MAKE | BUILD | HACK | CREATE

HackSpace

TECHNOLOGY IN YOUR HANDS

hsmag.cc | August 2022 | Issue #57

Robot Roundup

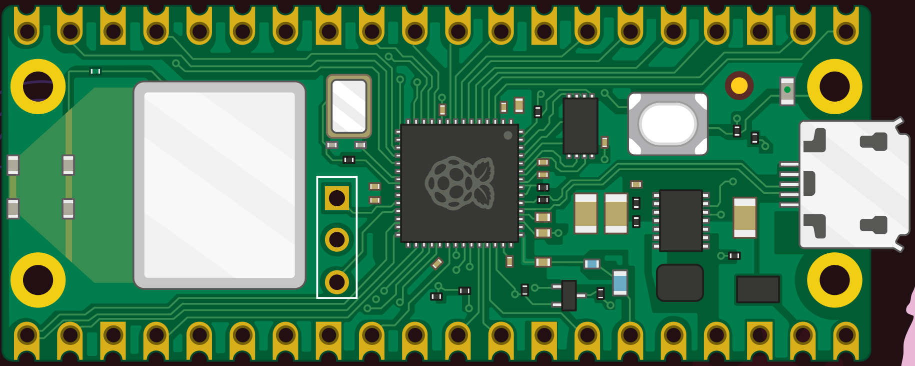
Make your Raspberry Pi move

Beer BBQ

Turn a keg into an outdoor grill

RASPBERRY PI

PICO W PROJECTS



Make internet-connected builds with the **NEW** $6 microcontroller board

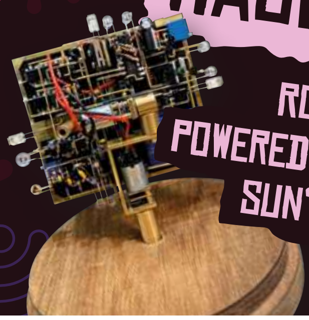
Aug. 2022
Issue #57 £6

Time-lapse Photography

Watch life on fast forward

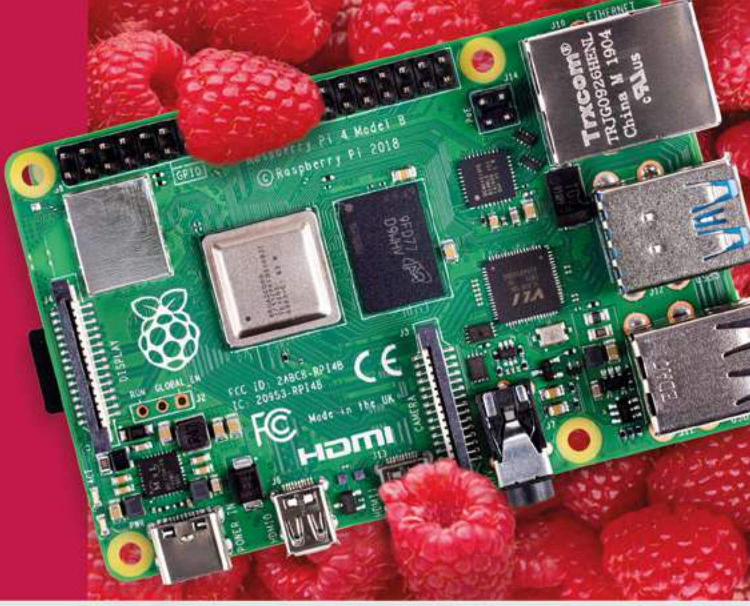
BEAM MACHINE

ROBOTICS POWERED BY THE SUN'S RAYS



LASER CUTTING KNITTING WELDING DOGS

American
Raspberry Pi
Shop



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

Partner and official reseller for top Pi brands:

adafruit sparkfun ELECTRONICS PIMORONI HiFiBerry and many others!

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

USA PiShop.us

Canada PiShop.ca

Raspberry Pi APPROVED RESELLER

Welcome to HackSpace magazine

When Pico came out in January 2021, there was one feature requested more than any other – wireless networking. The problem is, while there was some spare space on the Pico board, there wasn't really enough for wireless. There also weren't any spare GPIOs that could be used to connect the wireless controller. These two obstacles didn't stop the Raspberry Pi engineers, who ploughed on regardless.

The result of their hard work is the Pico W. It's (almost) pin and form factor compatible with Pico – the only changes being moving the debug pins and a change to the LED. This month, we're putting the Pico W through its paces with three projects to get you started on your own connected projects. Whether you want to build an IoT device, control a robot, or just flash some lights, we can help.

This month, we're putting the Pico W through its paces with three projects to get you started **on your own connected projects**

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 44 **FREE PICO** WHEN YOU SUBSCRIBE

EDITORIAL
Editor
Ben Everard
ben.everard@raspberrypi.com

Features Editor
Andrew Gregory
andrew.gregory@raspberrypi.com

Sub-Editors
David Higgs, Nicola King

DESIGN
Critical Media
criticalmedia.co.uk

Head of Design
Lee Allen

Designers
Sam Ribbits, Olivia Mitchell, Ty Logan

Photography
Brian O'Halloran

CONTRIBUTORS
Jo Hinchliffe, Marc de Vinck, Andrew Lewis, Mike Bedford, Nicola King, Rosie Hattersley, Jack Spiggle, Rob Miles

PUBLISHING
Publishing Director
Russell Barnes
russell@raspberrypi.com

Advertising
Charlie Milligan
charlotte.milligan@raspberrypi.com

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

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

To subscribe
01293 312189
hsmag.cc/subscribe

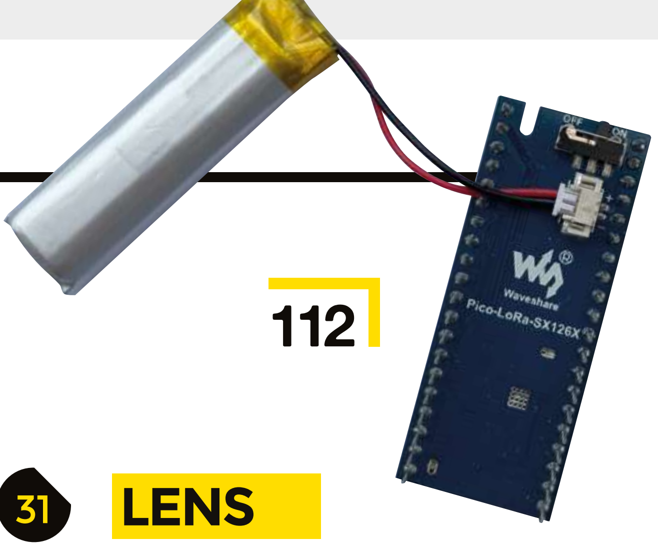
Subscription queries
hackspace@subscriptionhelpline.co.uk

