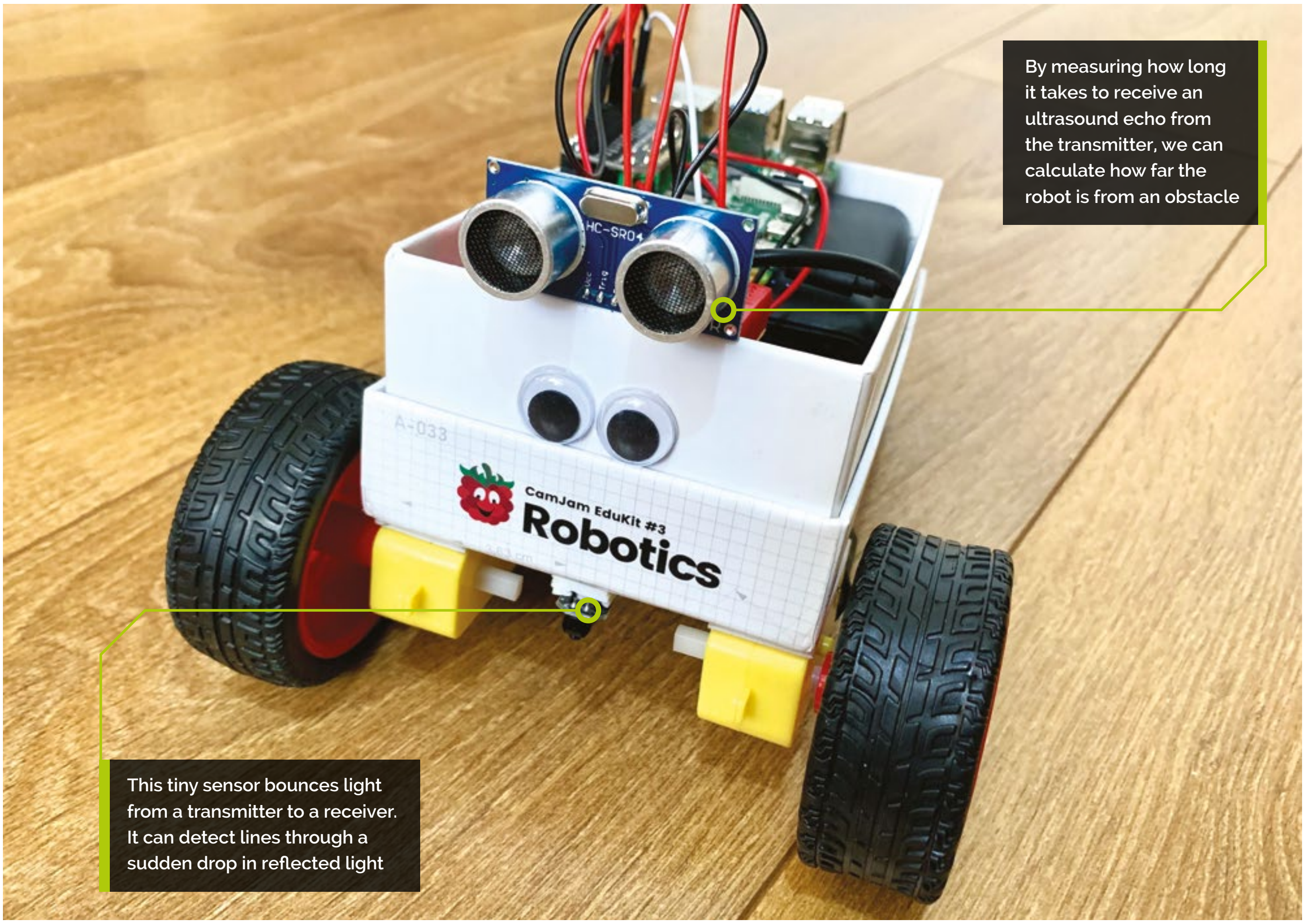


Figure 1 Wire up the line sensor to the HAT. Use the breadboard to create a 'ground rail'



By measuring how long it takes to receive an ultrasound echo from the transmitter, we can calculate how far the robot is from an obstacle

This tiny sensor bounces light from a transmitter to a receiver. It can detect lines through a sudden drop in reflected light

to connect circuits together without soldering, so we can quickly prototype circuits and correct mistakes. You'll need three wires: two plug-to-socket, and one plug-to-plug. Wire everything together as shown in the diagram, checking and double-checking. There are three connectors on the light sensor for power, grounding, and data. These all need to match up with the connector on the robot HAT connected to your Raspberry Pi.

03 Mount the sensor

The sensor needs to be in a sensible place on the box chassis, and that would be in the centre at the front on the base. However, this is also the highest point of the body, and the further away the sensor is from the ground, the less accurate it will become as ambient light leaks into the sensor. Start by making a hole off-centre towards the front of the chassis and feeding the three wires through it. Connect the wires to the sensor as shown in **Figure 1**, then mount the sensor to the body with sticky pads. You may find a couple of LEGO bricks will sufficiently lower the sensor to the ground.

04 Testing time

The sensor will not work without a little code to help it on its way. Enter the code from the **test_line.py** listing, overleaf (or download it from the GitHub repo: **magpi.cc/testlinepy**), and save it in your home directory as **test_line.py**. Now run the code:

```
python3 test_line.py
```

Don't worry, your robot won't move at this point. What we want to do is check the sensor is working correctly. Using a sheet of paper with a thick black line through it (**magpi.cc/testlinepdf**), hold your robot carefully and pass the paper under the sensor. If all is working well, you'll see messages on-screen that the line has been detected.

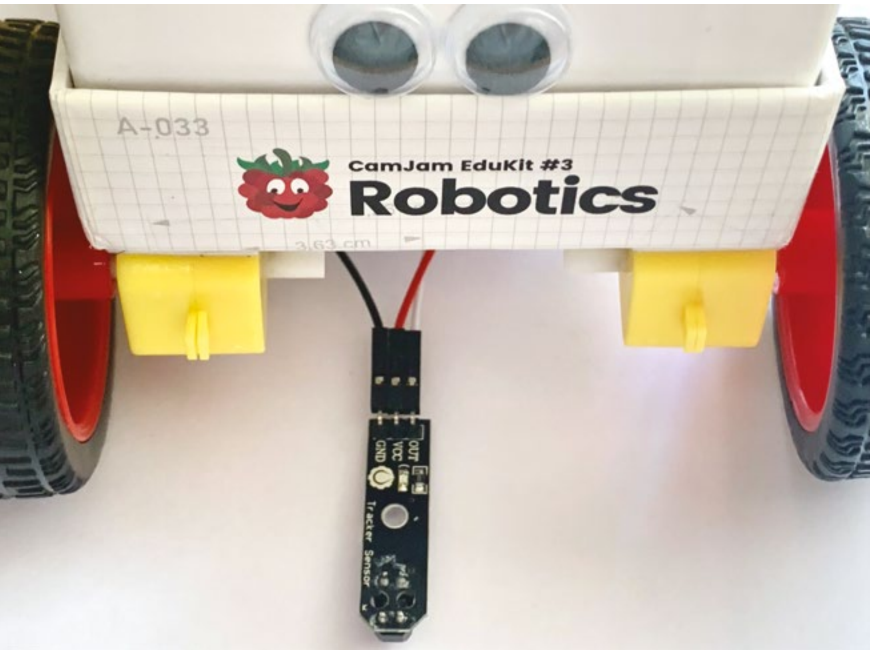
05 Follow that line

Now our little pal can detect a line, we can make them follow it too! By combining code to drive the robot forward and steer it left and right, we can make corrections as we go. This listing is a →

Top Tip

Safety First

Whenever connecting wires to your Raspberry Pi computer, ensure it is completely powered-down. A mistake when the computer is on can cause permanent damage.



Here is the wiring on the line sensor. The lower the sensor can be mounted to the ground, the more accurate the result will be

little longer, so download it to your home directory from **magpi.cc/linefollowerpy** and try it out. Place your robot at the start of the line and then run:

```
python3 line_follower.py
```

Hopefully, your roving friend will scoot along the length of the page. Do you notice how it's slower and more controlled? We're using pulse-width modulation (PWM) to slow the motors down. You can play with the setting by changing the `leftmotorspeed` and `rightmotorspeed` variables.

06 Make a course, of course

OK, so our new friend can follow a line. How about an entire course? If you've got access to a big roll of white paper, try mapping out a course for the robot to follow. If not, you could stick pieces of A4 paper together. Make it as big as you can, without any tight corners to which the robot may not be able to respond. Use a pen such as a Sharpie to create the line to follow, and make it nice and thick, like the one on the printout. Now watch as your newly smarter robot follows the line in circles.

07 Looking into the distance

If we want our robot to be able to move around a room on its own, there's a significant problem: walls. Our final modification is to add a distance sensor to the robot, so it can take avoiding action when it gets close to an obstacle. The sensor works by transmitting an ultrasonic pulse and detecting when it is returned. With a bit of maths, we can use the time taken ('time of flight') to calculate how far away the obstacle is from the robot. The wiring is a little more complicated for this as the sensor needs 5 V to work, but must only return a 3.3 V signal to avoid damaging your Raspberry Pi 4.

08 Wiring up for safe voltage

Study the **Figure 2** wiring diagram carefully. Mount the sensor to the breadboard along the long edge, so each connector has its own column (or 'rail'). Move the two existing ground connectors for the line sensor so they are on the same rail as the GND pin for the distance sensor. Connect TRIG to #17 on the CamJam HAT. Finally create a 'voltage divider' to reduce the return voltage to 3.3 V. Do this by adding the supplied 470 Ω resistor to bridge the GND rail to any spare rail. Now add the 330 Ω resistor to bridge that spare rail to ECHO. Finally, link the the spare rail to #18 on the HAT.

Top Tip

Colourful resistors

Resistors are identified by coloured stripes on their body and can be used either way around. The 470 Ω resistor is Yellow, Violet, Black, Black, Brown; the 330 Ω is Orange, Orange, Black, Black, Brown.

test_line.py

> Language: **Python 3**

DOWNLOAD THE FULL CODE:

magpi.cc/testlinepy

```
import time
from gpiozero import LineSensor

# Set variables for the GPIO pins
pinLineFollower = 25
sensor = LineSensor(pinLineFollower)

def lineseen():
    print("Line seen")
def linenotseen():
    print("No line seen")

# Tell the program what to do with a line is (un)seen
sensor.when_line = lineseen
sensor.when_no_line = linenotseen

# Repeat the next indented block forever
while True:
    time.sleep(10)
```

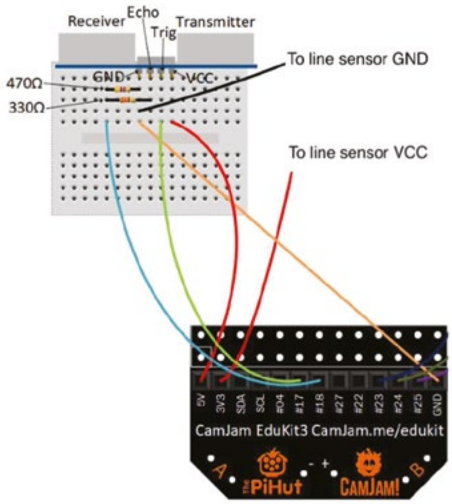


Figure 2 Build this circuit to get readings from the distance sensor. Always triple-check everything before powering-up!

09 Facing forward

To get an accurate reading, the ultrasonic sensor needs to be mounted facing forward in the centre. You may have to get a little creative to find the best way to attach the breadboard so it fits. We used a bit of double-sided sticky tape on the breadboard to hold in place, so the sensor sat over the edge of the box. A small cardboard or plastic box for the board to sit on would also work well. Another option is to remove the sensor from the breadboard altogether and use four jumper wires to reconnect it, allowing the breadboard to sit on the base.

10 Testing from a distance

Let's create another test file. In your home directory, create a new file called **test_distance.py** and add the code from the listing here (or download it from **magpi.cc/testdistancepy**). As before, run this code from the command line:

```
python3 test_distance.py
```

Watch the output from the screen and move your hand towards the sensor. If everything is working, you'll see measurements of the distance from your hand to the robot. This is calculated by taking the output from the sensor (the elapsed echo time in seconds), multiplying it by the speed of sound (34,326 cm per second) and then halving, as it has made an outward and return journey.

11 You own autonomous robot

Congratulations! Your robot build is now complete. Let's combine all the parts of the robot in one last Python program. It's a bit long, so you can download it here: **magpi.cc/avoidancepy**. The code will move the robot forward until it detects an upcoming obstacle. It will then back off and turn right. It will then advance forward. If the obstacle is still in place (or a new one is found), it will repeat the process until the obstacle is cleared. These simple instructions will result in a robot that will happily trundle around the room until you stop it or the batteries wear out!

12 Make it your own

You are the proud custodian of a line-following, obstacle-detecting robot. The good news is, that's not the end of your, or your robot's, journey. Now you have the basic building blocks of code, try and make your robot do more. Could you add dynamic speed control based on distance from an obstacle? Are there other sensors you could add? How about 3D-printing Daniel Bull's custom-designed chassis (**magpi.cc/robotchassis**)? Or, use this project as a starting point and design and build your own robot. Try replacing the CamJam HAT with a motor controller board and upgrading to four-wheel drive. Whatever you decide, have fun.

