MAKE | BUILD | HACK | CREATE

HackSpace

TECHNOLOGY IN YOUR HANDS

hsmag.CC | August 2023 | Issue #69

DISPLAYS **PLA** PICO CLUSTER

Free eBook!

Download your copy from

hsmag.cc/freecadbook

Welcome to HackSpace magazine

Here in the northern hemisphere, at least, the days are longer, warmer, and brighter. The great outdoors is a little greater, and it's a time when makers traditionally emerge from their sheds, basements, and hackspaces to engage in the world. Of course, they don't leave their tools behind. They use their maker skills to monitor, hack, and play with the world around them. This month, we're celebrating these projects that bring together the heady days of summer and geek tech. We're helping plants grow, spying on wildlife, and much more. Put on some sun cream and join us!

BEN EVERARD
Editor ben.everard@raspberrypi.com

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

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

GET IN TOUCH

hackspace@raspberrypi.com
hackspacemag
hackspacemag

ONLINE

hsmag.cc

Available on the App Store

GET IT ON Google Play

PAGE 30
FREE PICO W
WHEN YOU SUBSCRIBE

EDITORIAL

Editor
Ben Everard
ben.everard@raspberrypi.com

Features Editor
Andrew Gregory
andrew.gregory@raspberrypi.com

Sub-Editors
David Higgs, Nicola King

DESIGN

Critical Media and Raspberry Pi
criticalmedia.co.uk

Head of Design
Lee Allen

Designers
Sam Ribbits, Olivia Mitchell, Sara Parodi, Jack Willis

Photography
Brian O'Halloran

CONTRIBUTORS

Marc de Vinck, Andrew Lewis, Jo Hinchliffe, Derek Woodroffe, Phil King

PUBLISHING

Publishing Director
Brian Jepson
brian.jepson@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

recycle

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. Except where otherwise noted, content in this magazine is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported (CC BY-NC-SA 3.0). ISSN: 2515-5148.

Contents

06

06 SPARK

21 LENS

Cover Feature

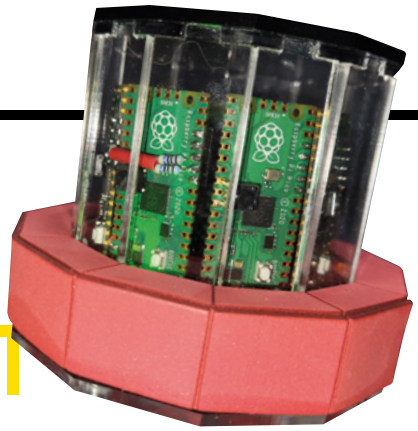
22

Tutorial

Retro gaming

72 Breathe new life (and old games) into a dusty old Atari ST

16



32

96

86

Tutorial

Wood working

78 Sustainable, strong, and cheap: recycled wood is perfect for makers

Interview

Uri Tuchman

40 Proof that when you love the tools, the tools love you back

49 FORGE

85 FIELD TEST

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

REGULAR

Geared mechanical box

By 3DTechDesigns

hsmag.cc/GearedMechanicalBox

Much as we adore the hand-crafted sensibilities that inform the Mythic I, the magic of 3D printing is that it can enable anyone to print objects that would be impossible for the average person to make by hand. This box, with its rotating gears opening six doors simultaneously, looks like it's come from a video game. It's so impractical, and so ornate, that it should hold a gemstone, or an extra life, or a portal to the next level in a game.

Assembly is easy, because the parts snap-fit together, and there's an instructional video included in the download.

Right
Not all printers are equal, so if you're building one of these yourself, you may want to have some glue on hand

Pieca

By **Tea and Tech Time**

hsmag.cc/Pieca

This snazzy-looking box is a Pieca: a 3D-printed enclosure that combines a five-inch touchscreen, a Raspberry Pi, and a High Quality Camera Module to create a digital camera. What's not immediately obvious for non-camera experts is that it's also fully compatible with Leica's M-mount lenses, opening up a world of high-end photography.

Above
Leica's been making lenses for over a century

Mythic I

By Keegan McNamara

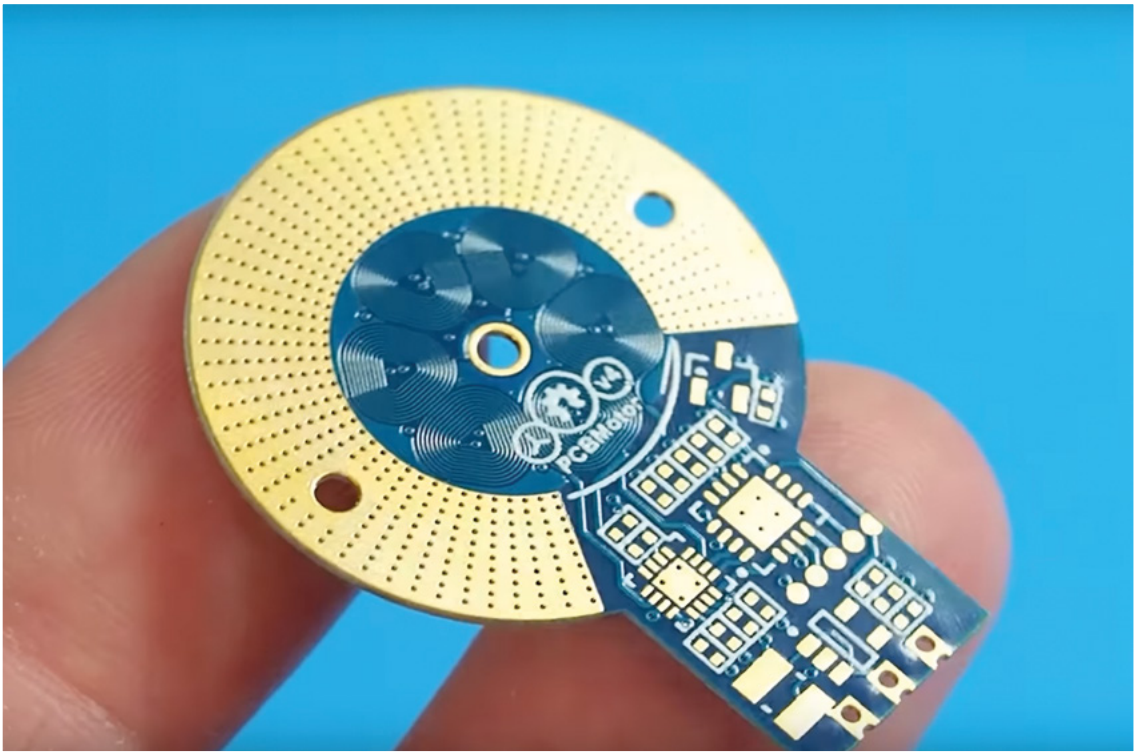
mythic.computer

The Mythic I, by Keegan McNamara, is a highly unusual computer. Keegan's gone to great lengths on his website to explain the thought processes behind its creation. Actually, thought process is the wrong phrase; philosophy is more accurate. If you've ever read *In the Beginning…Was The Command Line* by Neal Stephenson, you should expect something similar.

The Mythic I is built to be the best it can be. No corners have been cut, with a body made out of carved hardwoods, stuck together with animal-hide glue and wrapped in the finest full-grain, vegetable-tanned leather. As a computer it does one thing, and one thing well – the only software on here is a non-networked text editor (as Keegan puts it in his essay on the development of the Mythic I, "the past 50 years have shown us that unrestrained networking is a very dangerous Promethean fire"). We agree wholeheartedly.

Right
Keegan came up with the shape using modelling clay rather than CAD, to retain the hand-brain connection


Backspace



PCB Motor

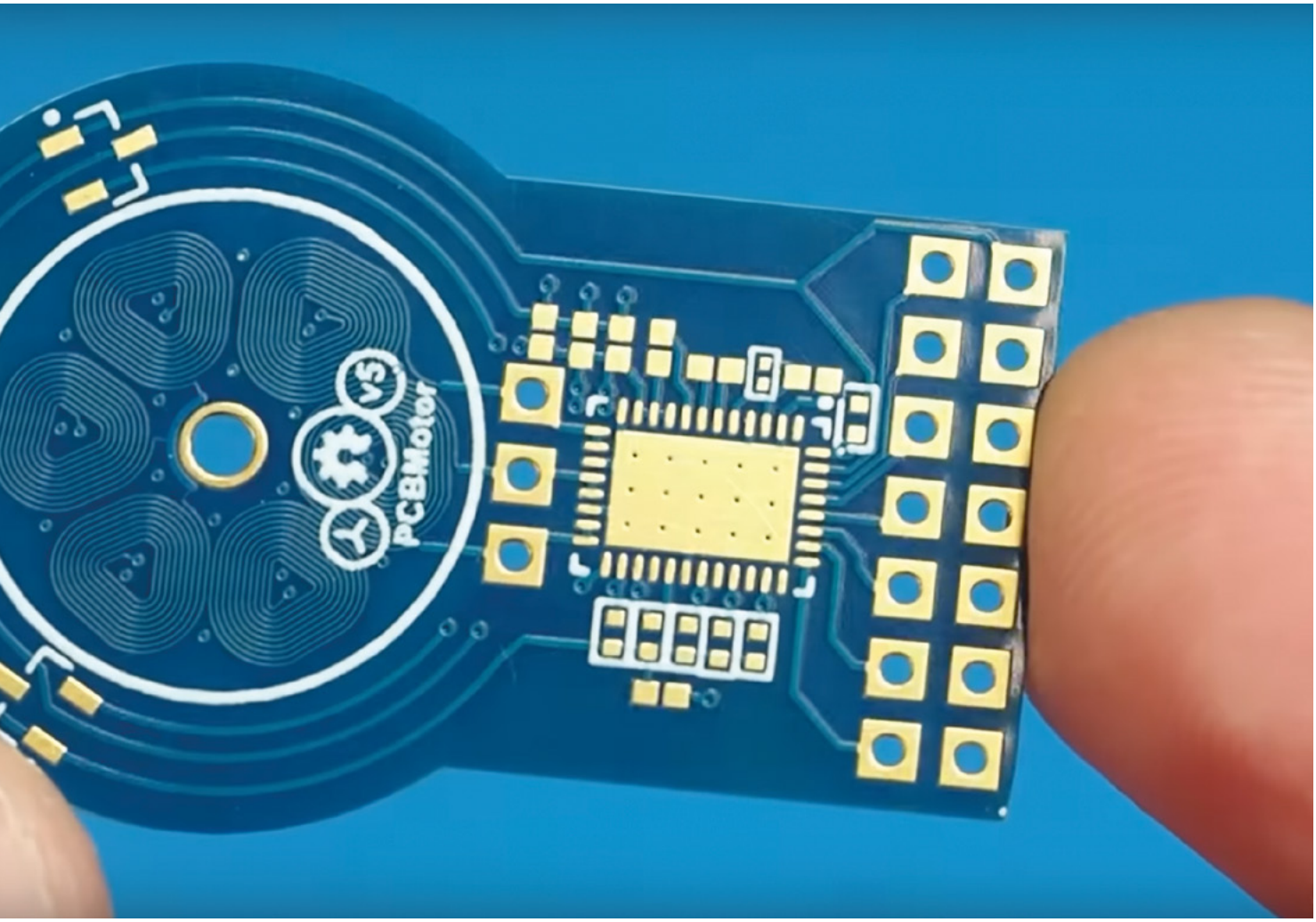
By **Carl Bugeja**

hsmag.cc/PCBMotor

We've looked at Carl Bugeja's PCB motor several times since he started work on it over five years ago. His innovation was to ditch the wrappings of copper wires that make ordinary motors too heavy, and to embed that functionality in a PCB. It's enabled him to create brushless motors that are smaller and lighter, and he's even branched out into the field of flexible PCBs to create different types of actuators.

The latest iteration of the Bugeja PCB motor features an upgraded speed controller, enabling the tiny electromagnets to turn on and off faster than before to achieve a top speed that's now 30,000 rpm faster than his previous model. We're in awe of Carl's dedication to this project, and we can't wait to see what he comes up with next.

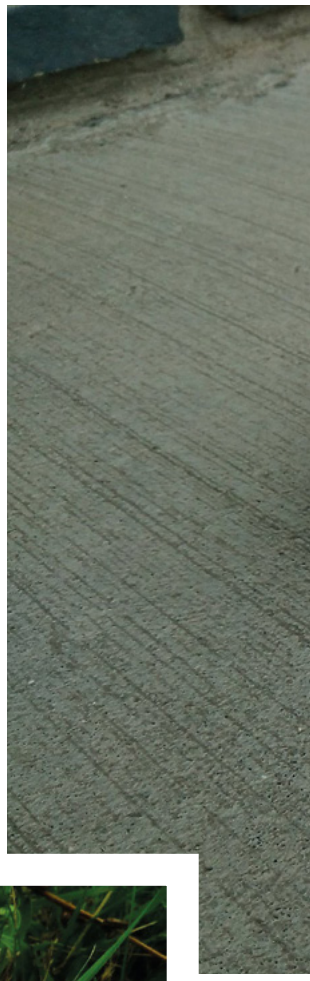
Below
The development boards that Carl needed to make his latest motor were a little expensive at €186 each – so he designed his own



Remocon Rover

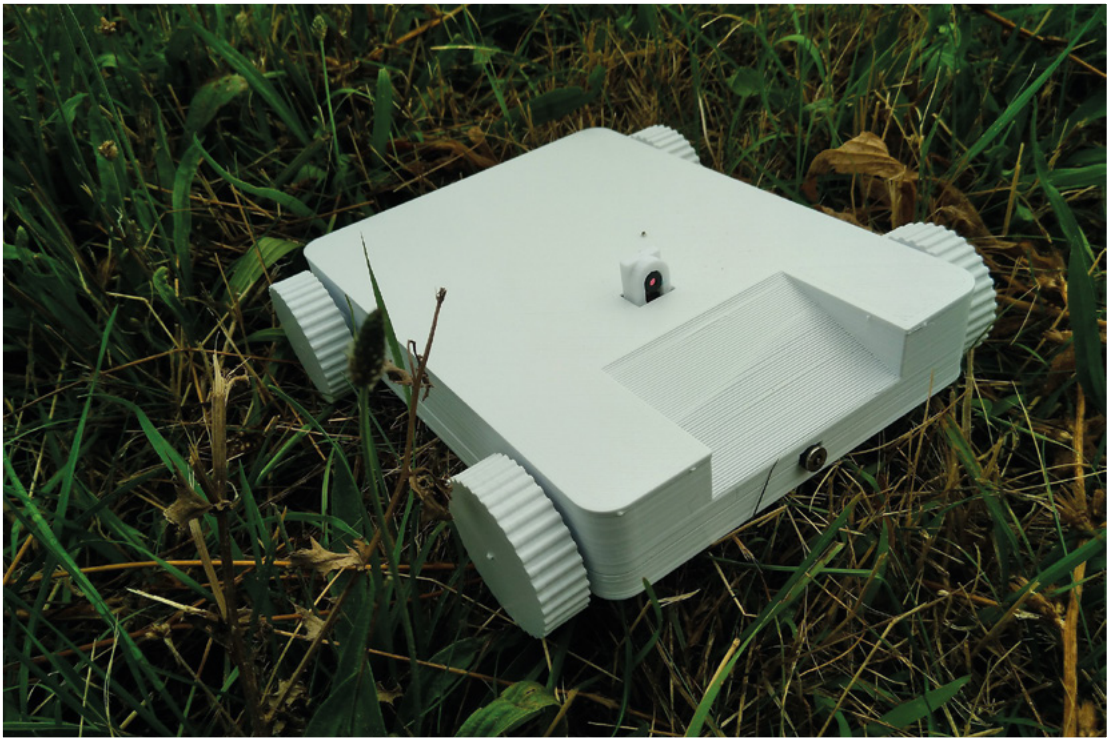
By **Remocon**

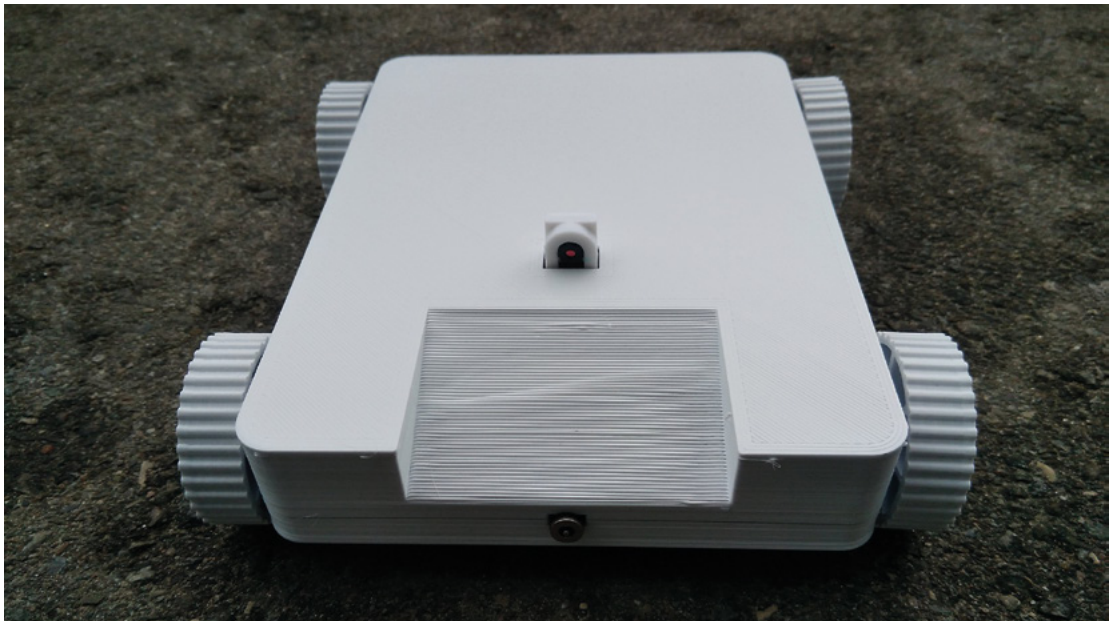
hsmag.cc/Remocon



The internet is full of devices that should never have been connected to it. Refrigerators, baby monitors, and even toilets are all out there on the Internet of Things (IoT), many of them still with the factory preset password. Just because we can connect something to the internet, doesn't mean that we should.

We most emphatically should connect things like *this* to the internet. It's a remote-controlled rover, complete with a camera, that you can control from anywhere in the world over the internet – it's a bit like one of Amazon's smart doorbells, except that it's got four wheels and you can use it to chase the dog around the house when you're not in.





Above
Home surveillance meets a 3D-printed body in this brilliant device

Objet 3d'art

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

Feast your eyes on the M.U.S.E.: the Most Unusual Sentence Extractor. Featuring a glorious 3D-printed body inspired by the Olympia Traveller de Luxe typewriter, this device was built by Brendan Charles in the hope that it would drive him to write more. It features a custom PCB for the keyboard, a 10-inch LCD screen, and a Raspberry Pi at its heart. It's also simultaneously an ideal tool for writing documents to the cloud, as well as being the most 1960s object we've ever seen. Do you want to know more? We'll be looking in depth at the process Brendan went through to create this machine in next month's HackSpace magazine, so tune in then.

hsmag.cc/MUSE


Share
Today we take a look at the Nilson Institute For Talented Youngsters, and the curious case of Toby Gershwin.
Unlike his exceptional classmates, Toby Gershwin is completely, utterly and excruciatingly... normal.
Students of the Nilson Institute are accepted for their superior grades, impressive skills or famous families. Toby got in when Grace at the front desk grabbed the wrong stamp.
So far, Toby has slipped by unnoticed, but he can't help feeling like a fake. A fraud. A pretender.
For show and tell, Tia Cappelli brought in her Junior Nobel Prize for science and Summer Snide brought in her dad, famous rockstar Sid Snide.
Toby brought a weird rock he found behind his house.
In music class, Hurry Langstrom plays the sax with soul, and Teo Moreno shreds the electric guitar.
Toby on the other hand, has trouble managing the triangle.
He's even outshined in gym class, where Tamara Park can pull off a triple backflip, and Greg Thunderberger can bench press the lunch lady.
Toby can put the ball in the net, but that just doesn't seem good enough.
Toby tried talking to his parents about his problem, but all they could say was how wonderful and special he was.
One day, Toby's worst fears come true. Mr. Nilson himself makes an announcement on the intercom, "Toby Gershwin to the principal's office, please."

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**

SWIRLING

I've been 3D printing for years and, while I love the technology, I always feel a bit guilty about the amount of plastic I'm using. Thanks for your article on recycling PLA. I'm going to give it a go, and see if I can finally turn all that spaghetti into something useful.

Jane
Norwich

Ben says: PLA isn't the worst plastic around. It's not from oil and, depending on your waste stream, may not end up in landfill. That said, if we're going to create a sustainable future, we have to get the most out of everything and that means using it more than once. What's more, the results can look incredible. This month, we're looking at laser cutting the sheets into jewellery, but there are many more ways it can be used.

MACRAM-YAY

I'm a fiddler. I just can't help it. I twiddle pens, jingle change, and generally play with anything that I can get my hands on. While I think this is a fairly innocent pastime, it drives my partner wild. Occasionally I slip and fling something across the room, or just make a bit too much noise.

I've given macramé a go and it satisfies my fiddling instinct, while being quiet and contained. I can knot together some string while we're watching TV in the evening, and it doesn't make my partner want to throttle me.

Saeed,
London

Ben says: Glad you're enjoying yourself. I can't keep my fingers still either. While I've not done too much with macramé yet, I've been a keen knotter for many years and can happily convert a length of string into a tangled ball. Perhaps if I took up macramé, I could put this habit to some use.

A SECRET LIFE

OMG Tim Hunkin! I haven't thought about *The Secret Life Of Machines* in years. It was an absolute inspiration to me growing up. It's great to see that Tim's still alive, well, and making cool stuff.

Mark,
Nottingham

Ben says: I have similar fond memories of his TV shows back in the day (no, I'm not going to look up how long ago they were, as it'll just make me feel old). If you've not watched his YouTube series, then you're in for a treat. It's both packed with information, but also gentle in the way TV used to be before algorithms made everyone shout and have cliffhangers and try to surprise you.

INSPIRATION
STARTS HERE

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

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

Digi-Key ELECTRONICS

Digi-Key is a franchised distributor for all supplier partners. New products added daily. Digi-Key and Digi-Key Electronics are registered trademarks of Digi-Key Electronics in the U.S. and other countries. © 2023 Digi-Key Electronics, 701 Brooks Ave. South, Thief River Falls, MN 56701, USA

ECIA MEMBER
Supporting The Authorized Channel

HackSpace
TECHNOLOGY IN YOUR HANDS

LENS

HACK | MAKE | BUILD | CREATE

Uncover the technology that's powering the future

SUMMER PROJECTS

They say, make hay while the sun shines; we say, MAKE PROJECTS!

Phil King

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

Make the most of the great outdoors by working on a project this summer. If you're a sporty type, you can up your game with a robotic basketball hoop, running or cycling tracker, badminton shuttle launcher, or putt speed measurer, among others.

Tidy up the garden with DIY devices such as an automatic weeding robot, mower, or lawn sprayer. Help your plants to thrive with a soil moisture monitor and automatic irrigation system.

Summer is also a great time for observing wildlife, whether with a smart animal-detecting camera or listening out for birds and bats to identify species.

You'll also want to keep an eye on the weather and make sure you're well-equipped for a hiking trip. And when night comes, you can spot some meteors. With a little effort and ingenuity, this could be the best summer ever.

GOOD SPORT

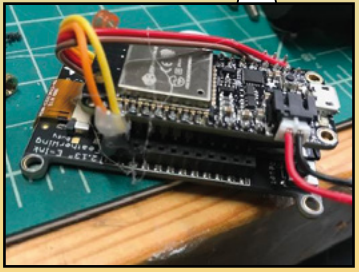
Have fun exercising this summer with the help of these sporty projects

BIKEEVEREST

The concept of 'Everesting' on a bike involves repeatedly climbing any nearby hill to achieve a total ascent of 29,029ft. You could even do it gradually on your regular rides using an altitude-tracking device like BikeEverest to tot up the cumulative distance climbed.

Maker 'rabbitcreek' has put together an Instructables guide (including code) to show how to create the handlebar-mounted device. Controlled by an Adafruit HUZZAH32 Feather (you could use an alternative ESP32-based board), it features a BMP388 precision barometric pressure sensor and altimeter, along with a 2.3″ e-ink screen to display the data and images.

hsmag.cc/BikeEverest



Portable power is supplied by a 600mAh rechargeable battery with an on/off switch, while the device is housed in a 3D-printed case (find the STL files in the Instructable).

ROBOTIC BASKETBALL HOOP

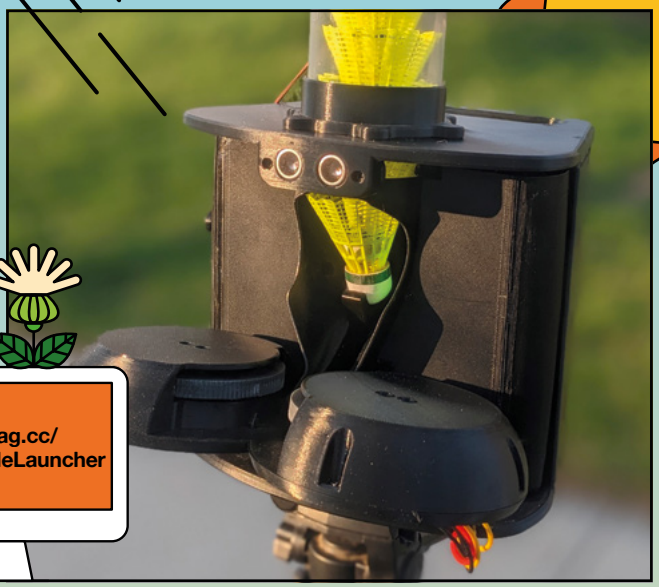
Keep missing the hoop? No problem with this robotic backboard that moves to deflect your shots into the basket! After building a simple parabolic backboard for the same purpose, maker Shane (of the Stuff Made Here YouTube channel) upped his game with this super-smart device that tracks the ball's trajectory using an Xbox Kinect camera.

A major challenge involved making the required calculations and movements in the mere 600ms it takes for the ball to arrive at the hoop. Controlled by an Arduino, three large motors provide enough power (about 0.2hp) to move the ultra-lightweight backboard, attached with push-rods, into position in time.

While the mechanical design is impressive, and numerous custom parts were 3D-printed or machined, Shane spent far more time writing the software, as it needs to be pretty complex to track the ball and angle the backboard optimally. →

hsmag.cc/RoboticHoop

BADMINTON ACE SHUTTLE LAUNCHER



Practising badminton without a playing partner is nigh on impossible. Or at least it was until the arrival of Peter Sinclair's Ace shuttle launcher. A work in progress, the project's aim is to "create a relatively low-cost multi-axis robot capable of launching badminton shuttles in user-defined patterns for teaching and training."

hsmag.cc/AceShuttleLauncher

Connecting to the ESP32 microcontroller-based launcher via Wi-Fi, the user can select and adjust the shot placement and pattern to learn or practise new badminton return techniques. A web dashboard features an interactive 3D court with controls for rotation and trajectory, adjusted using two servos.

Shuttles are stored in a plastic tube and released in turn by a servo-controller gripper. The fallen shuttle is then pushed by an arm between two rapidly spinning wheels to launch it into the air at an adjustable speed and angle.

RIOT BRICK

A popular use of technology in sports is as a route-tracking device. Seeking a way for his relatives around the globe to track his progress in a gruelling ultra-marathon – the 58-mile Brecon Beacons Ten Peaks Challenge in Wales – Alan Peaty came up with the RIoT Brick.

A 3D-printed shell houses a Raspberry Pi Zero W equipped with BME280 temperature/pressure/humidity and BH1750 light sensors to log environmental data. For IoT communication and location info, respectively, it features nRF24L01+ transceiver and u-blox NEO-6 GPS receiver modules. Powered by a 20,000 mAh battery pack, the RIoT Brick also receives additional GPS info from smaller ESP32-based trackers.

Stored in an SQLite database, the data is uploaded – via LoRaWAN and occasional 3G/4G – to Amazon Web Services, which hosts a website dashboard for followers to view the runner's progress and receive notifications.

hsmag.cc/RIoTBrick

HUD SNOW GOGGLES

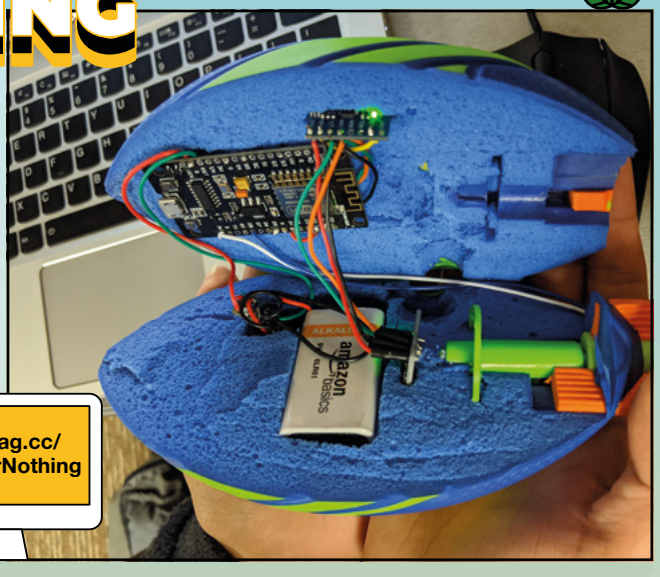
In the northern hemisphere at least, you'll need to head for the highest mountains (or artificial slopes) to ski and snowboard during the summer months. In which case, smart goggles with a built-in heads-up display may come in handy.

hsmag.cc/HUDSnowGoggles

While made back in 2013, Chris's proof-of-concept project illustrates the possibilities and is considerably cheaper than commercial options. Powered by a Raspberry Pi connected to a GPS module, the HUD – made from a pair of MyVu glasses – shows useful data including the time of day, speed, and altitude, along with temperature if a sensor is connected.