recycle

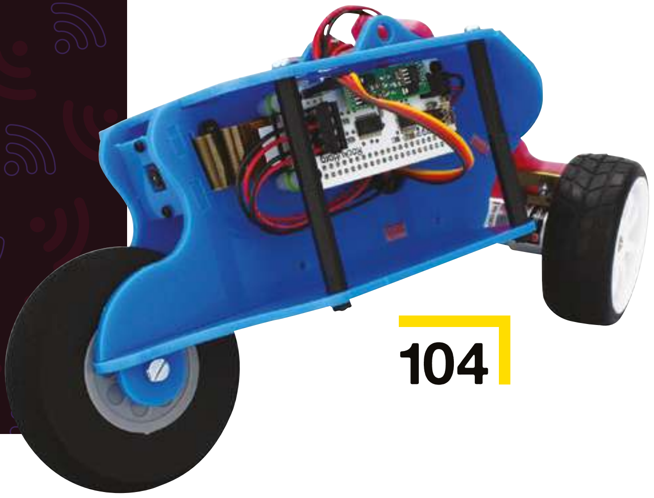
This magazine is printed on paper sourced from sustainable forests. The printer operates an environmental management system which has been assessed as conforming to ISO 14001.

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



112



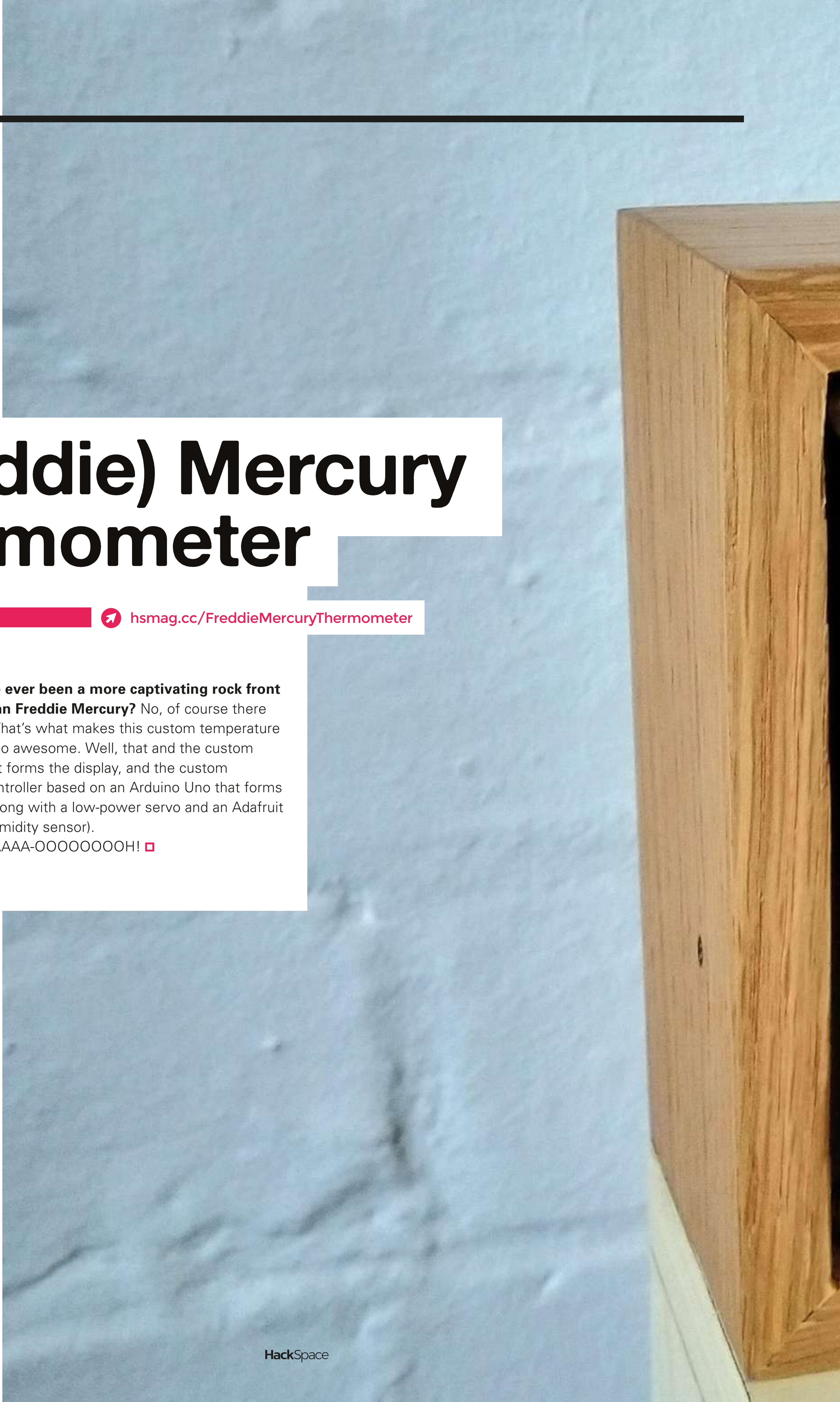
104

96

18

06

The Mercury Thermometer
Put Out The Fire 35°C
Keep Passing the Open Windows 30
It's a Beautiful Day 25
Doing All Right 20
Cool Cat 15
Hang On In There 10
Stone Cold Crazy 5
Ice Ice Baby 0
A Winter's Tale -5°C
Humidity
80–100%
60–80%
40–60%
20–40%
0–20%