Many thanks to Mike Horne and Tim Richardson of CamJam for their help sheets and code that informed this tutorial.

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**

test_distance.py

Language: **Python 3**

DOWNLOAD THE FULL CODE: **magpi.cc/testdistancepy**

```
import time
from gpiozero import DistanceSensor

# Define GPIO pins to use on the Pi
pintrigger = 17
pinecho = 18
sensor = DistanceSensor(echo=pinecho,
trigger=pintrigger)

# Check the distance every half a second
while True:
    print("Distance: %.1f cm" % (
sensor.distance * 100))
    time.sleep(0.5)
```

Building a web app for Pico W or Automation 2040 W

Turn almost anything into a wireless device

Ben Everard

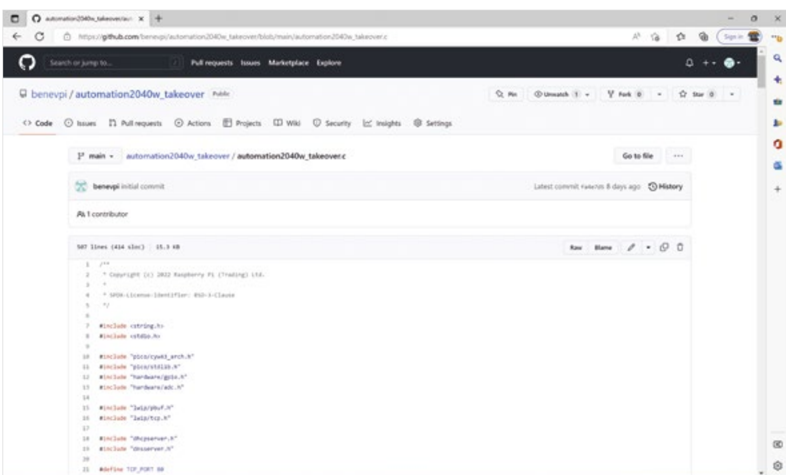
@ben_everard

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.

Raspberry Pi Pico W is a fantastically powerful development board, but, like many, it's limited to a puny 3.3 V, and the GPIOs can only handle a smattering of microamps. This is fine for blinking LEDs, but if you want to run motors, or interact with more powerful components, you're going to need extra circuits. There's a whole host of options for this, including relays, motor drivers, and power MOSFETs, and you could add them to Pico W yourself. Fortunately however, the good folks at Pimoroni have already done this for you.

In this article, we'll look at working with the bare Pico 2040 W board and the Automation 2040 W. The latter beefs up most of the components: the analogue inputs, four input IOs, three output IOs, and three relays can all handle up to 40 V. In other words, it's a great device for integrating with a huge range of equipment.

Below You can grab the code from GitHub



The Automation 2040 W doesn't come with any firmware, as it's up to you to write code that makes sense for your project. Obviously, this could be a huge range of different things, depending on what you use the board for. To give you a sense of what's possible, we'll look at building a web interface that runs on Pico W and controls the inputs and outputs.

While there is quite a bit of circuitry on this board to handle higher voltages and currents than a bare Pico W, we control all this with GPIO pins, just as if we were controlling lower voltages. So the same technique will work on other Pico W-based boards.

In fact, we've also built a firmware for Pico W. The slight difference here is that, unlike Automation 2040 W, any pin can be either input or output. We've arbitrarily split the pins roughly in half, but you can change this so that whichever pins you like do whatever you like.

Before diving into the project, let's think a little about what our code should do:

- It should work with or without an existing WiFi network
- It should be able to read and display all the digital inputs
- It should be able to read and display all the analogue inputs
- It should be able to control all the outputs
- It should be able to control all the relays
- You should be able to control the outputs easily from a program running on another machine.

In this tutorial, we'll work backwards. To start, we'll show you how to flash the finished firmware onto your Automation 2040 W, and then we'll show you how it all works.

First, download the firmware for Automation 2040 W from **hsmag.cc/Automation2040**; or for Pico W, from **hsmag.cc/picowtakeover**. In both cases, the compiled firmware that you want to use (at least at first) is **xyz_wireless_ap.uf2**. You can flash this to your board by unplugging the USB (if connected), then holding down BOOTSEL, and plugging it back into your computer before releasing BOOTSEL.

You should then see a wireless network called **Picow_test**, and you can log into this with the password **password**. This isn't the most secure option available, and should really only be considered a test mode. You can change the SSID or password in the code and recompile if you want to.

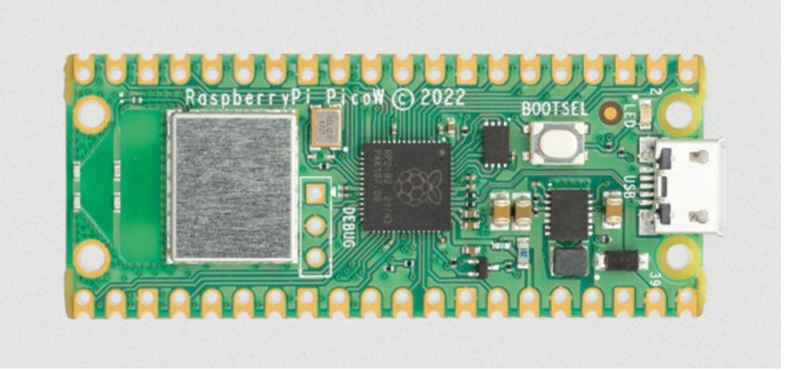
Once you've logged into that network, point your web browser to **192.168.1.4** and you'll see a web page with the basic stats of the IOs. You can see if the inputs are high or low, if the outputs are on or off, and what the voltage at the analogue inputs is.

> **"The very simple web server running on Pico doesn't** really pay much attention to the URL you send to it**"**

The very simple web server running on Pico doesn't really pay much attention to the URL you send to it. **192.168.1.4** is the IP address of the server, so you have to head there. It only listens for an HTTP request – it doesn't really care about the actual location you're requesting.

However, it does listen for parameters. In URLs, these come after a question mark, and multiple parameters are separated by an ampersand. The two parameters this server is particularly looking for are GPIO and state. Visiting the page **http://192.168.1.4/?GPIO1&state=1** will turn GPIO 1 on; visiting **http://192.168.1.4/?GPIO1&state=0** will turn it off. For general use, you don't need to know this, as there are links on the page to turn the IOs on or off. Note that the GPIO parameter refers to the actual GPIO number on Pico W rather than what is printed on Automation 2040 W.

So far, so straightforward. There are a couple of things that you might want to change: the look of the web page and the networking. To do either of these, you need to recompile the code. You can do this either by downloading and installing the toolchain, see **hsmag.cc/PicoSDKSetup** or by running it in a Google Colab environment (see box).



Above Pico W is a great device for quickly adding wireless connectivity

The CMake builds are called **automation2040w_takeover** and **picow_takeover**.

Now you've got a way of compiling the code, let's take a look at how you might want to change it. First, let's examine how it generates the HTML for the page. There's a framework for the HTML that's stored in the **macro HTML_SOURCE**:

```
#define HTML_SOURCE "<html><head><title>Automation
2040W Takeover</title></head><body><h1>Automation
2040W Takeover</h1> \
      <p>%s</p><h2>ADC</h2><p>%s</p>\
      <h2>GPIO Inputs</h2><p>%s</p>\
      <h2>Relays</h2><p>%s</p>\
      <h2>GPIO Outputs</h2><p>%s</p></body><html>"
```

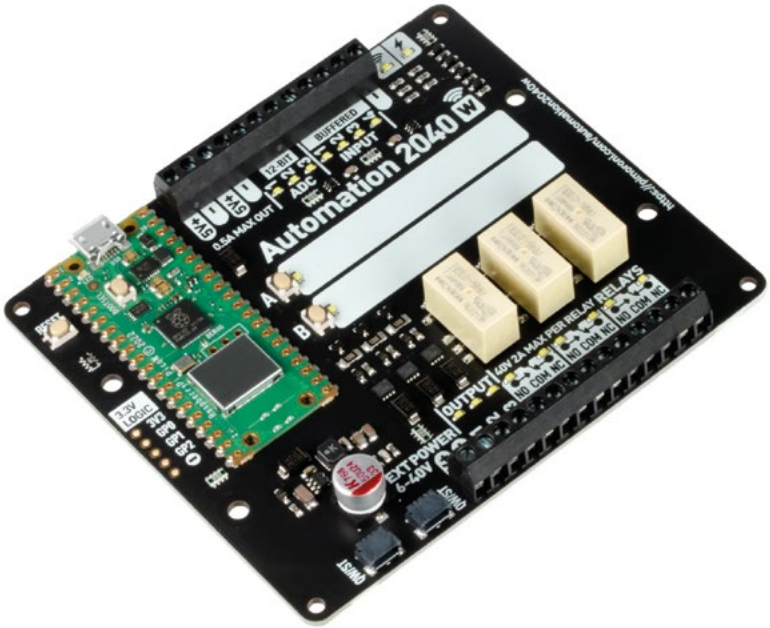
For those of you familiar with C, this should be fairly straightforward. If you're not, it might look a bit odd, but the basic thing you need to know is that macros created with a `#define` command will insert the string wherever `HTML_SOURCE` is in the code – in essence, it acts a lot like a variable (albeit a constant one). The `%s` is part of C's string formatting and identifies where we want other strings to be placed. →

GOOGLE COLAB

Installing the Pico SDK on macOS or Linux is straightforward, but on Windows it's a bit of a pain. Fortunately, there is a solution – run it in the cloud. Google's Colab environment gives you an interface to an online virtual machine running Linux. You can open a page setup for this at **hsmag.cc/BuildingBlink**.

This walks you through how to compile the pico-examples, but you can use exactly the same process to compile your own code provided that you keep it on GitHub (or another online Git repository).

How long this process will work for depends on how long Google keeps Colab going. Given the firm's history of killing projects, it's possible that you might find this suddenly no longer works in the future.



Above The Automation 2040 W can take a wide range of voltages

These strings are generated a bit later in the code. First, there is a string with what (if any) GPIOs were updated during the page load:

```
int update_state = snprintf(update, 32, "<p> GPIO
%d updated to %s</p>", gpio, state?"on":"off");
```

Then we generate text for the analogue inputs:

```
#define NUM_ANALOGUE_GPIOS 3
int gpios_analogue[] = {26,27,28};

…

char gpio_analogue_lines[500] = "";
const float conversion_factor = 45.0f / (1 << 12);
counter = 1;
for(int i=0; i<NUM_ANALOGUE_GPIOS; i++) {
        if (gpios_analogue[i] == 26) {
                adc_select_input(0);
        }
        if (gpios_analogue[i] == 27) {
                adc_select_input(1);
        }
        if (gpios_analogue[i] == 28) {
                adc_select_input(2);
        }
        if (gpios_analogue[i] == 29) {
                adc_select_input(3);
        }

        uint16_t result = adc_read();

        snprintf(line, 64, "ADC %d has an
analogue voltage of %f<br>", counter,
result*conversion_factor);
        strcat(gpio_analogue_lines, line);
        counter++;

}
```

As you can see, we build up line-by-line the text that we want to go in the analogue output.

The `snprintf` function is used to insert bits of text into a string. It's based on a format string which can include format specifiers where we want to put in a bit of data. Previously we used `%s` to insert a string. Here, we're using `%d` for an integer and `%f` for a floating-point number. The full parameters are (output_variable, format string, data, …).

We use the `strcat` function to add one string to the end of another, so here `gpio_analogue_lines` is built up from all the bits of text joined together. We won't go through all the code here, as it is fairly similar, but the output GPIOs are:

```
#define NUM_OUT_GPIOS 3
int gpios_out[] = {16,17,18};

…

char gpio_out_lines[500] = "";
char line[64];
int counter = 1;
for(int i=0;i<NUM_OUT_GPIOS;i++){ // get current
GPIO values NOTE -- WORK OUT WHICH ONES are
causing a problem
        if (gpio_get(gpios_out[i])) {
                snprintf(line, 64, "Output %d
is on, turn <a href='/?GPIO=%d&state=0'>off</
a><br>", counter, gpios_out[i]);
        }
        else {
                snprintf(line, 64,"Output %d is
off, turn <a href='/?GPIO=%d&state=1'>on</
a><br>", counter, gpios_out[i]);
        }
        counter++;

        strcat(gpio_out_lines, line);
}
```

Here, you can see how we add an **<a>** tag to the line to create a link with the parameter to toggle the GPIO depending on whether it's currently on or off. All these bits are then brought together with:

```
char html_string[2000] = "";
```

```
snprintf(html_string, 2000, HTML_SOURCE, update,
gpio_analogue_lines, gpio_in_lines, gpio_relay_
lines, gpio_out_lines);
```

You can easily tweak these lines to output whatever HTML you like, but you need to be aware of a couple of things. Firstly, strings in C have a fixed maximum length. This is what's in the square brackets at where the variable is defined. Strings are really just arrays of individual characters.