Alternatively, if you just want to look cool on the slopes, you could try adding an LED matrix to your goggles to show animations, as Josh Stapleton did (**hsmag.cc/LEDSnowGoggles**).

NERFORNOTHING



Life tip: when hosting a children's birthday party, don't leave any Nerf dart guns lying around or there could well be tears. A far safer bet is to let them play with Nerf foam football. Not content with throwing one around, you could make it smarter.

Timothy Kanarsky and his team opted to add a built-in tracking system to a Nerf football as a final project for their Physics 4AL course. By measuring the acceleration of the ball and the centripetal force of a point on it, they could calculate the distance thrown and angular velocity.

For this purpose, they sliced the football in half to add the electronics, including two accelerometers connected to a NodeMCU ESP8266 board. It's powered by a battery via a buck converter, while a push-button is used to start and stop data logging.

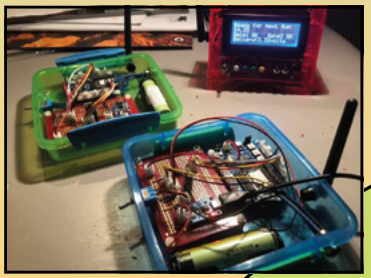
hsmag.cc/NerfOrNothing

HORSEBACK ARCHERY TIMING GATES

Think how difficult archery is, then multiply that by about a thousand when you have to hit the target while mounted on a galloping horse! Horseback archery may be a niche sport, but it seems it's growing in popularity around the world.

Whether you're an equestrian toxophilite or not, pnaybour's low-cost homemade timing gates could be useful for other sports where you need to time competitors between gates.

Each of the two gates (start and finish) is equipped with an infrared beam sensor to detect the passing rider. When the beam is broken, a signal is sent via XBee radio to a control unit based around a SparkFun Pro Micro microcontroller board to calculate the time difference. LEDs show the status of the gates, while an LCD displays the time.



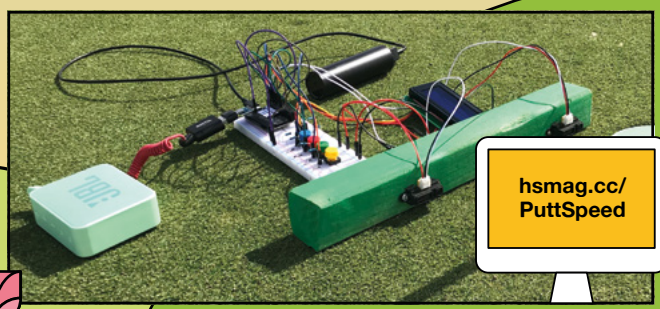
hsmag.cc/HorseArchery

PUTT SPEED TRACKER

While Mark Twain may have described it as "a good walk spoiled", golf is still a very popular pastime that has attracted inventors to create all sorts of gizmos to help players improve their game. No need to buy one, however, when you can build your own.

Grace Wilson's Putt Speed Tracker does what it says on the tin. Placed parallel to the line of the putt you're attempting, it uses two proximity sensors spaced apart to measure the ball speed as it nears the cup. Data is logged and displayed on a mini LCD. Helpful audio clips, such as "Too fast", are also played via a speaker.

According to Grace, the PocketBeagle-powered device works best for level, straight putts that are 4–10 ft from the hole. The aim is to achieve an optimum speed of 60–80 cm/s to reach the cup without going too far beyond if you miss it. →



hsmag.cc/PuttSpeed

GARDENING

Help your plants thrive while growing your own knowledge

ROKTRAK

As any gardener knows, keeping weeds at bay is a never-ending task. If you don't fancy a whole load of hoeing, you could build your own automatic weeding robot like Yuta Suito's impressive Roktrak.

Based on a Raspberry Pi 3, it uses a wide-angle camera and YOLO (You Only Look Once) algorithm to detect the pylons at the vertices of the area to be weeded. The robot heads for a pylon and then, when it gets closer, turns to look for the next one; by speeding up the turns, it makes ever smaller laps until it reaches the middle of the area you want to mow.

Equipped with two blades, Roktrak can tackle weeds and grass up to 15 cm long. Extra-large 'eccentric tyres' on the outer front wheels also enable it to traverse uneven terrain. By adding a pulley system, it can even handle banks of up to 60°. It's controllable by an Android app, too.

hsmag.cc/Roktrak

HERBIE_BOT

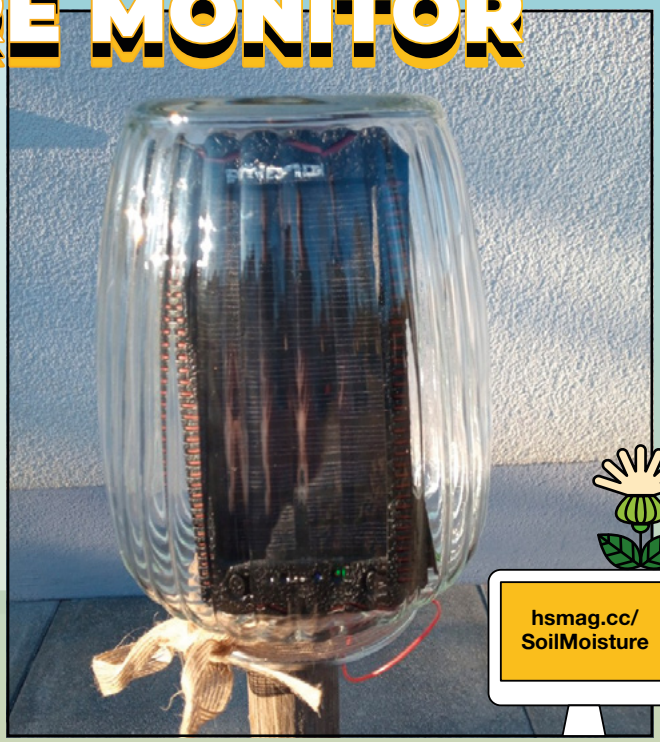
Weeding a lawn can be very laborious. After finding that spraying a lawn-safe herbicide manually was making his hand and arm cramp up, Russ Hall reckoned he could get his yard robot to do the job. By rigging it up with a camera and spray wand, he created Herbie_Bot.

With its wooden chassis and trolley-like appearance, it may look a little primitive, but this AI robot can detect and spray the weeds on a lawn. As Herbie_Bot trundles along, its downward-pointing Movidius OAK-D camera looks out for weeds using OpenCV computer vision; when one is spotted, a servo rotates the battery-powered spray wand to target it.

A Raspberry Pi 4 interacts with the camera and runs most of the processes using ROS (Robot Operating System), while an Arduino Mega is used to detect the robot's speed from its wheel encoders.

hsmag.cc/Herbie_Bot

SOIL MOISTURE MONITOR



Making sure your plants are kept watered is essential for their well-being, but there's also the possibility of overwatering and wasting precious water. So a soil moisture monitoring system can come in handy.

Having planted a flower garden with his wife, Joseph Eoff sought a more accurate way of checking the state of the soil than the rather primitive commercial system he was using. So he set up a network of 21 Bluetooth-connected sensors, three ESP32 modules, and a Raspberry Pi to track soil moisture and nutrient levels.

The ESP2 boards are solar-powered and transmit sensor data via MQTT over Wi-Fi to the Raspberry Pi. The latter runs Joseph's custom software, written in Django, to generate a graph and animated 'heat map'. The project is a work in progress, but the garden is looking good.

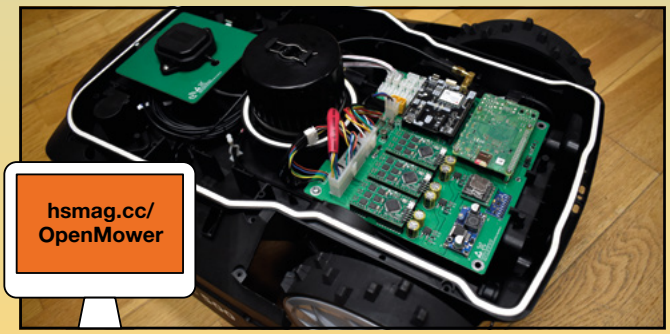
hsmag.cc/SoilMoisture

OPENMOWER

As part of his long-term aim to build a general robotics hardware prototyping platform, Clemens Elflein created an open-source automatic mowing machine. Adapting an off-the-shelf model, he replaced its electronics to make it a whole lot smarter.

His OpenMower robot features both a Raspberry Pi 4, mounted on a custom PCB, and a Pico. Low-level, real-time tasks are handled by the Pico, enabling the Raspberry Pi 4 to focus on high-level logic such as positioning, navigation, and coverage path calculation.

An RTK GPS board enables accurate lawn mapping, achieved by Clemens driving the robot manually around the lawn. "When it's time to mow," he says, "the OpenMower software internally uses the Slic3r library, which is basically a tool path planner for 3D printers, to plan a coverage plan for the recorded area." You can see the robot in action in his demo video: **hsmag.cc/OpenMowerVideo**.



hsmag.cc/OpenMower

IRRIGATION SYSTEM

In addition to monitoring soil moisture, you may want to set up some automatic watering. Finding that the plants in his vegetable garden often ended up withering during a hot dry summer, Ben Finio created a Raspberry Pi irrigation system.

His setup uses a weather API to determine whether it has rained; if not, the Raspberry Pi triggers a solenoid valve via a relay switch to water the garden using a network of plastic pipes. Naturally, you need to ensure that electronics and solenoid are protected from any splashes by enclosing them in weatherproof boxes.

Sure, there are commercial garden irrigation systems available, but it's more satisfying to create your own and customise it to your needs. There are countless online tutorials, some using an Arduino or other microcontroller, and also dedicated DIY kits to make it simpler. →

hsmag.cc/RPiIrrigation

WILDLIFE WONDERS

Tech projects to help you engage with nature this summer

WILDLIFE CAMERA

By setting up a camera trap, using a PIR sensor to detect movement, you can capture close-up shots of birds and other wildlife – try luring them with food. You'll need a weatherproof enclosure; you could use a transparent plastic box or something purpose-built such as the Naturebytes Wildlife Camera Case – also available as a fully-fledged kit including a Raspberry Pi and rechargeable battery pack.

With an infrared camera and IR LED illumination, you can even take photos and videos of nocturnal creatures. One option is to mount an IR camera in the roof of a bird box (out of roosting season), ready to view its eventual occupants via a YouTube live video stream: **hsmag.cc/IRBirdBox**.

BIRDNET-PI

You can identify birds from their calls, too. Patrick McGuire's BirdNET-Pi system runs on a Raspberry Pi. Once set up and connected to a mic, it listens 24/7, extracting bird songs and sounds, then analysing them to identify species in real-time.

A web dashboard shows which birds have visited your garden, with recordings of their calls. Other tools include daily charts and 30-day detection stats. You can also receive notifications and share your detections with the BirdWeather citizen science project.

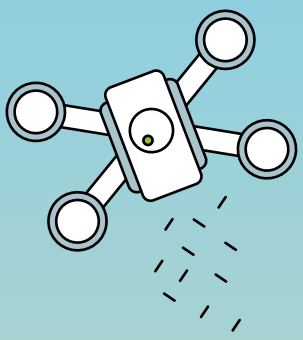
hsmag.cc/
BirdNETPi

DRONECORIA

As well as being fun to fly, drones have many practical purposes in the great outdoors. For instance, Dronecoria is a collection of tools for the aerial sowing of seed balls with efficient microorganisms, to make green large-scale landscapes at low cost.

Drones can also be used to spot wildfires. Based on off-the-shelf hardware controlled by a Raspberry Pi 3A+, the autonomous Hot Spotter drone (**hsmag.cc/HotSpotter**) can survey a 1600 m^2 area twice during a ten-minute flight, generating a heat map to show smouldering fires.

hsmag.cc/
Dronecoria



OTHER IDEAS

Some extra inspiration for projects to try this summer

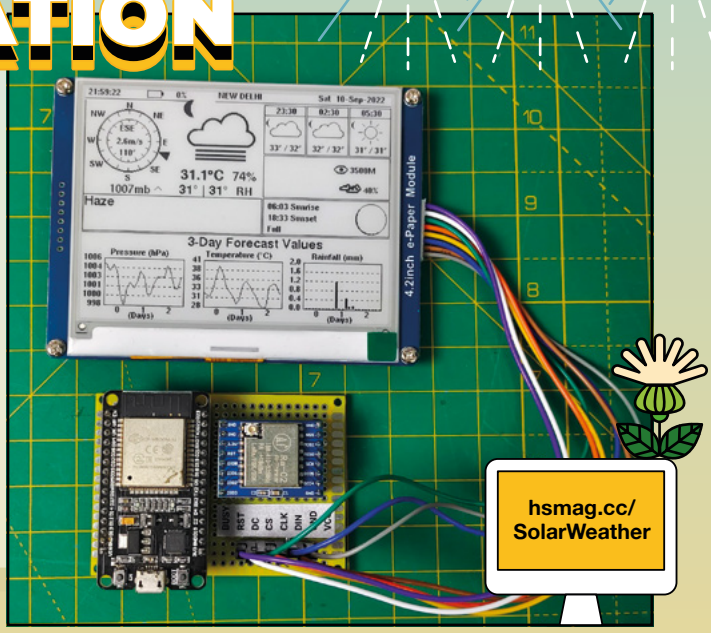
WEATHER STATION

There are countless ways to go about building a weather station, using different boards, sensors, and add-ons, which can be confusing and costly. The open-source Solar-Powered WiFi Weather Station V4.0 project aims to make it simpler and more affordable.

It comprises two nodes: a 'sender' deployed in the field (with no internet) and a 'receiver' indoors for uploading weather data for monitoring and analysis. The two nodes feature an ESP32-based custom PCB and communicate via LoRa modules.

The sender can be connected to a range of weather sensors, to measure temperature, pressure, humidity, wind speed/direction, rainfall, soil temperature and humidity, UV levels, and air quality.

Or, you could use a Raspberry Pi with Pimoroni's Weather HAT and upload the data to a web dashboard – see this author's tutorial in The MagPi issue 119 (**hsmag.cc/MagPi119**).



WATER DISTILLER

If you're going hiking or camping, a water distiller can come in handy. The 3D-Printed Water Distillation Doodad "can extract safe drinkable water from anything containing moisture: pond water, mud, moss, sweaty socks, rotten fruit, any live plant matter, etc."

The Doodad is screwed onto a small (300 to 500 ml) plastic pop bottle that fits inside a (split) larger 2l bottle to create a solar-powered water distillery – you put the 'dirty' water in the bottom and it gradually evaporates, with the resulting condensation draining into the smaller bottle.

hsmag.cc/ WaterDistiller

To avoid getting lost, you may also be interested in a Hiking Tracker (**hsmag.cc/HikingTracker**). Based around an Arduino and OLED screen, it tracks the compass heading, altitude, temperature, pressure, humidity, time, distance and GPS location.

SUBSCRIBE TODAY

FOR JUST £10

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

hsmag.cc/FreePico

Subscription will continue quarterly unless cancelled

NEW INTERNATIONAL OFFERS!

Get 6 issues plus a **FREE Raspberry Pi Pico W** for just...

$43 USA / **€43** EU

£40 Rest of the world

Head to **hsmag.cc/subscribe** to get yours today!

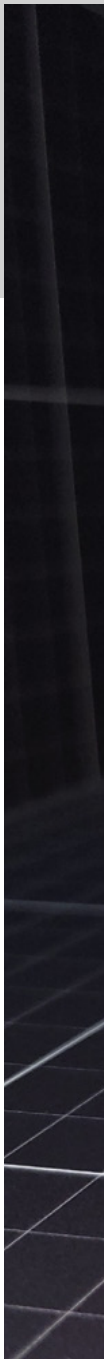
FEATURE

HOW I MADE

By Derek Woodroffe

THE PICOCRAY

Distributed computing with Picos



Every time a new computer board comes out, shortly after DOOM has been ported to it, there is usually the appearance of 'the cluster'. The Raspberry Pi Pico was released in January 2021 and, as I've seen no distributed computing projects, I thought it was time to see what could be achieved with a cluster of Raspberry Pi Picos.

Of course, the Pico has a few things not going for it, for easy clustering:

1. It doesn't natively run Linux
2. There is no Ethernet interface or easy access to an IP stack (except the Pico W's Wi-Fi)
3. It isn't really that powerful (compared to, say, a Raspberry Pi 4)

But they are inexpensive, and I had ten. The lack of a multitasking OS or IP stack means we will have to natively do the communications and the process control but, for something simple, that shouldn't be too hard.

PROOF OF PRINCIPLE

I started with three Picos connected by their I2C ports – one as an I2C controller and two I2C processors, all powered by individual USB cables and connected via a strip of Veroboard. A quick test in MicroPython proved that it was workable but, as we are aiming for speed, I quickly swapped to C. Luckily, there is example code for the Pico I2C Slave in Raspberry Pi's C GitHub, and this formed the basis for the Processor-to-Processor communication.

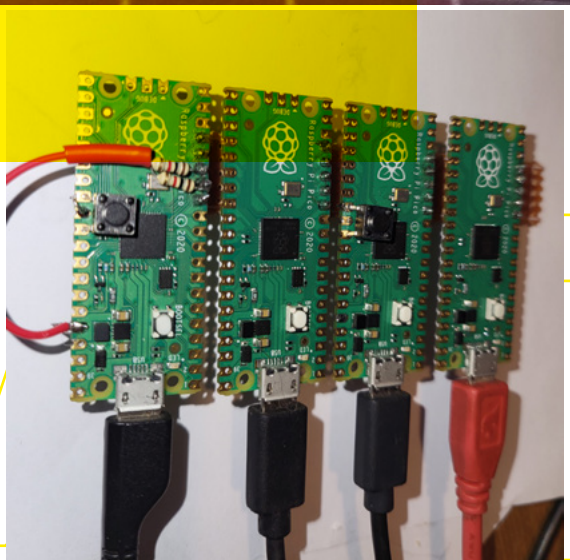
I quickly got the C code working, and this allowed me to have a 256-byte area in each Processor using I2C 1-byte addressing and

readable/writable from the Controller. One byte in this area was designated as a status for that Processor.

ADDRESSING

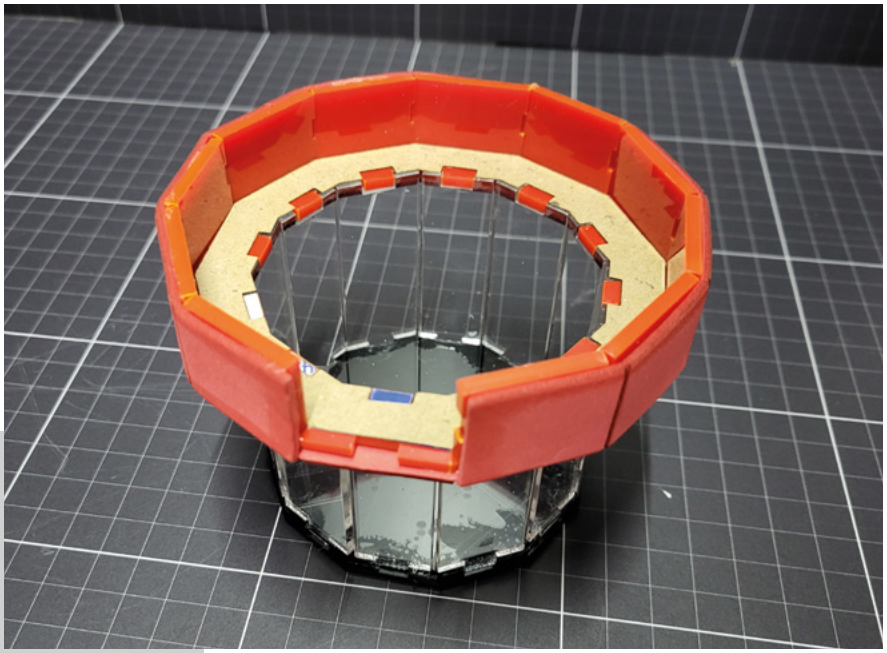
I quickly realised that scaling this up beyond a few Picos would mean that I'd need to set the address of all the Processors individually, which I could do in software, but this would mean programming each differently, or I could use switches or links on each board, which would be error-prone, not scalable, and require a lot of wiring up. I decided on a more elegant solution.

Defining the Controller was easy – for this, I used a single GPIO pin →



Above
Almost 50 years after the Cray-1, this is almost a 50th of the size

Left
You don't have to use a Cray-inspired enclosure, but why wouldn't you?



Above
The empty enclosure, ready for Picos

pull-up enabled. Grounding this pin, the Pico became a Controller; while left unconnected, it was a Processor.

For the Processors, I devised a scheme where each Processor would wait a random amount of time before connecting to the I2C bus. When the Controller saw the Processor on a default I2C port, it would allocate that Processor an I2C address, and the Processor would restart its I2C code with the new address.

> ***"I2C BECAME THE PROTOCOL OF CHOICE"***

This worked fine with two Processors, but adding more led to conflict as there is no way the new Processor knew if the I2C bus already had a device listening on the default address. To resolve this, I added a separate Assert line which was common to one GPIO pin on all Processors.

Now, when a new Processor comes online, it can check the Assert line. If it's low, the Processor takes the Assert line high and adds itself to the I2C bus. If the Assert line is already high, it does nothing and waits a random period before trying again.

When the Processor has had an I2C address assigned by the Controller, the Processor lowers the Assert line, allowing another to claim the default address and sets its status to READY.

After a few seconds, all the Picos will have a unique I2C address and have registered their statuses as READY.

And the best bit – they now all run the same code.

WHY I2C?

I chose I2C initially because I'm lazy, and connecting three wires to each of the Picos seemed the quickest way to get them all connected, but I stuck with it.

USB would have been great, but it's complicated to do host/Controller and still allow programming of the Pico. I discounted it as too difficult.

SPI would also be great, the bus speeds are up to 20 times faster than I2C, but you would need separate CS lines for each Processor. This means no auto address assigning and a limit on Processors, dictated by the spare GPIO pins of the Controller.

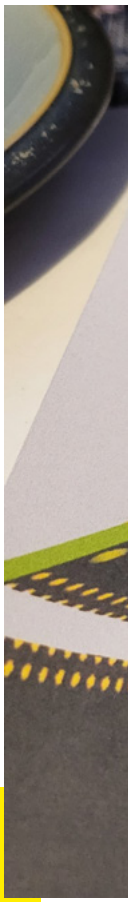
Serial/UART has two issues. There is no inherent addressing, so you would have to listen to and process all the messages on all the Processors. Also, there is a limit of around 1MBd (Async), although synchronous may be worth exploring later.

So, I2C became the protocol of choice.

THE PCB

Adding more Picos to the Veroboard caused the cabling to become increasingly difficult to manage – a PCB was in order. To keep the size of the PCB down, I came up with the idea of mounting the Pico's USB connector downwards, as this would give power to each Pico and, as there were only four connections between Picos (GND, I2C Click, I2C Data, and Assert), it should then be easy to connect a daughter board between them. I then thought it would be great to make an 8-port USB hub to enable me to program them all at once, too.

There are very few 8+ port USB hub ICs available that are easily hand-solderable. I eventually decided on the FE1.1S, which is only a 4-port hub. You need three of these devices to make an 8-port hub, as

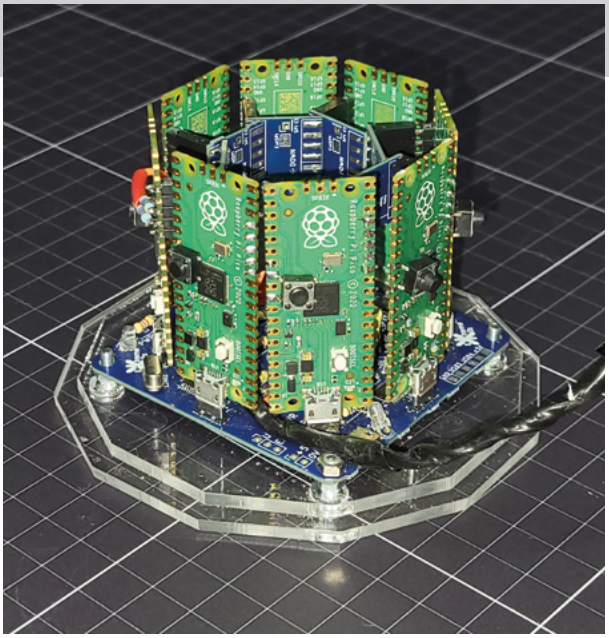


each chip has four downstream ports and one upstream. If you want to end up with a single upstream port, you need to use a third USB hub chip to combine these two upstreams into a single upstream connection to your computer. This was all a mighty risk, as I'd never played with USB hubs before, but if only the power to each socket worked, it would be a neat way of connecting all the Picos.

With the PCB, I made eight (and some spares) daughter boards to connect the GND, I2C, and Assert lines in parallel to all the Picos at a jaunty angle. Annoyingly, I missed off the I2C pull-up resistors, so these are added to the last PCB in the chain.

MANDELBROT SET

With a solid hardware base, I could now start to put all this awesome Pico power to work. I'd been looking for a program to use as a demonstration, ideally a problem that could be easily carved up into multiple tasks, and I rediscovered the Mandelbrot set.



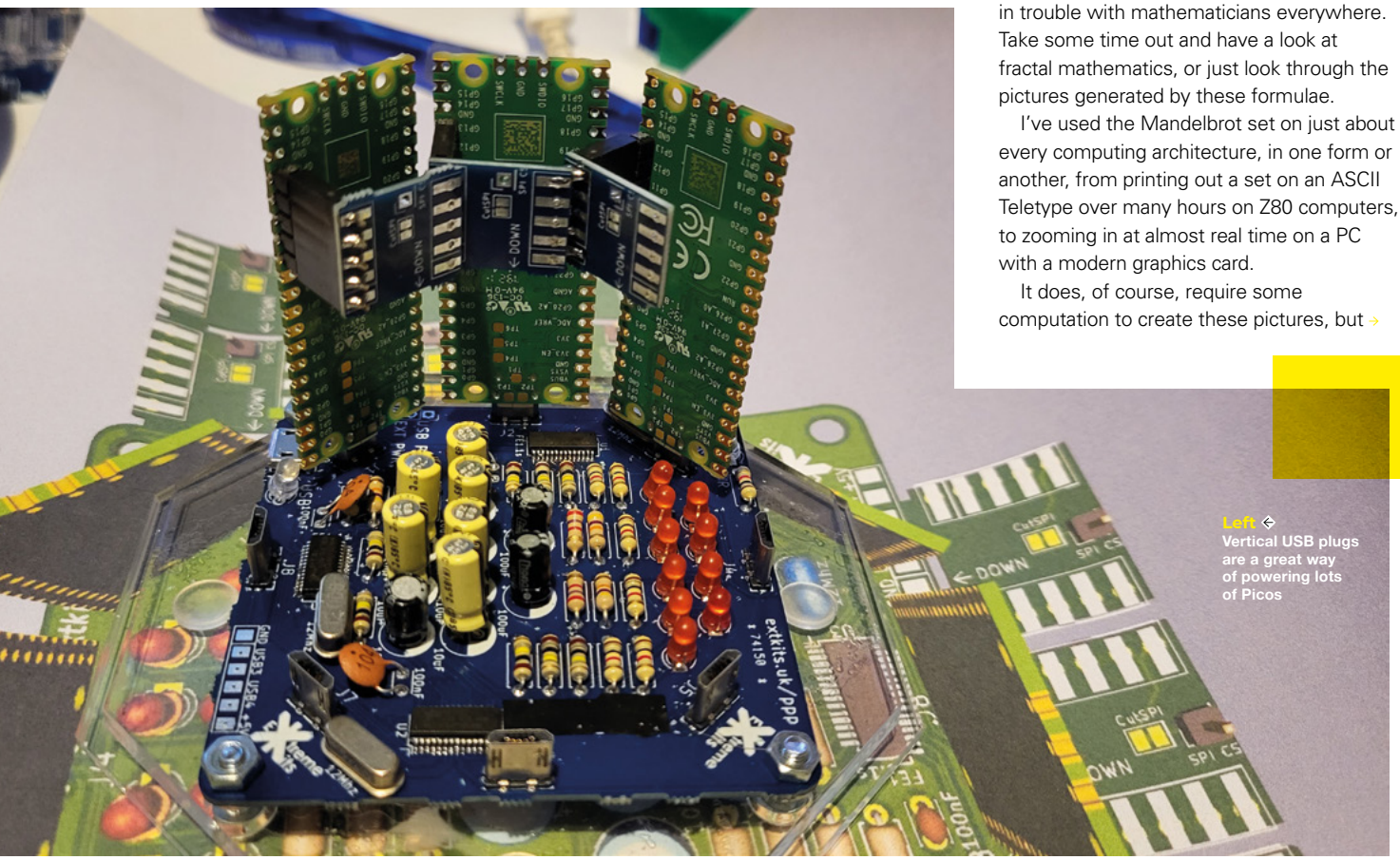
Above
Octagons are bestagons

The Mandelbrot set is a mathematical function that creates a complex pattern from a relatively simple algorithm. What is brilliant about this fractal is, by controlling the range of numbers sent to the function, you can 'zoom in' to the pattern and reveal more and more detail.