(Freddie) Mercury thermometer

By **TurboSnail**

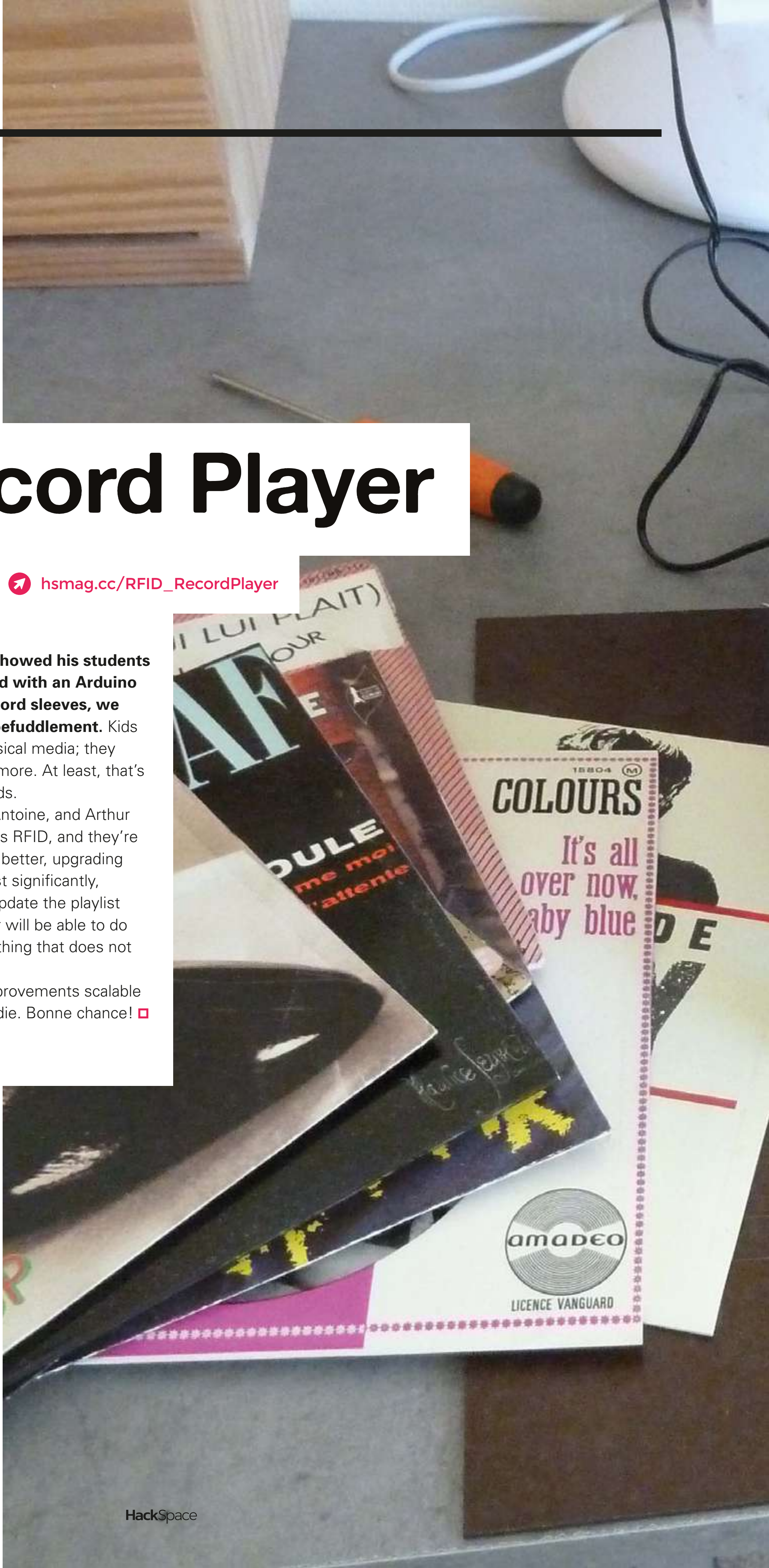
hsmag.cc/FreddieMercuryThermometer

Has there ever been a more captivating rock front man than Freddie Mercury? No, of course there hasn't. That's what makes this custom temperature display so awesome. Well, that and the custom PCB that forms the display, and the custom microcontroller based on an Arduino Uno that forms the guts of the build (along with a low-power servo and an Adafruit AHT20 temperature/humidity sensor).

All together now: DAAAA-OOOOOOOOH!

Right Freddie points at the temperature with his resin-printed arm

The Mercury Thermometer
by
TSD EE
Put Out The Fire
35°C
Keep Passing the Open Windows
30
It's a Beautiful Day
25
Doing All Right
20
Cool Cat
15
Hang On In There
10
Stone Cold Crazy
5
Ice Ice Baby
0
A Winter's Tale
-5°C
Humidity
80-100%
60-80%
40-60%
20-40%
0-20%



RFID Record Player

By **Ibrahima, Scott, Antoine & Arthur**

hsmag.cc/RFID_RecordPlayer

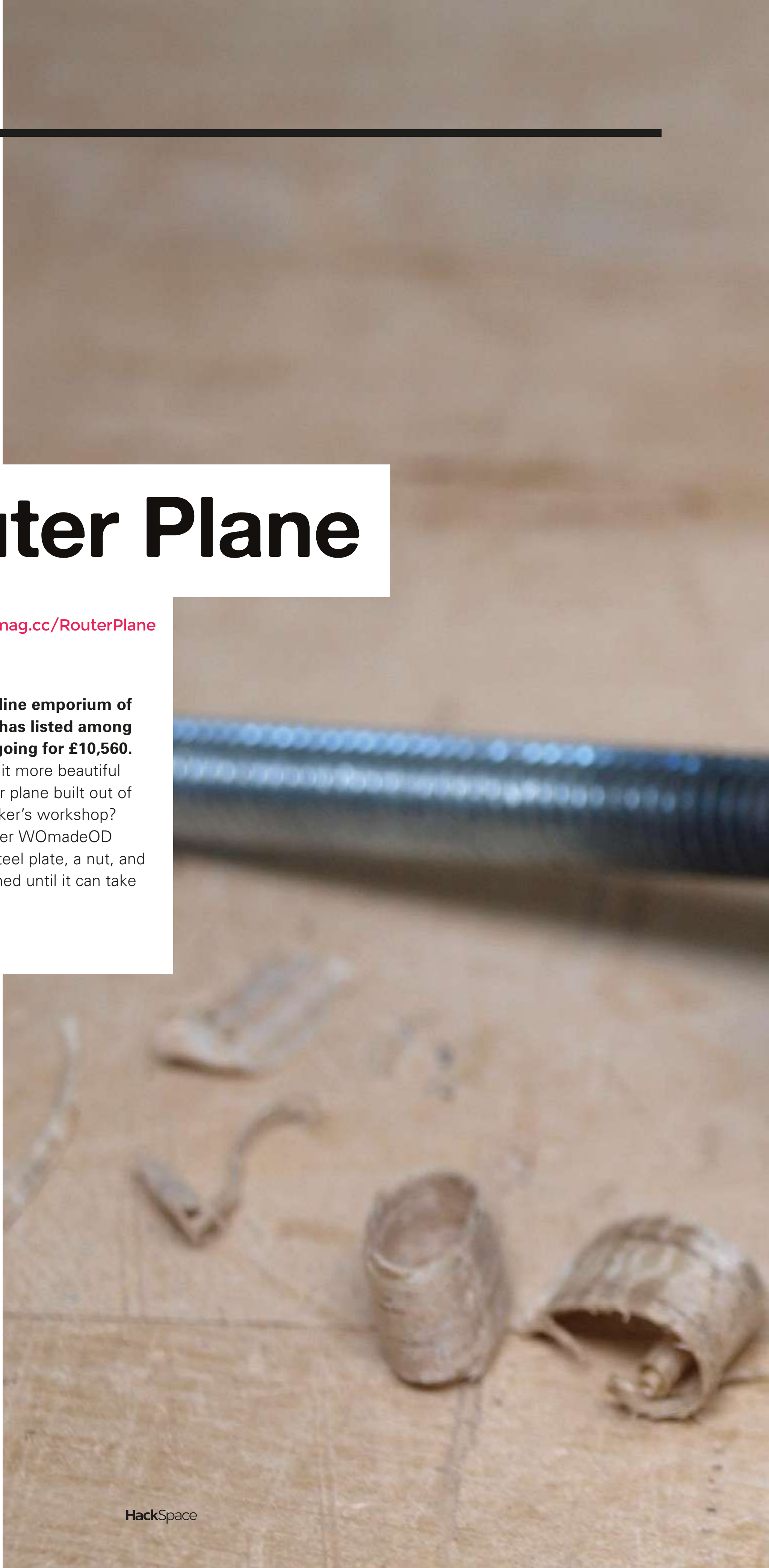
When teacher Jean Noël showed his students a record player upgraded with an Arduino to read RFID tags on record sleeves, we can only imagine their befuddlement. Kids these days don't use physical media; they don't even download anymore. At least, that's what we're told. We don't actually know any kids.