Secondly, there's also a maximum length for the output buffer. This is the **`result`** element here:

```
typedef struct TCP_CONNECT_STATE_T_ {
    struct tcp_pcb *pcb;
    int sent_len;
    char headers[128];
    char result[2048];
    int header_len;
    int result_len;
    ip_addr_t *gw;
} TCP_CONNECT_STATE_T;
```

NETWORKING

By default, the program will create its own wireless network. The code that does this is:

```
#ifdef WIFI_SSID
    printf("Connecting to WiFi...\n");
         cyw43_arch_enable_sta_mode();
    if (cyw43_arch_wifi_connect_timeout_ms(WIFI_
SSID, WIFI_PASSWORD, CYW43_AUTH_WPA2_AES_PSK,
30000)) {
        printf("failed to connect.\n");
        return 1;
    } else {
        printf("Connected.\n");
    }

         printf("Starting server at %s \n",
ip4addr_ntoa(netif_ip4_addr(netif_list)));

#else
    const char *ap_name = "picow_test";
    const char *password = "password";

    cyw43_arch_enable_ap_mode(ap_name, password,
CYW43_AUTH_WPA2_AES_PSK);

    ip4_addr_t mask;
    IP4_ADDR(&state->gw, 192, 168, 4, 1);
    IP4_ADDR(&mask, 255, 255, 255, 0);

    // Start the dhcp server
    dhcp_server_t dhcp_server;
    dhcp_server_init(&dhcp_server, &state->gw, &mask);

    // Start the dns server
    dns_server_t dns_server;
    dns_server_init(&dns_server, &state->gw);
#endif
```

As you can see, this is wrapped in a preprocessor **`if`** statement. For people not familiar with C, the preprocessor is a text formatter that runs before the main compiler. What this says is that if we define **`WIFI_SSID`**, then include the code up until the **`#else`**; if we don't, include the latter part.

The define of **`WIFI_SSID`** is done in two stages. First, we can send definitions to CMake when we create our build files with a line like:

```
cmake .. -G"NMake Makefiles" -DPICO_BOARD=PICO_W
-DWIFI_SSID=yourssid -DWIFI_PASSWORD=yourpassword
```

We can then get these defines to flow through into the C preprocessor with the following in the CMake file:

```
if ("${WIFI_SSID}" STREQUAL "")
       message("Wifi SSID not defined, so will work
in AP mode")
else()
         message("Wifi SSID defined, so will work in
station mode")
         target_compile_
definitions(automation2040w_takeover PRIVATE
        WIFI_SSID=\"${WIFI_SSID}\"
        WIFI_PASSWORD=\"${WIFI_PASSWORD}\"
        )
endif()
```

All this means that you can select your network to connect to when you run CMake, but if you don't send any details there, it will create its own access point. If you connect to an existing network, your IP address won't be **192.168.1.4** – instead, it'll be assigned by your network. If you connect to the USB serial connection, you'll see the IP; alternatively, if you can access your router, you should see the device listed there with its IP address.

Automation 2040 W and Pico W are great options for adding connectivity to non-connected devices. Here, we've shown one way to easily let you control them from your laptop or mobile phone, but there are loads of other options. □

Below Flip the Automation 2040 W over for details of how to connect it

Flat-pack rockets

An aviation device fit for a Swedish furniture superstore

Jo Hinchliffe

@concreted0g

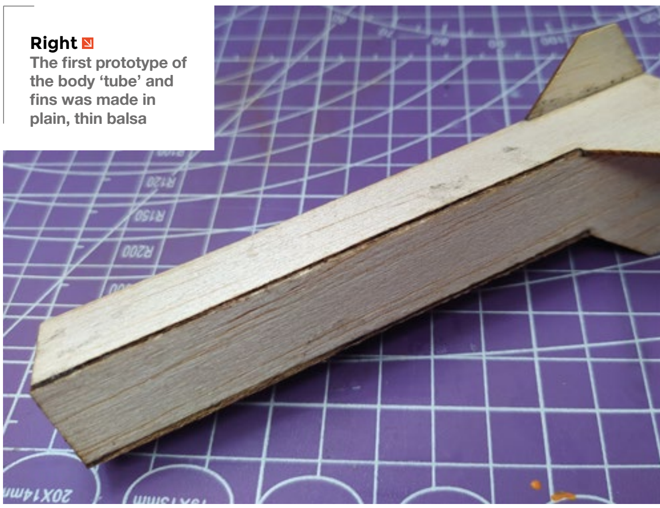
Jo Hinchliffe is a constant tinkerer and is passionate about all things DIY space. He loves designing and scratch-building both model and high-power rockets, and releases the designs and components as open-source. He also has a shed full of lathes and milling machines and CNC kit!

Above Take-off of the EXO-B, the larger of the two flat-pack rocket prototypes

Working on the EXO-S swing-wing rocket glider project, which was featured in issue 56, saw me working with balsa wood, and I started thinking more about building rockets with sheet materials rather than tubes. Mainly an interesting challenge for fun and learning, but it struck me that there are advantages to using sheet material, as well as a fair number of problems to solve!

Lightweight tubes are delicate in one axis – they stand up well in compression along the length of them, so resist the general flight forces well. But, if you take a model rocket and give it a squeeze, it's incredibly easy to crush them, buckling the body tube quickly beyond repair. The buckling issue actually comes up a lot of the time when mail-ordering kit. After their journey through a postal system, they often end up a little damaged. It struck me early on in this project that a 'flat-pack' rocket kit from sheet materials could be easy and affordable to mail to people, with a good chance of survival.

I set about making sketches. My first thought was to make a long, thin four-sided pyramid, which I still think is an elegant idea, but it is a challenge to make parts of a pyramid separate, like a nose cone. I quickly came to the idea of a square-sided body, and it struck me that it seemed silly to cut and place fins on each face of the body when I could simply extend the fin out of each side of the body. This creates offset fins, but this should still create a balanced airframe in flight. The other area I considered was the nose cone. I went through numerous sketched ideas looking for the simplest solution. I considered a tapered,



Right The first prototype of the body 'tube' and fins was made in plain, thin balsa

Left Both the large and small rockets use internal bulkheads to hold the motor in position

flat-sided cone first but, again for simplicity, I moved towards the idea of a nose cone that was curved in one axis and flat-sided. As I was building the prototype, I was thinking I would make the entire nose cone hollow and bend a thin wall to form the curved side. However, I ended up going with an even simpler idea. Other early decisions were to make a really small prototype, essentially to see how the panels would align and generally work, and then to make a larger rocket, about the average size of a beginner's model rocket. I'm a sucker for a tiny rocket though, and decided that the small prototype would have to be a flying model!

The other challenge with creating rockets away from standard tube components is how to create a successful design that will fly well. If you are designing a more standard rocket, you can use excellent open-source tools like OpenRocket, but there aren't tools available for more unconventional designs such as these. I've scratch-built a fair few rockets over the years, so a lot of my decisions were based on rules of thumb and previous experience. I did use a few older trusted techniques from the rocketry community to make some estimations and help inform the design. I used Inkscape to create the design files, and I used either my budget CNC diode laser rig (reviewed in issue 47) and a CO_2 laser I have access to at a local makerspace to cut the panels. However, don't let a lack of laser put you off. You could print out your designs, cut out card templates, and then use these to mark out and cut your panels.

> **I'd decided to use balsa sheet in various** thicknesses for the majority of the airframe

I'd decided to use balsa sheet in various thicknesses for the majority of the airframe as it's widely available, affordable, and lightweight.

Internally, I needed to have some bulkheads that would act as a motor mount. Again, in more standard designs, tubes are sold in a variety of materials which match the outer diameter of standard motors, but I wasn't allowing myself to stray from the flat-pack mantra! I went with a tubeless design that used a series of bulkheads that allowed a motor to be slid into place. These were cut from 1.5 mm plywood as they needed to be strong enough to resist the thrust forces from the burning motor. The bulkhead at the base of the rocket has small bolts glued into position so a further external bulkhead can be bolted on to hold the motor in. Inside the rocket, the upper bulkhead stops the →

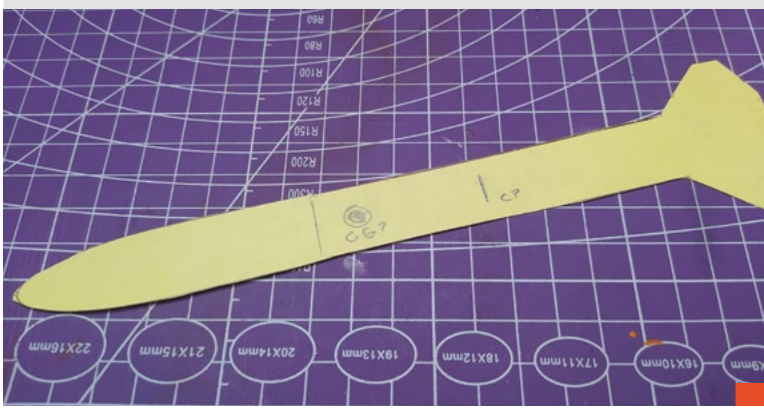
QUICK TIP

The very first piece that I contributed to HackSpace magazine was a walkthrough tutorial on how to design and simulate a model rocket using OpenRocket, back in issue 12.

Right Adding laminating film reinforces the lightweight balsa and adds a glossy low-drag finish. Sublimation printing adds fabulous decorative finishes

PRESSURE PUSHING DOWN ON ME

If you are designing a rocket using software design and simulation tools such as OpenRocket, you will probably know about the 'centre of pressure'. The centre of pressure is a point on the rocket about which all the forces on the rocket centralise. Therefore, the centre of pressure is most affected by the dimensions and sizes of the rocket airframe components, the diameter of the tube, the length of the nose cone, the size of the fins, and more. With our odd-shaped square rocket, that is trickier to simulate. We can use an older technique that gives us an estimation of where the centre of pressure is. The technique is to cut out a cross-section representation of the rocket design, and cut it out of cardstock. For this project, that's pretty easy to achieve from the drawn designs. With our cardboard cut out, we then find the point at which the cut-out balances – this point is the estimation of our centre of pressure. On the real rocket, the centre of pressure should be ideally somewhere between one and two calibres behind the centre of gravity. A calibre, in this instance, is the diameter of the body tube. Of course, our tube is actually a square, so we need to make an estimation there as well!



motor being pushed into the rocket during flight, but allows the ejection charge that pushes off the nose cone to pass through.

With my first few parts cut for the lower section, I set about assembling the first prototype. Gluing the first two sections with fins at 90 degrees to each other is probably the hardest part, and I used some square-sided blocks to help me keep things true. Once a couple of the internal bulkheads are positioned and checked for squareness, it all gets a little easier when adding the third and fourth panels. For this smaller design destined to fly using a tiny 13mm diameter Estes motor, I used balsa sheeting that was 1.3mm in thickness. I find sheet balsa a little variable in terms of stated thickness, so I went through my pile and graded sheets by taking numerous measurements with a set of callipers. Once assembled, I was pleased with the prototype in terms of lightness and I was also pleased with the rigidity of the square section around the bulkheads. But, I was very concerned with the strength of the fins which were very flimsy; also, away from the bulkheads, it was getting a little too flexible.

A LAYERED APPROACH

A popular approach for increasing the strength of balsa is to create a composite structure by adding a layer of another material. At this tiny scale, that might be gluing tissue paper, or thin card, or even super-fine fibreglass. However, I had an idea I wanted to try! I'd seen in RC plane-building communities that people use laminating film, which is essentially the same as office laminating pouches, to cover foam models as a reinforcement layer. This also collided with me researching craft communities using laminate films as a recipient for sublimation printing – meaning you can heat-press really nice artwork onto laminate-covered objects.

Some research ensued as I wanted to find a brand of laminating film that would be laser-safe for laser cutting. Some laminating pouches may contain vinyls which release chlorine gas when burnt and, as such, are very hazardous for human health when laser-cut, but also can cause corrosive damage to machines and extraction systems. However, there are also many laminating films that exist that are made from HDPE and PET layers, which are both considered laser-safe. Finding laminating film products that have their material contents listed, or in a datasheet, is pretty uncommon though, so it might take a while!

I pressed some laminating film onto a piece of the thin balsa using a modelling iron set to around 140°C, and pressed each section for around ten seconds.

Figure 1 For both the smaller and the larger rocket, I went with a stack of balsa profiles glued together to create the nose

Left The EXO-B micro prototype on the launch pad, ready for its first flight

Initial results were great, in terms of increasing the resilience of the material without adding much weight. There are a few challenges, however: as the laminate film cools, it can warp the balsa material. For the small prototype, I laminated both sides of the very thin balsa, but later, for the larger rocket, I laminated only a single side. Either approach could create a little warping. After some experimentation, I found that a few approaches helped. First, heating/ironing the balsa on both sides before adding the laminating film helped to remove any moisture, and then adding the laminating film seemed to make it a little more stable. If warping did occur still, I found covering the part in greaseproof paper, heating it for another ten seconds, and then placing a heavy, flat weight on it and allowing it all to cool slowly for five minutes helped get a stable flat part. I was already extremely pleased with the structural qualities of the laminate/balsa composite, and felt the fins would definitely survive the violence of a launch, so I set about experimenting with adding graphics by thermal-transferring sublimation print designs to them. Massive thanks to the Ffiws Maker Space in Porthmadog, North Wales, for allowing me to use the sublimation printer for these experiments. I printed out some colourful patterns using an online digital camouflage generator to create the images, and then had a few attempts at transferring the print using a range of heat and pressure settings. As the laminating film is not particularly designed for this, I was using guesses at first and then adjusting. I found that a pretty high amount of pressure, and a temperature of 130 °C for around 20 seconds, was pretty good in terms of transferring the design. You would probably get more depth of colour if you pressed for longer, but you run the risk of creating heat bubbles in the balsa/laminating film material.