This is a seriously simplistic explanation of a truly fascinating set of formulas and patterns that is out of the scope of this article, and will probably get me in trouble with mathematicians everywhere. Take some time out and have a look at fractal mathematics, or just look through the pictures generated by these formulae.

I've used the Mandelbrot set on just about every computing architecture, in one form or another, from printing out a set on an ASCII Teletype over many hours on Z80 computers, to zooming in at almost real time on a PC with a modern graphics card.

It does, of course, require some computation to create these pictures, but



Left
Vertical USB plugs are a great way of powering lots of Picos

it's easy to split this into areas that can be handed out to each of the Processors for them to work on their own little part, so for this project it's ideal.

Above
An extra Pico controls everything

Below
Is this the most colourful mathematical function?

DISTRIBUTED COMPUTING

Splitting up the Mandelbrot is quite simple. The set is broken up into 'lumps' of a line of 120 pixels by the Controller.

Once there is at least one Processor with status READY, the Controller will send a task to the Processor – this is an X and Y coordinate and a step value (3 doubles) for the Mandelbrot calculation and how many calculations to do, which is the lump value. The Controller sets the Processor's status to GO to start the computation. This is acknowledged by the Processor changing its state to BUSY. A note of which Processor and starting X and Y coordinate is also saved.

The Controller continues allocating tasks until there are no READY Processors.

To fully use the available processing power on each Processor, the calculations are split over the two Pico cores. When the Processor has completed the calculations in both cores, it sets its status to DONE.

The Controller continues to scan each Processor for its status. If it sees a status of DONE, it will download the results, currently a lump of unsigned chars representing the iteration value of the Mandelbrot calculation. When this has been transferred, the Controller sets the Processor status to READY.

CONSOLE DISPLAY

The Controller uses the received data and the saved X and Y coordinate to assemble the image. A palette is applied to the 8-bit data to give 16 bits of colour – this is sent to the display, either directly or via DMA. The display is a 240×320 pixel colour module with touch, and is connected to the Controller via SPI.

Touching the display starts the Mandelbrot at the next zoom level, centred around the area that was touched. A few different colour palettes can be chosen from, although they can only be changed at compile time. The console has a set of three buttons: Reset, Back (undoes the last zoom), and Again (redoes the last zoom).

CRAY CASE

The original Cray1 was a 1970s supercomputer, with a distinctive dodecagon shape and a settee around the base. They were originally made in various colours and with some transparent panels, ideal as a

Left
Everything together and working away

case/shape for this project. The case was cut from transparent and red acrylic, and glued together using an acrylic welding solution. To keep the shape together whilst the acrylic welded, the inside of the non-transparent pieces were backed with cardboard.

It would not be a Cray without some form of settee, and this is replicated by using red 2 mm thick foam. The foam shape is laser-cut and the seat cushion detail was laser-engraved.

IMPROVEMENTS

Although the project works very well, there are some ideas I'd like to pursue.

I'd like to do something other than compute a Mandelbrot.

The communication speed could be improved by using SPI, 4-bit SPI, some form of X bit I2C maybe, USB or something else – there are a lot of free PIO processors looking for a job.

The Processor's results could do with more than 256 bytes in shared/accessible memory too.

There are some unexpected USB problems, although not all caused by the PCB. They occur when twelve devices are plugged in simultaneously into a computer. There is a known issue in the Pico's tiny USB that causes enumeration problems. This can be partially mitigated by software changes at compile time.

Some issues are caused by the host computer, and I'm still working through them. Due to this, I only really use the PCB for power, although it does mean I have to update the software on the Picos one by one, which is a pain.

CONCLUSION

I have achieved some level of distributed computing spread over eight or more Picos and a Controller, and there is a rough format in the code for allocating and structuring the data to and from multiple Processors so other algorithms can be implemented.

But I'm not sure if any of it is of any practical use, other than a learning exercise, unless you want a small inexpensive multiprocessor system with a touch display, 18 cores, and 208 free GPIOs.

Basically, I have created a solution looking for a problem.

FURTHER READING

Extreme Electronics website:
hsmag.cc/ExtremeElectronics

GitHub code and circuit diagram:
github.com/ExtremeElectronics/PicoCray

Wiki Fractals:
en.wikipedia.org/wiki/Fractal

Wiki Cray-1:
en.wikipedia.org/wiki/Cray-1

Raspberry Pi GitHub I2C slave example:
hsmag.cc/PicoExample

Twitter:
@extelec

Mastodon:
mstdn.social/@Extelec

sinclair

Commodo

the

COMPUTERS THAT MADE BRITAIN

OUT NOW

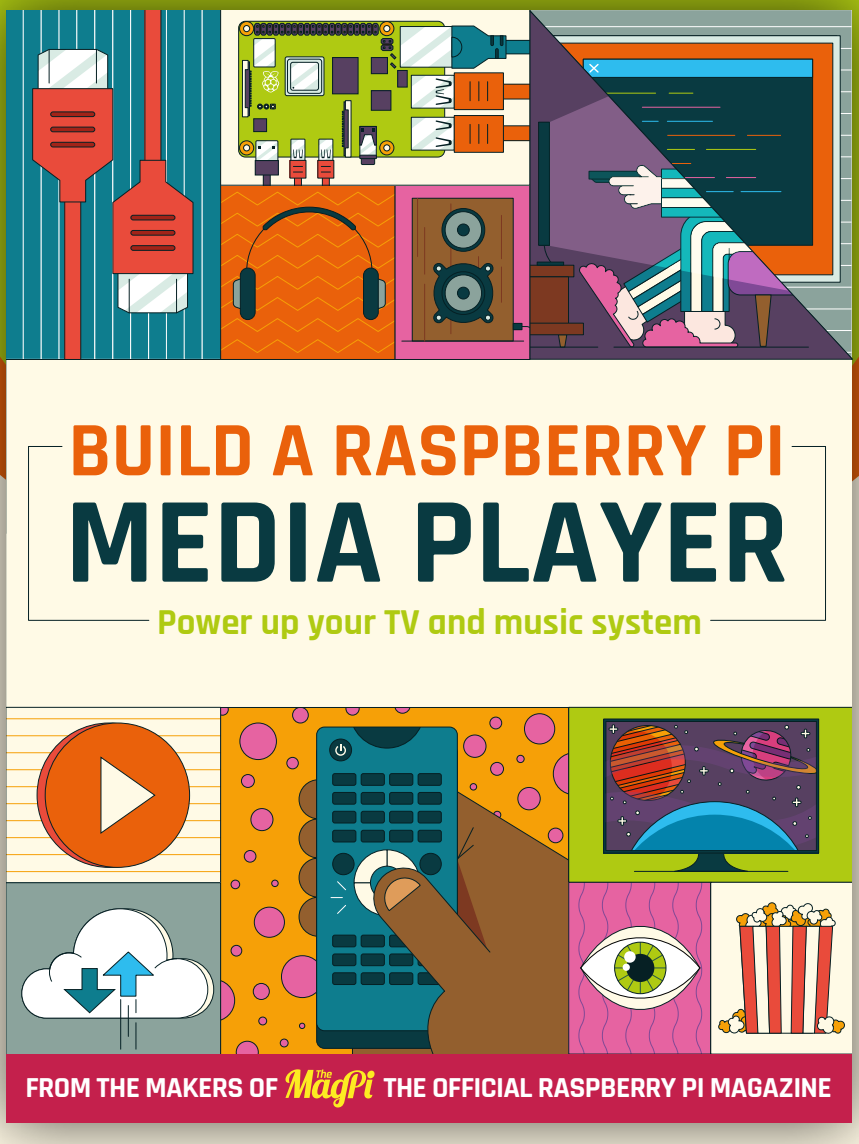
"The Computers That Made Britain *is one of the best things I've read this year. It's an incredible story of eccentrics and oddballs, geniuses and madmen, and one that will have you pining for a future that could have been. It's utterly astonishing!"*

- **Stuart Turton**, bestselling author and journalist

Buy online: **wfmag.cc/ctmb**

Available on amazon

Your FREE guide to making a smart TV



magpi.cc/mediaplayer

HackSpace magazine meets...

Uri Tuchman

Keeping traditional crafts alive by making beautiful things

Uri Tuchman is an artist, metalworker, and tool-maker. He's also made some of the most beautiful things on the internet, at first from the corner of his Berlin apartment, then from his workshop. If he can't find a tool he wants, he'll make it himself; in fact, even if he can find the tool he wants, he'll probably make one himself and use that instead, as he says that you get a better connection between your hand and the tool that way, and we believe him.

If you ever despair at the ugliness of the modern world, don't worry, just go and visit Uri's YouTube channel at **hsmag.cc/UriTuchman** and let his unique blend of slow and steady craftsmanship, pigeons, and hand tools soothe your troubles away. →

HackSpace You've done almost too many things to talk about. But what really catches my eye is the brass, the carving, the mechanisms that you use all look so old-fashioned. So, I'd like to go into why you seem to hate the Industrial Revolution so much. The reason I ask that is that most people who are making cool things use CNC or 3D printing. Hand-machined brass and carvings I would associate more with the 18th century, before the Industrial Revolution.

Uri Tuchman Let me try to answer a more general question about modern tools, more advanced technologies and things like that, which I deal with a lot. It's always a balancing act.

There is a very strong come-back to hand tools, especially in wood working; there's something very therapeutic about it, something very nice. There are lots of reasons why you would want to use more traditional tools than the newer ones, some of which aren't that obvious. But the tool really dictates the outcome sometimes – I'm not talking about the nice finish that you get from a CNC machine, or a grinder or something like that, but rather that different tools dictate what kind of project you take upon yourself, like what cuts do you want to make.

Aesthetic features, like carvings, like intricate engraving of flowers and such, they came hand in hand with the tools you had at the time. If they had had CNC machines back then, I'm not sure they would tackle the same aesthetics, the same kind of organic shapes.

Sure, you can do organic shapes today as well, but it coincides with the way you craft things with your hands. And so, if you want to use a CNC to make flowers, engravings, it feels like you need to force the machine to do that – like you're telling it to do something that [it] is not really made to do. It can, but it's actually probably easier and faster to do it by hand.

If you want to mass-produce, that's a different story; if you want to have millions of copies, CNC machines will win every time. But if you want to do a one-time piece, a lot of the time, hand tools have the advantage.

And there's also the human touch: you interact with the thing that's right in front of you, it's a hand-to-hand interaction. And so, if I had a CNC machine, if I had a laser cutter or something like that, then it would change the things that I make. I don't have anything against modern technology, it's just that they don't fit the kind of things that I want to make.

HS Do you feel a different connection with machines and tools that you've made yourself?

UT Yes, of course. Just like an oiled machine is nice to work with; you tend to give it more attention and time to make nice things. If you have a machine that is limping a little bit and isn't very well-tuned, you tend to rush through the work to get it done, because it's not as pleasing to use. So you make a bad job with it.

> "It's just that they don't fit the kind of things that I want to make"

I think a good-looking machine also gives you the same feeling that you want to spend time with it. And you want to take things more slowly and have more patience with them. Making my own tools is a great joy.

I'm actually working on a new tool now, and I'm putting a lot of effort into making it better than anything on the market. It's going to look great. But there's also a thing that happens to a lot of makers: you start making tools for your shop to make nicer tools for your shop. And you just keep on going in this loop, so you never actually make things with your workshop, right?

HS You use a pigeon as a maker's mark on a lot of your work. Where does that come from?

UT When I was in [the] last year of my time in art school, as my final exhibition, I made a whole bunch of things in the exhibition, including electric violins and all sorts of weird stuff that was hanging in a lobby-type environment, a waiting room. I made a photo book for the waiting room, which was supposed to be the most generic photo book ever. So I called it the *Big Book of Black and White Pictures of Pigeons*.

And because I wanted some sort of volume in the book, I really didn't care about the quality of the photographs; the photographs weren't the main thing. The main thing was that I have a book of photographs. So I just took a whole bunch of pictures of pigeons, and that was the first kind of project with the pigeon. I also painted the pigeon on a snooker table for the same exhibition, I think.

Why pigeons? I think it's a great animal. There's something very cool about the fact that it's adapted to being in the city. It's not a very natural animal, weirdly enough, it's just like a city animal by now. They're not elegant any more. People call them flying rats, and yeah, it sounds kind of a bad thing.

But I don't know, what does that say about us? We also live in the city, right? And we also throw garbage everywhere. And so there's something that I can relate to in those pigeons, in that we all have to live in this environment that we created for ourselves.

HS As well as hand tools you've also worked with milling machines and lathes. Do you have any mechanical or

Above
Uri's handmade astrolabe calculates the date from the position of the sun

Right
If you're anywhere near the Tel Aviv Museum of Art, pop in and see this typing automaton

Above
A lot of Uri's tools look like they come from the golden age of invention, like this machine for cutting gears

engineering background, or are you just looking at things from the perspective of an artist?

UT I never studied engineering. I worked for six months or so with an industrial designer, so I've learned the basics of SolidWorks, though I'm still pretty bad at it. Having said that, I've always tried to use it again, or use Fusion 360 or something like that. But I never get around to it. It's too involving for me to actually start designing the project this way.

I guess you can see that in my products as well. I do learn from one project, and the next one I do get slightly better and more methodical. But you know, I make a whole bunch of mistakes, and that's how you learn which ways you want to connect joints or things like that. CAD works well enough for robots, but it's not for me.

HS There's a phrase you use in one of your build videos to describe a mistake as a "happy little accident." That's a Bob Ross saying, isn't it? You get to follow the art along as the artist is making it, rather than having it presented to you as a fait accompli.

UT I work in a way that is very direct; I make one thing after another, which has its downsides. Planning ahead can be very useful and very efficient. The comparison you can make with Bob Ross is that his method is called direct painting, meaning that he does a painting in one sitting, so to speak. He goes to the canvas, and he paints everything immediately – that's going to be the final picture. OK, maybe there's a few layers that he's going to put on top. But basically, he works with just wet paint on the canvas and on the palette, and he works on them together. As opposed to more of a kind of Venetian painting, or Dutch paintings where you [add] layer after layer after layer. You wait months in between for the layers to dry, and then you get optical effects and so on. Today, nobody has time for that. Bob Ross is very much the essence of working in one sitting and working just with wet paint.

HS I guess that brings you back to SolidWorks and not wanting to spend too much time planning things when you just want to make a start.

UT Yeah. Even though I sometimes really feel like I would be faster if I planned ahead, I just like to get my hands dirty. So I need to think about that as well.

HS Do you have a favourite object that you've made?

UT One of my favourite-looking pieces is the callipers, with the hand and the foot. This is one of my favourites for sure. And the astrolabe, I really loved the astrolabe. It's a beautiful item but it's less my design. Yes, I put my ideas into it, but it's more of kind of a universal design for the astrolabe. Yeah, I would say these two are two of my favourites, of course – the small automata I love as well. Yeah, these three are my top.

> "I work in a way that is very direct; I make one thing after another, which has its downsides"

For the astrolabe, in particular, I had to do some research on the celestial side of things. I wanted to get the basic concept of how this works, which is fascinating in itself. But most of the research in the astrolabe was the translation of the linguistic stuff, like the names of the stars and the constellations.

I know Hebrew, and my Arabic is pretty garbage I'm afraid to say, but almost all, maybe 98% of astrolabes, were written in Arabic, with a few in Latin, and on rare occasions you find them also in Hebrew. Because a lot of goldsmiths and metalworking guys were Jews living in the Islamic world. Nowadays, we're more familiar with [Jews being associated with making jewellery], and it kind of coincides. That's why you see some of them with Hebrew texts, because maybe a rich Jewish person in Spain ordered all the way from Jordan, or something like that. And it was in Hebrew, but normally, they would do it in Arabic. The cool thing is that they would write it in Hebrew, but it's actually Arabic – it's only phonetically written in the Hebrew script. And the weird thing is that the Arabic is ancient Arabic.

It's a huge process to understand all this, and it's thrown up all kinds of interesting stuff. The best example I can think of right now is that you have the star signs – the crab, the goat, whatever. And so, you have the twins. But instead of saying 'twins', they point to the two stars that are at the bottom of the legs of the twins, and it just says 'leg leg', one after the other. Those ancient metalworkers had very little real estate to work with. →

Above
It's unergonomic, it's heavy, but with this mouse you would be invested in every click

HS Your other favourite, the animatronic hand, has made it into an exhibition at the Tel Aviv Museum of Art. How did that happen?

UT I'm not sure who threw my hat into the ring. But it's a group exhibition with three artists, one of them Chaya Hazan. She had an idea for an installation, and she was looking for someone who could make the mechanism she had in it… And so, yeah, she called me and we talked and we got things rolling. It was almost a year in the process, just like thinking and working things out and figuring out what it will look like and how it will function. And then, yeah, then I start making the project at some point.

HS How much time do you spend engraving? It looks like a slow, deliberate process.

UT It very much depends on how intricate you want to make it; even then, it's not that straightforward. You'll see master engravers talk about something called background removal. Background removal is the least creative work… but it's also the most time-consuming, and that's why it makes the projects much more expensive. Background removal can quadruple your time spent on the engraving, or even more sometimes – it depends, because I do it by hand. So that background removal just takes days. But if it's just engraving with minimal or no background removal, just the scroll-work and the elaborate things like that, and if you already have the design printed and everything in place, a credit card-sized workpiece can take three or four hours, or even less if you're in the groove.

That's in theory: in practice, engraving always sucks in more time than you think. You need to stop to go to sharpen the tool, and to take a coffee-break and do something else, then you refine the design. Then, all of a sudden, three days have passed.

HS If you were just taking your first steps in brass engraving, where would you start?

UT I would say get your sharpening right, because if the tool is not sharp, you will struggle. It will feel impossible. But just a little bit of adjustment on the sharpening can be night and day. All of a sudden it's like the best thing ever.

So, if you don't manage to get the sharpening right, you won't understand what's happening and you're not going to have the confidence to know that it's just about the tool. And so, you need to eliminate problems and only concentrate on your own practice of engagement.

The second thing is a good vice helps a lot. After you get the hang of the strokes, you'll want to turn the device effortlessly. Even with a swivel vice, there are angles that you cannot reach, and as it turns around the pivot, a swivel vice takes the work away from you. And it will become a headache very, very quickly. Weirdly enough, the ball vice that I use is actually not the best thing for hand engraving. So I normally don't rush to recommend this for people who want to start engraving… one alternative is to buy a bowl, just a metal bowl and fill it with pitch. This is a kind of clay material that you can melt and then embed the workpiece in it. Usually you use it for embossing, to make three-dimensional shapes in metal. But for engraving, it's also fantastic; you can turn it and it's very solid. It's much cheaper than a ball vice for sure. I don't know why I haven't done that yet.

> "I would say get your sharpening right, because if the tool is not sharp, you will struggle"

HS And you've also been working on a guitar build. How did that come about?

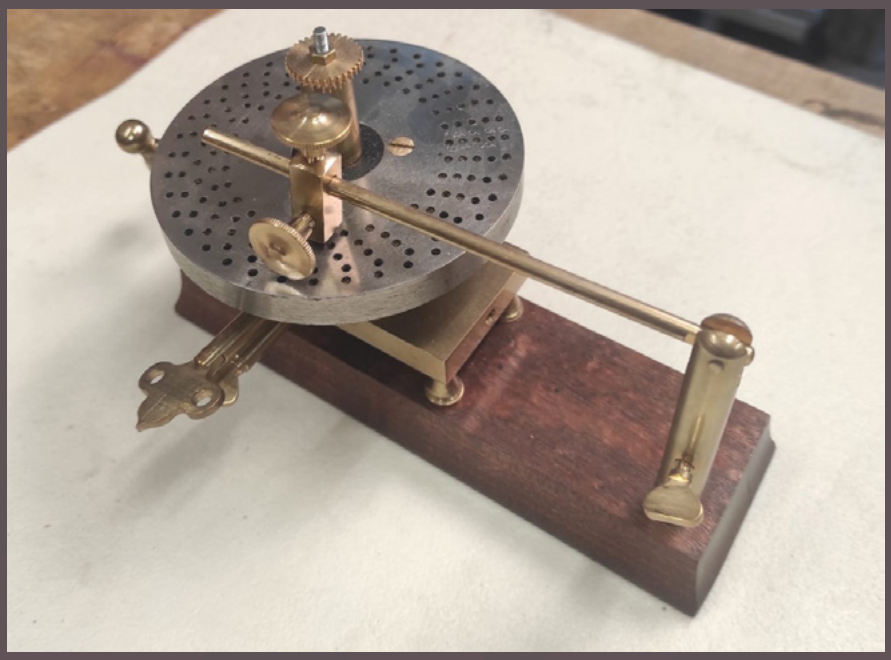
UT The square guitar? Ben from Crimson Guitars sent me an email out of the blue, asking me if I want to get involved in a guitar-building competition. He's a super-nice guy. It's been a lot of work, this guitar. I'm not a guitar maker.

HS I'm only just starting to make a guitar now. I'm finding it very liberating, because I know that the first one is going to be not very good, so it's taken away the fear of making mistakes. So far, the hardest thing has been getting hold of the right saw.

UT That's not a bad approach, but you need to be careful, because if you already tell yourself that it's not going to be great, I don't know how much motivation you'll have. You know what I mean? Maybe you'll not want to put as much effort as you should into the fine details, if you know that the product is not going to be the best thing. But I don't know, maybe that's how I feel at least when I do projects like that, I suppose. Right?

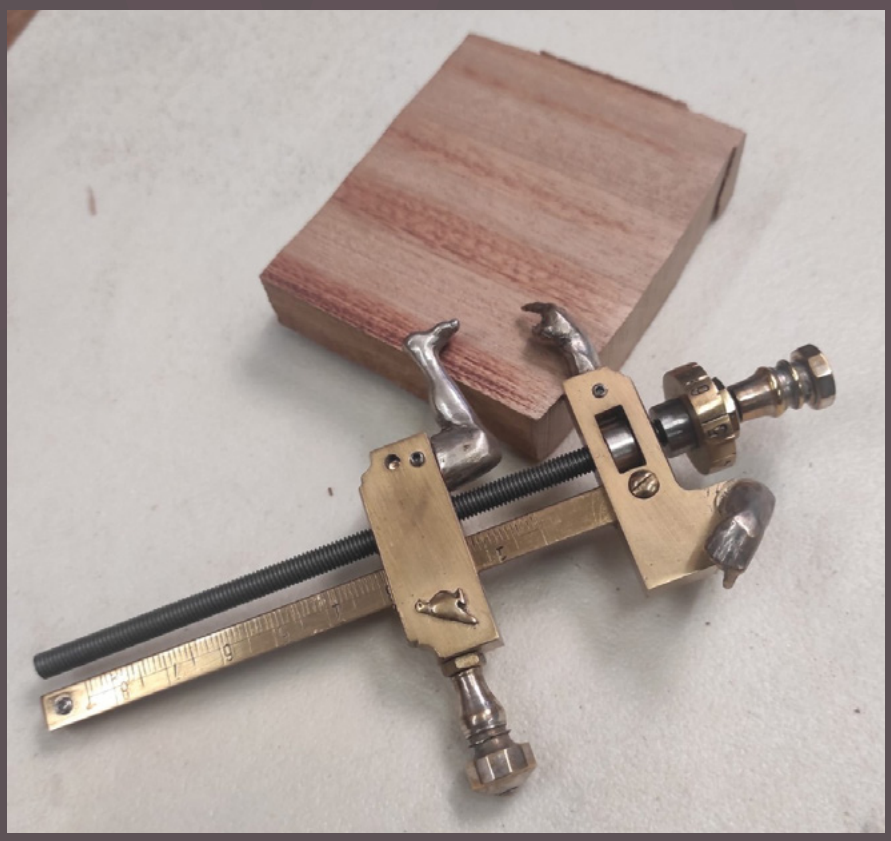
I visited this guitar maker, like top-of-the-line, amazing handmade, hand-carved guitars. They're amazing. And he has a wall covered with all of his tools. And it feels like none of the tools that he uses are standard tools. All of them are special, especially for guitars. So, yeah, it can [become] quite expensive really quickly.

I think the coolest thing he showed me was a set of files. Of course, you have to have funky files for guitars. And some cool chisels: chisels that have a very thin body, and that open out wide at the blade like a square. Weird tools are the best. □



Left
We strongly recommend Uri's YouTube channel. There's something wonderful about watching a proper artist at work

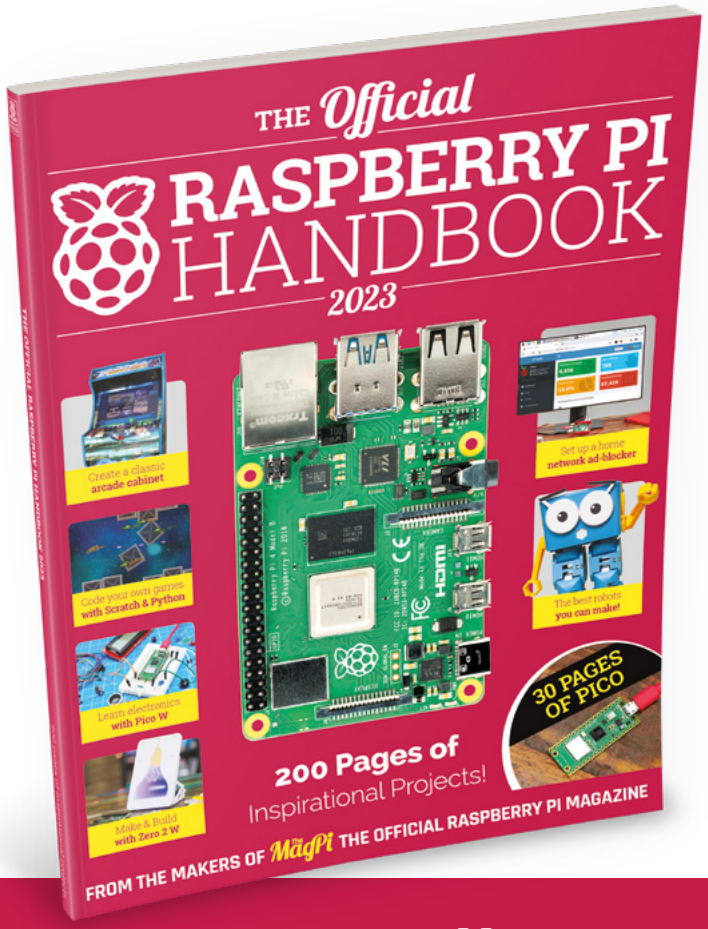
Below
Callipers, meet Monty Python



THE *Official* RASPBERRY PI HANDBOOK 2023

200 PAGES OF RASPBERRY PI

- QuickStart guide to setting up your Raspberry Pi computer
- Updated with Raspberry Pi Pico and all the latest kit
- The very best projects built by your Raspberry Pi community
- Discover incredible kit and tutorials for your projects



Buy online: magpi.cc/store

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

KiCad: using a PCB assembly service

Create KiCad projects that can be fully assembled by popular PCBA companies

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!

We've covered a lot in the previous three parts of this series. If you've worked through them all, you should be at a point where you can create simple board designs pretty well. In the next part of this series, we are going to look at a more complex board, building a minimal RP2040 board example. The RP2040 chip itself comes in a QFN-56 package, and whilst that can be soldered at home using reflow or hotplate soldering techniques, for many, that will be a challenge too far. To avoid this, we are going to use a PCB assembly (PCBA) service.

There is a lot to look at in making an RP2040-based board, so in this article, we will prepare a simpler design for manufacture to help us learn the PCBA approach. Designing for PCBA adds complexity in that we have to create numerous files, not only defining the PCB design, but also choosing and placing known components on the board. These components have to be available to the PCB assembly house, which again adds a little complexity. It's fair to say the first few times you do this process, it will seem like a lot of work compared to simply uploading your project file to OSH Park for it to create and send you a PCB!