The students in question – Ibrahima, Scott, Antoine, and Arthur – did recognise that the underlying technology is RFID, and they're working on making their teacher's project even better, upgrading the user interface to make it easier to use. Most significantly, instead of uploading a new Arduino sketch to update the playlist (something that requires coding skills), the user will be able to do it by renaming MP3 files on the SD card (something that does not require coding skills).

The students eventually plan to make the improvements scalable as a kit form that they will be able to sell on Tindie. Bonne chance!

Right
Other improvements in the pipeline include an RGB LCD screen to give information to the user and show some cool light animations

#TeamArduinoCC
ARDUINO
SUPPORT THE REAL ARDUINO
BIG Saund "2000" automatic Records
VOLUME
Reject



Mini Router Plane

By **WOmadeOD** hsmag.cc/RouterPlane

Workshop Heaven, the online emporium of woodworking delights, has listed among its wares a hand plane going for £10,560. It's a beautiful tool, but is it more beautiful than this handmade router plane built out of scraps of wood in the maker's workshop?

We'd say not. This build, by Instructables user WOmadeOD comprises a chunk of hardwood, a square of steel plate, a nut, and a bolt, which is shaped, hardened, and sharpened until it can take off thin slivers of wood.

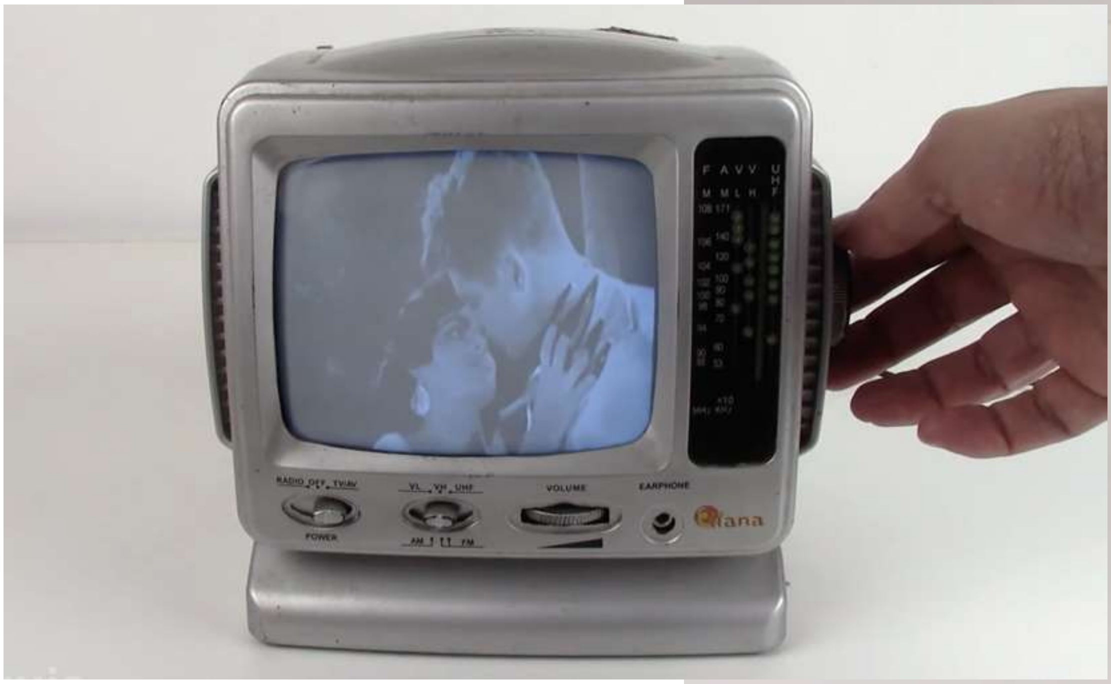
Right
To adjust the depth of the plane, just turn the bolt a fraction of a turn

Raspberry Pi TV simulator

By Pakéquis

Pakequis.com.br

In this age of LED everything, it's hard to imagine the tiny CRT televisions that were once marketed as 'travel' TVs. That's what this is, only this has been retrofitted to include a Raspberry Pi playing a selection of old black-and-white films. Even better, when you change the channel, it produces an authentic screen of white noise, just like TVs used to do. □