> " **I printed out some colourful patterns using an online** digital camouflage generator to create the images "

Alongside working on the laminating and sublimation parts of the project, I had preliminarily designed a nose cone idea. I guessed that the overall design would need some weight at the nose cone end to create a stable airframe, so decided to make the top half of the nose cone out of a stack of balsa sheets glued together (**Figure 1**). I also wanted to include a payload section, an accessible area into which I could add an altimeter to tell me how high it flew. Below the stack of nose cone parts, the rest of the nose cone is closed with a sheet of thin balsa glued on one side and then also has a hinged side made with a piece of tape →

Above Cut sections of the larger EXO-B model in the laser cutter

Below right An altimeter and logging device packed into the larger EXO-B payload bay

YE OLD SIMULATOR

One of the difficulties with odd-shaped rockets is that you can't always use existing software simulation packages to check out the stability of the design. However, thousands of stable model rockets were designed, made, and flown way before free simulation software existed. One technique which is well worth learning and practising is the swing test. To perform a swing test, you need to attach a piece of string to the rocket at its centre of gravity. It's important that you do this with the rocket fully loaded with a motor and recovery system and any payload, so it's as it will be at a real launch. Find the point at which the rocket balances level and attach the string here. I tend to tie a loose loop around the rocket and then hang it and adjust until the rocket is balanced, and then add some tape to keep the loop on the centre of gravity. Gently begin to swing the rocket around in a big circle. I start off with about a metre of string and slowly let some string out so it ends up flying in a circle around 1.5 to 2 metres away from me. If the rocket is stable, the nose of the rocket should point in the direction of flight and it should fly level. Even if you start with the rocket the wrong way around or at an angle, it should correct into nose first, level flight. If it's unstable, the rocket may sit at an odd position or may oscillate and act chaotically. If unstable, adding nose weight usually helps. For swing tests, I often have some plasticine handy so I can temporarily add a little weight to experiment with.

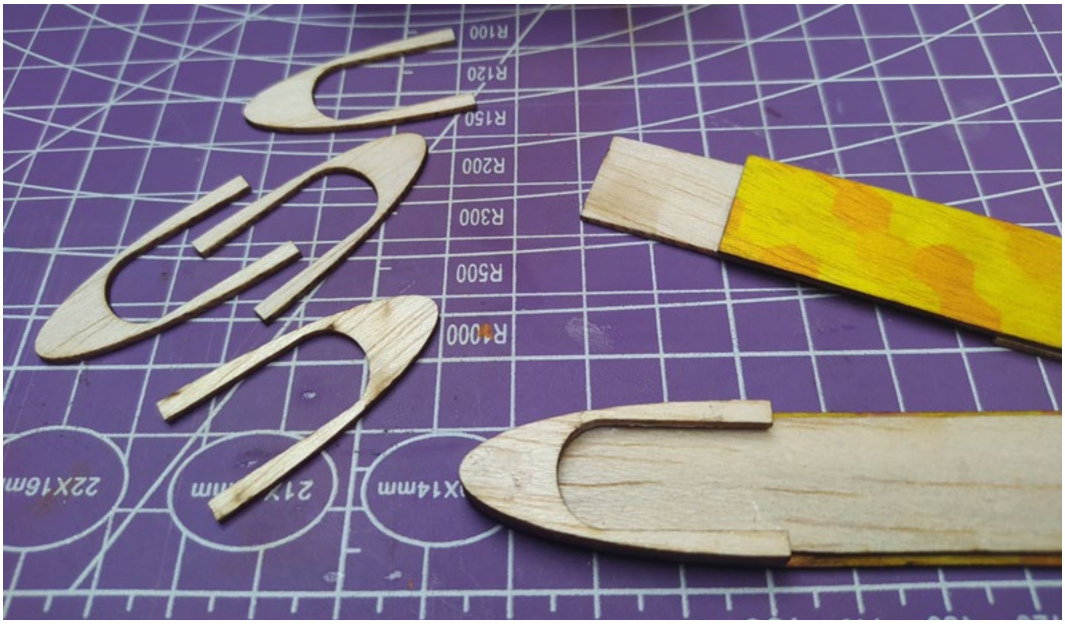
QUICK TIP

I hold the current UK altitude record flown on the same motor, which was 703 ft (214 m). With the EXO-B micro getting two-thirds of this performance in a very under-optimised design, that's an excellent result.

and closed with a tiny M2 bolt and a nut epoxy glued into a small balsa block inside the payload bay. The lower section of the nose cone, or 'shoulder' that fits inside the lower section, has some small plywood bulkheads added, to which the recovery system shock cord is attached. This keeps the rocket parts and the parachute or streamer all connected, so you hopefully get it all back. For the prototype, I simply cut the nose cone parts from non-laminated balsa to work out how it would all go together and wasn't worried about appearances. But it gave me a complete rocket that I could test and then go on to launch!

With the nose weight added and the tiny PerfectFlite FireFly altimeter in the payload bay, it was time to set out and test-fly the EXO-B micro prototype. The motor of choice was a tiny 13 mm Estes A3-4T motor. On a nice calm day, we set up the launch equipment and pressed the button. Tiny rockets like this disappear very quickly off the pad and this one didn't disappoint. We managed to follow it up visually and it deployed nicely, the two halves of the rocket separating well. The small Mylar streamer we had included, which creates drag but is more packable than a parachute, unfurled and helped us track the rocket back down to the ground. Recovering the EXO-B micro, it was completely undamaged and, on examining the altimeter which logs the peak altitude and the maximum velocity, it showed very promising results! The peak altitude was 498 ft (151.79 m), and the maximum speed was 141 mph (227 kph).

Moving to the larger EXO-B version, which flies using 18 mm diameter motors, it was largely a case of scaling up the design. I sublimated the digital camo pattern onto the balsa as I thought it looked cool, but I really should have gone with a brighter, more visible colour scheme! There's only a couple of differences in the construction of the larger EXO-B. A small obvious part is that I used only a single side of laminate onto the thicker and sturdier 3 mm balsa sheets. I also laminated and sublimation-printed the larger sheets of



balsa and then cut all the parts out of the sheet, rather than sublimation-printing directly onto the cut parts. The payload bay was made in a similar fashion to the EXO-B micro, but I used a bolt at each end to secure the cover. I installed some larger altimeters, as this rocket required the extra weight and I had a larger area to use, but I ran the battery connector cables out into the area below the payload bay to allow me to connect the power on the launch pad and not have to screw and unscrew the cover. Again, this larger airframe needed more weight in the nose cone, so I drilled a 10 mm hole into the centre of the tip and superglued a similar-sized ball bearing into it. As I had a little spare of the laminated printed material, I cut a matching 10 mm disc to patch up the hole on the outside.

BLOCKED-UP NOSE

One revision for both sizes is to change the stacked layer of balsa pieces that form the nose cone. I want to expand the payload area forwards into the nose cone, meaning weight can be pushed forward. This should help to create the stability the rockets need, but without adding as much ballast, meaning that the weight placed for stability is useful weight, like an altimeter.

The larger EXO-B version's first flight was on a slightly breezy day and it turned into the wind a little, which tells me it's a little over-stable. On an Estes B6-4 motor, which is a go-to motor for first flights, it achieved 366 ft (112 m). It's large enough to contain a parachute rather than a streamer, and I used a bright red one to try and help find the camouflaged rocket after landing! It's on the edge of being a little underpowered, and I'll probably test it again with a slightly more powerful motor. I also think that I could build another this size, but set it up for 24 mm motors which would increase the power availability a lot!

So, I've learnt loads and definitely proven that we can make decent rockets from flat materials. Although they should definitely be considered works in progress, I've published the part designs on this repository, should anyone like to build their own – **hsmag.cc/FlatSheetRockets**. For me, I'm already working on an improved version of the 13 mm micro one, which has the hollow nose cone idea implemented. I'm also considering if I could build a 13 mm motor one that is so lightweight it might get close to my current UK altitude record. Watch the skies for updates!

Above
For the next build of the EXO-B micro, I've adapted the nose cone stack to create more room to position nose weight and payload

Below
Building another EXO-B micro in a pleasing hexagon-patterned livery

Tin can plant monitor

Build a plant monitor with servo-driven indicators instead of LCDs, LEDs, or TFTs

Dr Andrew Lewis

Dr Andrew Lewis is a specialist fabricator and maker, and is the owner of the Andrew Lewis Workshop.

It's easy to fall into the maker trap, where your projects are driven by the availability of tools and parts rather than the desire to create something that serves your particular aesthetic. It might be more complicated, but choosing the path less travelled is a great way to build your skills and think outside the project box. In this article, you'll see how you can provide feedback about soil moisture and temperature with servos rather than the now ubiquitous LCD or TFT displays. You'll also get to play with recycled junk and build a model to house your project.

When anyone mentions a servo motor, makers tend to think of remote-control aeroplanes, walking robots, or animatronic toys. Hardly anyone thinks of car dashboards or clocks, but not every servo is a big, powerful motor, and chances are that the gauges in your car are actually servo-controlled. Light duty 9g servos (so called because their weight is 9 grams) are great for making dials or indicators because they're small and light, but they can still react quickly while moving a reasonable amount of weight. You'll use two 9g motors in this project to create different types of gauge, and you'll also learn a few basic model-making tips to keep things interesting. You will build a simple plant monitor to report on the moisture level of the soil, and get a general idea of the ambient temperature by reading the thermistor on Raspberry Pi Pico. The code for this project is available at hsmag.cc/issue60.

PAINT HIDES MANY SINS

Before you start creating dials and gauges, you're going to need somewhere to put them. You could just use a simple plank of wood, or you could get

creative and start sticking some bits and pieces together to make a strange contraption. In this article, the base object is a steampunk-esque model engine created with a sardine tin, a small tomato puree tin, a grinding bit box, and some air fittings from a compressor and a fish tank pump. Glue the pieces together to create a shape you like, and then apply a coat of undercoat to bring the pieces together visually. This sounds like a strange step, but it's very difficult to get an eye on the real shape of a model until all of the pieces share a base colour. Once that's done, it's easy to see which bits need more detail and which bits need to be removed or changed. You probably aren't working on a deadline for this project, so you have time to play around. Remember that you'll be adding some dials and electronics to your base, so don't paint everything yet, because you'll probably want to drill, glue, or otherwise modify things as you add the mechanisms.

If you're feeling particularly crafty, you can make a stand to hold your plant monitor and plant pot. Glue a plant saucer to a board, and mark out an area large enough for your plant monitor to stand. You can use air-drying clay to build up the base and create walls, stairs, or flagstones. Once the clay is dry, you can paint it and apply lichen and other dressings to make the landscape look more realistic.

The first type of dial mechanism you will create is a linear indicator. A linear indicator takes the rotary motion of the servo and converts it into linear motion by using a simple rack and pinion gear system. If you're not familiar with the term, a rack and pinion gear uses conventional gear and an 'unrolled' gear to create linear motion. As the conventional gear turns around, the linear rack gets pulled along in a straight line. You can 3D-print, laser-cut, or handmake your rack and pinion gears, or just steal some from your LEGO Technic™ kit. You'll be using the linear motion created by the rack and pinion to indicate the ambient temperature. Since a 9g servo typically rotates through 180 degrees, you will want to use a rack with slightly more than half of the number of teeth that the driving pinion has. You'll need to enclose the rack into a channel so that it doesn't →

MICROPYTHON

This project uses a Raspberry Pi Pico running MicroPython to control servos and read data from sensors. If you haven't installed MicroPython before, you can find instructions here: **hsmag.cc/WhatIsMicroPython.**

Above Adding a layer of undercoat to a project makes it much easier to get a feel for shapes and scale. For supposedly metallic items, a base coat of black followed by a chrome or copper coat makes a fair base to start adding more detailed block colours, then washes of thin colours (washed on then partially wiped back off again with a rag) to accentuate certain features. An almost dry brush loaded with a metallic colour can be used on edges to add the appearance of long-term wear, while semi-transparent gloss coats can be used to give the impression of oil or grease patches

Above
You can create a small model feature to hide the moisture sensor in the plant pot. Slices of a tree branch make a nice flat surface for a rustic model table loaded with small screws and other objects, or you could create something entirely different instead

Right
Air-drying clay shrinks as it hardens. That means cracking is a common problem. You can simply fill any cracks with more clay, or leave them alone and make them a feature of your landscape instead

Left Agricultural grit mixed with PVA glue makes an excellent gravel effect. If you don't like the slightly slick and glossy look of the dry PVA, sprinkle a layer of unglued grit on top of the wet grit/PVA mixture. The top layer will still stick in place, but there won't be any glue on the top of the stones

STAY STUCK

This is the sort of project where glue plays a major role, so it's worth taking a quick look at the different types of glue that you might encounter. For model work, there are two types of glue that you'll use, and these are high viscosity and general-purpose superglues. High viscosity superglue is thick, and less likely to run if applied to a vertical surface. High viscosity superglue usually starts to bond in about 15 seconds. It's great for filling gaps and making big, bold joints along edges. General-purpose superglue is its thinner sibling, which sets more quickly (in as little as five seconds, depending on the temperature and humidity of the room). Thinner glue is great when you can't quite reach the target that you need to apply glue to, and need to let it drip into place. It's also a good way to hold strings or knots into place. Very low viscosity superglue is available for use in special cases, but it is very difficult to apply accurately, and in inexperienced hands, it will cause absolute chaos and leave you looking like you're trying to do a live-action role-play of Katamari Damacy. Waterproof PVA glue is very useful in model-making projects to keep things like fine grit or gravel in place. Simply stir the PVA and the stones together, and then lay them into place on your model. When the glue sets, the stones should be an immovable feature of the landscape. You can add acrylic tints to PVA to create the effect of muddy ground or shallow water.

come out of alignment with the pinion. You can easily create a channel by covering your rack with a couple of layers of masking tape to increase its thickness slightly, then gluing a channel together out of lollipop sticks or balsa wood. Remember, if you're painting the channel, the paint will add thickness and make the inside of the channel smaller, so it's best to be generous with the masking tape and use thin paints.