So, for this exploration of PCBA services, we've laid out a small design in KiCad for a little H-bridge circuit prototype using four N-channel MOSFETs. We aren't going to step through the board design, particularly as we've covered the approach in earlier parts of the series. You can check out the project files here: **hsmag.cc/issue69.**

We're going to use the popular and reasonably affordable JLCPCB assembly service to manufacture and assemble our boards. This means that we need to consider what parts we are going to use on our board, as each part needs to be available in the JLCPCB

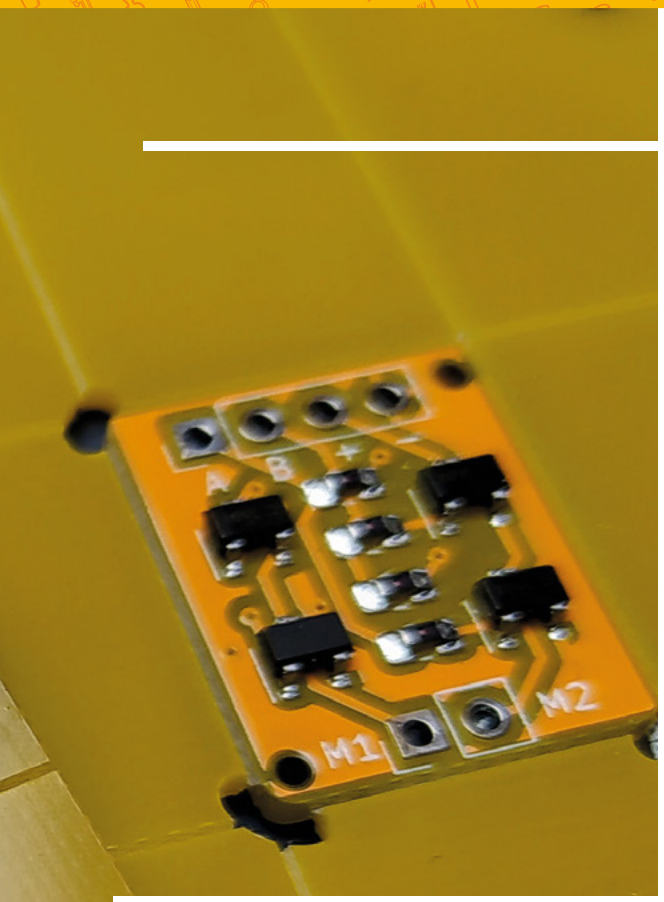


Figure 1
Our PCB design successfully assembled by the PCBA service

parts library. The first thing to do is to head over to JLCPCB (**jlcpcb.com**) and register for an account. You can explore the JLCPCB parts library using the search function and filters – you will see parts' cost and availability. In fact, you can also advance purchase components so that they are held in your own virtual warehouse ready to be used on your board designs in the future. One thing of note is that available components fall into two distinct groupings: 'basic parts' and 'extended parts'. Basic parts will be added

> These components have to be available to the **PCB assembly house, which again adds a little complexity**

to your board at the price listed. So, if you are adding five basic part resistors at 0.07 dollars each, then they will cost 0.35 per board. However, if that part was listed as an extended part, then that part will still cost the same unit price, but there will be a one-off $3 setup cost of including that part in your project, as the part will have to be manually retrieved from storage and loaded into the pick-and-place machines.

As you peruse the JLCPCB parts library, make sure that if you spot a component that you are likely to use, you make a note of the part number – these usually start with the letter 'C' and are listed on the main component landing page. For our small H-bridge board example, we are only interested in having →

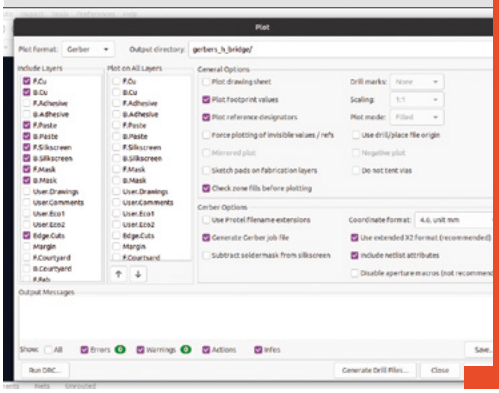
PCB ASSEMBLY

In earlier parts of this series, we used OSH Park to manufacture our PCBs – they make it super-easy by accepting KiCad PCB file upload directly. If you want to use other PCB fabrication services, or indeed as we are in this article, a PCBA service, it's much more likely you will need to plot Gerber files for your PCB and also files containing drilling information.

Gerber and drill files have some variables, and your fabrication service should give you some information regarding what they need in terms of Gerber files and drill files. For example, JLCPCB has a page (**hsmag.cc/gerberdrillkicad**) which outlines the settings required by the Gerber plotter in KiCad that their service needs. It's a little out of date in that the screenshots are from KiCad 5.19, but you can find all the same options on the KiCad 'Plot' dialog. To access the latter, you click File > Fabrication Outputs > Gerbers from the PCB Editor.

One thing of note is that plotting Gerbers creates a bunch of files for different layers of your PCB design, so make sure that at the top of the 'Plot' dialog, you create a folder for your Gerber files or else they all end up mixed into the main root folder of your project. You can also create the drill file from the 'Plot' dialog after you have plotted your Gerber files. Clicking the 'Generate drill files' button will launch a drill file dialog. For JLCPCB, place your drill file into the same folder as you did your Gerber files and, finally, compress this folder into a zip file for upload to JLCPCB.

Again, most PCB fabrication houses will have guidance on what format they require – JLCPCB list the settings needed on the link above. Don't worry too much if you upload something that doesn't work: it will show as an error and you can always ask the online chat service for help or guidance as to what settings to change.



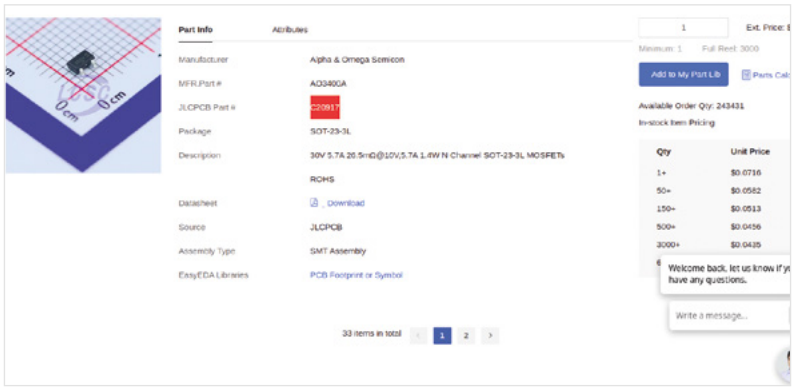


Figure 2 Searching for parts in the JLCPCB part library

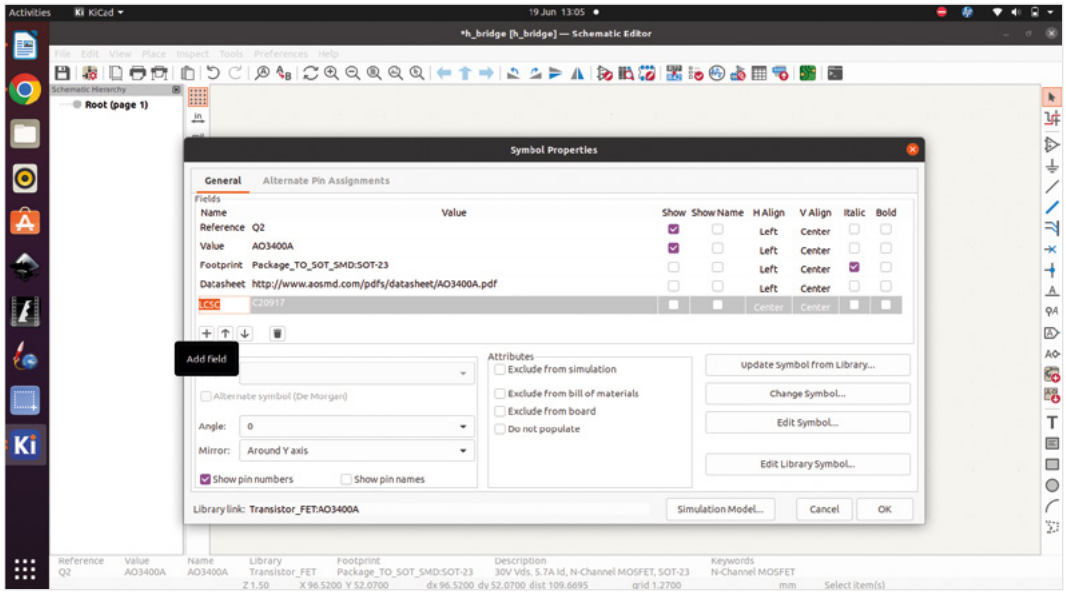
Figure 3 Adding an extra LCSC field to Symbol Properties enables a correct BOM to be created

JLCPCB add the SMD components, so we chose some 10K resistors (part number C49122), and the four MOSFETs are going to be AO3400 chips (part number C20917) (**Figure 2**). We need to add these details to an extra field added to the Symbol Properties so that later, when we generate a bill of materials (BOM), these specific components will be identified. In order for JLCPCB to ignore the through-hole components in our design, we've simply not added this extra LCSC field to their schematic symbols, and therefore they won't be included in the assembly process later on.

> You need to do this for every component you **expect JLCPCB to add to your board**

To add these details, in the Schematic Editor, highlight a component and press the **E** key to open the Symbol Properties dialog. Click the + button, which is labelled 'Add field' when you hover over it (**Figure 3**). You should see a new field line appear. In the 'Name' column, we need to label this field 'LCSC', and then in the 'Value' column, we need to add the CXXXXX number we researched for that component. For small projects, such as this example, you can do this manually for each schematic symbol. You need to do this for every component you expect JLCPCB to add to your board; you can't just add this field to one 10K resistor and expect it to work out to add the same part for the others. An alternate approach for larger projects is that you can edit the symbol at the library level, or copy the symbol to a custom library with the LCSC field and number populated at the symbol level. This means that whenever you place that custom symbol in the schematic, you have the correct LCSC part number ready for the BOM. Before creating the BOM, you still have to assign the footprints to the symbols as you would for any PCB design.

KiCad has a tool to generate a BOM – the tool icon is accessible from the Schematic Editor and shows 'Generate a bill of materials from the current schematic' when you hover over it. If you click this tool, it will open the Bill of Materials dialog. On the left-hand side of the dialog, you'll see an area titled 'BOM generator scripts'. There are some scripts already installed, but there is an excellent script written by arturo182 which specifically creates a JLCPCB-formatted BOM. To use



QUICK TIP

We covered creating and editing custom schematic symbols and libraries in the third part of this series.

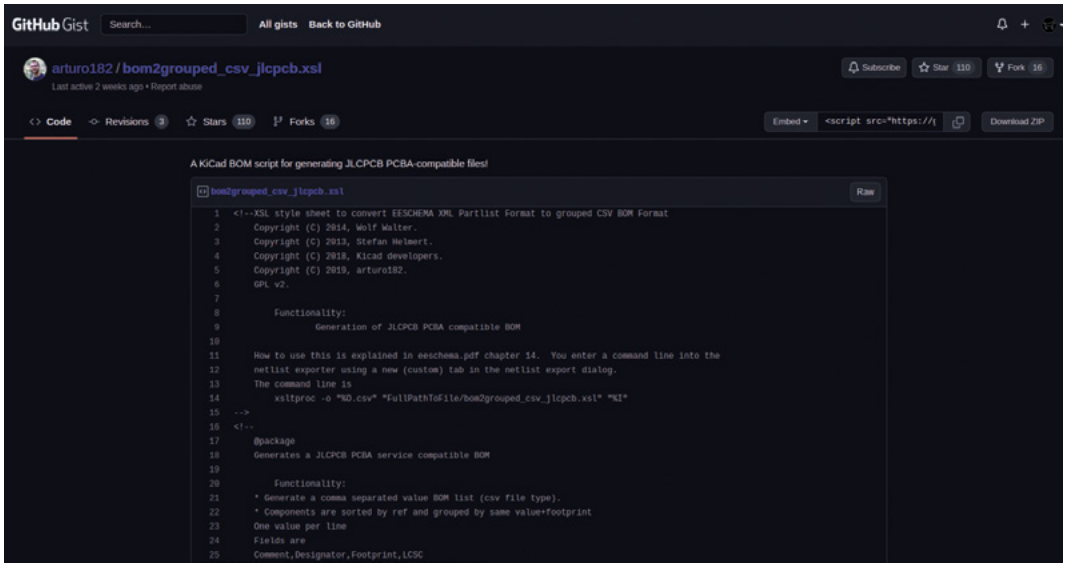


Figure 4 Downloading the custom JLCPCB BOM generator script

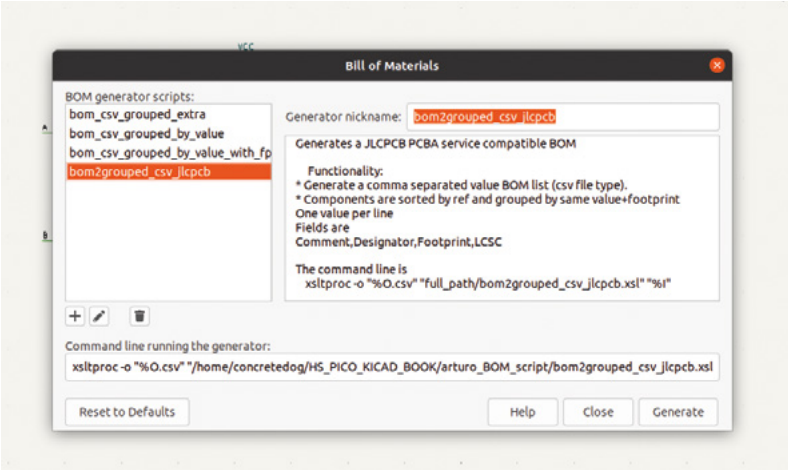


Figure 5 The BOM dialog with the custom JLCPCB generator script added and selected

this script, go to the GitHub repository, then in the upper right-hand corner, you will see a Download ZIP button. Click it to download, then extract the zip file (**Figure 4**). In the Bill of Materials dialog, click the + button and navigate into the folder you just unzipped and select the **bom2grouped_csv_jlcpcb.xsl** file. Clicking the OK button will add this handy script to the list.

BOM SQUAD

Once your board is complete and you have all the LCSC numbers attached to the schematic symbols, you can reopen the BOM dialog. Select the 'bom2grouped_csv_jlcpcb' script, then click Generate to create a BOM that the JLCPCB service will be able to use (**Figure 5**).

The final piece of the PCBA puzzle is to generate a footprint position file, also referred to as a centroid file. Similar to the BOM file, this is essentially a spreadsheet which contains details of each placed component, showing both coordinates and the rotational angle of the part. To generate this, in the PCB Editor, we need to select File > Fabrication Outputs > Component Placement. Make sure that your settings match the dialog box shown in **Figure 6**. Note that if your project contains through-hole components that you want JLCPCB to include, you need to uncheck the 'Include only SMD footprints' option and include the LCSC numbers for those through-hole parts in the BOM. Note that for this component footprint POS file and the BOM file we generated earlier, we don't want those files inside the zip file of Gerbers, as they are uploaded separately. Once you are ready, click the Generate Position File button. Notice that this will generate two POS files: one for the upper layer and one for the lower layer. As our design is single-sided, we are only interested in – and later, need to upload – the upper layer file. With the upper layer POS file generated, we need to make a few alterations to the spreadsheet column header titles for it to work properly for JLCPCB. Open the file you have generated in your spreadsheet program. We use LibreOffice Calc, but MS Excel or Google Docs →

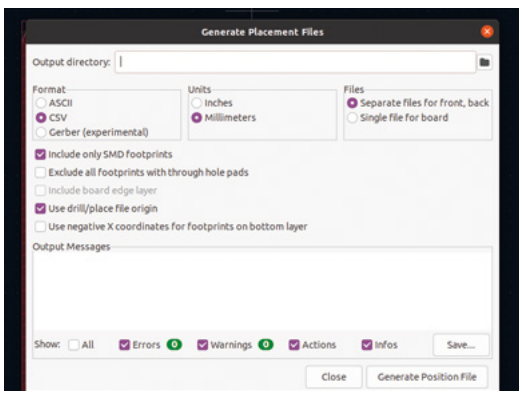


Figure 6 The Generate Placement Files dialog

Figure 7
Using LibreCalc to edit the generated positional file column titles

File Edit View Insert Format Styles Sheet Data Tools Window Help

Liberation Sa 10

A1 Designator

	A	B	C	D	E	F	G	H
1	Designator	Val	Package	Mid X	Mid Y	Rotation	Layer	
2	Q1	AO3400A	SOT-23	96.078	-50.1165	90	top	
3	Q2	AO3400A	SOT-23	104.4	-50.3705	90	top	
4	Q3	AO3400A	SOT-23	96.078	-55.5475	90	top	
5	Q4	AO3400A	SOT-23	104.328	-55.5475	90	top	
6	R1	10k	R_0805_2012Metric	100.4335	-49.022	0	top	
7	R2	10k	R_0805_2012Metric	100.4335	-53.848	0	top	
8	R3	10k	R_0805_2012Metric	100.4335	-56.134	180	top	
9	R4	10k	R_0805_2012Metric	100.4335	-51.308	180	top	
10								
11								

should work. We need to make the following changes: the title of the first column, Ref, should be changed to Designator; PosX and PosY should be changed to Mid X and Mid Y; Rot should be changed to Rotation, and finally, Side should be changed to Layer (**Figure 7**). Save the POS file with these alterations – we now have everything we need to upload to the JLCPCB service.

With everything ready, we can now head to the JLCPCB website. Click on the Standard PCB/PCBA tab to upload our Gerber zip file. After a short upload, you should see a render of the upper and lower sides of your board in the preview window (**Figure 8**). You can make changes to the board type, material, thickness, and more on this initial page. However, apart from changing the colour of the board to yellow, we left everything at the default setting.

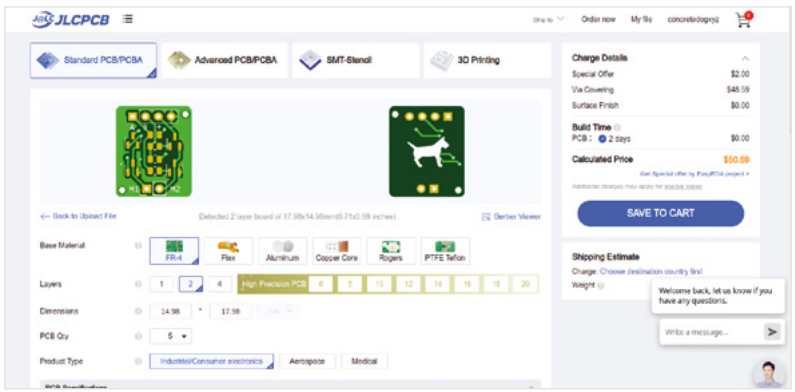
> You should see a render of the upper and lower sides **of your board in the preview window**

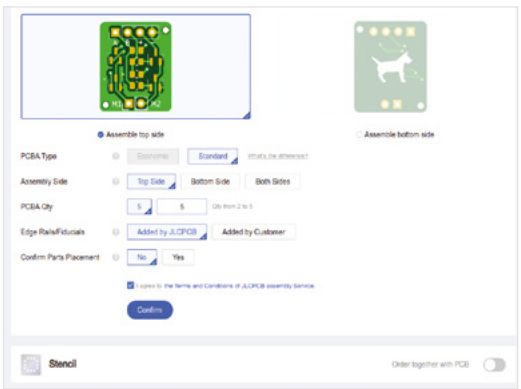
At the bottom of the page, you can click a button to add/expand the PCBA services. In **Figure 9**, you can see that we have opted to have the top side assembled only (as we only have components on this side). We've also ensured that the Edge Rails/Fiducials entry has the Added by JLCPCB option highlighted. This indicates that for our very small boards, JLCPCB services will create any needed panel layouts. With all that selected, click Next.

You'll get another larger preview of the PCB layout generated from the Gerber uploads. Check it carefully and then click Next to move to the next tab. This will look like **Figure 10**. It's reasonably self-explanatory. Click the Add BOM File button and upload the BOM file we created earlier, then click the Add CPL File button and upload the top layer CSV positional file we made and edited earlier. Clicking

Figure 9
The preliminary choices for the PCBA service

Figure 8
If the Gerber zip file uploads correctly, you'll be rewarded with the first of many preview images of your board





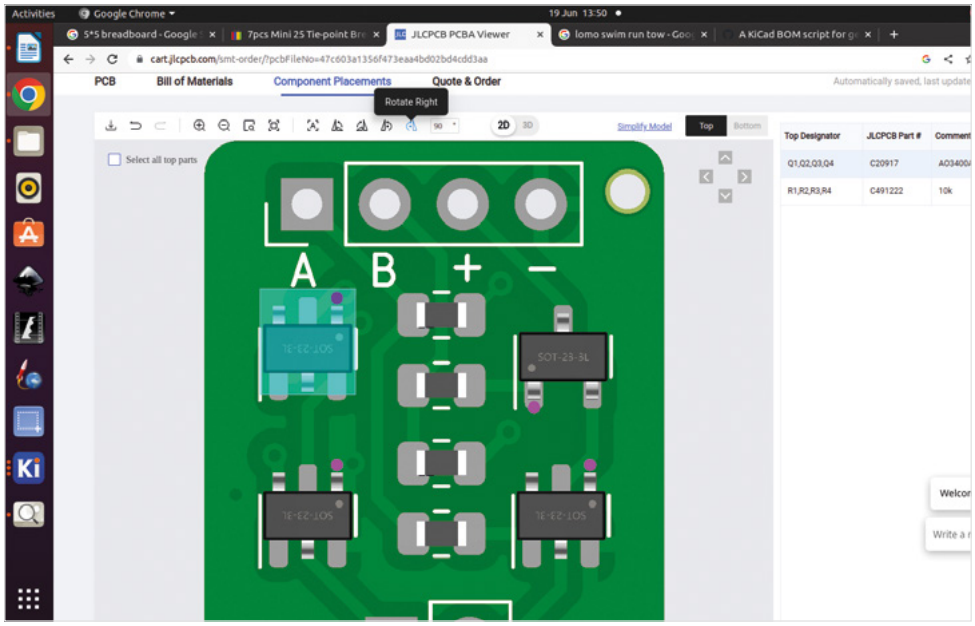


Figure 11
Correcting footprint rotational position can be done in-browser as part of the JLCPCB order process, or you can edit your positional file offline to create correct values

Next, these will be uploaded and processed, which may take a few minutes, and you should see a render with the board and the components placed.

SPIN ME ROUND

Often, at this point, you will find that components are not rotated correctly on the footprints. There are two ways to correct this, if your components are at standard angles, you can left-click to highlight a component in the render image and, when highlighted, use the rotation tools above the PCB render in the JLCPCB web page. You can continue to do this until your design looks correct. Clicking Next will save these orientations and take you to the Add to Basket ordering page. Whilst this is a fine approach, another approach is to open the positional file we created and uploaded offline and edit the rotational value of the components that have appeared incorrectly in the render. In our experience, either way is fine, and the JLCPCB engineers will question if a component isn't sitting on a footprint correctly.

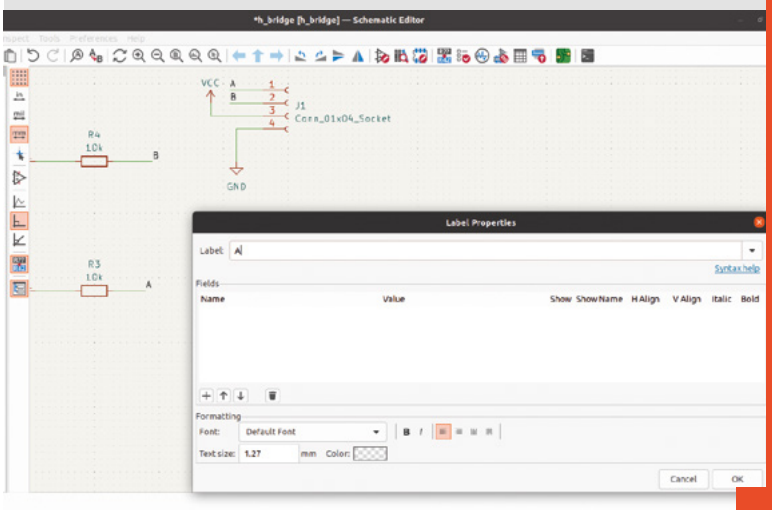
All that's left to do is to add the order to your shopping basket and pay! Once the order is confirmed and paid, you get regular updates on the order listing and, if it's a complex design, it's worth checking back into your account four to six working hours after placing the order to check the DFM analysis regarding component placing in the order history details.

Once everything is ordered, all you have to do is wait! However, not for that long! We started this simple H-bridge motor driver design in KiCad and had the assembled PCBs (**Figure 1**) in our hand eight days later. □

Figure 10
Adding the BOM files and the positional file we made earlier

CASTING A NET

In the schematic for our H-bridge circuit, you'll notice that we haven't directly wired everything together and that the connections between the gate pins on each MOSFET are labelled 'A' or 'B'. In turn, there are also two pins on the connector marked 'J1' labelled 'A' and 'B'. These are 'net labels' and, to make one, you simply click the 'Add a net label (L)' tool icon. You then left-click in the schematic and the 'Label Properties' dialog will appear. Into this you simply type the name of your net label, so, for example, type the letter 'A'. You can change the font and size, but when you are happy, click the OK button and you can now place your 'A' label into the schematic. Notice that it has a small square connector. You can then connect a wire to the 'A' label as you would to any other component using the 'Add a wire' tool. You can recreate another 'A' label using the same method to attach to the other end of your wireless connection, or you can copy and paste the original net label. If you have more complex descriptive net label names, it can be a good idea to use copy and paste as if you create a net label with a spelling mistake, it will not connect and may take you a while to discover the error. In this simple H-bridge example, we didn't really need to use this technique but, in the next article with a more complex design, using net labels can really help to keep a schematic cleaner and more readable.



Laser-cutting PLA

Turn old 3D prints and support material into boxes, jewellery, and other items

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.

Below
It can be challenging to get to pieces with balanced colours

We looked, last month, at how to recycle PLA from old 3D prints and support materials into sheet material. The resulting sheet is about 1 to 3 mm thick. In the previous issue, we also looked at slump-moulding this to create bowls and, in this article, we'll start with those same recycled sheets and look at how to laser-cut it.

PLA has some slightly unusual properties that make it great for 3D printing; the main one being that it softens at a very low temperature. When laser-cutting, this means that the bit that the laser hits obviously vaporises (as happens with other laser-cuttable plastics).

However, the heat from this spreads out and softens much more plastic than you would usually expect. It's quite common to find that the part is still soft when you've finished cutting. The make-up of the laser bed is really important. If there's not enough support, you may find that your parts sag under their own weight once they soften from the heat. You might also find that some of the softened and liquidised PLA sticks to the print bed and can make it tricky to get small parts off. A bit of a bash usually gets things unstuck, but the more delicate the part, the more of a problem this is likely to be. Sharp corners tend to get a bit rounded off as it just melts away – the same goes for thin branches. If you're particularly good with your laser control software, it might be possible to plan the job so that cuts on either side of a hole or branch are done at different times, to give the parts more time to cool down, but this isn't something that we've tested.

We've also had problems where small parts were cut out and then blown slightly by the air-assist, and rewelded themselves to the main piece. Most of the time, we've had success prying them out afterwards but, again, the smaller and more delicate the part, the harder it is. On the whole, we've had very few problems with parts (or holes) over about 2 mm × 2 mm, and things get a bit tricky once you go smaller than this. That said, with a bit of post-processing, we've had some much smaller parts come out fine.

> "It might be possible to plan the job so that cuts on either **side of a hole or branch are done at different times**"

The final part that makes things a bit tricky is that the sheets we've made aren't perfectly uniform, and can vary in width by about 50%. With care, you can minimise this but, unless you upgrade to a purpose-built sheet press, you're probably not going to be able to eliminate this. Obviously, this means that you need your power and speed settings calibrated for

FLAME-POLISHING

Flame-polishing is the process of heating up the part quickly so the outer surface melts, then cooling it down. This process is very effective on PLA because of its semi-crystalline structure. You can end up with a very glossy surface that really brings out the colours.

We've had good results using an electronics head gun with a fine nozzle. Set the temperature to about 260°C and low to moderate airflow. The aim is to just melt the outer surface, so you don't want to be in any one place for long. Keep moving and keep checking on the shininess. More melt will give you more shininess, but also means more bubbles and risk of deformation (see below). However, it does have some downsides:

- The chances are that you've got some bubbles trapped under the surface of your part. If you heat it up, these bubbles expand and can pop in the liquid PLA. If you manage the temperature well, this often isn't a big problem. On parts with lots of detail, the bubbles often don't really show up. On parts that have larger smooth surfaces, the bubbles can be really noticeable.
- Thin branches can go soft and get blown around. You can poke them back a bit if they don't get stuck. This is really noticeable on geometric designs, and less obvious on more organic designs.

This means that you need to think about the appropriate finishing method at the design stage. If you're hoping to flame-polish your work, make sure these two issues won't be problems for you.

the thickest part of the material. This means you're overpowering some of the cuts.

There are some potential benefits to how PLA reacts to the heat as well. When cutting finger joints, we found that if we took the parts straight off the print bed and fitted them together, there was enough softness from the heat to both help tight-fitting parts get together, and then weld themselves slightly together, ensuring a really solid part.