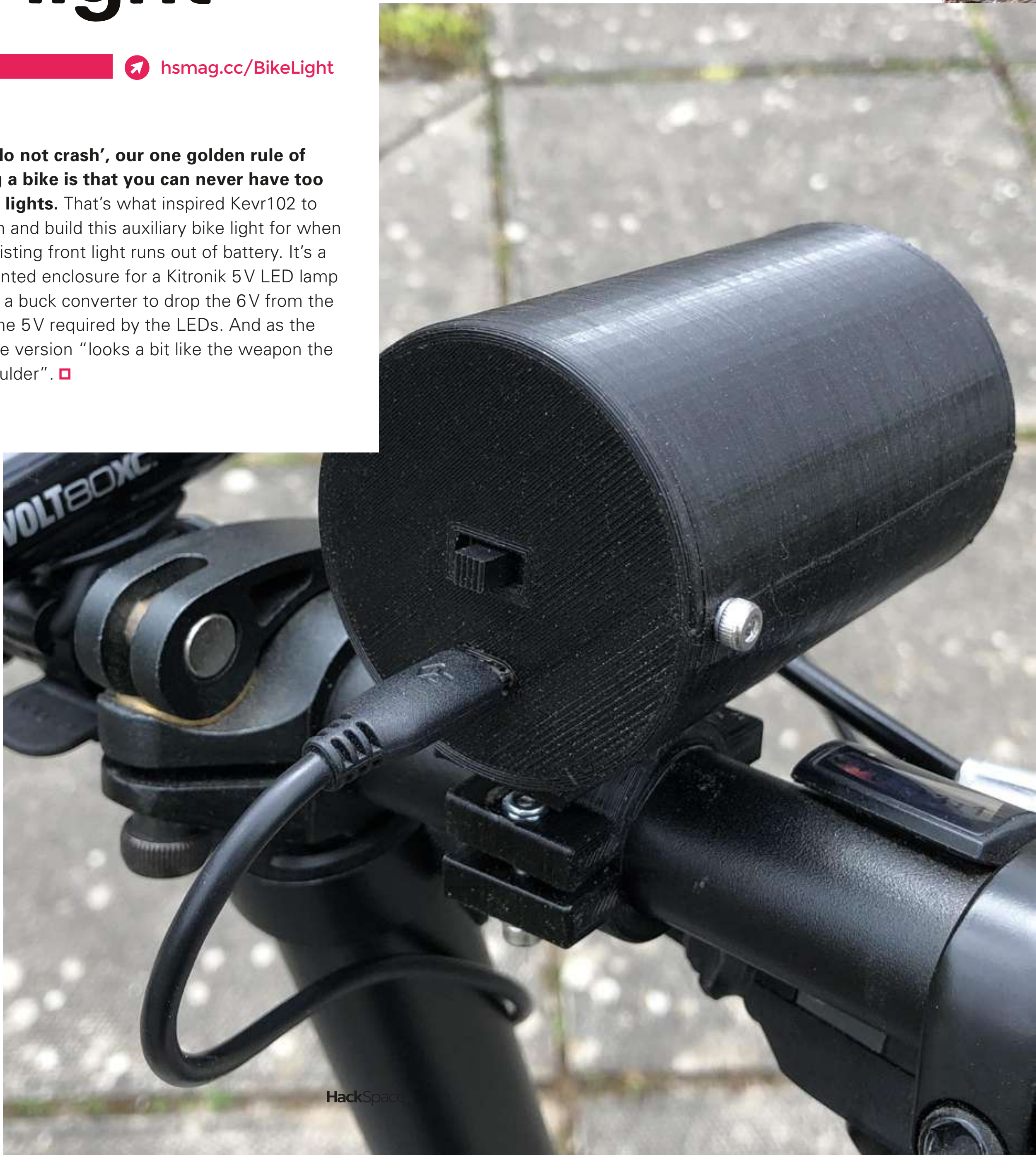
Right
Movies are stored on the Raspberry Pi's SD card, and there's an Arduino Pro Mini to provide voltage that the TV can use

FIRST NATIONAL PICTURES, INC.
Presents
Sir Arthur Conan Doyle's
Stupendous Story of Adventure
and Romance
THE LOST WORLD
By arrangement with
Watterson R. Rothacker
RADIO OFF TV/AV
POWER
VL VH UHF
AM FM
VOLUME
EARPHONE
Diana

3D-printed bike light

By **Kevr102** hsmag.cc/BikeLight

After 'do not crash', our one golden rule of riding a bike is that you can never have too many lights. That's what inspired Kevr102 to design and build this auxiliary bike light for when his existing front light runs out of battery. It's a 3D-printed enclosure for a Kitronik 5 V LED lamp kit, with the addition of a buck converter to drop the 6 V from the AA batteries down to the 5 V required by the LEDs. And as the maker notes, the square version "looks a bit like the weapon the Predator has on its shoulder". □



Right Kevr102 designed this bike light in Fusion 360

Power (5V)

3D-printed power loom

By **Fraens** hsmag.cc/PowerLoom

This issue, we've gone into the coding behind knitting and crochet, and raised a hat to the Jacquard loom, which took instructions that could've been programmed anywhere, by anyone. It wasn't Turing-complete, but it was, almost, a computer.

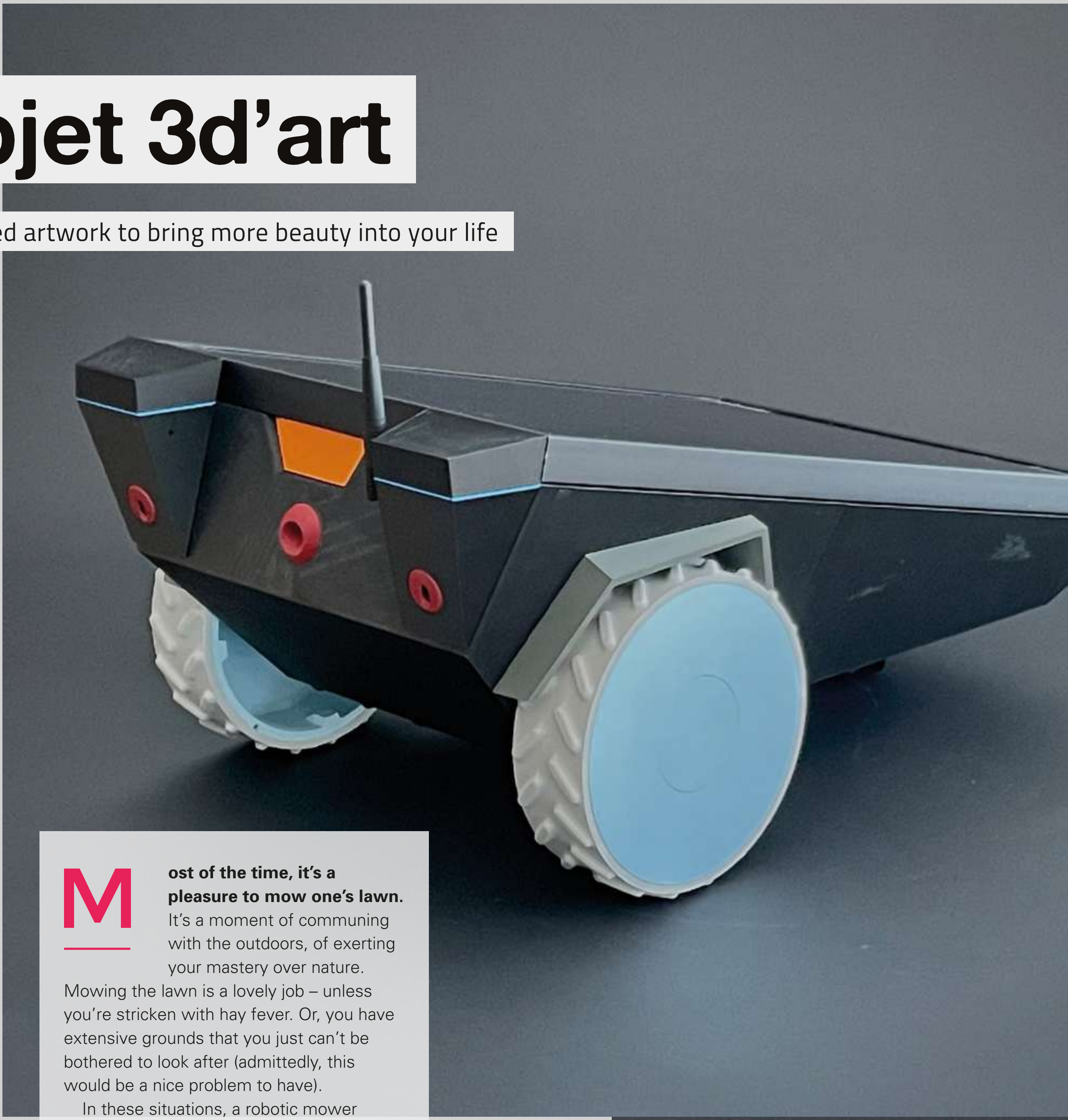
This 3D-printed model pays tribute to the power looms that laid the way for the Industrial Revolution and, indirectly, computer science.