> "You'll need to move the servo to figure out the maximum and minimum limits **that it can move to within the constraints of the model you've built**"

When you're first setting up a servo with a gear system, you should make sure that you know the position that the servo is in by powering it up and moving it to a sensible known position. Once the gears are attached, you'll need to move the servo to figure out the maximum and minimum limits that it can move to within the constraints of the model you've built. If possible, keep gears loosely fitted to the servo so that they're free to slip if excessive force is applied. It's better to slip gears on a shaft →

QUICK TIP

Bicarbonate of soda makes superglue set pretty much instantly, and it's also great when used with superglue for filling larger gaps in a model.

Above Copper is a soft metal, so a copper linkage is likely to bend rather than transmit enough force from the servo to damage something important. It's also useful that you can add a little ball of solder to the end of a copper rod to keep it from slipping out of its mounting hole

PULSE WIDTH

Most servos expect to receive a PWM waveform on their signal wire (which is usually yellow, white, or orange), with the remaining two wires being used to power the servo. The Raspberry Pi Pico can generate PWM signals on most of its outputs, so connecting a servo is often as simple as picking a digital pin and plugging it in. The PWM signal used to control the servo is a typical square wave with a variable duty cycle and (for 9g servo motors) a frequency of 50Hz. The frequency is set in MicroPython using the easy-to-understand function 'machine.pwm.freq(50)'. The duty cycle is slightly more complicated to explain. In MicroPython, the duty cycle is set with the 'machine.duty_u16()' function. MicroPython uses a 16-bit resolution for PWM signals, so the range between 0% and 100% duty cycle is 0–65535 (note that in reality, the Pico hardware uses 12-bit resolution which is upscaled to 16-bit by MicroPython, so true 16-bit accuracy is not possible). To operate a 9g servo through its full 180-degree range using this scale, the duty cycle would range between 1000 and 9000, with 1000 representing 0 degrees and 9000 representing 180 degrees.

than break apart your nice new model by typing the wrong number into your code.

The second dial mechanism used in the plant monitor is a rotary gauge, and it is used to indicate the moisture level of the soil. However, since the 9g servo is a rotary servo anyway, it feels a little bit too easy to just stick an arrow onto the servo shaft and call it done. Instead, you can use metal linkages to control the bar of the moisture level scales remotely. If you're accurate enough, you can cut the linkage from a single piece of thick gauge wire. If you're not confident that you'll be able to get the measurements close enough, use two pieces of wire and join them with solder or a wire connector block once they're in place. Using linkage wires means that it's possible to transfer the motion from the servo to a remote location, effectively hiding the servo from view. In the plant monitor, the linkage wires pass from the servo on the inside of the sardine tin on the bottom of the model, through the puree tin, and through the air fixture at the top of the model. The scales pivot on a piece of wire glued to the side of the air fixture, and the linkage wire connects a few millimetres to the side of the pivot point. As with the first servo, the range of motion must be calibrated to make sure that the servo doesn't bend the linkages or pull something apart.

QUICK TIP

A stiff toothbrush or wire brush can be used to create a grass-like texture in clay. Simply stipple the brush into the surface to get the desired effect.

Left Choosing a meaningful icon to represent a particular concept is a difficult task, but words can be less effective when viewing something at a glance. The idea of dry twigs and green leaves hanging on the scale could be interpreted in two different ways, confusing a user who doesn't understand that the scales should be balanced

ACCURACY IS RELATIVE

The moisture sensing part of the plant monitor doesn't display an absolute value; it simply reports the sensor readings as one of five possible states: very dry, slightly dry, ideal, slightly wet, and very wet. The scales will tip or balance based on where in that list the reading falls.

> "The real point of this project **is to have fun and make something unexpected**"

When thinking about the technology used in this project, there isn't really much to see beyond a standard analogue moisture sensor, a Raspberry Pi Pico, and two PWM servos. The real point of this project is to have fun and make something unexpected out of the contents of your recycling bin. It's amazing how much fun you can have by sculpting, painting, and finishing a project like this. □

GETTING WET

The important thing about moisture sensors is that they generally don't directly measure moisture. The two most common types of moisture sensor actually measure either the resistance or capacitative response of contaminants in water. The value returned by these sensors is relative rather than absolute, and it varies depending on the composition of the soil, the root density of plants in the soil, the presence of stones or wood, and many other factors. This means that getting an accurate value from a sensor requires calibration for the particular soil and water combination that it's been placed in. To short-cut the full calibration process, the code for the plant monitor assumes that the moisture sensor value returned when the monitor is first powered on is the ideal moisture level. The scales on the device will indicate whether the soil is drier or wetter than when the sensor was first queried.

If you look at the code for the plant monitor, you might notice that power to the sensor isn't provided constantly. The sensor is powered via a GPIO pin that switches the sensor on before a reading is taken, and then turns it back off again when the reading has been taken. This project uses a capacitative sensor, and having the sensor powered isn't a real problem because capacitative sensors don't directly interact with the soil in the same way that a resistive sensor will. Capacitative sensors don't require a direct connection, so they are less prone to corrosion. Resistive sensors work by passing an electric current through the soil and measuring the resistive drop between the sensor probes. As you can imagine, passing a constant current (however small) through the probes will lead to very rapid corrosion. To mitigate the problem, it's generally considered good practice to power the sensor only when needed, and also to alternate the current when possible.

TUTORIAL

Pinhole cameras and camera obscuras

Cameras don't have to feature advanced electronic circuitry. In fact, they don't even need a lens

Mike Bedford

Despite loving all things digital, Mike admits to being a bit of a Luddite, vinyl records and all.

Back in 1816, French inventor Nicéphore Niépce captured an image on light-sensitive material. Images didn't last long – they blackened very quickly – but this was the birth of photography. Niépce's magic built on a much earlier development. Its origins are lost in the mists of time, but Leonardo da Vinci wrote about a device called a camera obscura and it became popular from the 17th century onwards. A camera obscura (from the Latin for dark chamber) is a darkened room with a lens in one wall looking out over the surrounding area, and a white screen on the opposite wall. It allowed observers in the room to see an image of the world – albeit an inverted one – projected onto the screen. Much smaller, hand-held or table-mounted versions allowed people with minimal artistic skill to trace the image onto tracing paper. But here's an interesting thing – you don't need a lens. Camera obscuras can have a tiny hole where the lens would otherwise be and, although the image is a lot dimmer, it still works. Here, we'll see how to make a camera obscura with just a pinhole

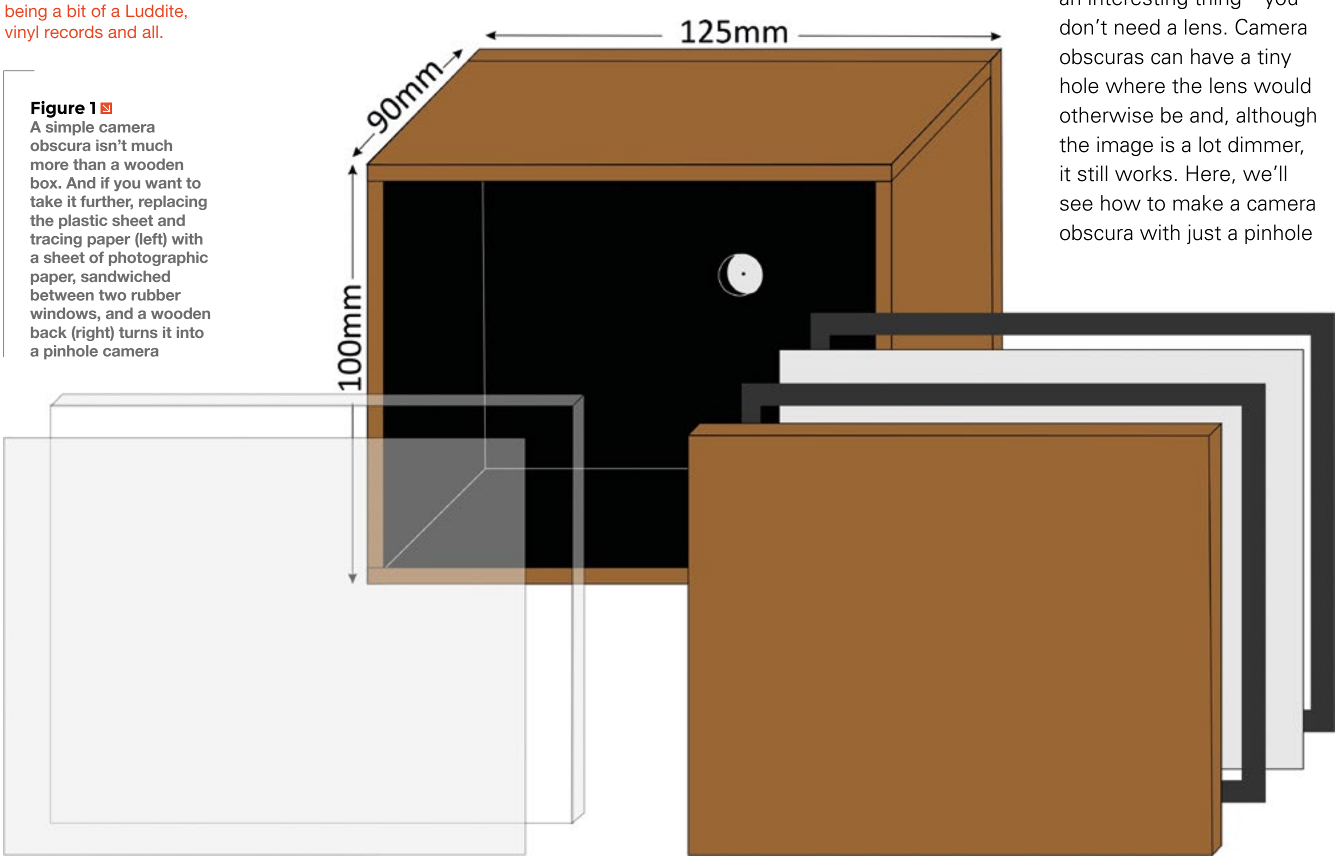
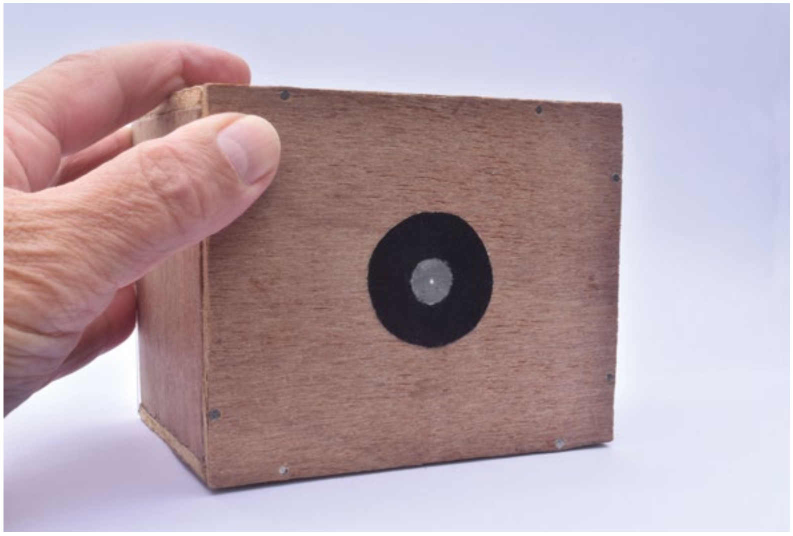


Figure 1
A simple camera obscura isn't much more than a wooden box. And if you want to take it further, replacing the plastic sheet and tracing paper (left) with a sheet of photographic paper, sandwiched between two rubber windows, and a wooden back (right) turns it into a pinhole camera



Above A simple camera obscura is a quick and easy build

and follow this up by looking at several ways of using a pinhole to capture a photo, rather than just seeing a fleeting image. So, if you thought a lens was essential to photography, think again.