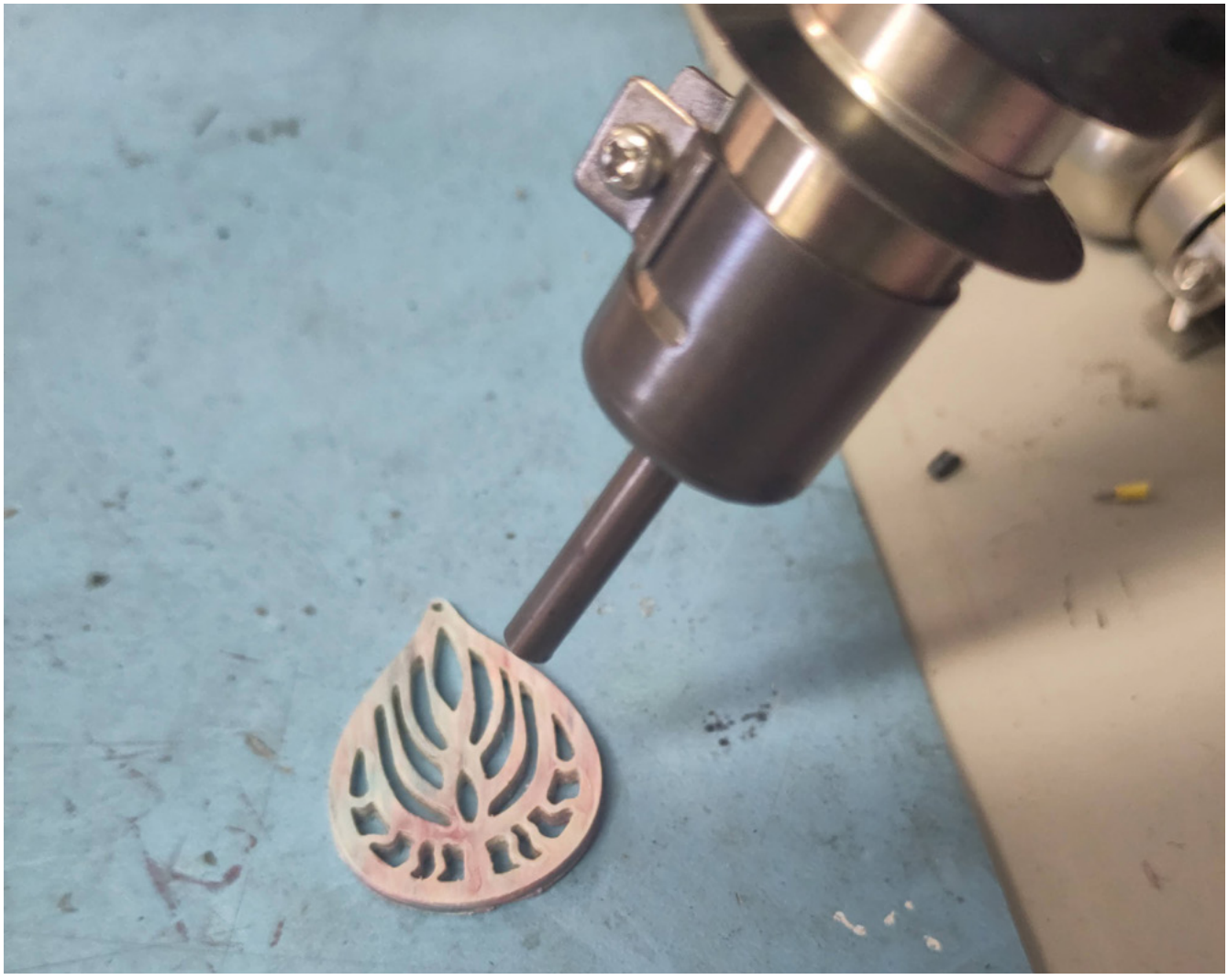
JEWELLERY MAKING

We really like the look of the swirls of colour you get in the recycled PLA sheets, so we tested the process out by making jewellery.

The first problem is that with this method of pressing sheets, you end up with an irregularly shaped sheet. We'd recommend the method mentioned last month – of scooping up the molten plastic, twisting it into a blob, and then re-pressing it. Not only does this mean the colours swirl together much more pleasingly, but it also means that you end up with a circular sheet. Most CAD programs are designed to work with rectangular sheets, but at least a circle is sufficiently regular to be easy to work with. Use a tape measure to find the smallest distance across, then draw a circle of this diameter in Inkscape (or your CAD program of choice), and whatever fits in this circle, should fit on the sheet. You can delete the circle before cutting.

As we've mentioned, there are a few aspects that make small parts tricky to cut. We hedged our bets on this by cutting more than we needed. When making earrings, for example, we cut three of each shape and picked our favourite two. This →

Above
Just add a jump-ring and an off-the-shelf earring hook to turn your pieces into jewellery



Above
A quick blast with hot air can give a shiny finish

DESIGNS

Getting the right designs is key to getting good results. We've found two approaches that work well. The first is simple designs with areas to show off the pattern in the plastic. These can look great with a matt surface and minimal post-processing.

The second is organic designs with narrow, but not too narrow (around 1.5 to 3mm wide), branching features. These can look great flame-polished.

Obviously, it depends on your personal taste, but these are the general areas that we've had the most success with.

Unfortunately, there doesn't seem to be a good repository of open-source designs for laser cutting in the same way that there is with 3D printing. There are many sellers on Etsy selling laser-cuttable templates for jewellery of varying quality (and we don't know of any way of actually verifying if a seller actually has a licence to sell the designs that they're selling). Alternatively, you can design your own.

also helps because the pattern is quite complex, so it can be hard to place designs to get two earrings with equally balanced colours. By cutting three, you have a better chance of finding two that work well together. This might sound wasteful, but don't forget that any you don't use can just be put back in the recycling mix for the next sheet.

Once you've got your parts off the cutter, the next challenge is the finishing. The laser leaves quite a lot of residue. Some of this is smoke and some of this is molten plastic that's been blown away from the cut line. This is exacerbated by the fact that PLA can have two very different finishes, depending on how it cools. Sometimes you'll find it ends up matt, sometimes really shiny. We won't go into the technical details, but you often end up with a shiny outline next to the cut and a matt finish elsewhere. This can leave it looking a bit untidy.

How big a problem this is depends on exactly how the part has come off and what it'll be used for.

Above
Different patterns can give very different effects

Let's examine a couple of examples of how to get parts looking good. The first star pendant design is quite large and has space to show off a lot of the pattern from the plastic. Some negative space adds an opportunity to give it a bit more detail.

The relatively simple design of this means that there aren't many lasered edges, so not much needs removing. A quick sanding down with 180 grit sandpaper removes almost all the imperfections. There are still some slight blemishes on the final piece if you look closely enough, but it's pretty hard to make these out.

> **By cutting three, you have a better chance of** finding two that work well together

The second piece is more organic-looking. A lot of details have been cut with the laser, which means that it needs more work. We start with a heavy sand with 60 grit paper. This removes most of the waste material, but leaves it rough and with a lot of burrs in the details that are hard to remove. We go through the grits to 180 grit paper. This removes the worst of it, but leaves it looking very muted. A flame-polish (see box) brings out the colours with minimal bubbling, but some parts warp a little.

Laser-cutting recycled PLA is a great technique that can lead to really interesting-looking designs that are also kind to the environment. However, it's not a universal technique that can be used to make anything look great. You need to think about your design and take the time to post-process if you really want to end up with something that looks great.

Above
Larger, flat surfaces show off more of the pattern, but also more defects

SAFETY

As far as we can tell, PLA is a relatively safe plastic to laser-cut. The big caveat here is fumes. While PLA isn't particularly bad for fumes, generally you don't want to breathe in any fumes from laser cutting, so an enclosed laser cutter with an extraction system is very strongly recommended. The second caveat is that, while we believe there aren't any particular risks associated with laser-cutting PLA, it's not something that we've been able to get a lot of information on.

For us, working with PLA and an enclosed laser cutter with fume extraction fits within our risk tolerance, but it is experimental and, before working on this, you need to decide for yourself if it's acceptable for you.

Part 03

Raspberry Pi Pico inputs and outputs

Create a buzzer game whilst learning about how to use inputs and outputs on Raspberry Pi Pico

MAKER

Stewart Watkiss

Also known as Penguin Tutor. Maker and YouTuber that loves all things Raspberry Pi and Pico. Author of *Learn Electronics with Raspberry Pi.*

penguintutor.com

twitter.com/stewartwatkiss

You'll Need

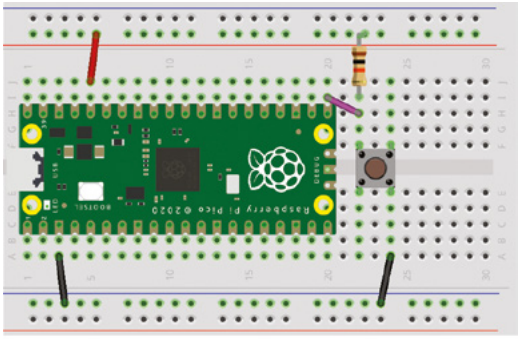
- 6V buzzer **magpi.cc/buzzer6v**
- Arcade buttons with 5V LED **magpi.cc/arcadebuttons**
- 2N7000 MOSFET **magpi.cc/2n7000**
- 470Ω and 10kΩ resistors **magpi.cc/575resistorkit**
- PCB switch **magpi.cc/tactileswitchbuttons**
- ULN2803 integrated circuit **magpi.cc/uln2803**
- Various wires, including crimp connectors

Inputs and outputs are the ways that a Raspberry Pi Pico can interact with the outside world. In this tutorial, you will learn about pull-up and pull-down inputs and how to connect arcade buttons to a Raspberry Pi Pico. You will also learn about driving a buzzer using a MOSFET transistor.

You will then learn how you can swap the MOSFET for an integrated circuit to create an interactive game to play with friends.

01 Switches as inputs

The digital GPIO pins on Pico can have two different values. One is a high input where the input is at, or near, 3.3V. The other is a low value when the input is at, or near, 0V. What about if the pin isn't connected to anything? You may think that without any input then it would be low, but that isn't necessarily the case. Instead, the input is treated as 'floating'. With a floating input, it can pick up stray signals from passing radio signals or static electricity resulting in it sometimes being high and sometimes low. To avoid this, we need to use pull-up or pull-down resistors.



▲ **Figure 2** Pull-up example circuit showing a pull-up resistor and switch connecting to ground

02 Pull-up and pull-down resistors

Pull-up and pull-down resistors are a way to set a default value on the input pin. If it's pull-up, that means that with no other input the value will be high; with pull-down, it means that the default value will be low.

To change the value, you need to apply a stronger input that overrides the pull-up or pull-down resistor. For a pull-up, you provide a connection to ground; for pull-down, you provide a connection to 3.3V.

For example, connect a 10kΩ resistor between a GPIO pin and the 3.3V source on pin 36. This sets the input high. Connect a button switch from the same pin down to ground. Whenever you press the button, the lower resistance in the switch results in the voltage being dropped across the resistor and no voltage across the button providing a low input. This is shown in the **Figure 2** wiring diagram.

03 Pull-up and pull-down inside Pico

For most circumstances, you don't need to use external resistors and you can instead use internal resistors within the Pico. This is achieved by using the pull argument when you define the pin.

To enable a pull-up resistor, use:

```
button1 = Pin(16, Pin.IN, Pin.PULL_UP)
```

The third parameter enables a pull-up resistor for that pin. You can now remove the external resistor.

The same can be done for pull down by using the value `Pin.PULL_DOWN`.

To detect if the button has been pressed, use the following line:

```
button1.value()
```

This will return 1 if the input is high (button not pressed) and 0 if the input is low (button pressed).

04 Handling outputs

Using the GPIO ports as an output gives a different problem. The GPIO pins can only provide around 12mA from each of the pins. This is enough to pass a signal on to another device and can also be used to light up a standard LED, but it cannot switch large currents needed by a buzzer, high-power LED, or many other devices.

A common way of switching larger loads is with a MOSFET transistor.

05 N-channel MOSFET

A MOSFET is a semiconductor device which is a type of transistor. It is like a switch which is controlled by an electrical signal instead of pushing it down. The MOSFET has three pins: a drain and source, which is where the main current will flow, and a third pin known as the gate which controls whether the MOSFET is turned on or off. For the N-channel MOSFET, if a positive voltage is applied to the gate then current can flow from the drain to the source.

MOSFETS can be used to switch bigger loads than integrated circuits are capable of. An example of a MOSFET is a 2N7000, which can switch up to 200mA. See the schematic symbol and pin layout in **Figure 3**.

06 Making noise with a buzzer

The buzzer used is a self-contained miniature buzzer designed for 5 or 6V. This type of buzzer is the easiest to use and will make a buzzing noise whenever a voltage is applied to it.

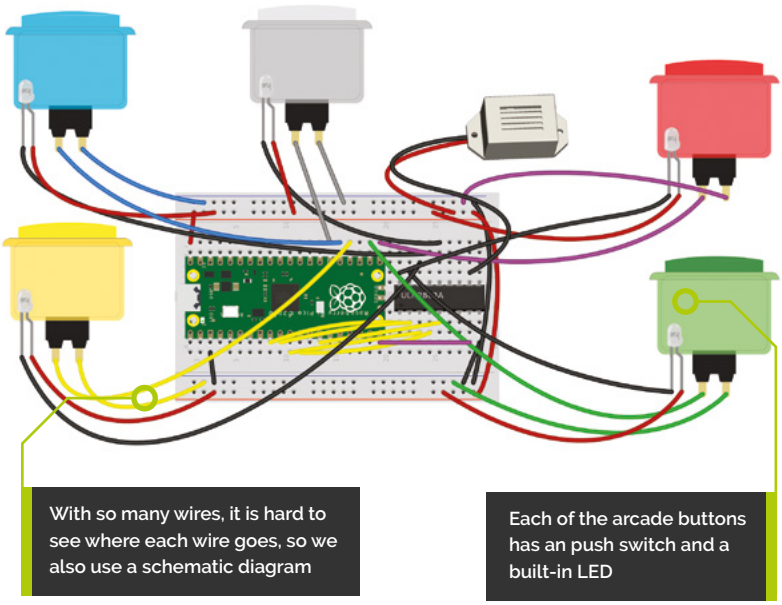


Figure 1 The wiring diagram for the quiz buzzers system

Some other types of buzzers, such as a piezo buzzer, need to have a rapidly changing signal, which would need to be included in the code.

The buzzer uses 25mA, which is too much for Pico's GPIO pins, but within the range for the 2N7000 MOSFET. It also needs to be powered from a 5V power supply. This can be taken from the VBUS connection on pin 40 of your Pico, which is connected to the 5V connection on the USB connector.

Top Tip

MOSFET pinouts

Different MOSFETS can have different pinouts. Check the datasheet for your MOSFET for more details.

07 Wiring up the MOSFET

Different MOSFETs can have different pinouts. The 2N7000 is in a TO-92 package. With the flat part of the MOSFET at the front, the source is on the left (connected to ground), the gate is in the middle (connected to the output from the GPIO pin), and the drain is on the right (connected to the buzzer's negative terminal).

The breadboard layout for the circuit can be seen in **Figure 4**. A resistor is used between the

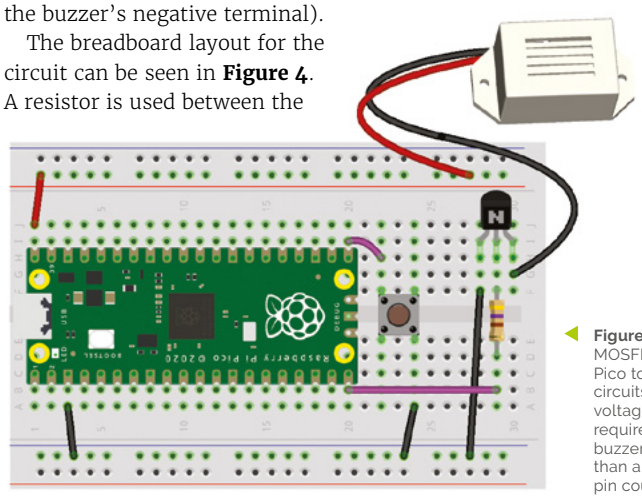


Figure 4 The MOSFET allows Pico to control circuits with a higher voltage or current requirement. The buzzer needs more than a standard GPIO pin could provide

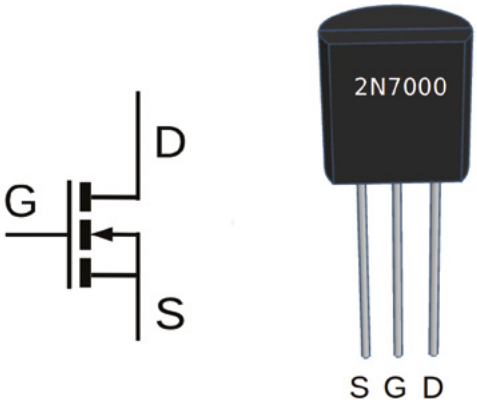


Figure 3 Circuit symbol and pin layout for a 2N7000 MOSFET. The gate acts like an electronic version of the button on a switch

GPIO pin and the gate of the MOSFET. The reason for this is that when a MOSFET is first turned on, there can be an initial inrush current flowing into the gate. Once the MOSFET is fully turned on, the current will be very small. A 470Ω resistor reduces the inrush current.

08 Making some noise

With the wiring complete, the buzzer can be activated by setting the output of the GPIO pin high. A simple `while` loop can be used which detects if the button is pressed:

```
if button1.value() == 0:
```

And if so, then turn the GPIO output pin high.

```
buzzer = Pin(15, Pin.OUT)
buzzer.value(1)
```

09 Using an integrated circuit

To create the game, you need to add arcade buttons with built-in LEDs. Many arcade buttons are designed for 12V, but to use with Pico's power, you need ones designed for 5V.

The button LEDs could be controlled through a MOSFET as described above. This would need five MOSFETs and five resistors to control all the LEDs. This would also take up a lot of space on the breadboard. An alternative is to use an integrated circuit that combines these into a single chip. A suitable integrated circuit is the ULN2803. This is actually a Darlington driver chip, which is based around a bipolar transistor – see the schematic in **Figure 5**. The ULN2803 has eight Darlington drivers (as shown in **Figure 6**) and can also be used to replace the MOSFET for the buzzer.

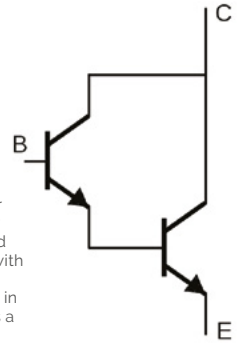


Figure 5 The Darlington driver uses two bipolar transistors paired together. Used with a digital circuit, they can utilised in the same way as a single MOSFET

10 Connecting the buttons

These arcade buttons have four pins, two of which are for the switch and two are for the LED (see the datasheet or supplier information to identify which is which). The buttons and their LEDs have spade connectors. These can be connected using pre-made wires, or you can create your own using a crimp tool (see **magpi.cc/crimp**).

For the arcade buttons, the switch pins should be connected between the GPIO pin and ground. For the LED, connect the +5V terminal to the 5V power supply and the ground terminal to the appropriate pin on the ULN2803.

The wiring is shown in **Figure 1**, but the number of crossing wires makes it hard to see. A better diagram is the schematic in **Figure 7**.

11 Programming the arcade buttons

The outputs for the LEDs are defined in the same way that the buzzer is. Rather than creating five separate variables for the inputs and five for the LEDs these have been combined into lists.

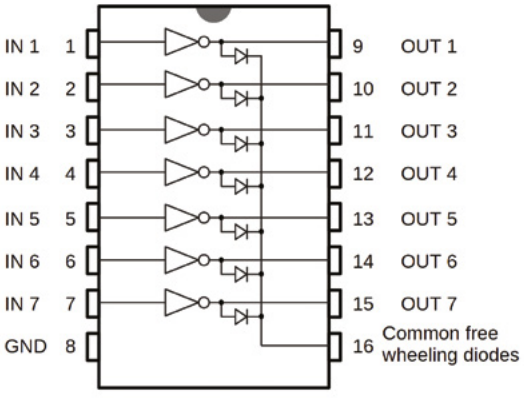


Figure 6 The ULN2803 has eight Darlington drivers, which share a common ground. These can be used for eight separate devices. The large triangle symbols represent each Darlington driver

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**

Top Tip

Darlington driver

The Darlington driver uses two transistors connected together. This gives a very high gain, allowing a much higher output current compared to the input.

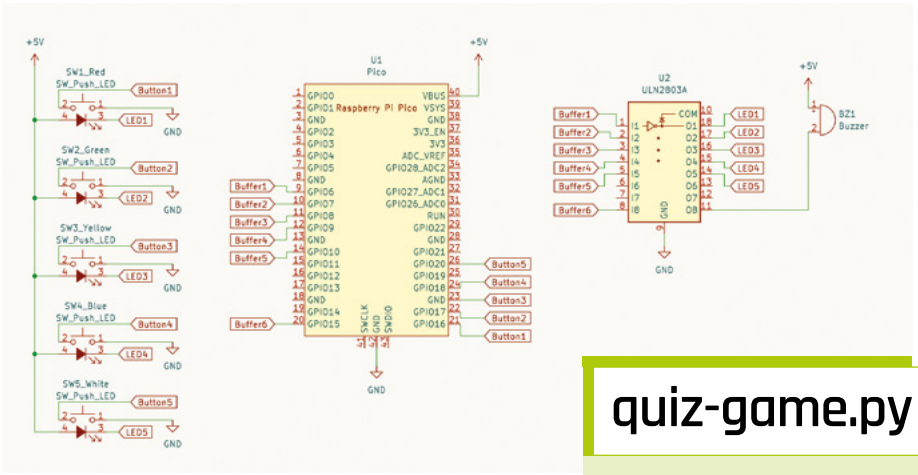


Figure 7 The schematic diagram makes it easier to see which components are wired together. To avoid crossing lines, labels have been used; labels with the same name are connected together

A `for` loop is used to go through each of the pin numbers and assign them to the buttons list for the inputs and the `leds` list for the outputs.

The code provided allows for the buttons to be used in a 'finger on the buzzer' quiz game. Each of the buttons' elements are checked and, if a button is pressed, then the LED is turned on and the buzzer sounds.

12 Finishing the game

You can play the game with the players holding the switches in your hand, or you can create an enclosure to hold the circuit and mount the switches. If you have a 3D printer, then you can download the STL file from **magpi.cc/penguinquizgame**. If not, you can create your own by drilling appropriate holes into a plastic storage box.

There are different games that could be played by changing the code. The code included here is for a quiz game buzzer. In addition, the GitHub repository also includes a reaction game.

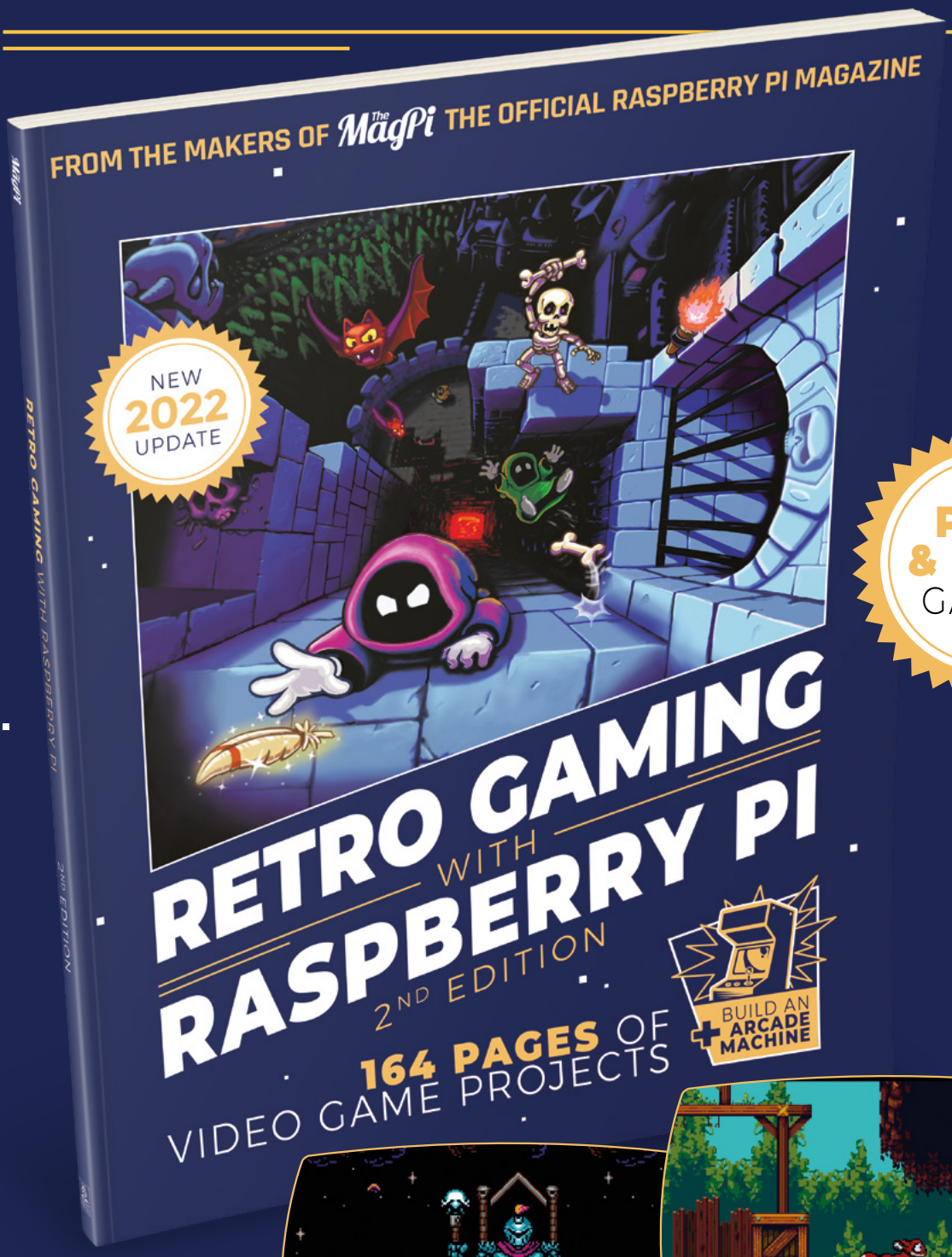
Figure 8 The finished game can be installed into an enclosure. This one is 3D-printed, but you could just use a plastic storage box and drill holes for the buttons

quiz-game.py

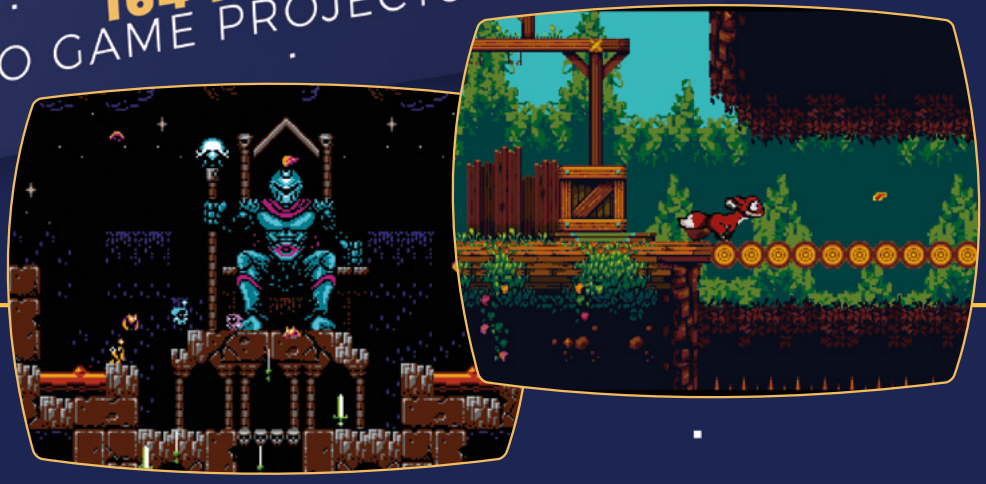
DOWNLOAD THE FULL CODE:
magpi.cc/arcadebuttonsgit

Language: **MicroPython**

```
## Quiz game
# Detects the first to press their button
# Sounds the buzzer for 1 second and lights the
# button pressed for additional 2 seconds
from machine import Pin
import utime

buzzer = Pin(15, Pin.OUT)
# Red, Green, Yellow, Blue, White
button_pins = [16, 17, 18, 19, 20]
led_pins = [6, 7, 8, 9, 10]
buttons = []
leds = []

for i in range (0, len(button_pins)):
    buttons.append (Pin(button_pins[i], Pin.IN,
Pin.PULL_UP))
    leds.append (Pin(led_pins[i], Pin.OUT))

while (1):
    button_pressed = False
    # check for which buttons pressed
    for i in range (0, len(buttons)):
        if buttons[i].value() == 0:
            print ("Button {} pressed".format(i))
            leds[i].value(1)
            button_pressed = True
        else:
            leds[i].value(0)
    if button_pressed:
        # Sound buzzer for 1 second
        buzzer.value(1)
        utime.sleep(1)
        buzzer.value(0)
        # Leave LEDs on for 2 more seconds
        utime.sleep(2)
        # Reset all buttons
        for i in range (0, len(buttons)):
            buttons[i].value(0)
```



PLAY & CODE GAMES!