Fraens has made the STL files available on Printables and Thingiverse, and there's a full list of materials used on his YouTube page, including screws, ball bearings, and a 12 V motor. Apparently, one of the hardest things to get right was the weight of the shuttle: too heavy and it won't move; too light and it gets snagged easily.

Right The Luddites would never have smashed a machine they could have printed themselves

Objet 3d'art

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



Most of the time, it's a pleasure to mow one's lawn. It's a moment of communing with the outdoors, of exerting your mastery over nature. Mowing the lawn is a lovely job – unless you're stricken with hay fever. Or, you have extensive grounds that you just can't be bothered to look after (admittedly, this would be a nice problem to have).

In these situations, a robotic mower comes into its own – and TGD Consulting has come up with a 3D-printed, Raspberry Pi-controlled solution: PiMowBot. Like the BEAM project featured on page 46, it takes power from the sun, so the whole top panel is a solar panel. You can get the design files from Cults3D below. □

hsmag.cc/PiMowBot

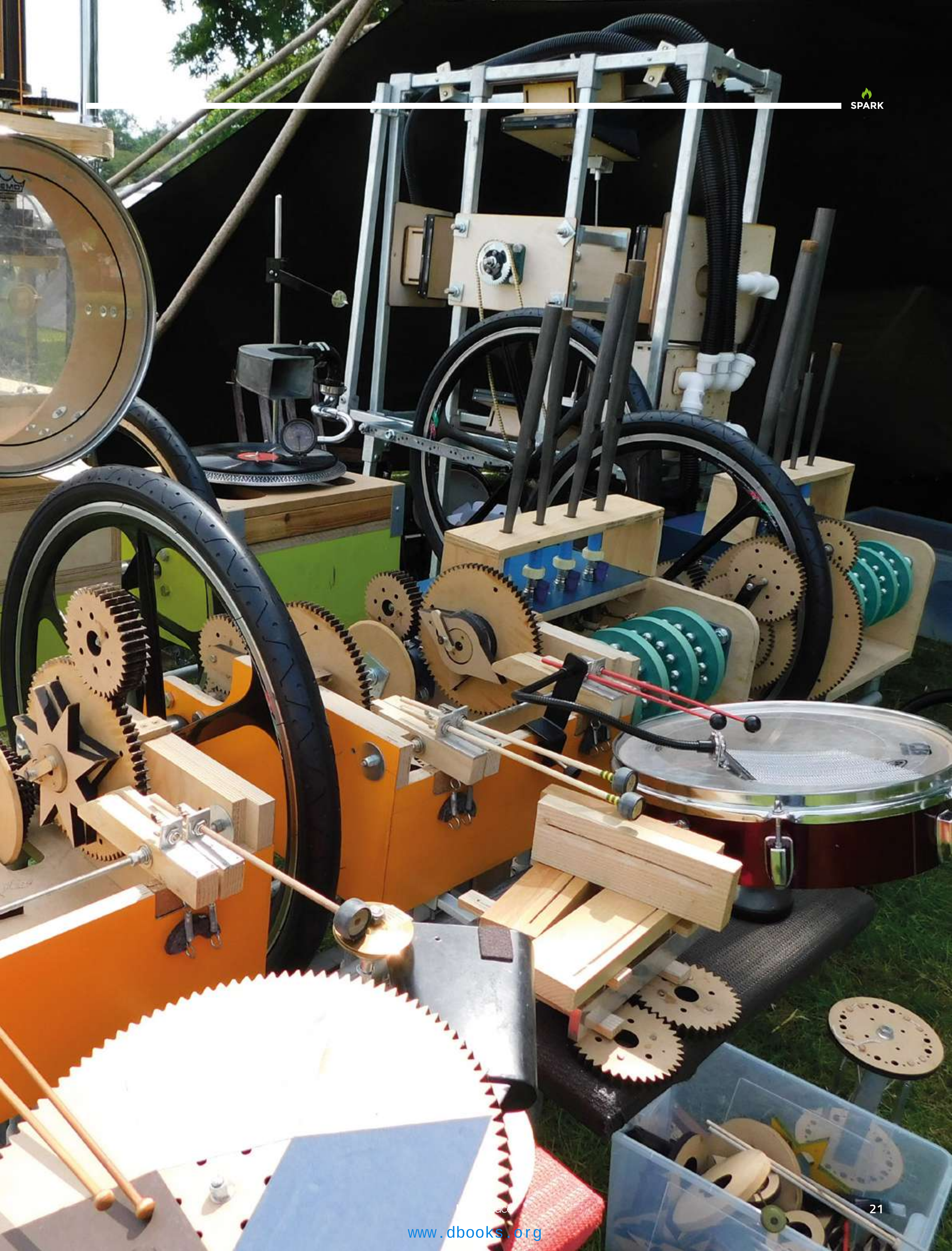
Electromagnetic Field 2022

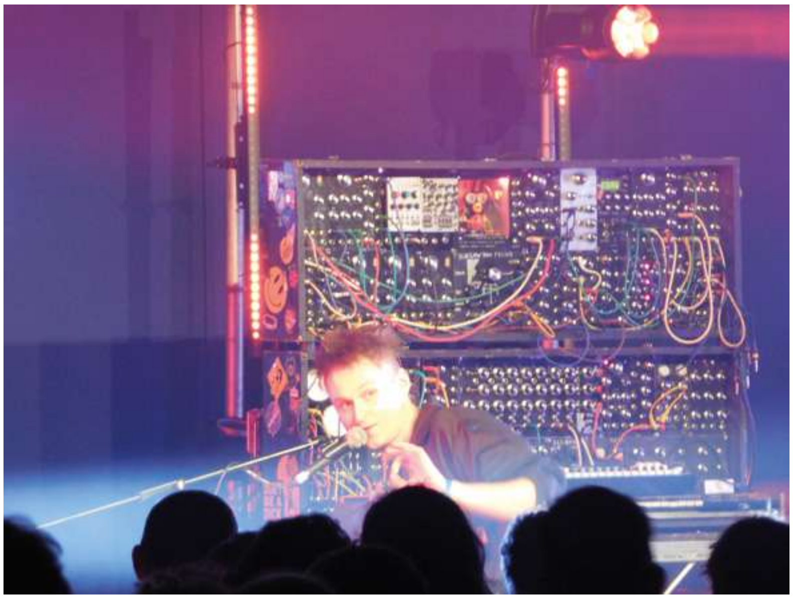
Our roving reporter visits the UK's biannual maker festival

Every two years, the British summer is lit up by Electromagnetic Field (also known as EMF Camp). This maker festival brings people together to show off makes and talk about the things they've been doing. Since 2018, this has been held in Eastnor Castle just outside Ledbury in Herefordshire (a county forever blighted by Robert De Niro's mispronunciation in the film *Ronin*: for our international audience, it's pronounced approximately hair–re–ford–shire).