OLD-SCHOOL CAMERA OBSCURA

Making a simple pinhole camera obscura isn't difficult – it's not much more than a wooden box with a translucent back; see **Figure 1**. Sizing isn't critical, but the dimensions in the drawing work well. Make it from 5 mm plywood, glued and pinned, and make sure the joining edges are perfectly flat so you get a good light-tight joint. If you're not too sure about your handiwork, run a bead of sealant

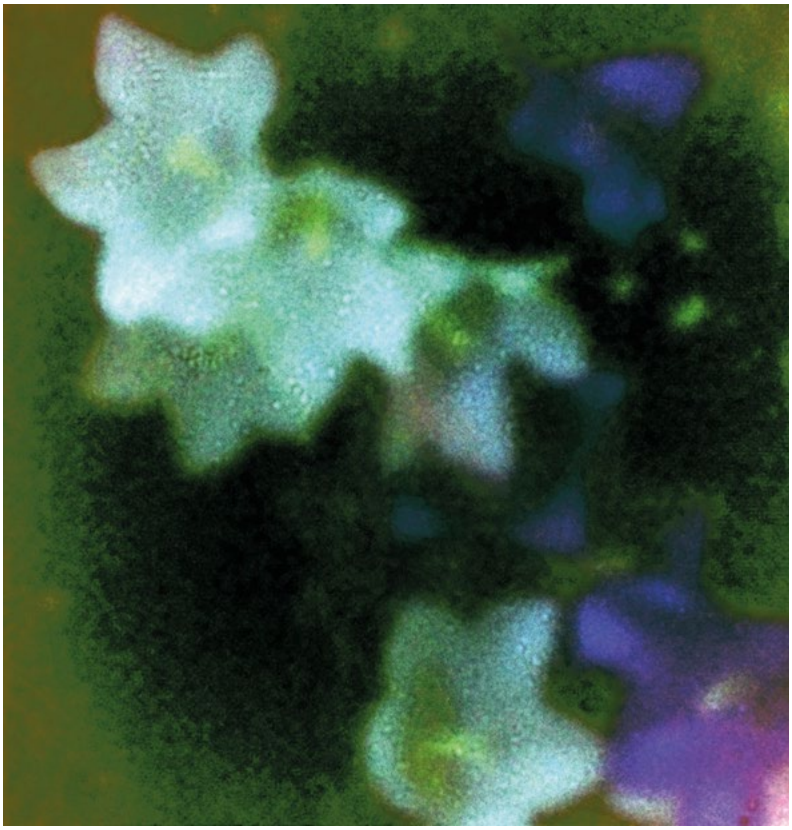
> **"Before using the camera obscura, tape some tracing** paper over the transparent screen"

along the inside of each joint. Now paint the inside with matt black paint. Make the back from thin (2 mm-ish) transparent plastic sheet, attaching it to the plywood with sealant. Now drill a hole, about 20 mm in diameter, in the centre of the face opposite the transparent screen, and attach a pinhole over that hole using adhesive tape – see 'Make a Pinhole' box overleaf.

Before using the camera obscura, tape some tracing paper over the transparent screen. If you use it to view a bright object, either in a darkened room or outside at night, you'll probably be able to see the image as soon as your eyes become dark acclimatised, but you'll notice it's upside down. In daylight, though, the image will be extremely dim, even on a bright day, so you'll probably need to tape some black cloth onto the camera and drape it over your head, just like photographers of old. Since the tracing paper screen is replaceable, you can draw the scene displayed on it by tracing over it with a pencil.

Instead of a pinhole camera obscura, you might just like to build this as a proper camera instead of an experimental project. The main difference is that, instead of the transparent window opposite the pinhole, you'll have a wooden rear panel on which →

Left Using a Raspberry Pi HQ Camera provides a good introduction to digital pinhole photography



you'd mount the film or, better still, so you can load it in red light rather than in total darkness, photo-sensitive paper. If you use the dimensions shown in the diagram, it'll take 4" × 5" (100 mm × 125 mm) film/paper. The paper will be held in place when you attach the back, but use four 90-degree spring-loaded toggle latches so it's easy to remove the back for swapping the paper. And to make sure it's light-tight, use 'windows' of thin rubber sheet either side of the paper, and seal along all the edges with black tape while you're still in the dark. You'll also need to devise some means of blocking the pinhole until you want to take a photo. And because exposure times will be long, use a tripod.

This camera doesn't have a viewfinder, so framing your image is a bit of a guessing game. You could add some sort of simple viewfinder – and there are several options – but an easy solution is just to draw a couple of angled lines on the top of the box to represent the angle of view which, if you use our dimensions, will be 65 degrees horizontally. With practice, these will give you some idea of what'll be in shot. Exposure times will be very long and, while you might need to experiment, 15 seconds would be a good starting point for paper on a bright day, increasing to 15 minutes on a dull, cloudy day, although film could be 15 times shorter. You'll have to develop the film or paper yourself using traditional

Above
Our first Raspberry Pi HQ Camera pinhole photo wasn't exactly sharp – pinhole photos rarely are – but it does have a strange sort of appeal

Right
Having experimented with a wooden box and the Raspberry Pi approach, if you have a DSLR, it'll provide the easiest solution to pinhole photography

darkroom techniques. But, if you use positive paper, you will get a proper positive image from the start, rather than having to then make a contact print onto ordinary (i.e. negative) photo-sensitive paper.

RASPBERRY PI HQ PINHOLE CAMERA

If using chemicals and a red light in a darkroom isn't your thing, we've got a better solution for pinhole photography. It captures the image digitally, using a Raspberry Pi and Camera Module. That Module must be an HQ Camera because the lens is removable. Don't fit any of its lenses but, instead, drill a hole about 10 mm in the screw-on cap that the camera was supplied with, tape a foil pinhole over the hole (see 'Make a Pinhole' box below), and screw it onto the camera. Exposure times will be long, so be sure to use a tripod. If you don't intend to do more than try it out by taking a few photos inside, or through the window, you really don't need to do much more, although we do recommend reducing the strain on the ribbon cable between the camera and the Raspberry Pi using either an off-the-shelf mounting plate, or a DIY one made from a sheet of plastic, four plastic spacers, four nuts and bolts, and four screws.

If you want to take your Raspberry Pi pinhole camera into the big wide world, you're going to need to make it more portable and durable, by building the HQ Camera and Raspberry Pi into a case, together with an LCD panel and a power source, and some suitable software. Projects like this abound, but mostly for the original or V2 Raspberry Pi Camera Modules, so you might have to do a bit of a redesign.

> "You could build it without a screen and devise some sort **of primitive viewfinder for the true pinhole experience**"

Alternatively, you could build it without a screen, and devise some sort of primitive viewfinder for the true pinhole experience.

DSLR PINHOLE PHOTOGRAPHY

The easiest way to creating pinhole photos digitally is to use a DSLR. Use the camera body's cap, drilling →

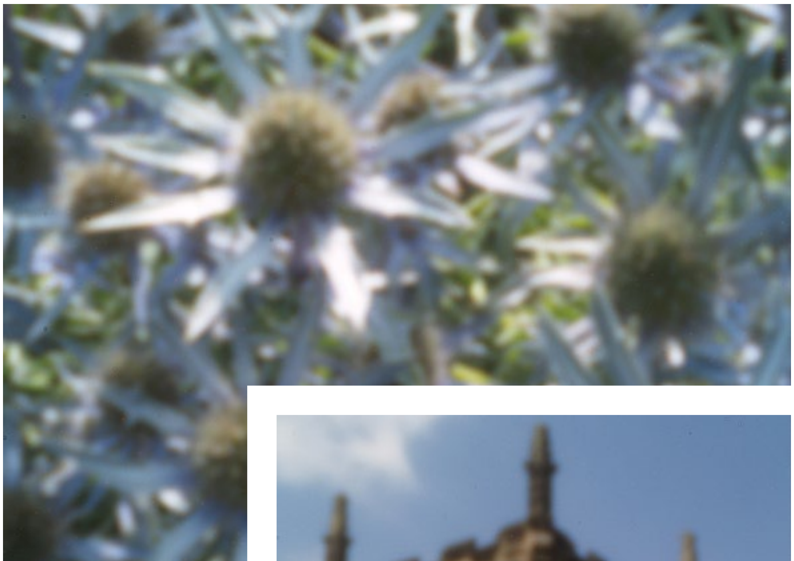
FOCAL LENGTH

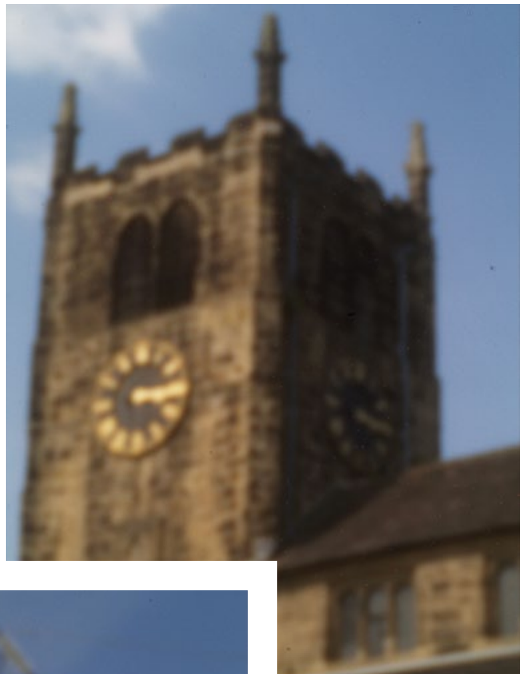
The focal length is a measure of the zoom setting of a lens – the longer the focal length, the more it's zoomed in. It normally depends on the characteristics of the lens, but with a pinhole camera, it's simply the distance between the pinhole and the sensor or film. The focal length alone doesn't define the amount of magnification, because that also depends on the size of the sensor or film – the smaller the sensor, the greater the magnification. For that reason, we often talk about a 35 mm equivalent focal length, because 35 mm film cameras were so popular for so long. In this scheme, 50 mm gives you an image that looks the size your eye sees a scene. A greater focal length is telephoto, and a smaller focal length is wide-angle.

MAKE A PINHOLE

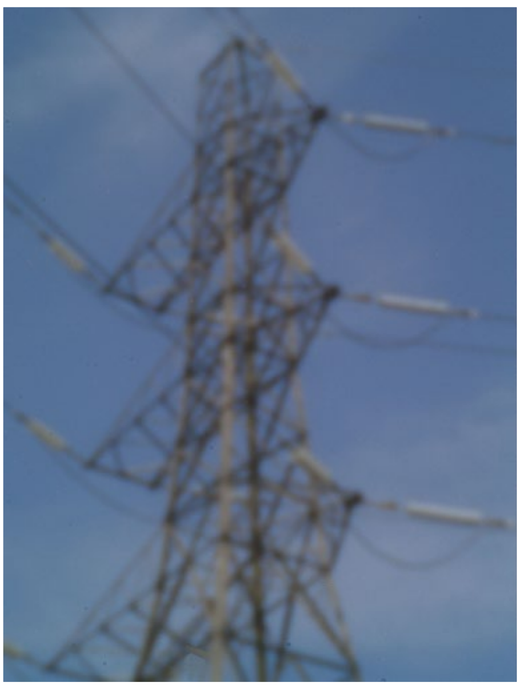
To make the pinhole, flatten a piece of heavy-duty aluminium foil of the type used in food containers, and use a needle to make a hole. Use a magnifying glass and fine sandpaper to get rid of any burs, because these will affect the image quality; you may need to use the needle again if sanding partially blocks the hole. Your image won't be as sharp as it could be – although pinhole photos are never super-sharp – if the hole is either too large or too small. As a guide, the diameter should be 0.037 multiplied by the square root of f, where f is the focal length (see box above). With a DSLR pinhole camera, this will be about 0.25 mm. For a camera obscura, as opposed to a camera, an ideally sized pinhole will often give you a very dim image. So use a bigger hole – perhaps 1 mm in diameter or more – sacrificing sharpness for brightness. It's not hard to make a new pinhole if you get it wrong, so do experiment.

Below Making a pinhole isn't high tech, but it'll be a trial and error process to get it just right





Right We'll leave you to figure out what subjects make the best pinhole photos, but here are the results of our endeavours, with a variety of scenes. They all have that characteristic blurry nature, so we don't anticipate abandoning lenses in the near future, but it's quite astonishing how such a low-tech solution can reap any results at all



a hole in it and taping a pinhole to it. The focal length (see 'Focal Length' box) will differ between cameras, but it will probably be about 40 mm, which, in the case of a consumer DSLR, will be slightly telephoto. If our cameras are typical, the viewfinder will be so dim you won't be able to preview the image. We have a solution, but you will have to do some experimenting.

With the camera on a tripod, take a test shot using the pinhole, and then capture the same scene using a proper lens at a variety of zoom settings. This way, you will be able to find out the focal length of your pinhole. Now, whenever you use your pinhole DSLR, first fit the lens, zoom it to the appropriate focal length, frame the shot and then, without moving the camera, fit the pinhole and then take the photo.

> **Some people actually like the pinhole effect for its** somewhat dreamy, blurry nature

Automatic exposure modes don't seem to work too well with a pinhole; also, with one of the cameras we tried, unless the camera was set to manual exposure, it complained there was no lens and refused to take a photo. So you're probably going to be using manual exposure, but that can be a bit hit and miss, and exposures are going to be long. We found that, with the camera set to a speed of ISO 100, exposure times were about 20 seconds even on a bright sunny day, so use a tripod.