RETRO GAMING WITH RASPBERRY PI

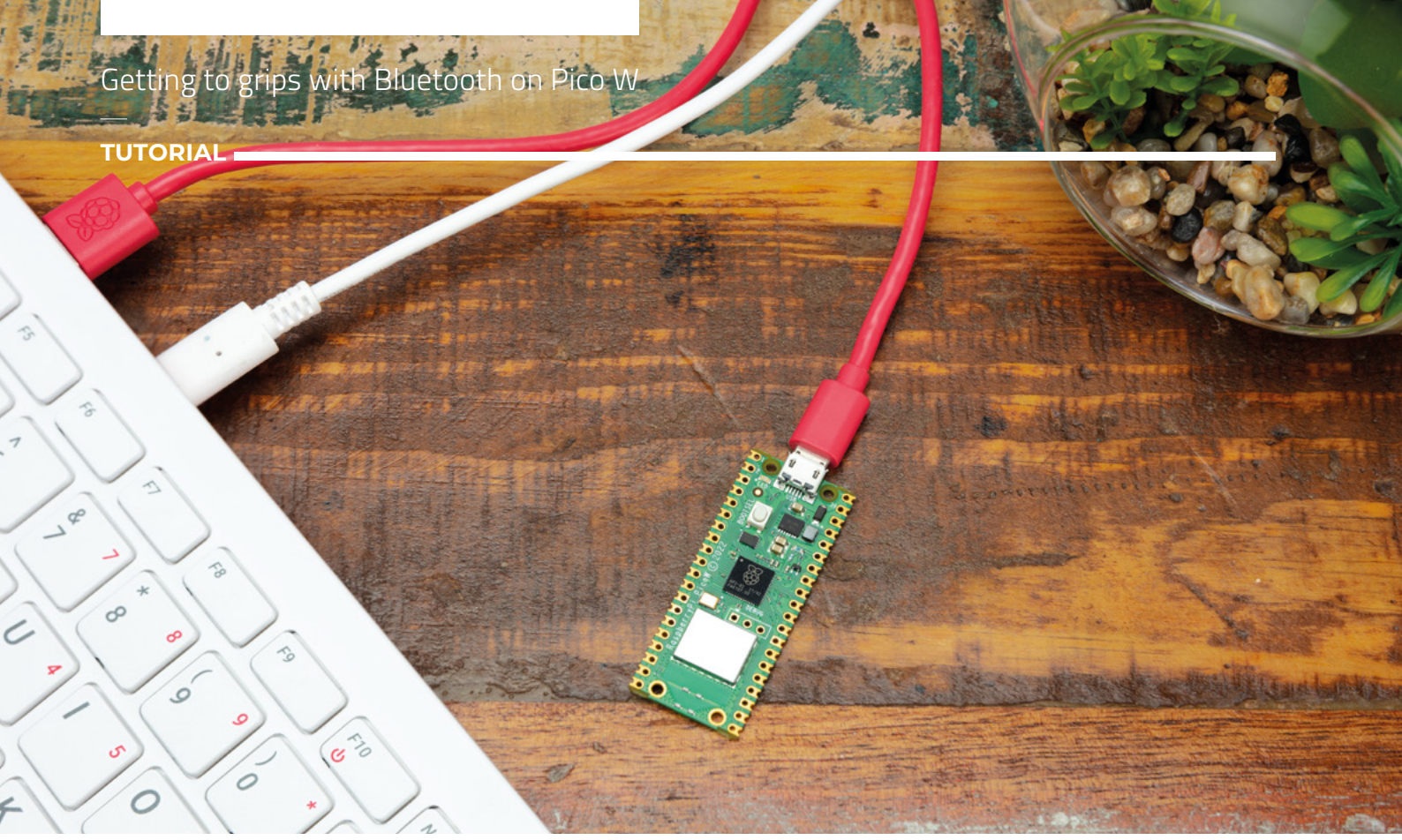
2ND EDITION

Retro Gaming with Raspberry Pi shows you how to set up a Raspberry Pi to play classic games. Build your own games console or full-size arcade cabinet, install emulation software and download classic arcade games with our step-by-step guides. Want to make games? Learn how to code your own with Python and Pygame Zero.

- *Set up Raspberry Pi for retro gaming*
- *Emulate classic computers and consoles*
- *Learn to code your own retro-style games*
- *Build a console, handheld, and full-size arcade machine*

BUY ONLINE: **magpi.cc/store**

TUTORIAL



Getting to grips with Bluetooth on Pico W

Communicate wirelessly easily and without wasting power

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.

Pico W now supports Bluetooth both with MicroPython and C. But what is this, and why should you care?

The chances are you've used Bluetooth before, be it a keyboard, headphones, or sensor of some sort. It sends data back and forth via wireless radio signals. It typically has a range of a few metres to maybe tens of metres.

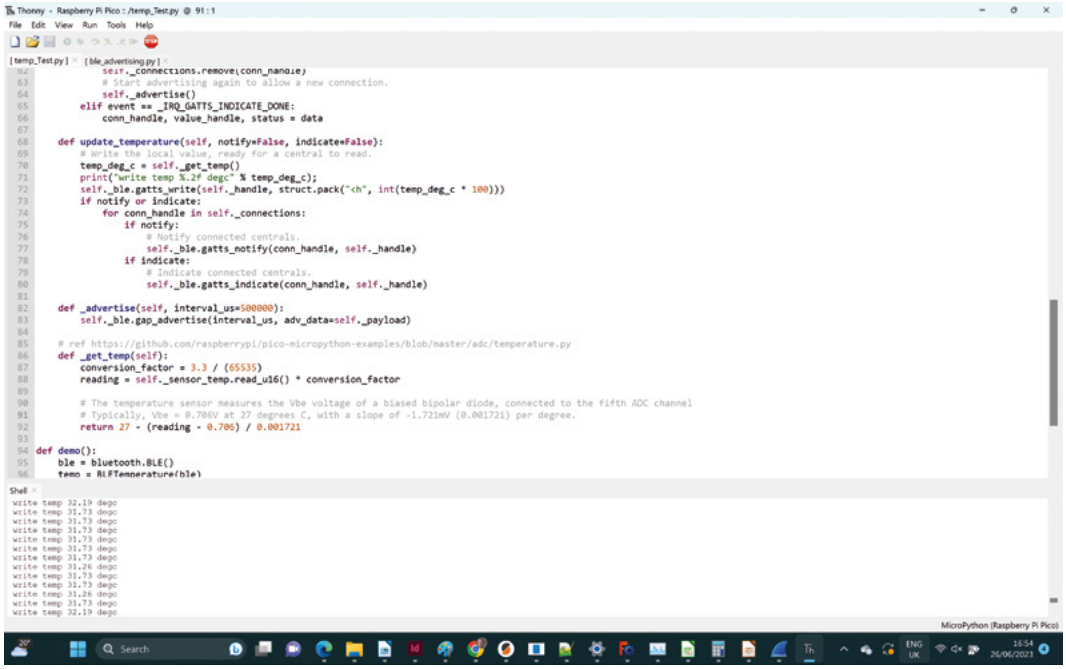
When you first start a Bluetooth device, you usually want it to start broadcasting some data. Often, this is for pairing, but it can also send a small amount of data to any other device in range. This is done using the Generic Access Profile (GAP).

GAP defines two roles: central and peripheral. Central devices are typically phones or computers, and they receive data from peripherals which tend to be sensors. Pico W can be either a central or a peripheral device.

You can simply continue sending data using the GAP, however, it only allows one-way communication, and each payload can only contain 31 bytes of data. You can send data both ways, gain more security, and generally get more features by connecting with a Generic Attribute Profile (GATT). The GATT defines the services and characteristics. In BLE terminology, a characteristic is a piece of data, and a service is a collection of characteristics. To use services, a device has to have a GATT that defines which services and characteristics they offer. These GATTs are predefined – you can find a list of them at **hsmag.cc/bluetooth_specs**.

If you want to create a Bluetooth peripheral, the first thing you need to do is decide what GATT you want to use.

One crucial part of this is that the receiving software has to be expecting the type of data that your device is sending. For example, if you



Left
The temperature data from the Pico W on-board sensor is often a few degrees too hot

have a Bluetooth UART app on your phone, that won't be able to communicate with a Bluetooth temperature sensor.

Profiles, services, and characteristics are all identified using Universally Unique Identifiers (UUIDs). A full list of all the numbers assigned to different things in Bluetooth is documented here: **hsmag.cc/bluetooth_assigned**.

Now we know a little about what's going on, let's take a look at an example. Note, you'll also need to save the **ble_advertising.py** program to your Pico (under the same name).

We'll use the example **pico_ble_temperature_sensor.py** from the **pico-micropython-examples** GitHub Repository (**hsmag.cc/pico_ble_ex**).

In order to use this, you'll need to flash the latest version of MicroPython to your Pico and you will need to save both the **ble_advertising.py** and **pico_ble_temperature_sensor.py** programs to Pico.

Here's the code from the **pico_ble_temperature_sensor.py** program, which we'll dissect to find out what's going on.

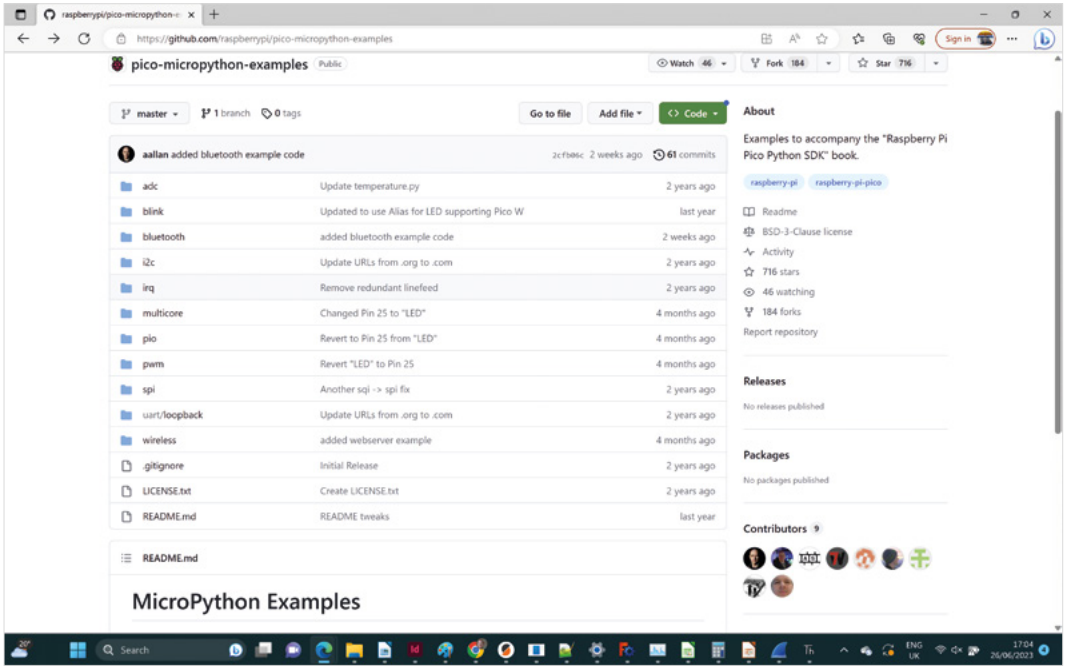
```
# This example demonstrates a simple temperature
sensor peripheral.
#
# The sensor's local value is updated, and it
will notify
# any connected central every 10 seconds.

import bluetooth
import random
import struct
import time
import machine
import ubinascii
from ble_advertising import advertising_payload
from micropython import const
from machine import Pin

_IRQ_CENTRAL_CONNECT = const(1)
_IRQ_CENTRAL_DISCONNECT = const(2)
_IRQ_GATTS_INDICATE_DONE = const(20)

_FLAG_READ = const(0x0002)
_FLAG_NOTIFY = const(0x0010)
_FLAG_INDICATE = const(0x0020)
```

CUSTOM GATT PROFILES

In this article, we've looked at using a predefined GATT. For a lot of purposes, this is fine. There are loads of predefined profiles out there, and using a standard one means your code will work with other code.

However, you might reach a point where there's just no GATT that will do the job. For this, you need to create your own profile. In Bluetooth-speak, this is a vendor-specific profile.

The main thing you need to do is make sure you use a 128-bit UUID for your IDs. You can use uuidgen or an online tool to generate these for you. Because they're so large, the chance of two bits of software accidentally using the same UUIDs is low.



Right
The official MicroPython examples guide you through using the hardware

```
# org.bluetooth.service.environmental_sensing
_ENV_SENSE_UUID = bluetooth.UUID(0x181A)
# org.bluetooth.characteristic.temperature
_TEMP_CHAR = (
    bluetooth.UUID(0x2A6E),
    _FLAG_READ | _FLAG_NOTIFY | _FLAG_INDICATE,
)
_ENV_SENSE_SERVICE = (
    _ENV_SENSE_UUID,
    (_TEMP_CHAR,),
)

# org.bluetooth.characteristic.gap.appearance.xml
_ADV_APPEARANCE_GENERIC_THERMOMETER = const(768)

class BLETemperature:
    def __init__(self, ble, name=""):
        self._sensor_temp = machine.ADC(4)
        self._ble = ble
        self._ble.active(True)
        self._ble.irq(self._irq)
        ((self._handle,),) = self._ble.gatts_
register_services((_ENV_SENSE_SERVICE,))
        self._connections = set()
        if len(name) == 0:
            name = 'Pico %s' % ubinascii.
hexlify(self._ble.config('mac')[1],':').decode().
upper()
        print('Sensor name %s' % name)
        self._payload = advertising_payload(
            name=name, services=[_ENV_SENSE_UUID]
        )
        self._advertise()

    def _irq(self, event, data):
        # Track connections so we can send
notifications.
        if event == _IRQ_CENTRAL_CONNECT:
            conn_handle, _, _ = data
            self._connections.add(conn_handle)
        elif event == _IRQ_CENTRAL_DISCONNECT:
            conn_handle, _, _ = data
            self._connections.remove(conn_handle)
            # Start advertising again to allow a
new connection.
            self._advertise()
        elif event == _IRQ_GATTS_INDICATE_DONE:
            conn_handle, value_handle, status =
data
```

WHAT IS IN A NAME?

Bluetooth gets its name from the medieval King of Denmark and (briefly) Norway, Harald 'Bluetooth' Gormsson. The translation Bluetooth isn't quite accurate. He was actually called Blátǫnn, and blár in Old Norse means dark blue or black.

For those of a more British persuasion, Harald Bluetooth was the great-grandfather of King Harold II (who died at the Battle of Hastings). Probably – disentangling 1000-year-old family trees is a little tricky.

Harald Bluetooth consolidated the Kingdom of Denmark and brought the warring tribes together in much the same way that the protocol could bring incompatible wireless devices together. Bluetooth wasn't intended to be the name of the protocol – it started out as the codename for the project. However, no more suitable name came up, so Bluetooth stuck.

The Bluetooth logo merges the runes analogous to the modern Latin alphabet letters 'h' and 'b'; ᚼ (Hagall) and ᛒ (Berkanan) together. It's Harald Bluetooth's initials.

FORGET TCP/IP

If you've worked using TCP/IP or HTTP or similar technology to send data between computers before, it might be tempting to think that things will work similarly with Bluetooth. It doesn't. It's so different that any comparison is probably unhelpful. The best advice we can give you is to forget everything you know about traditional networking when working with Bluetooth. They're completely different beasts that work at different levels to achieve different goals.

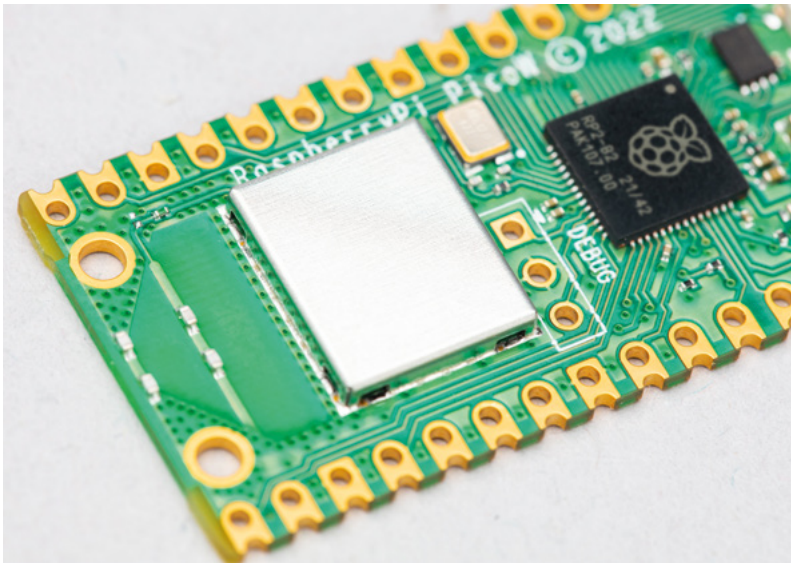
```
    def update_temperature(self, notify=False,
indicate=False):
        # Write the local value, ready for a
central to read.
        temp_deg_c = self._get_temp()
        print("write temp %.2f degc" % temp_
deg_c);
        self._ble.gatts_write(self._handle,
struct.pack("<h", int(temp_deg_c * 100)))
        if notify or indicate:
            for conn_handle in self._connections:
                if notify:
                    # Notify connected centrals.
                    self._ble.gatts_notify(conn_
handle, self._handle)
                if indicate:
                    # Indicate connected
centrals.
                    self._ble.gatts_
indicate(conn_handle, self._handle)

    def _advertise(self, interval_us=500000):
        self._ble.gap_advertise(interval_us, adv_
data=self._payload)

    # ref https://github.com/raspberrypi/pico-
micropython-examples/blob/master/adc/temperature.
py
    def _get_temp(self):
        conversion_factor = 3.3 / (65535)
        reading = self._sensor_temp.read_u16() *
conversion_factor

        # The temperature sensor measures the Vbe
voltage of a biased bipolar diode, connected to
the fifth ADC channel
        # Typically, Vbe = 0.706V at 27 degrees
C, with a slope of -1.721mV (0.001721) per
degree.
        return 27 - (reading - 0.706) / 0.001721
```

```
def demo():
    ble = bluetooth.BLE()
    temp = BLETemperature(ble)
    counter = 0
    led = Pin('LED', Pin.OUT)
    while True:
        if counter % 10 == 0:
            temp.update_temperature(notify=True,
indicate=False)
        led.toggle()
        time.sleep_ms(1000)
        counter += 1

if __name__ == "__main__":
    demo()
```

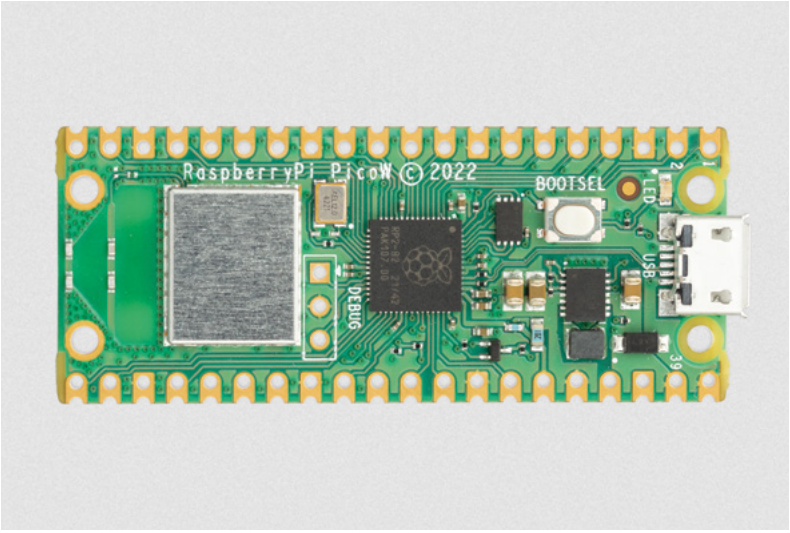
We'll look at the code in a bit more detail shortly, but let's first set up a computer to receive the data.

Thanks to modern web browsers' ability to interact with BLE through the Web Bluetooth interface, you don't need to install anything to get the data. There's example code for getting the Temperature characteristic from the Environmental Sensing service at **hsmag.cc/bluetooth_temp**. Click on 'Start Notification' and you should see a box pop up listing the available Bluetooth devices. Select the one starting 'Pico' and you should (after a few seconds) see some data start to appear.

It might look a bit cryptic – there'll be two sections starting with 0x. The number is a little-endian integer (the 0x at the start indicates that it's being shown in hexadecimal format), so you need to convert it from the hex string shown. →



Below
The metal enclosure holds in the wireless magic



Above
Pico W is an almost drop-in replacement for Pico but with Wireless LAN and Bluetooth

Remove both 0x figures, then paste the other digits into a converter such as the following: **hsmag.cc/hex_to_float**. Divide the result by 100, and you have the temperature of the Pico.

At this point, you could be entirely justified in wondering exactly how you are supposed to know that the number is a 2-bit little-endian integer. Fortunately, like most things Bluetooth related, it's all in the documentation. In this case, it's in the GATT Specification Supplement available from **hsmag.cc/gatt_ss**. Bluetooth is, in general, well-documented, but it's a complex set of protocols with a lot of options, so finding the things you need in the massive pile of available documents can be a challenge. We'll try to guide you to the appropriate places.

If this all seems a bit convoluted, it's because Bluetooth really works best when it has specific code for both sending and receiving, but we're using some generic code for receiving data. Since Pico can be a central device as well as a peripheral, there's code for using a second Pico to receive the data at **hsmag.cc/pico_bluetooth_reader**, but we won't look at that in detail in this article.

DIGGING INTO DETAIL

Now that we can read the temperature, let's go back and take a look at how this all works.

The first part of this code defines the identifiers we're using:

```
_FLAG_READ = const(0x0002)
_FLAG_NOTIFY = const(0x0010)
_FLAG_INDICATE = const(0x0020)

# org.bluetooth.service.environmental_sensing
_ENV_SENSE_UUID = bluetooth.UUID(0x181A)
# org.bluetooth.characteristic.temperature
_TEMP_CHAR = (
    bluetooth.UUID(0x2A6E),
    _FLAG_READ | _FLAG_NOTIFY | _FLAG_INDICATE,
)
_ENV_SENSE_SERVICE = (
    _ENV_SENSE_UUID,
    (_TEMP_CHAR,),
)

# org.bluetooth.characteristic.gap.appearance.xml
_ADV_APPEARANCE_GENERIC_THERMOMETER = const(768)
```

Most of this is building up the `_ENV_SENSE_SERVICE` data, which is used as an input for `gatts_register_services`. This method is documented at **hsmag.cc/gatts_register_service**. The key part of this is the single parameter which is (according to the documentation) "a list of services, where each service is a two-element tuple containing a UUID and a list of characteristics. Each characteristic is a two- or three-element tuple containing a UUID, a flags value, and, optionally, a list of descriptors."

The available flags are then listed as:

```
_FLAG_BROADCAST = const(0x0001)
_FLAG_READ = const(0x0002)
_FLAG_WRITE_NO_RESPONSE = const(0x0004)
_FLAG_WRITE = const(0x0008)
_FLAG_NOTIFY = const(0x0010)
_FLAG_INDICATE = const(0x0020)
_FLAG_AUTHENTICATED_SIGNED_WRITE = const(0x0040)

_FLAG_AUX_WRITE = const(0x0100)
_FLAG_READ_ENCRYPTED = const(0x0200)
_FLAG_READ_AUTHENTICATED = const(0x0400)
_FLAG_READ_AUTHORIZED = const(0x0800)
_FLAG_WRITE_ENCRYPTED = const(0x1000)
_FLAG_WRITE_AUTHENTICATED = const(0x2000)
_FLAG_WRITE_AUTHORIZED = const(0x4000)
```

In this case, you can see that `_ENV_SENSE_SERVICE` is a two-element tuple containing first `_ENV_SENSE_UUID`. We've previyously defined this as 0x181A, which is listed in the **hsmag.cc/bluetooth_assigned** document as Environmental Sensing Service. In the same document (section 6.1), it lists the range of allowable characteristics for this service. We can pick and choose any of these for our particular implementation, and we've selected temperature (defined in the same document as 0x2A6E).

The final set of constants:

```
_IRQ_CENTRAL_CONNECT = const(1)
_IRQ_CENTRAL_DISCONNECT = const(2)
_IRQ_GATTS_INDICATE_DONE = const(20)
```

are event codes from the MicroPython Bluetooth module. They are detailed here: **hsmag.cc/mp_ble_events**.

That's the basic data you need to create a Bluetooth temperature controller. Let's now take a more detailed look at the code.

Let's work backwards from the `demo` method that's kicked off when we run the script. This creates a `BLETemperature` object called `temp`. By creating it, this kicks off the `__init__` method which sets everything up.

```
        self._ble.irq(self._irq)
        ((self._handle,),) = self._ble.gatts_
register_services((_ENV_SENSE_SERVICE,))
        self._connections = set()
        if len(name) == 0:
            name = 'Pico %s' % ubinascii.
hexlify(self._ble.config('mac')[1],':').decode().
upper()
        print('Sensor name %s' % name)
        self._payload = advertising_payload(
            name=name, services=[_ENV_SENSE_UUID]
        )
        self._advertise()
```

The first line here tells the Bluetooth module to call the object's `_irq` method when any event happens. This lets us handle things such as connections, disconnections, and if a central device has responded to us sending data with `indicate`.

After this, it sets the relevant data for the Bluetooth module and finally calls `_advertise` which itself just runs:

```
self._ble.gap_advertise(interval_us, adv_
data=self._payload)
```

This obviously starts advertising. It's this that makes the device available for pairing. When you tried to read the temperature from your web browser, you would have seen a pop-up with the available devices. This, in essence, just means the devices that are currently advertising.

Once we've started this, we can get back to our `demo` method. From this point on, we can kind of ignore most of the Bluetooth stuff – it doesn't really bother us. Advertising and pairing all happen in the background. We loop through and update the temperature using the method in the class:

```
    def update_temperature(self, notify=False,
indicate=False):
        # Write the local value, ready for a
central to read.
        temp_deg_c = self._get_temp()
        print("write temp %.2f degc" % temp_deg_c);
        self._ble.gatts_write(self._handle,
struct.pack("<h", int(temp_deg_c * 100)))
        if notify or indicate:
            for conn_handle in self._connections:
                if notify:
                    # Notify connected centrals.
                    self._ble.gatts_notify(conn_
handle, self._handle)
                if indicate:
                    # Indicate connected centrals.
                    self._ble.gatts_
indicate(conn_handle, self._handle)
```

The first part of this is just getting the data in the right format, which, as we've looked at previously, is two little-endian 2-bit numbers that, when combined together, give the temperature in 100ths of a degree Celsius.

The format string `"<h"` is little-endian, 2-bit signed integers – the first character is the endianness, and the second is the number format (you can see a full list of other options at **hsmag.cc/python_struct**).

Sending the data is done in two parts. First, we write to the handle (we got the handle when we initialised the service in the `__init__` method), and then either `notify` or `indicate`. There's no difference between the two at this point, but if we did `indicate`, there would then be an event when the central confirmed it had received the data (see box). This code uses `notify` (it's set in the `demo` method), but would work equally well using `indicate`. □

CLASSIC AND BLE

When we use the word Bluetooth, we're talking about an umbrella term for a range of technologies. These technologies are usually broken down into Bluetooth Classic and Bluetooth Low Energy (BLE). A lot can be said about this, but the short version is: Classic is mostly used for audio, and BLE is everything else. Some more modern headphones do support BLE, and more may in the future.

This is different from the version number, since Bluetooth 4 onwards includes specifications for both Classic and BLE.

While Pico W can use both Classic and BLE, this article (and probably everything else you read about Bluetooth on Pico W) refers to BLE.

16 bits of Retro

Emulate an Atari ST or another computer from inside

Dr Andrew Lewis

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

Like the Commodore Amiga, the Atari ST was an iconic home computer, offering many people their first glimpse of the 16-bit digital world. After almost 40 years, it's still an engaging design, although the internal hardware may need a few repairs and upgrades to fit in with the modern standards. In this article, you'll see how to replace the internal electronics of an Atari-ST with a Raspberry Pi, while retaining the original keyboard. None of the changes performed require modifications to the original case or to the circuitry of the Atari, making the process completely reversible in the future.

Simulating a classic system on new hardware is nothing new, and after 40+ years of technological advancement, emulating home computers, classic mainframes, and even full-blown arcade machines is well within the reach of a small computer like the Raspberry Pi. What desktop emulation doesn't have is the classic 'feel' of the original machine. A modern keyboard and joystick don't have the same tactile impact as a classic machine, and if you want to recapture that feeling but have access to modern system tools, then you'll need to consider repurposing an existing machine.

TALK TO THE KEYS

The classic Atari ST case with the built-in keyboard and parallelogram F key design is one of those features you can't emulate in software, but it isn't too difficult to interface original Atari hardware with a modern microcontroller, and you don't even need to solder any wires to do so. Remove the screws from the base of your Atari ST, and remove the lid. Remove the motherboard and all internal components, keeping the keyboard and outer case to one side. This isn't a difficult process, and there are no hidden screws or clips to worry about. You can safely store all of the components that you don't need so that the whole process is reversible.

The original Atari keyboard has its own microcontroller built in to process key presses.

Depending on your model of Atari, the keyboard will be connected to the motherboard by a single connector with either 8 or 18 connections. On earlier Atari machines, the 18-pin connector was used, routing joystick and mouse connections through the keyboard's 6301 processor. On later machines like the 1040ST, only the keyboard signals were routed through the connector, so fewer pins were required.