"For some, it's about how they built a particular project; **for others, it's about a technology they've been working with**"

The bulk of the festival is taken up by talks. Across three stages, people from the makersphere give talks about almost any topic. For some, it's about how they built a particular project; for others, it's about a technology they've been working with. Others are, well, about almost anything. To give you a bit of a flavour, here are some that we went to see: →





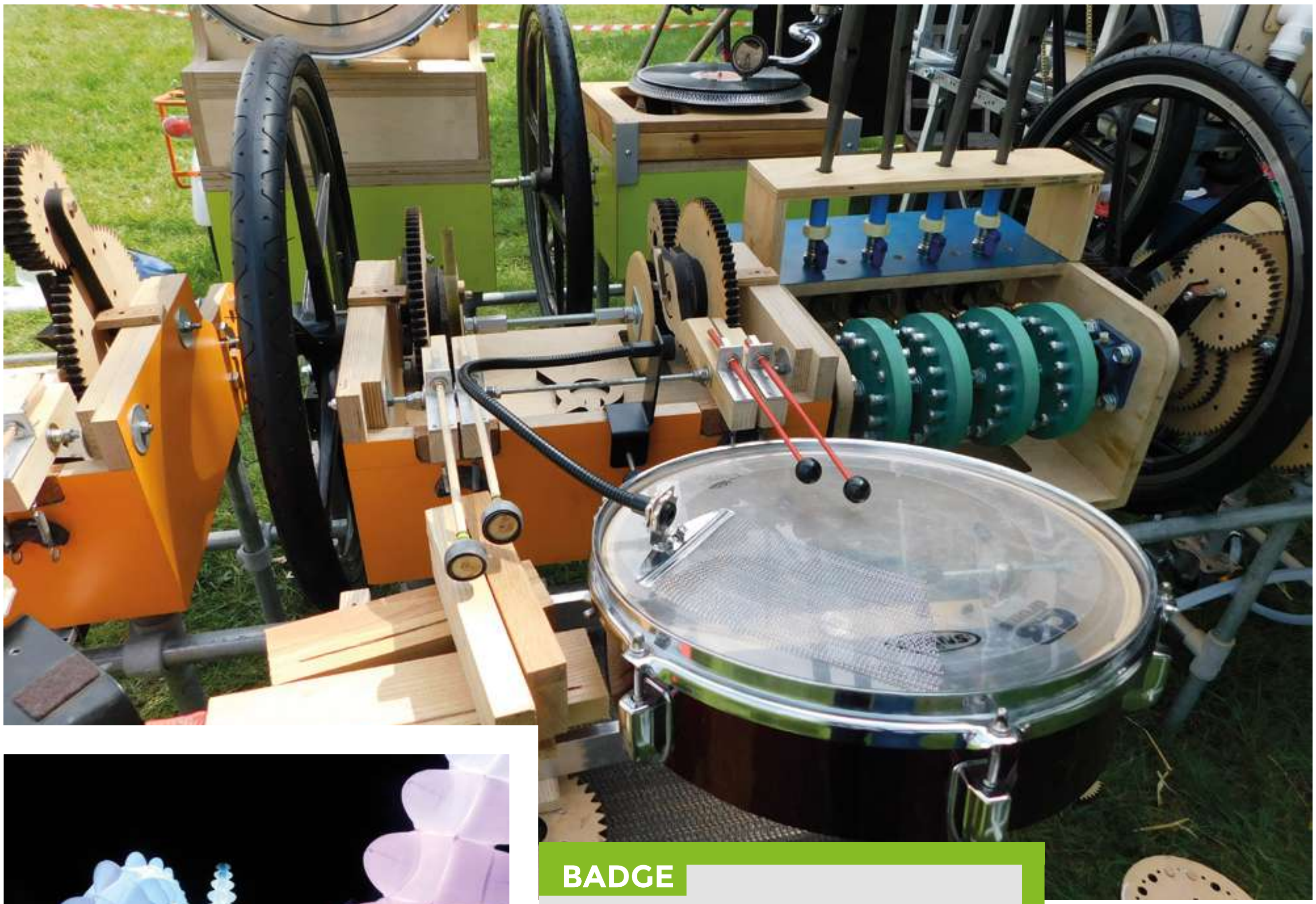
Above Sam Battle valiantly pressing on through some technical hiccups

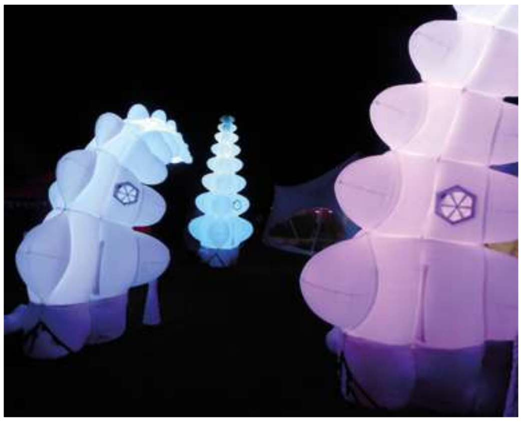
Solarpunks assemble
Terence Eden talked us through his experiences running domestic solar setups on three of the houses he's lived in, what's worked well, and why he thinks we should all be running domestic solar setups.

Wearable fire art
After being invited to join a fire art group at Burning Man, Jeff Gough decided to build a flaming top hat. Jeff talked us through the process of designing and making something that's small enough to fit on your head, but also safe and fun. He wore the result – a hat that emitted occasional puffs of burning gas throughout the talk.

Being YouTubers
James Bruton and Matt Denton (Ruth Amos was meant to join in as well, but had to cancel at the last minute due to Covid) talked us through their experiences building YouTube audiences, and bonding over their different builds of BB-8.

Rewilding human-computer interaction
Artist Tim Murray-Browne thinks that the way we work with computers has become too sanitised. While he didn't provide an ultimate solution, he's been working with dancers to create a movement-based interface. In this interface, machine learning (ML) translated movements into sound in a non-





Above The Mammoth Beat Organ mechanical modular instrument in action

BADGE

Every attendee got a programmable electronic badge. Known as 'TiDAL', this featured an ESP32 processor (with WiFi and Bluetooth), a small TFT screen, joystick, two buttons, three-axis accelerometer, three-axis magnetometer, and a battery.

You can create and install MicroPython apps from the App Store. You can see the available software at **2022.badge.emfcamp.org**. Options include a very basic oscilloscope, some games, and a weather forecast.