More than most types of photography, pinhole photography is all about experimentation. So do just that and have fun. You won't get sharp photos, but some people actually like the pinhole effect for its somewhat dreamy, blurry nature. Even if you don't share that view, don't lose sight of the fact that you're using a lens made, quite literally, from nothing.

DEPTH **OF FIELD**

Ordinary cameras tend to have an aperture that can be varied between about f/2 and f/22, the larger number representing the smaller aperture. The smaller the aperture, the longer the exposure will be, but you end up with a better depth of field. If you're not familiar with that term, a small depth of field means that only objects at a narrow range of distances from the camera will be in focus; a large depth of field means that objects over a larger range of distances will all be in focus.

The aperture is calculated by dividing the diameter of the lens or pinhole by the focal length (see 'Make a Pinhole' box). Because pinholes are so small, the aperture will also be small – f/250 is typical. The upshot of this is that the depth of field will be huge. That's not all good, though. First up, saying that everything will be equally in focus isn't strictly true. Pinhole cameras are never sharp, so saying that everything is equally blurred would be more accurate. Next, if you want to shoot someone against a blurred background, you won't manage it with a pinhole camera. Another disadvantage is that if you have specks of dust on the image sensor, they will often be visible on your photos, whereas with an ordinary lens they'd be so out of focus that they'd be invisible. You might want to read up on methods of safely cleaning the sensor.

Below You can make a telephoto pinhole simply by moving it further from the film or sensor like this. Be warned, though: if you thought ordinary pinhole photos lacked sharpness, just wait until you try a telephoto pinhole. Oh, and exposure times will be even longer

HackSpace
TECHNOLOGY IN YOUR HANDS

FIELD TEST

HACK | MAKE | BUILD | CREATE

Hacker gear poked, prodded, taken apart, and investigated

PG 110

GAME BOY PRINTER

A WiFi network for your retro gamer

PG 112

BITTLE

Build your own robot pet

PG 102

BEST OF BREED

Fine-tune your finger setup

ONLY THE BEST

Keyboard mods

The best bits for your fingertips

By **Marc de Vinck** @devinck

Just about a year ago, we covered the topic of keys and keyboards. The topic was getting popular, as components like microcontrollers were getting harder to source. And to be quite honest, we thought some things, like the supply chain, would have gotten better a year later, but they haven't. Trying to source components for your next project is still difficult. Trying to find a microcontroller? In many instances, it can be nearly impossible to find, or at least find one at a reasonable price.

However, one small niche of the DIY electronics community seems to be thriving during these times: the custom keyboard scene. Some of the reasons might just be because it's a fun and interesting hobby. But we think there are other factors for its increased popularity. Mainly, the parts are just simply available and available at a very reasonable price. Yes, some of the prices have gone up, while others have stayed relatively the same. And that's not to say that all the parts needed to build your keyboard are easily sourced, but it does seem like you can always find a good source of key caps, switches, and PCBs. And with the introduction of the new RP2040 microcontroller from Raspberry Pi, which makes for a great keyboard controller, many people have taken on the challenge of building their own.

In this second look at customising keyboards, we'll cover a few new products, some things we wish we had covered last year, and some interesting components to make your next keyboard build really stand out. If you're going to DIY your own keyboard, you might as well add some cool sensors and inputs! And most importantly, everything in this roundup is currently available, and we hope will continue to be readily available in the future.

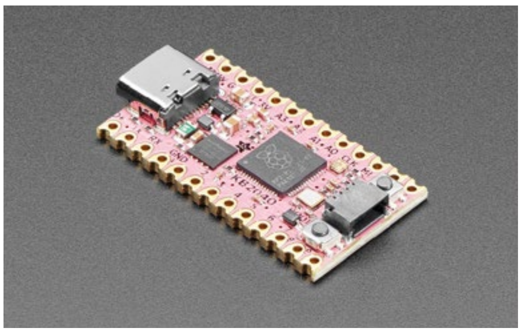
Adafruit KB2040 – RP2040 Kee Boar Driver vs PICOkeyboard

ADAFRUIT $8.95 | adafruit.com

BOBRICIUS $19.99 | tindie.com

The Adafruit KB2040 is a drop-in replacement for a typical Arduino Pro Micro, which happens to be a common part of custom keyboards and controllers. This board allows you to take advantage of its built-in dual-core 32-bit Arm Cortex M0+ processors and 264kB of RAM, and 8MB of SPI flash in your next build. With all that power, and the 20 GPIO pins, you can have up to 100-keys matrices. That makes for a great keyboard!

The board also has castellated edges for easily adding it as a daughter board to your PCB, or you can use the STEMMA QT connector and skip the soldering altogether! A few other nice features are the use of a USB-C connector, breakouts for alternative USB connectors if needed, and a colourful little RGB NeoPixel. Head to the Adafruit website to learn more about building your own keyboard with the RP2040 Kee Boar Driver.



Left More boards should come in pink

Below The tiniest custom keyboard?

The PICOkeyboard by Bobricius is an interesting little set of PCBs for making your own keyboard. Unfortunately, you can't get a complete set, so you'll need to bring your own buttons and other various components to complete the build. But, there are a few things that we really like about the two PCBs that are included. First, they are just cool-looking! We just like the look of gold and black together for a keyboard. Next, the creator uses a second PCB as a shield over the first PCB to hide all the solder joints and button housing. It makes for a beautiful-looking final build.

The PCB combo also allows them to easily plug into a Raspberry Pi Pico, and there is room for adding an RFM95 for long-range communications. There are also pads broken out for adding any standard 8-pin ST7789 IPS 320 × 240 display. Check out the product on Tindie to learn more, and all the extra components you will have to source yourself. It may take a little extra time, but it looks to be well worth the effort.

VERDICT

Adafruit KB2040 - RP2040 Kee Boar Driver

Another great micro for building a custom keyboard.

10/10

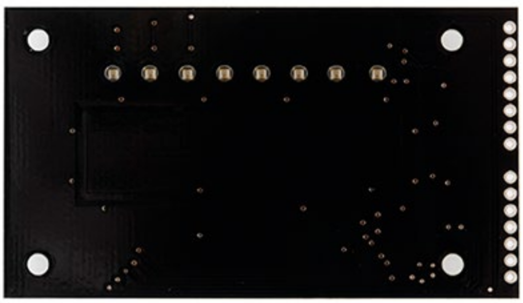
PICOkeyboard

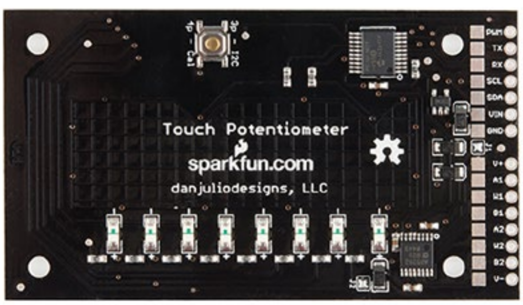
A clean-looking keyboard.

7/10

SparkFun Touch Potentiometer

SPARKFUN ◆ **$16.95** | **sparkfun.com**

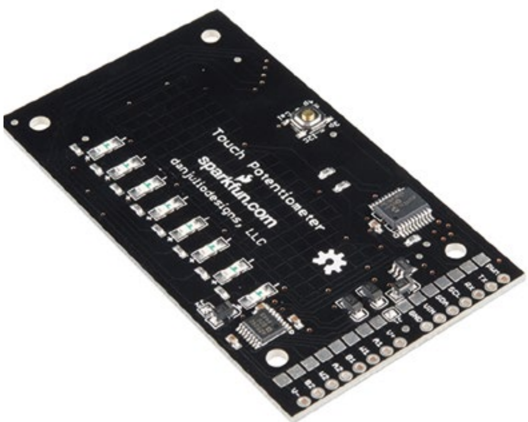




And while we're looking at cool sensors that you could add to your custom keyboard build, let's look at the SparkFun Touch Potentiometer. This prototyping board features an intelligent linear capacitive touch sensor, and it allows you to have 256 different positions. You can use this as a drop-in replacement for a standard standalone potentiometer as it emulates the wiper of a mechanical version, but it also has some more interesting features.

> "**We think adding tactile swiping on a DIY keyboard** would be an interesting and useful feature"

You can also use the PWM output from the board, or serial output, and connect it to any microcontroller or Raspberry Pi. We think adding tactile swiping on a DIY keyboard would be an interesting and useful feature. Check out SparkFun's website for more information and a great 'getting started with touch potentiometers' guide.



Left ◆
Create a touch interface easily

VERDICT

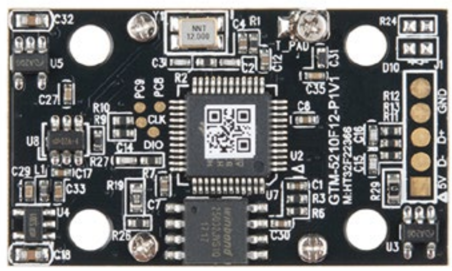
SparkFun Touch Potentiometer

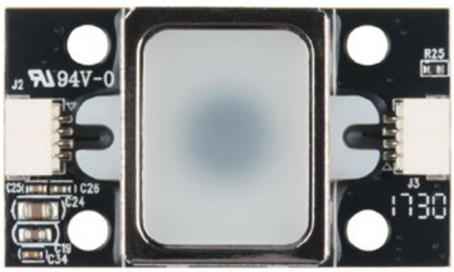
Prototype a unique input.

10/10

SparkFun Fingerprint Scanner

SPARKFUN $38.50 | sparkfun.com





Left
Lock down your machine

Sure, fingerprint scanners are common on commercial keyboards, but it's not very often that you see a DIY keyboard with a built-in one. It's most likely because that technology was difficult to implement, but not anymore! With the SparkFun Fingerprint Scanner, you can easily add an all-in-one scanner that can scan your fingerprints at 400 dpi, and with an error rate of less than 0.001%. It's also quick, reading your fingerprint in less than 1.5 seconds.

The sensor is small and has several mounting holes for easy installation. It also features a built-in 32-bit Arm Cortex M3 processor, so all the calculations are done on board, leaving your microcontroller free for other tasks. And if this version is too big, they make another one that is much smaller. Head on over to SparkFun's website to learn more about its range of fingerprint scanners.

VERDICT

Fingerprint Scanner

Another unique option for a custom keyboard.

9/10

KAILH SWITCH SOCKETS

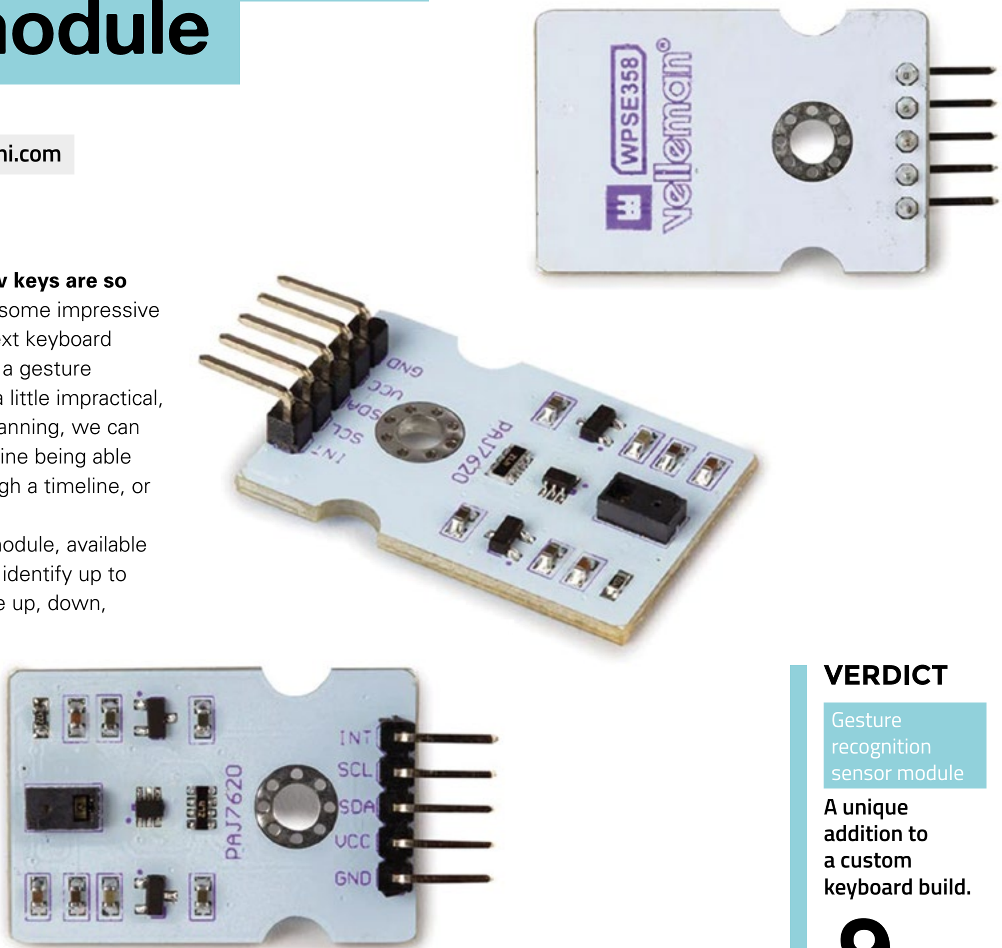
ADAFRUIT $4.95 (pack of 20) | adafruit.com

The Kailh Switch Sockets are for the true DIY keyboard enthusiast. With this little component and the proper footprint on your custom PCB, you can easily add a no-solder socket for your MX switches. Perfect for your next build where you'd like the option of building multiple boards with different types of switches. Just keep in mind that you will still need some kind of mechanical support for snapped-in switches. A little hot glue or epoxy works well. A pack of 20 for just under $5!