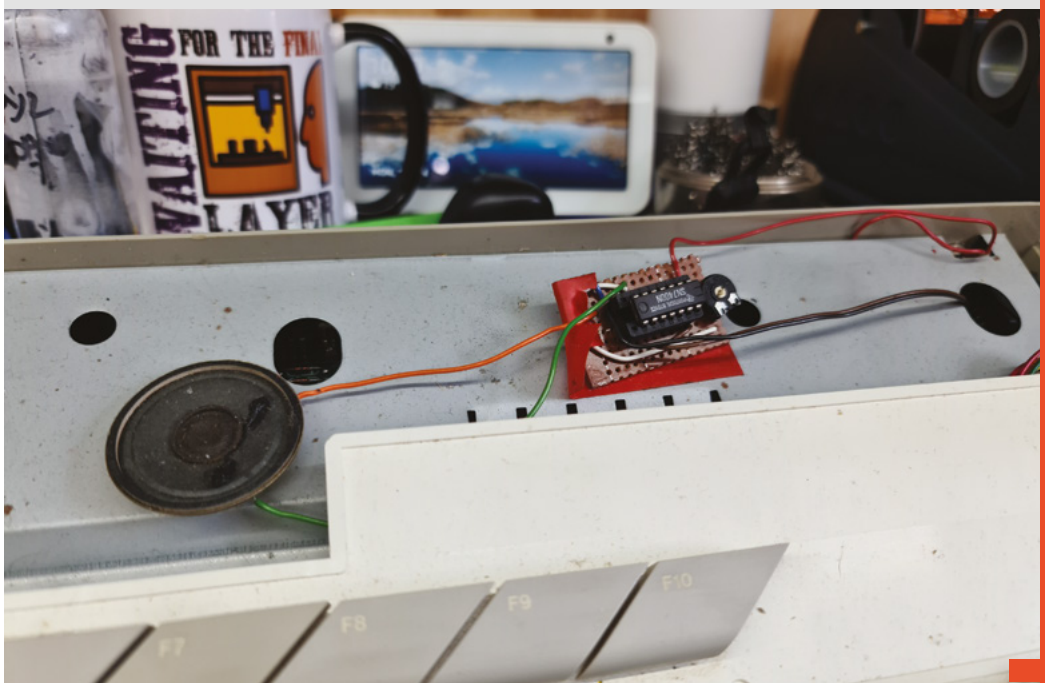
1040ST Pinout

1	Ground
2	NC – Hole is blanked off to facilitate correct orientation of the plug
3	NC
4	5V
5	RX
6	TX
7	Reset
8	NC

> What desktop emulation doesn't have is the classic **'feel' of the original machine**

DON'T SHOUT AT US

There are two ways that people look at classic computer (and car) modifications. Either they're giving old technology a new lease of life, or they're some sort of sacrilegious act that voids the value of the item by replacing it with non-original parts. This project treads a path somewhere in the middle, where old technology can become useful again, but no permanent changes are made to the original parts of the computer. The stock computer can be returned to its original state in just a few minutes with nothing more technical than a screwdriver and a pair of scissors, and no evidence that anything was done beyond a broken warranty sticker. It's quite probable that some collectors will still get shirty about this, so it's worth remembering that computers like these were not pristine when seen in the wild. People would modify them, overclock them, dye the cases, and even add internal features like audio amplifiers and breakout sockets for video or audio editing. The project shown here is nothing more than an extension of what hackers were already doing back in the day.



Left
The Atari was hacked at some point to add a simple audio amplifier with an internal speaker, and an audio output on the rear of the case

Right
The classic Atari ST came without many of the features found in its successors. Smaller than later models, although it sported the classic Atari keyboard design, the early ST had no built-in floppy drives or internal power supply to bulk up its size

520ST Pinout

1	Ground
2	NC – Hole is blanked off
3	Joystick 0 Right
4	Joystick 0 Left
5	Joystick 0 Down
6	Joystick 0 Fire / Mouse Right Btn
7	Joystick 0 Up
8	Joystick 1 Right /Mouse YB
9	Joystick 1 Left / Mouse YA
10	Joystick 1 Down / Mouse XA
11	Joystick 1 Fire / Mouse Left Btn
12	Joystick 1 Up / Mouse XB\
13	+5V
14	RX Data
15	TX Data
16	Reset
17	NC
18	NC

The full pinout is shown here for completeness, but to interface with the keyboard, only five pins are actually necessary. The 5V and Ground lines are needed to power the keyboard's microprocessor, and the RX/TX pins are used to communicate. The Reset pin is needed to initialise the keyboard,

> On early Atari ST computers, the first joystick **socket was shared with the system mouse**

and contrary to what you might read online, failing to connect this pin can lead to some very strange behaviour from the keyboard. Talking to the keyboard with a Raspberry Pi is just a matter of connecting these five pins to a suitable microcontroller that can be configured as a USB HID, reading the outputs from the keyboard's microcontroller using a serial connection, and mapping the outputs to a modern-day USB keyboard

equivalent. The keyboard communicates using a slightly unusual 7812 bps connection, and the connections are made RX to RX, TX to TX, which is frustrating but not uncommon. The process of translating the keyboard signals has been made easier, thanks to the work of Kevin Peat, over at **hsmag.cc/Atari-HID-keyboard**, where you can find code for the Arduino Leonardo. The connection between the keyboard and the Arduino Leonardo can be accomplished with male DuPont connectors, meaning that no soldering or wire-cutting is necessary. A small dab of hot glue or Kapton tape will hold the connection together, and can be easily removed in the future.

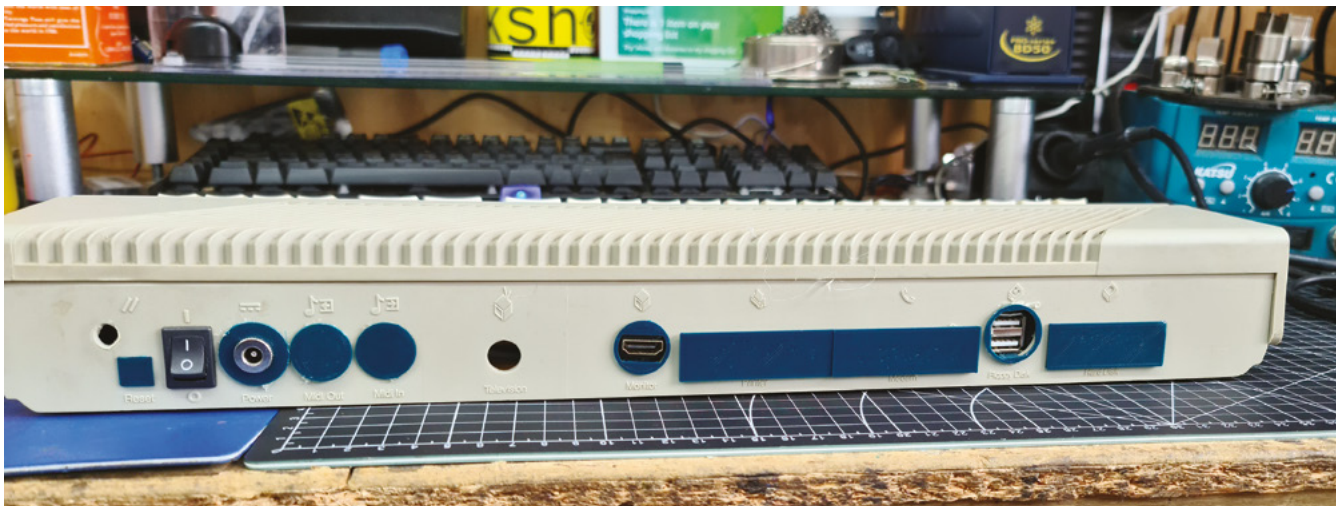
WAGGLE YOUR JOYSTICK

After the keyboard, the most iconic thing about Atari computers (and, likely, the first thing that people think of when you say the name Atari) is probably the classic black joystick. Originating with the Atari 2600, the joystick is digital, with five internal switches connected to a 9-pin D-sub plug. This 9-pin joystick connector remained in use across many computers and consoles for many years, with some changes to the internal wiring. On early Atari ST computers, the first joystick socket was shared with the system mouse. On later versions of the ST, the 9-pin sockets were replaced with a 15-pin variant that allowed for more connected devices. Connecting a classic joystick to a modern computer is a common task, and there are a variety of different ways that you can accomplish the task. One option is to use an off-the-shelf converter that has the correct socket fitted to a USB plug. This is the easiest solution, and requires zero tinkering to get working. However, from an aesthetic point of view, it requires USB cables going into the classic case, and it can look a little bit odd. For a classic Atari ST with side-mounted sockets, it's much neater to mount a USB joystick encoder inside the Atari ST, wired to D-sub connectors in their proper place on the side of the case. There is a 3D-printed mount provided in the files for this project that will let you mount the sockets to a board that matches the PCB layout of the Atari ST motherboard. This means you can place in a dummy PCB, and mount the 3D-printed socket adapter into the Atari without needing to use any permanent fixings. The pinout for the Atari joysticks is very simple, as you can see from the table here.

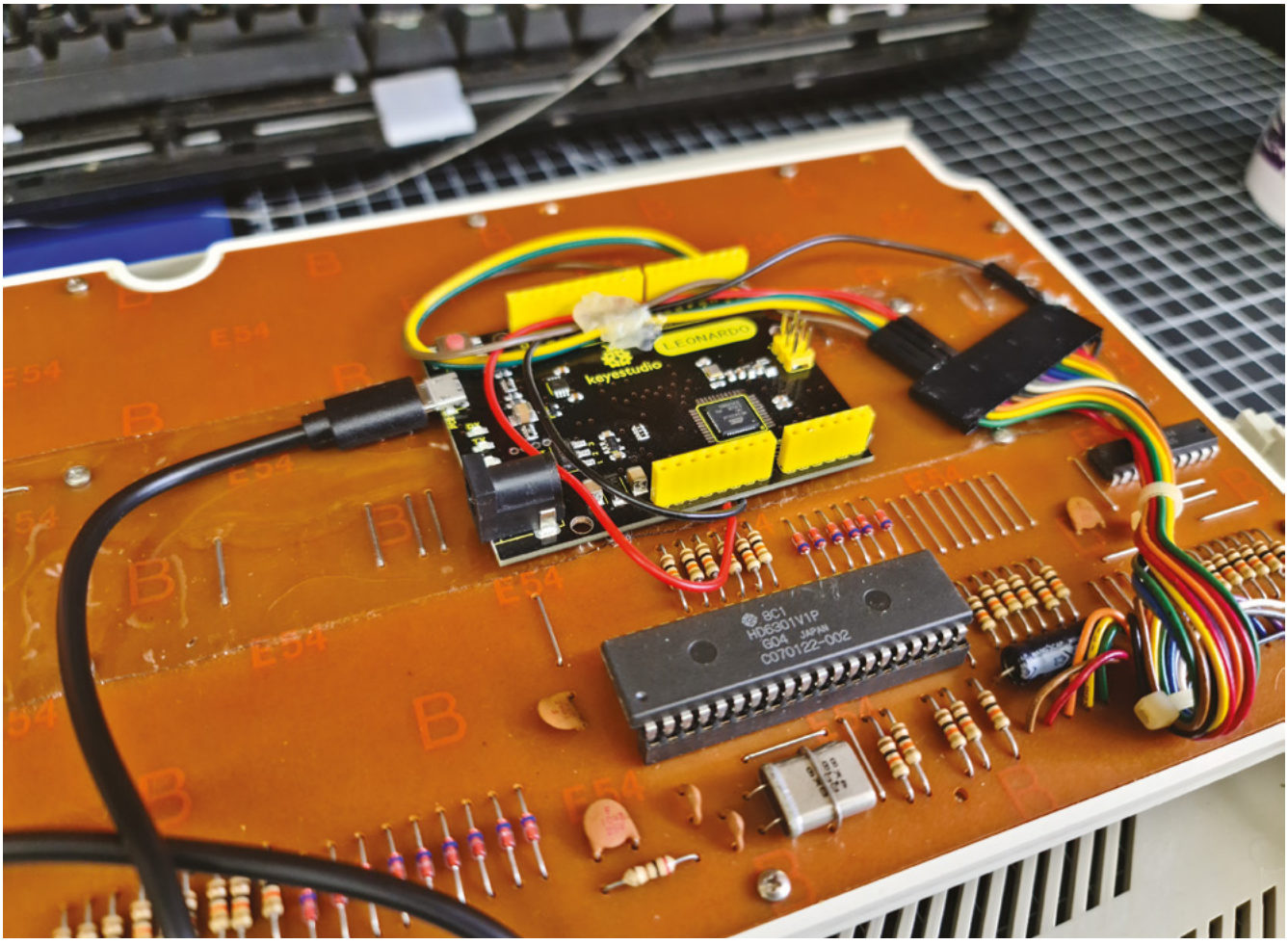
1	Up
2	Down
3	Left
4	Right
5	Reserved
6	Fire
7	+5V
8	Ground
9	NC

QUICK TIP

Use some thin plastic or copper-clad board to make a fake motherboard, with holes drilled in the same places as the original. This will make it very easy to fix your Raspberry Pi and associated bits of hardware into place, and will also make it easy to remove everything if you want to put the machine back into its original condition.



Below
You can add 3D-printed blanks for sockets on the case you're not using without needing to modify the case itself



Above
The Atari keyboard has its own processor to deal with the key matrix. Different versions of the Atari ST had different internal connectors. This keyboard has the 18-pin connection, which also routes the keyboard and mouse sockets through the keyboard processor. Later versions had sockets with a smaller number of connections

If you're wondering about the reason behind the Reserved and NC pins, those pins were used on the original Atari console for paddle controllers. In theory, you could use a joystick encoder to reproduce this functionality for emulation purposes, even though it isn't a function found on Atari ST itself.

MAKE MUSIC

One of the unique features of the Atari ST is the availability of MIDI ports to control musical instruments. The Hatari emulator supports MIDI devices, so if you have a USB MIDI controller, you can mount DIN sockets into the Atari ST case and use them as though it were part of the original hardware. Connecting to MIDI is explained in the manual for the Hatari Emulator at **hsmag.cc/Hatari-emulator**. Remember, however, that the Atari MIDI OUT socket was not a standard MIDI socket, but a combined MIDI OUT and THRU socket. This shouldn't cause a problem in most cases, unless you attempt to connect a non-standard cable in an existing MIDI setup.

OF MICE AND MENDING

The difficult choice at this point is what to do with the mouse socket. The traditional Atari ST mouse plugged into the side of the Atari next to the joystick. If you wanted to play a game with two joysticks, you needed to remove the mouse and plug it in or build an external switch-box. With a modern machine, this isn't an issue. You can either opt for a Bluetooth mouse, add extra USB sockets, or remain faithful to the original design and dedicate one of your sockets to being a mouse socket. If you're lucky enough to have a working Atari ST mouse and want to do this, then there are adapters around (such as **hsmag.cc/tinkerBOY-converter**) that will let you connect the mouse via USB in the same way that you can connect the keyboard or joystick. If you prefer to find out how the mouse works and have a go at implementing this yourself, you can start by having a look at **hsmag.cc/Shaft-encoder** for an explanation of how the Atari mouse works. However, even the most ardent fan of the Atari ST will admit that the ball mouse was a bit of a brick by modern standards,

OPERATING SYSTEM

If you're unfamiliar with TOS, it is the hardware operating system that runs most Atari computers. The most common TOS versions used for emulation are probably v1.02 for the standard ST, and 1.62 for the Atari STE. Other TOS versions are available for machines like the TT030 and Falcon. Like most game images, the operating system is subject to copyright. Legislation varies by territory, and it is up to you to make sure that the image you are using doesn't violate any local copyright laws. There is also an alternative incarnation of TOS known as EmuTOS, which has been created as a replacement for the original TOS. EmuTOS is free and can be found at **hsmag.cc/EmuTOS**. For all other flavours of TOS, a simple web search for 'Atari TOS images' should be enough to get you started.

and after decades of use, many old mice have failed microswitches or encoders. For the sake of simplicity, it's probably a good idea to rely on a modern Bluetooth or USB mouse to control the Raspberry Pi, and then dedicate the original mouse port exclusively for joysticks. If you'd like to add a more authentic look, you can use the STL file of an Atari ST mouse from **hsmag.cc/Optical-mouse** and 3D-print your own version.

In terms of hardware, there's little else to consider. Any holes left in the Atari ST case can be filled with 3D-printed plugs. No glue is required because the plugs are held in place with a rubber O-ring. A short HDMI extender fits neatly into the space previously occupied by the monitor socket, and a 2.5mm socket mounted in the old power plug location can be used to provide power to the Raspberry Pi. USB extenders can bring out the USB sockets to the outside of the case in the same way.

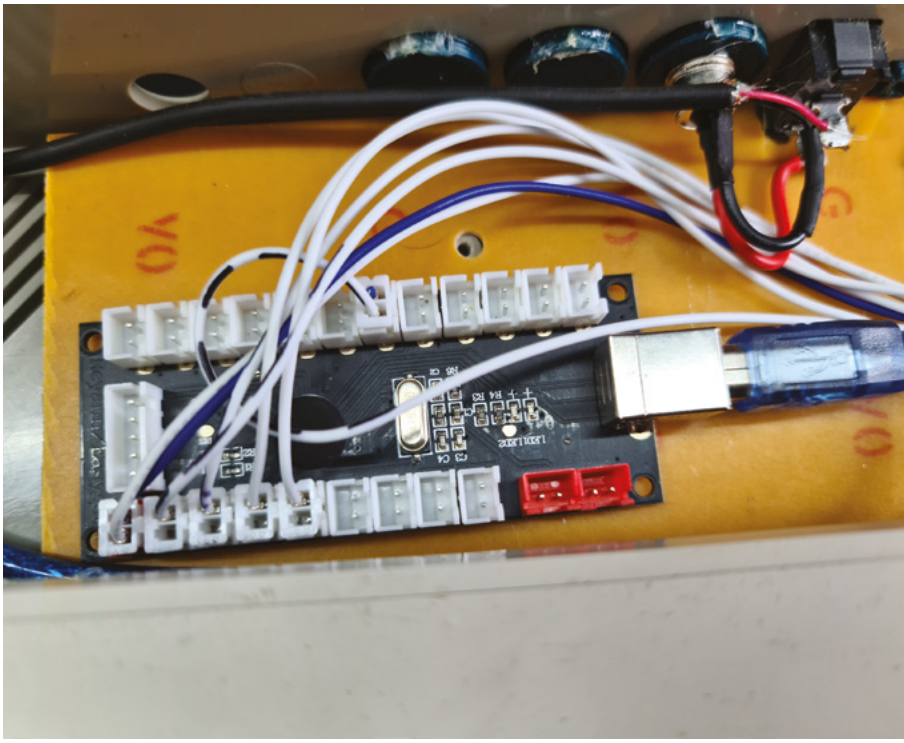
> Once RetroPie is installed, you'll need to do a little bit **of extra work to set up the Atari ST emulation**

Once you've connected up all of the necessary cables to your Raspberry Pi, it's time to start thinking about emulation. The standard option for Atari ST emulation on a Raspberry Pi is Hatari, but since you're already here, why not install support for something more than just the Atari ST?

RetroPie is a specialised system that brings together several emulation projects on the Raspberry Pi, including the Hatari emulator. You can either install it onto an existing system, or you can download an installation image for the Raspberry Pi that does the majority of the work for you. Installing RetroPie is as easy as downloading the appropriate image from **retropie.org.uk**, then write the image to a microSD card, insert the card into the Raspberry Pi, turn on the power, and follow the on-screen prompts.

The RetroPie website has excellent documentation, and **hsmag.cc/RetroPie-install** is the best place to start. Once RetroPie is installed, you'll need to do a little bit of extra work to set up the Atari ST emulation. The Hatari emulator needs a valid TOS image installed in order to work.

The TOS image needs to be installed to the folder **~/RetroPie/BIOS/tos.img**, while game images are copied to **/home/pi/RetroPie/roms/atarist**. The Hatari emulator uses disk images to load games from a virtual floppy drive (or drives). Starting the emulator with a disk image inserted into the drive will automatically load that image. Loading games that require multiple disks means that you need to open up the Hatari emulator settings, and change the disk inserted in the virtual drive. You can find more information on this in **hsmag.cc/RetroPie-docs** and also from **hsmag.cc/Hatari-user-manual**.



Above
A generic joystick board is the easiest way to connect Atari (or custom) controllers to the Raspberry Pi. The board shows up as a generic joystick device, and will return events when the joystick moves or a button is pressed. Although the board in this image supports modern analogue joysticks, it will work fine with a retro digital joystick

TUTORIAL

Scrap wood picture frames

Decorate your house with leftovers

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.

"Have you seen the price of wood recently?" This, we believe, is now the standard greeting whenever two wood workers meet. And, well, have you? Even manufactured wood, which used to be a cheap and affordable way to get making has rocketed in price. This month, rather than head to the shops, we've raided our scraps bin for offcuts.

Right
Hide the gaps with laser-cut decals

Picture frames are a bit of an odd thing because they're defined most by the thing that's not there – the gap in the middle where the picture goes. For small images, such as photos, you might find it easiest to simply start with a single sheet and cut out the middle to make a frame. However, if you want to frame a larger piece, you're going to end up with a lot of wasted material if you do this. It's far better to join four bits of wood together – one for each side.

At this point, we'll be honest. This author is not a highly skilled wood worker, but he's been trying to make a mitre-cut picture frame intermittently for years. Each one ends the same way – glaring gaps in the mitre joints. He's tried hand-cutting them and using various different mitre saws, but nothing seems to eliminate those pesky gaps. Recently, he was faced with a picture of unusual dimensions that needed a frame and decided to get by this problem once and for all. It's time to learn how to make a picture frame that is at least presentable, and to do it with the contents of the scrap pile. This isn't so much a tutorial as a tale of how the process went for us, in the hope that you can learn along with us. The starting point for this project was some scrap plywood. Obviously the wood doesn't have to be scrap, and it doesn't have to be ply, but we often find ourselves with narrow strips of ply that can be saved up for

Left
A half-lap joint is both (fairly) easy and satisfying

Below
Three different options for the back – reinforce, route before, and route after

framing. The important thing is that you have four continuous lengths long enough for each side, and that it's thick enough to cut a shoulder into for the picture to sit in (around 5 mm thick, or you can double-up if you have thinner wood).

The first thing we wanted to try was a classic mitre-cut frame. Mitre joints are where each side of the joint is cut at 45 degrees – it's the traditional joint for picture frames, and can create a particularly good

> There should be no gaps, and a blob of glue on **each joint should hold everything together**

impression when using wood with a pattern cut into it. When done well, there should be no gaps, and a blob of glue on each joint should hold everything together (though you can also use staples or biscuits). The word 'should' is hiding a lot of problems there.

The first step was to cut the wood into appropriately sized strips. The exact width of each strip is more of an aesthetic decision than anything. We went with 5 cm, and cut them on a table saw, but you should use whatever saw you have to hand. They need to be straight and consistent. →

Throughout this build, we sanded our cuts smooth. We won't go over every time we did this, but we kept it all splinter-free, at least in the parts that are on show.

Once you've got your strips, it's time to rebate them. Adding a shoulder to one side of the strips gives a space for the picture and backing to sit. The easiest way to do this is with a router and rebate bit. You could also do it using a table saw. If you don't have either of these, another option is to create the shoulder by using two strips of wood – one slightly narrower than the other, and sitting one on top to create the shoulder. If you're particularly serious about your hand tools, there is also such a thing as a shoulder plane that can create them, but that is well beyond this author's knowledge or skill.

Now it's time to cut the mitres. Honestly, you probably should think twice before taking advice from this particular author since he's never managed to cut accurate enough mitres, but you're here now, so let's plough on. You can use either a mitre block or a mitre saw to cut a 45-degree angle. An angled wedge and a shooting board is another option – to be honest, we're beginning to suspect that there are as many ways of cutting mitres as there are wood workers.

As well as the angle, the length is critical. Generally, with picture frames, the length doesn't have to be perfect in absolute terms, but you do need the two verticals to match as close as possible, and likewise for the two horizontals. Therefore, we cut one to length as accurately as possible, and measured the second one using the first. Line the front edge up and use the other edge to place the saw at the start of the cut and you should get very accurate lengths. If the opposite sides don't match, then the corners won't be 90 degrees and therefore the mitres won't fit, even if they are perfectly cut.

While doing this, you do have to make sure that the shoulder is on the correct side. At this point, you

Above
Half-lap joints are good and strong, even if there are little gaps

Right
No laser? No problem! You can hand-cut these details

can line your wood up in the shape of a square and see that there are small gaps in each corner. How did they get there? Who knows. Your options at this point are:

1. Start again and make a slightly smaller frame using the same bits of wood
2. Throw a hissy fit and use the parts for firewood
3. Accept your fate and try to hide the gaps

We've already tried options 1 and 2 several times and, this time, we're going to give option 3 a whirl.

To make the joints strong, we used a router to remove some wood from the back of each corner so we could glue in a small strip of wood to act as a reinforcer. This may not be necessary given that we're also adding something to the front, but it seemed prudent to make it as strong as possible, especially as our mitres weren't perfect. Given that

> **The problem with picture frames is that it's pretty** hard to hide anything on the front

we've got gaps, the best thing, then, is to make sure we can't see them. This is in the venerable tradition of hiding your mistakes. The problem with picture frames is that it's pretty hard to hide anything on the front. The simplest solution would be to fill them and paint the frame. Indeed, for centuries, framers used liberal amounts of gold leaf in their frames for the sole purpose (we suspect) of hiding their dodgy carpentry. No doubt they told the lords and ladies it was because only gold was good enough to adorn the grand halls, but we know better.

However, we like the look of wood, so rather than just filling them, we'll cover them up with more wood. Again, we searched through the scrap bin and found some 3 mm ply. The colour contrasted with the redder plywood of the underlying frame. We found some botanical designs and laser-cut out as many as we could fit on our scrap wood, and stuck them over the corner and along the side.

To mount the picture in the frame, you need a bit of backing material – we used cardboard cut from a delivery box. Place the picture in the frame first, then the backing and, finally, you can buy kits of frame backing clips that screw in and hold everything in →

Above
You can just see the gap on the inside, but only if you look closely

Above
Picture frame fittings are cheap and widely available online

place. Our kit also came with a hook for hanging on the wall.

The results do cover up our dreadful mitres, and add a little colour to the final design. We're sure that there are some wood workers currently shaking their heads at us, but that's OK. HackSpace magazine is about building cool stuff whatever your skill level, and this is where we are with woodwork.

There might also be some people reading this and shaking their heads because they don't have a laser cutter or a mitre saw. For those people, we had another go, but this time we used only straight cutting (we used a table saw and a bandsaw, but it will work just as well with other saws if you're careful).

At this point, we realised something that should have been blindingly obvious at the start. Rather than go to the effort of cutting mitres only to cover them up, we could just use the one joint that's even simpler: butt joints. In these joints, everything's cut at 90 degrees and you just butt each bit of wood up to the next and glue them in place. They're not the strongest joint, but if we stick something over the front of them to cover up our dodgy carpentry, that'll also provide enough strength.

Rather than laser-cut patterns, we went with something a bit more geometric (and easier to cut by hand). Again, we used a contrasting colour of plywood, and glued it on the front. We also rebated the back.

It's a bold, in-your-face design, but just needed a bit of scrap ply, a saw, and a router.

ONE STEP BEYOND

At this point, we admit, we got a bit carried away. This wood working thing seemed pretty easy. We'd just created two decent-looking picture frames (if we may say so ourselves). However, we'd exhausted our pile of scrap wood. Fortunately, there's a great option just round the corner from Bristol Hackspace

(where we were working on this), and that's the Bristol Wood Project. They salvage wood that's destined for landfill and have a warehouse full of everything from reclaimed pallets to hardwood flooring. It's roughly sorted, but you still have to rummage through to find the different pieces. We do have to admit, though, that shops where you have to rummage are our favourite kind of shops.

We got a piece of hardwood – we think it's oak, but for most items they don't identify the wood – that's about 10 cm × 10 cm and 2.5 metres long – a bargain at £5.

It was far too thick to make the frame out of as it is, so we used a table saw to split it down the middle and tidy up the sides. The result was two lengths of wood each long enough for one picture frame (and a massive pile of sawdust).

> **The joints aren't perfect, and some of the gaps were** a little larger than we'd hoped they'd be

In the previous two picture frames, we used mitre joints and butt joints. The downside of both of these joints is that there's very little area to glue, so if they're not perfectly aligned, the joint will be weak, especially on thin wood like the plywood. This time, we wanted to try something a little daring – not covering up the gaps! This meant that we had to use a joint that would be strong even if the wood wasn't perfectly flush.

The easiest strong joint, as far as we're aware, is the half-lap joint. In this joint, you remove half the material from one side of the joining bits of wood and half from the other. The two go together a bit like they're holding hands. Ideally, the two bits of wood will join flush, but even if there are some small gaps, the joint will be reasonably strong because there's a big area to glue.

You can cut half-laps a number of ways, depending on what tools you have available. A handsaw and chisel will do it, but we've got a table saw, so we used that. We set the blade height to half the thickness of the wood, then used a mitre sled to run the wood through the blade creating multiple cuts, and then cleaned the wood out with a chisel. Once that was done with all joints, we clamped and glued (taking care to ensure that the diagonals were of equal length).

This time we tried a slightly different order. Instead of rebating before we glued up, we rebated afterwards. The result was a slightly neater back, but the corners were rounded, so we couldn't fit the picture in perfectly.

Once the glue was dry, we used a router to create a shoulder in which the picture can sit. Finally, we added clips and a hanging hook. The joints aren't perfect, and some of the gaps were a little larger than we'd hoped they'd be, but overall, we're pretty happy with it. And we've got enough wood left over to have another go. The three frames we've made all hold pictures and, while they do look a little rustic, they are all charming in their own way. We're happy to have them on our wall.

Picture frames don't have to be hard to make to be interesting to look at, and they don't need much material, so are easy to make from scraps, even with limited tools. If you're looking for a project that doesn't cost too much and helps you work on your wood working skills, it's a great option. You don't have to make them the way we have (to be honest, we'd probably recommend you don't), but it's easy enough to figure out a way to make a rectangle with whatever bits of wood you have, and take it from there. If the joints turn out a bit ugly, just cover them up! ◻

SCRAP WOOD

There's a long and venerable tradition of making things out of scrap wood. Perhaps the most iconic is furniture made from old pallets. Old scaffolding board is another popular source of wood.

These are excellent options, particularly if you're still getting started and want to practice your skills without spending a fortune on materials. However, there are a few things that you have to watch out for:

Nails
These can be hiding in wood ready to cut you or blunt your tools. A cheap metal detector can help find the little blighters.