BEST OF BREED

Gesture recognition sensor module

PIMORONI ◆ $14.00 | pimoroni.com



Little joysticks and arrow keys are so yesterday! Want to add some impressive motion control to your next keyboard build? How about adding a gesture sensor? Yeah, it may be a little impractical, but if done with a little planning, we can see this being a great addition. Imagine being able to press a hot key, then swipe through a timeline, or adjust the volume.

The Gesture recognition sensor module, available from Pimoroni, allows you to simply identify up to nine predefined distinct gestures like up, down, forward, backward, and a few more. The module also features two types of inputs; normal and game mode, allowing you to adjust the speeds at which it senses your movements. It works best at a distance of 10 cm, and where the ambient light is less than 100 lux. It has built-in I2C, so connecting it to a Raspberry Pi should be straightforward.

VERDICT

Gesture recognition sensor module

A unique addition to a custom keyboard build.

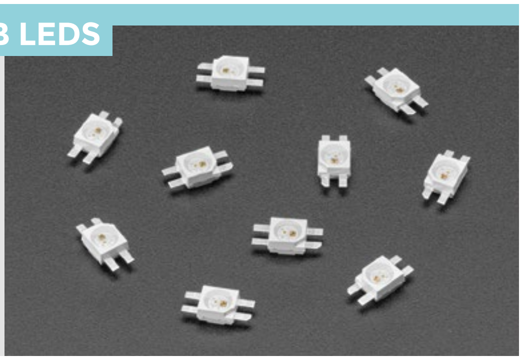
9/10

Right ↗
These are not the droids you are looking for

NEOPIXEL **REVERSE MOUNT RGB LEDS**

ADAFRUIT ◆ $2.95 | adafruit.com

And while you're designing that custom PCB for your mechanical keyboard, why not add a little light, or better yet some RGB light! With the NeoPixel Reverse Mount RGB LEDs, you can keep the top profile of your build as smooth as silk and mount the LED from the backside of the PCB. Just don't forget to add a cutout for the light to shine through!



Game Boy Camera Wi-Fi printer

Take retro-licious photos

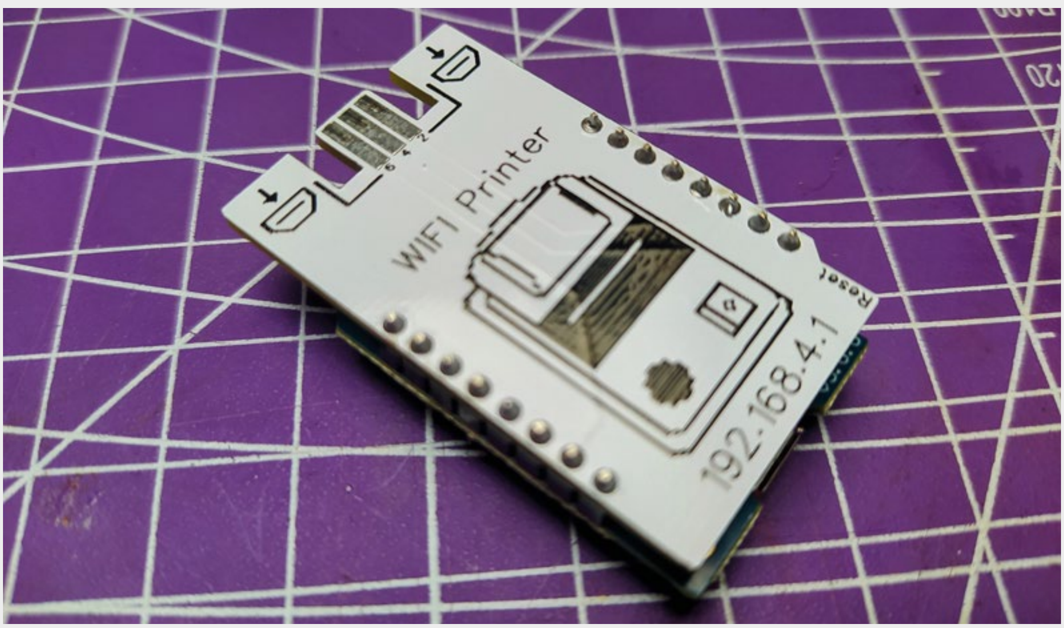
NILLTRONIC ◆ £26 | etsy.com

By **Jo Hinchliffe** @concreted0g

The Game Boy Camera (GBC) cartridge was a classic Game Boy add-on that allowed people to take photos and print lo-fi images. It also featured a random assortment of games, a chiptune classic application 'Trippy-H', and some of the most bonkers background music.

The add-on has remained popular and is very collectable. There was always the issue that, apart from buying the official Nintendo thermal printer accessory, you couldn't get your images off the GBC cartridge in a straightforward manner. We remember a long time ago writing a blog post about kludging a solution with an incredibly rare aftermarket cable, some software, and Windows XP, but it was clunky and prone to failure. So when this print-to-Wi-Fi solution crossed our path, we were intrigued.

Available on Etsy, this is essentially an ESP8266 riding on a custom PCB with a connector for a Nintendo thermal printer cable or a Game Boy Color link cable. It's simple to use; you power the device via a micro USB cable, plug in the Game Boy link cable,



Right The tiny PCB comes fully assembled with a nice silkscreen that helps you orient the link cable connector correctly

Below
I can look ridiculous in any resolution!

and then power on the Game Boy with the GBC cartridge inserted. You can then navigate through the camera game and use the print function to print the image to a file on the saved Wi-Fi device. To access the images, connect to the Game Boy printer Wi-Fi network on your device, and then navigate to the IP address listed on the PCB. At this point, you'll be presented with a small website served off the device which allows you to select and download your images. It not only allows you to select and download the images, but it also has a neat trick that allows you to change the two colours (black and white) to other colours which, whilst sounding limited, is actually quite good fun. On our phone, we found it easy to select images, tinker with the colours, and then long-click on the images themselves to download/save them.

One issue we encountered was that you have to be careful to keep the Game Boy link cable connector well aligned with the contacts on the Wi-Fi printer device, as the connector can wriggle around if you move the cable, and you'll lose connection. Not a problem though if you set it up on a sturdy work surface.

The images on the Wi-Fi printer device are stored in memory, so you can actually retrieve previously 'printed' images without having to have the Game Boy attached by simply powering up the board even after power cycling the device. There's room on the device to store over 200 images – these are held on the device until you delete them, so you can re-download an image previously stored on the device as often as you need.

Whilst perhaps the GBC is always going to be a niche interest, it's a joy to discover what images are stored on your old GBC cartridge, and it's also fun to take such lo-res photos. We'd forgotten how impressive the 'trick' lens effects are in the GBC – it's a great nostalgia trip.

Above
To transfer files from the Game Boy, you need to power the Wi-Fi printer via USB and insert a link cable. You can wirelessly connect to the printer without a Game Boy attached to view saved images

Left
When you connect to the Wi-Fi printer network, you are served a handy website to view and download images

VERDICT

A great device that does exactly what it's supposed to.

9/10

Bittle

Build your own robot pet

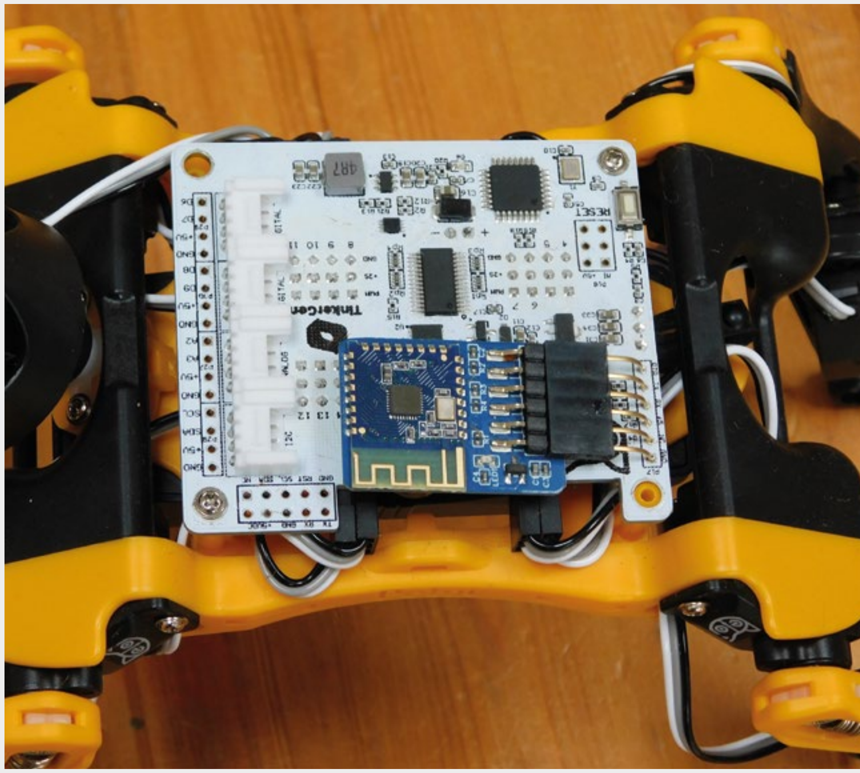
PETOI $299.00 | petoi.com

By **Ben Everard** @ben_everard

The vast majority of robot kits that you see are wheeled robots. Of those that have legs, the majority have six or more legs. This is because walking on four or fewer legs is hard – you have to balance in real time, since the robot isn't stable throughout the movement. You have to be able to move quickly and precisely to avoid falling over, and this means that you need quality parts and lightweight construction. Petoi's Bittle Robot Dog promises just this, and we put it to the test to find out more.

Out of the box, it's a fun curiosity. The build isn't completely straightforward, but shouldn't be particularly challenging for the average HackSpace mag reader. Once all together, it can connect to an app on your phone which lets you move it around, a bit like a remote control car – only with legs. The servos are quick and there's some spring in the joints to cope with shocks. Like this, it's fun to play with, but there's a limited number of uses for a remote-controlled dog, and it would be pretty hard to justify the price for this alone.

Right The dog's head isn't essential, but does make it a little less creepy



However, the real value of Bittle comes out when you start extending it. At a basic level, you can program it in a Scratch-like editor to walk, sit, crawl, etc. You can get interaction from various sensors, including an IR sensor and vision sensor (not included). There's a free online course to help you with this: **hsmag.cc/learnbittle**.

Go a step further, and you can get full control of the hardware using Arduino, or gain more processing power by adding a Raspberry Pi. At this level, you can take Bittle in any direction, such as working on the kinematics of movement to create your own Boston Dynamics-esque dance sequences (there's an inverse kinematic model and gait generator to help you get started: **hsmag.cc/bittlewalk**), or performing more advanced processing of the sensor inputs using AI. There aren't any artificial limits in terms of what you can do with Bittle, so it's up to you where you take your little electric dog – both literally and figuratively.

COST CONUNDRUM

Ultimately, $299 is a lot of money (though robotics kits can certainly get substantially more expensive). The most significant thing you get for that money (compared to other walking robots that are available) is speed. This comes from the lightweight, but powerful, servos and the sprung joints. Using these, you can get a reasonably sprightly trot out of the little beast. This speed really increases the usefulness of the robot in almost all applications. Slow walkers are frankly a bit boring, so, even if they're easy to use, people quickly lose interest. Slowness also limits the types of movement you can do, so the kinematics become less interesting, and interacting with the environment is less about interacting and more about crawling over. In contrast to the plodders we've used before, Bittle springs about in a much more engaging (and computationally interesting) way.

> "There are plenty of slow crawlers, but we've not **come across another quite as lively as Bittle**"

There are some great wheeled robotics kits around for just about any budget and feature set, but there isn't the equivalent range in walking robots. There are plenty of slow crawlers, but we've not come across another quite as lively as Bittle. It might seem a bit ridiculous to describe a $299 robot that doesn't have a specific function as good value, but robotics is an expensive hobby, and walking robots are an expensive type of bot. In Bittle, Petoi has done a good job of making a robot that's big and complex enough to interact with the real world, but still at a price point that is (just about) affordable for hobbyists. □

Above
The springs give a bit of flex to the joints, making movement more forgiving

Left
There are Grove connectors on the control board for add-ons

VERDICT

A capable and fun robot, with a price tag to match.

9/10

next month

issue
#61
ON SALE
17 NOVEMBER

PROGRAMMING FOR MAKERS

Also

- PICO
- 3D PRINTING
- MUSIC
- MINCE PIES
- AND MUCH MORE

DON'T MISS OUT

hsmag.cc/subscribe