Splinters
Let's be honest, there's usually a reason free wood is free (or cheap), and that's because it's not holding together particularly well. This isn't necessarily a problem, but be prepared to spend time sanding and finishing your projects, and you might have to design around problems with the raw materials.

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

SUBSCRIBE FOR JUST £10!

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

+ FREE RASPBERRY PI PICO W*

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

magpi.cc/subscribe

HackSpace
TECHNOLOGY IN YOUR HANDS

FIELD TEST

HACK | MAKE | BUILD | CREATE

Hacker gear poked, prodded, taken apart, and investigated

PG 92

ARDUINO UNO R4

The most famous microcontroller board gets upgraded

PG 86

BEST OF BREED

The finest displays available

PG 96

TRACE

Digify your sketches

ONLY THE BEST

Time to take another look at displays

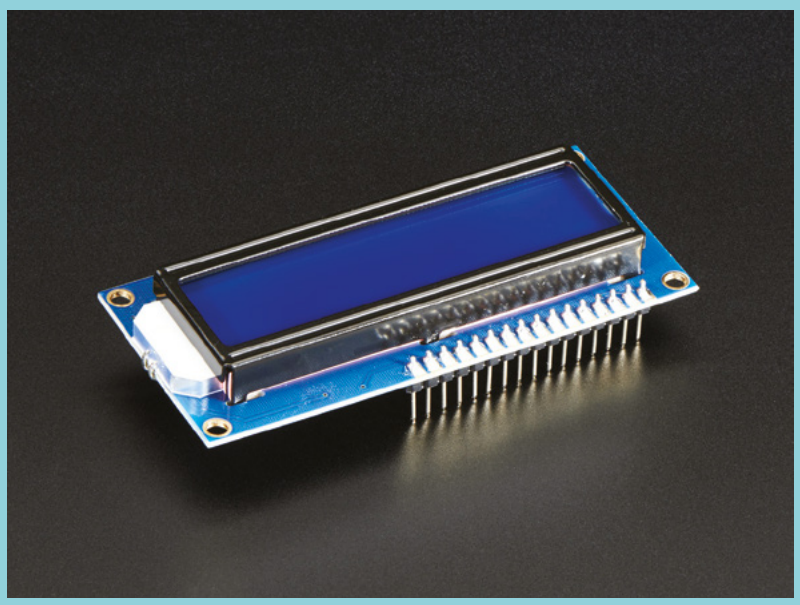
Show your data with style

By **Marc de Vinck** @devinck

One of the first Best of Breed reviews that I wrote for HackSpace was about the variety of displays available for DIY electronic enthusiasts. After countless other reviews, and many years later, I thought it was time to look at the subject again. A lot has changed over the past few years, but also a lot has stayed the same. For example, this variation (pictured) from Adafruit of the classic 16×2 LCD is still just as useful as it was when first introduced to the DIY market many years ago. Yes, you can pick one up with a few added features like an RGB backlight, or different colour characters, but they haven't changed much over the years. And that's a good thing! They work, and they work well.

But now, we also have innovative flexible displays, lots of OLED variations both large and small, e-ink, and plenty of other technologies both old and new. Displays are commonplace. Shortly after you learn to blink an LED or make a button work with your Raspberry Pi, you typically add an LCD display. We've all done it, and it's really satisfying to see your first 'Hello World'.

It's almost hard to imagine a project that doesn't require some kind of alphanumeric display. And many projects require more complicated graphical displays, but those types of products are also readily available and equally as affordable. In this Best of Breed, I'll be looking at a few of my favourite displays, both old and new, that will hopefully inspire you to add a little bit of additional functionality to your next build.



Tufty 2040 vs Pico Display Pack

PIMORONI $25.60 | pimoroni.com

PIMORONI $16 | pimoroni.com

The Tufty 2040 by Pimoroni is a much-requested LCD version of the firm's Badger 2040 board. It's managed to combine an RP2040 microcontroller with a beautiful little colour LCD screen, making for a great little badge. The board features five buttons, allowing you to scroll through menus easily and interact with the badge. It also features a light-sensing phototransistor to automatically adjust the brightness, a battery connector, and a slot in the top of the board for attaching a lanyard. It's available as the Tufty board with LCD and Raspberry Pi Pico built-in, or you can pick up the accessory kit version for a few extra dollars which includes a USB cable for programming, AAA battery pack, Velcro adhesive squares, and a lanyard. Either way, it's a great little kit!

Pimoroni has designed another beautiful accessory for the Raspberry Pi Pico – this time, it's the Pico Display Pack. The board features a 1.14˝ IPS LCD screen, four buttons, and an RGB indicator LED. This is a great solution for adding a colourful display to your Raspberry Pi Pico or Pico W microcontroller. You'll have 240×135 pixels to get your message across. Best of all, this board comes fully assembled and ready to use. Pimoroni has done a great job with including a detailed diagram of the pinouts, and a very clear 'getting started' guide with MicroPython code examples.

VERDICT

Tufty 2040

A great new badge.

10/10

Pico Display Pack

Beautiful, colourful, and crisp.

8/10

BEST OF BREED

HyperPixel 2.1 Round – Hi-Res Display for Raspberry Pi

PIMORONI $57.15 | pimoroni.com

What's better than adding a full-colour, high-resolution display to your Raspberry Pi? Adding one that is round! And the people over at Pimoroni have done just that with their HyperPixel 2.1 Round Display. There's something about round LCDs that just seems a bit magical. I'm not sure why. Maybe it's because most displays are rectangular? The HyperPixel Display features a brilliantly bright and crisp 2.1-inch touchscreen display with a pixel dimension of 480×480. Minus the corners, of course! It's compatible with all Raspberry Pi models, but works best with a Raspberry Pi Zero, since you can neatly hide the latter away behind the touchscreen. It makes for a beautiful, compact touchscreen interface for your next project.

VERDICT

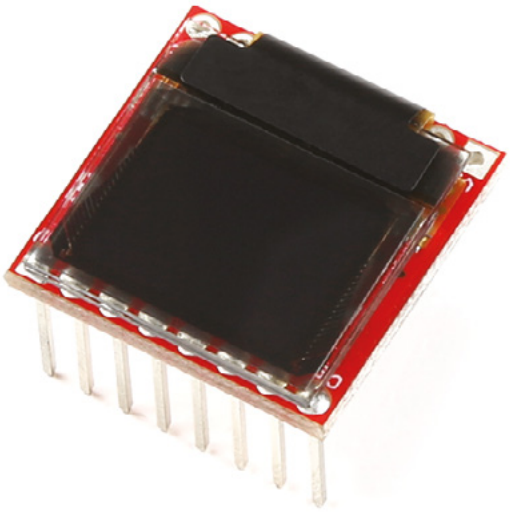
HyperPixel 2.1 Round – Hi-Res Display for Raspberry Pi

Hyper-colourful and hyper-fun!

9/10

SparkFun Micro OLED Breakout

SPARKFUN $17.50 | sparkfun.com



The SparkFun Micro OLED breakout with headers is exactly what the name suggests. The board features a very small OLED screen measuring only 0.66 inches across, with a resolution of 64x48 pixels. It's not a lot of room to fit lots of information, but sometimes you just need a simple graphic to convey your message. The board allows access to all 16 of the OLED's pins but, to make things simpler, the pins on the bottom of the board are used for parallel interface, while the other pins can be used to control the display via SPI or I2C. If space is limited, and you don't need to show a lot of information, this board might be a perfect fit for your next miniature project.

VERDICT

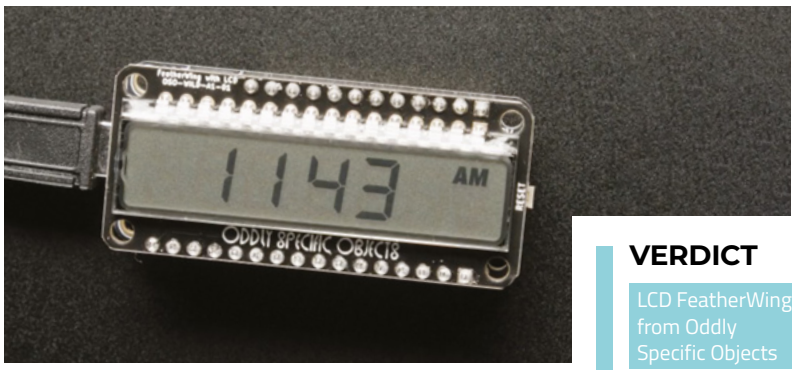
SparkFun Micro OLED Breakout

Good when you need a small display.

8/10

LCD FeatherWing from Oddly Specific Objects

ADAFRUIT $19.95 | adafruit.com



Sometimes you just need a little bit of retro in your build, and that's where the LCD FeatherWing, designed by Oddly Specific Objects and available at Adafruit, comes into play. This board features a low-powered CircuitPython-compatible display designed specifically for Adafruit's Feather line of microcontroller boards. The glass display has 48 segments, including five indicator icons, 57 segment digits, four decimal points, and an AM or PM indicator. If you are looking for a very low-power display, this is for you! The display itself only consumes micro-amperes of power, allowing battery-powered projects to last for months on end, even with an always-on display. Just don't forget to put your micro into low-power mode too!

VERDICT

LCD FeatherWing from Oddly Specific Objects

Retro tech made easy

9/10

BEST OF BREED

1.12” Mono OLED

PIMORONI $18.56 | pimoroni.com

Pimoroni always designs beautiful boards and, as much as this board is lacking in a typically cute name, the 1.12-inch mono OLED breakout board is another great addition to its line of products. The board features a bright and crisp 128×128-pixel monochromatic display. Despite its small size, you can still manage to display some detailed graphics. Pimoroni has some very good documentation, and the board is available in either an SPI version or I2C. The display is compatible with all models of the Raspberry Pi and Arduino. Head over to its website to learn more about this beautiful little OLED display.

VERDICT

1.12” Mono OLED

Tiny, crisp and well documented

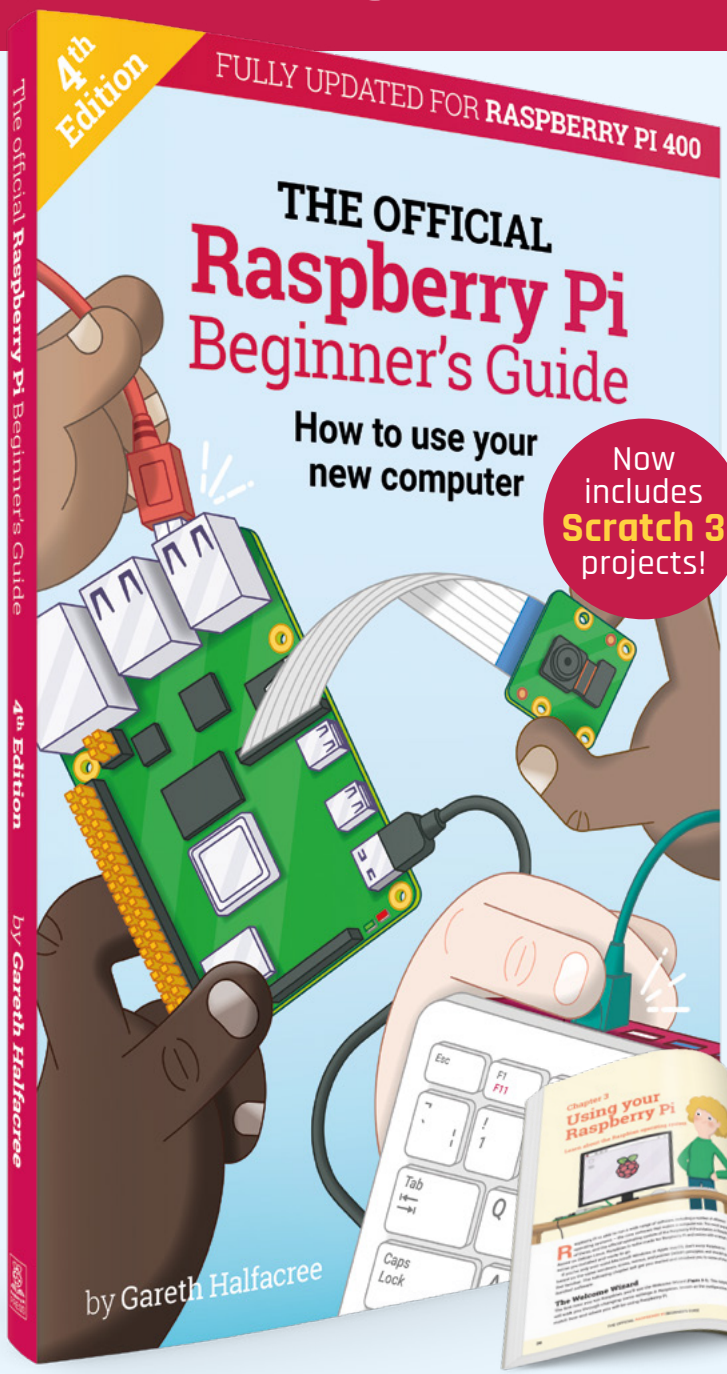
9 /10

STANDARD LCD 16×2

ADAFRUIT $9.95 | adafruit.com

Anyone who’s tinkered with any modern electronics almost inevitably has at least one 16×2 LCD screen in their toolbox. They are inexpensive, easy to use, and come as part of almost every starter kit. They are available in a variety of colours, some with dimmable backlights, and others are even full RGB. If you don’t have one in your toolbox, you certainly need to add it.

THE OFFICIAL Raspberry Pi Beginner's Guide



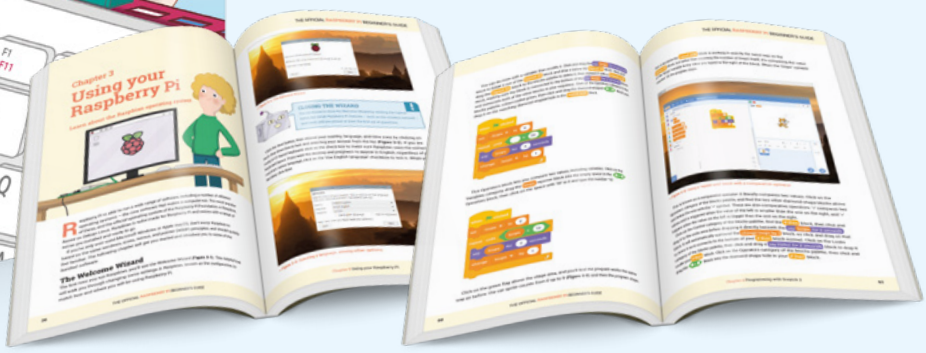
The only guide you need to get started with Raspberry Pi

Inside:

- Learn how to set up your Raspberry Pi, install an operating system, and start using it
- Follow step-by-step guides to code your own animations and games, using both the Scratch 3 and Python languages
- Create amazing projects by connecting electronic components to Raspberry Pi's GPIO pins

Plus much, much more!

Just £10



Buy online: magpi.cc/BGbook

REVIEW

Arduino Uno R4

La scheda famosa di Arduino arriva alla versione 4

ARDUINO ◆ **From €18** | **hsmag.cc/unor4**

By Ben Everard @ben_everard

Thirteen years ago, Arduino released its ninth board, and for reasons only it knows, it decided to call the board Uno. It quickly became synonymous with maker electronics projects.

This fourth iteration includes by far the biggest changes to date, not least because, for the first time, there are two options: the Minima, which is a fairly bare microcontroller board (in the same style as earlier Unos), and the Wi-Fi version that includes both a wireless networking controller and a grid of 96 LEDs.

Right ◆
Both boards have the same footprint

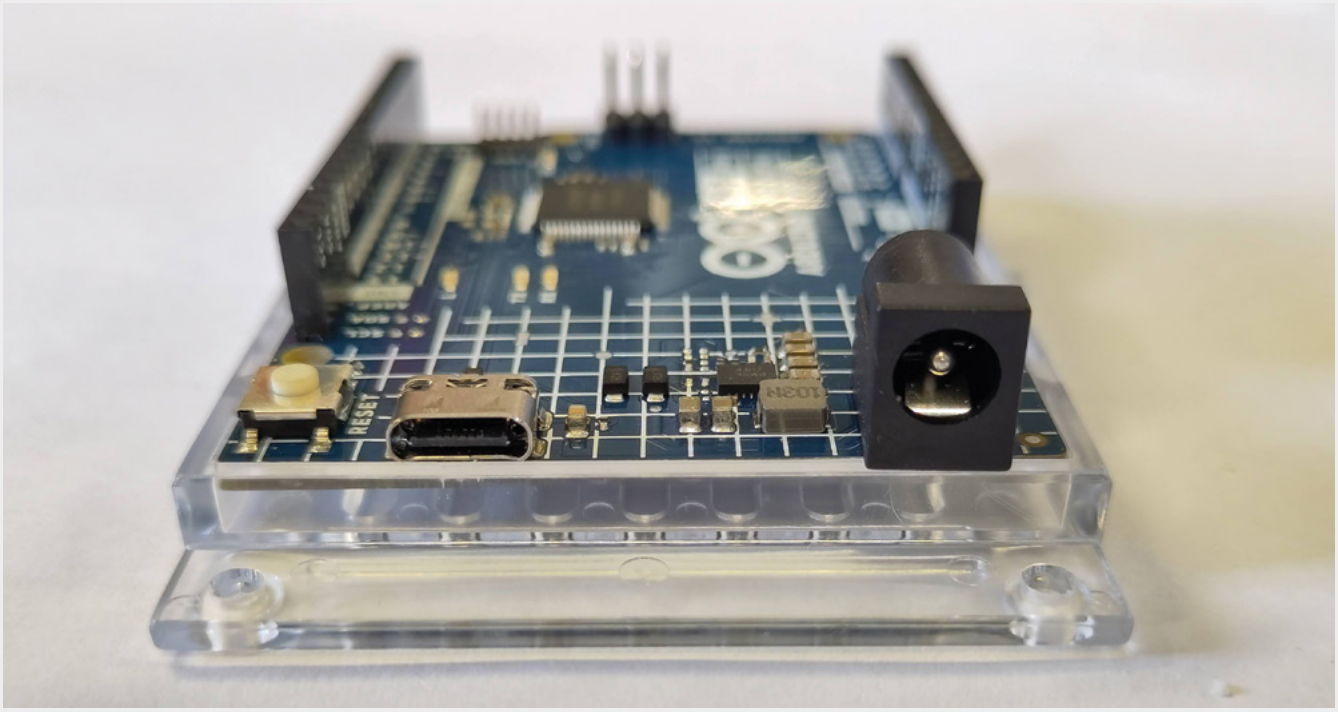
The other big change is that the microcontroller has been swapped. The AVR-based ATMega controller is replaced by an Arm-based Renesas chip. This is a major upgrade in almost every way. It's faster, has more RAM, more storage, more peripherals (including a digital-to-analogue converter). We don't have space to list every feature here, but it's generally more capable. However, when shifting not just the microcontroller family, but the architecture of the core, it's going to end up working a bit differently.

The Arduino libraries do quite a good job of abstracting away the hardware, and code can often run on different hardware without modification. However, some programmers seeking to squeeze every drop of performance out of the AVR boards use code that is platform-specific. As a result, not all code written for the Arduino Uno R3 will work on the R4.

This aside, though, Arduino has done a good job of ensuring compatibility with the R3. The pinout is the same (with some added peripherals), and the microcontroller operates at 5V (the vast majority of Arm microcontrollers operate at 3.3V). This means that, as long as the software works, the hardware should too.

While the 5V GPIOs are great for compatibility with existing Uno hardware, the world has moved on in the past 13 years, and most electronics now expect 3.3V. The R4 Wi-Fi does have a pair of 3.3V GPIOs in the Qwiic connector which should make it easy to connect to I2C hardware, especially those with a Qwiic or STEMMA QT connector. However, this isn't on the Minima. Otherwise, you need to ensure that any hardware you want to use is compatible with 5V.

One slightly unusual feature of this, and some other official Arduino boards, is that it comes with a transparent plastic mounting plate. While this might



Above The R4 now connects to your computer with USB-C

seem like a trivial thing, it does have some useful features. At the very least, it stops the contacts on the bottom from shorting out if you place the board on something conductive (such as the leg you trimmed off a soldered LED and missed when you tidied your desk).

The mounting plate has a few more tricks up its sleeve too. It mounts your Uno on your project quite securely, yet still lets you unclip it without screws. The screw holes on the clip match the mounting points on the Uno, so you can screw through and it acts as a standoff. It also has another set of sunk screw holes so that you can screw in the clip without interfering with the Uno. Additionally, it has clips to attach it to a breadboard (on either side) so that jumpers don't pull out when you accidentally knock the board.

> "Smallness isn't always a feature – it's a design choice **that comes with positives and negatives**"

While Arduino software support is generally excellent, it feels like, in recent years, boards have come with features with little to no documentation. In the case of the R4, that's the on-board OpAmp. It apparently exists but, at the moment, there's no information on how to use it. This isn't necessarily a big deal because OpAmps are hardly essential features of microcontrollers. However, given that it's an advertised feature (and one we'd be really interested in trying), it's a bit disappointing that there's no information on how to actually use it.

The R4 is a natural successor to the R3. Some people will complain that it should have stayed on an AVR microcontroller, but the gulf in performance between AVR and Arm chips means that this is increasingly not viable.

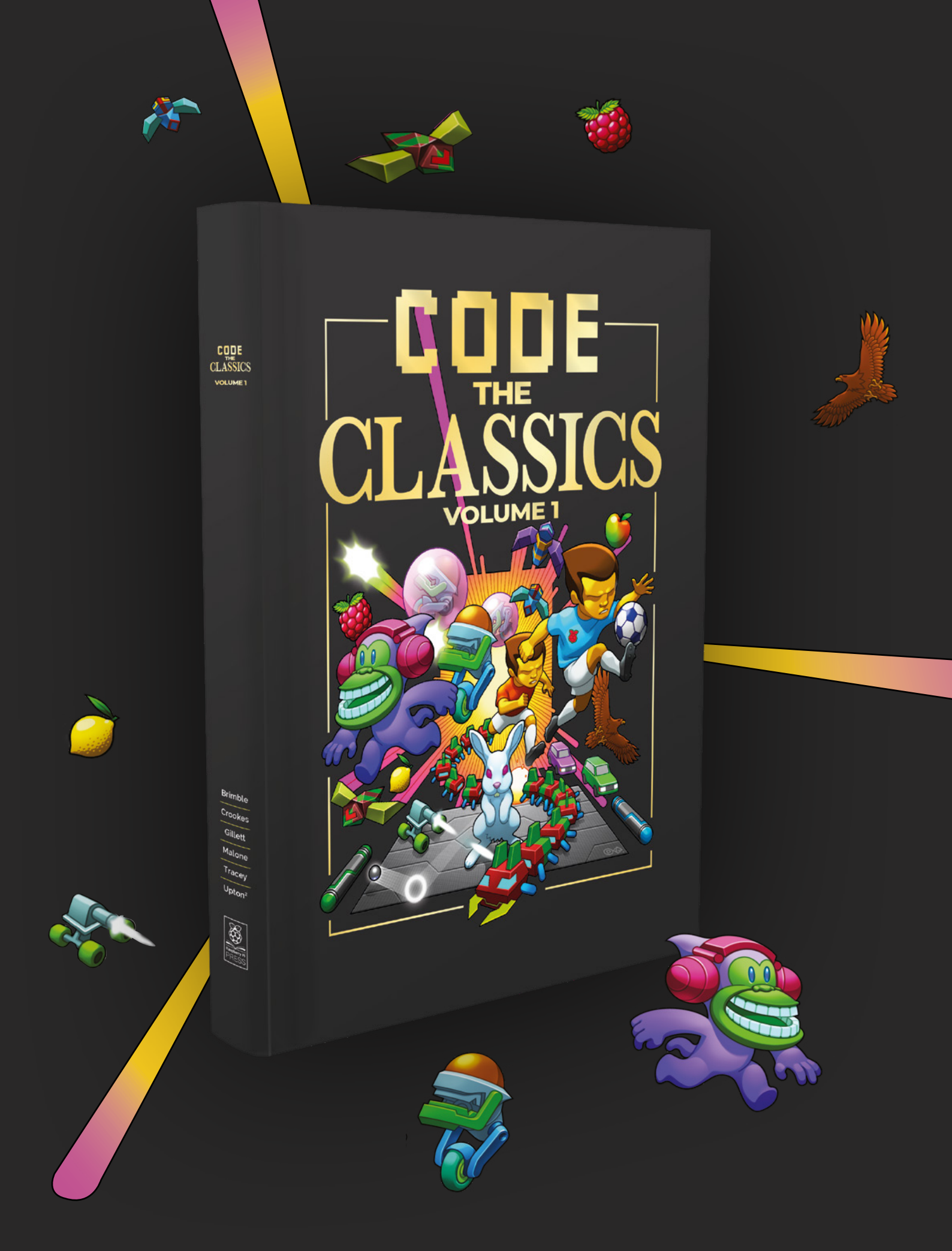
It's a reminder that smallness isn't always a feature – it's a design choice that comes with positives and negatives, and the trade-off isn't always worthwhile. In a world where many dev boards are compressed so much, pins are labelled in minute fonts. On the Uno, each pin is labelled three times to make sure you know exactly where you're poking the wires. It comes with a mounting plate that's actually useful.

When the original Uno was released, it quickly became the standard microcontroller because, frankly, the other options weren't great. We're now living in a golden age of microcontroller dev boards, where there's a myriad of choices. Many of them are really well-thought-out and have great software support. For the Uno to continue to stay relevant, it needs to stand out. If you're looking for a 5V microcontroller, a classroom microcontroller, or an Uno form-factor microcontroller, then the R4 is still a great choice. ◻

VERDICT

Still a classic, but now with more competition.

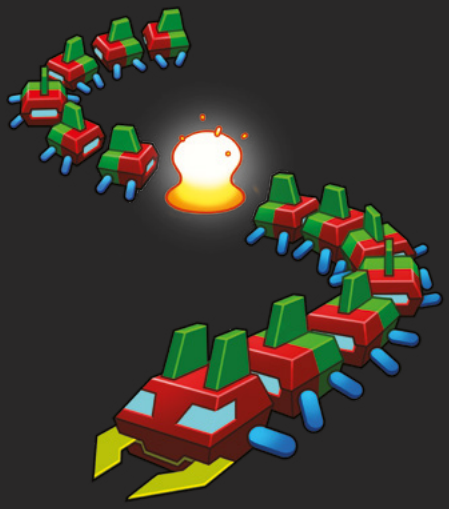
10/10


CODE
THE
CLASSICS
VOLUME 1
Brimble
Crookes
Gillett
Malone
Tracey
Upton²
CODE
THE
CLASSICS
VOLUME 1

CODE THE CLASSICS VOLUME 1

This stunning 224-page hardback book not only tells the stories of some of the seminal video games of the 1970s and 1980s, but shows you how to create your own games inspired by them using Python and Pygame Zero, following examples programmed by Raspberry Pi founder Eben Upton.

- ***Get game design tips and tricks from the masters***
- *Explore the code listing and find out how they work*
- ***Download and play game examples by Eben Upton***
- *Learn how to code your own games with Pygame Zero*



Available now **hsmag.cc/store**

CROWDFUNDING NOW

Shaper Trace

Convert your doodles into fabrication files

From $63 | **hsmag.cc/trace** | **Delivery:** Oct 2023

Most digital fabrication methods work with vector graphics, and the most popular format is SVG. From these SVG files, you can laser-cut, engrave, CNC, and even 3D-print your designs. But how do you get the SVG files in the first place? Vector graphics software can be a bit complex, and it takes time to learn. Wouldn't it be great if you could just sketch your idea and scan it into your computer? That's the idea behind the Shaper Trace.

It's a frame that you place over your drawings and snap a photo using the official mobile phone app. It will then automatically convert your scribbles into a dimensionally accurate SVG file. It's the dimensionally accurate bit that will be make or break for a technology like this. It's currently pretty easy to take a photo of black ink on white paper and convert this to an SVG. Getting the sizing accurate isn't impossible, but it does make it more difficult.

The crowdfunding campaign promises things like the ability to trace the outline of whole patterns to get a laser-cuttable file. If they can make this process quick and easy, then it'd be a great addition to a makerspace or home workshop.

The 'if' in the last sentence is critical. For this to have value, it has to be reliable and work all the time, not 80% of the time, or 90% of the time.

Shaper uses similar technology in its Origin CNC, which places the spindle in the correct position using optical recognition, so they certainly seem like a company that has the expertise to solve this problem well. We are looking forward immensely to getting our hands on one to test it fully. □

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.

nextmonth

issue #70 ON SALE 31 AUGUST

EBEN UPTON

Raspberry Pi: the inside story

ALSO

- → MICROPYTHON
- → MUSIC
- → 3D PRINTING
- → TOMBOLA
- → AND MUCH MORE

DON'T MISS OUT

hsmag.cc/subscribe

The sublime art of sublimation

The ability to apply repeatable designs to a surface that isn't paper can have a massive effect on your making and prototyping process. Glass, metal, fabric, wood, and other materials can all be printed onto, if you have the right bits of equipment. Like the sound of this? Then tune in next month, and we'll show you how it's done.

HiPi.io

PiKVM

Manage your servers or workstations remotely

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

PiKVM HAT
for DIY and custom projects

Pre-Assembled version

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

Available at the main Raspberry Pi resellers

core ELECTRONICS · PiShop.ca · RPishop.cz · Welectron. · elektorstore · PIMORONI

Pi-Shop.ch · ThePiHut · PiShop.us · CloudFree · PiShop.mx

Reseller suggestions and inquiries:
wholesale@hipi.io