The choice of MicroPython and an online editor meant you didn't have to install any software to create your own apps. Just plug in your badge (which featured a USB-C socket that slotted straight into a USB port on a computer – no cable needed), point your browser to **editor.badge.emfcamp.org**, and you were ready to start coding.

This toolchain-less setup made it much easier to start creating things while on site.

linear way. A simplistic movement (such as waving an arm) doesn't in itself result in a particular sound. Instead, the computer tracks a range of movements and converts them into sound using a dancer-specific ML model.

Alongside these talks were workshops where people could learn to solder, build, sew, blacksmith, and other maker skills. The workshops took place in 'villages' – these are self-organising groups of attendees. Anyone can start a village where they and others can camp together. Some villages included larger tents and gazebos which were used as meeting places and general areas to relax and chat.

Standing apart from the rest of the main festival was Null Sector. At the end of a track that leads you through the camping field stood a tower of button-activated flaming torches, and more lasers than had any right to be in a field, lighting up the sky across the whole event. Inside was a curious mash-up of nightclub, market, escape room, and interactive art display. Oh, and there was a scanning electron microscope as well. It's the sort of place that defies easy categorisation. →

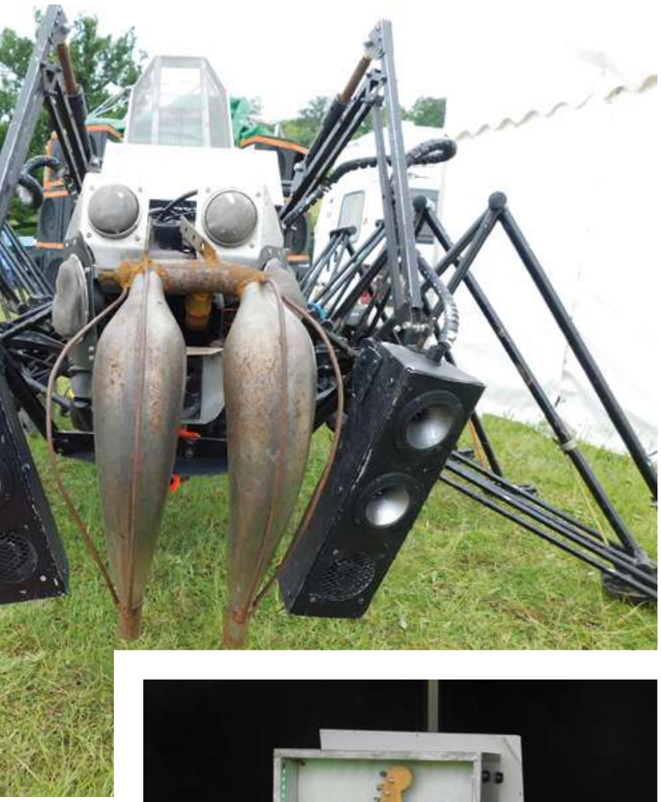
Right We're slowly building up the skills to make our own Hacky Racer

BY GEEKS FOR GEEKS

EMF Camp is unashamedly a festival run by geeks – the organisers and setup crew are all volunteers. Nowhere is this more obvious than in the infrastructure. There's power available to your tent, and WiFi coverage across the entire site. This alone would be quite impressive, but it doesn't stop there. There's also a DECT phone network that allows you to use a regular wireless phone – not a mobile, but the handset of a landline. There's an analogue phone network that lets you use regular landlines, as well as fax machines and dial-up modems. It's not just old-fashioned tech, though. There was also an experimental 4G cell.

If you're wondering what use a DECT network, or a fax-compatible analogue phone network is, then you're missing the point entirely. None of this is essential, and certainly not in a grassy field in Herefordshire. There is no need to have dial-up internet when WiFi is everywhere. But then WiFi doesn't go *gruuuuhhhh*, *hurrrrrrrrr*, *badoing*, *badoing*, *kruuurrrrr* when you connect, and that noise is, for a certain generation of geek, the sound track of our childhood. EMF Camp is a biannual chance to dust out the old computing equipment, don some rose-tinted glasses, and remember how things used to be.

Speaking of old equipment, there was an arcade machine full of games of yesteryear. A personal highlight was a 1979 version of Asteroids running on a vector display. Vector displays mostly came and went before I reached the age of arcades, so this was the first one I'd been able to play. While I've played many raster re-creations of this game, I was genuinely blown away by the vector graphics in a way I wasn't expecting. I think that perhaps my eyes have come to expect and ignore slight pixelation, so when greeted with a truly straight line from a cathode ray, my brain struggled to process the reality.



RACE-READY

One of this author's personal highlights this year was Hacky Racers. This is where people (or teams of people) build small novelty electric go-carts and race them around a field. It's broadly based on the Power Racing Series in the USA. Points are awarded for style as well as speed on the track, so designs included a giant Meccano vehicle, a cut-and-shut Little Tikes car, and an *Only Fools and Horses* yellow three-wheeler that appeared to be stuck in reverse.

I was invited by race organiser Mark Mellors to have a go in the 'Rule Zero' fire engine. The racing vehicles have a maximum width of 90 cm and length of 150 cm, which results in quite a snug fit. After wriggling in, I set about a few laps of the empty track. It was great fun. In the two weeks that I've been back home since EMF Camp, I've already bought two broken 'hoverboards' to salvage motors from in order to build my own. Look out for more on the build in future issues of HackSpace magazine.

It's difficult to sum up an event like EMF Camp in a few pages, but hopefully we've given you a flavour of what it was like. If you'll be in the UK in 2024, and are interested in technology or making things, we highly recommend you go. □

THANKS!

EMF Camp is the product of hundreds of volunteers. The people who give the talks, the people who run the bar and shop, and even the event organisers are volunteering. It's the passion that these people bring that makes the event special.

We'd like to give a heartfelt thanks to all who helped make this event great.

If you're thinking of attending in 2024, please consider volunteering, whether that's by giving a talk, running a workshop, or doing a shift in the bar.

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**

SUSTAINABLE MAKING

Alex Glow makes some very good points in your interview with her last issue. Whenever I make anything, I must go through a few iterations before I get anything to work – that means multiple PCBs made (with multiple boards shipped from the factory), multiple loads of solder, components, and, even when they do work, everything needs a battery, with all the pollution their production entails, and that battery needs electricity. That all sounds a bit grim, but one thing Alex said stands out: "It's not our fault, but it is our responsibility". We can try to do better without feeling guilty about the 60–70 years of oil extraction and plastics waste, because it's overwhelmingly the fault of a handful of companies. All we can do is try to do better, and not let the perfect be the enemy of the good.

James
Sunderland

Ben says: Alex has put together a great resource for makers who want to minimise their contribution to the forthcoming environmental apocalypse – green-ee.com. We've found it was worth a browse, even if you don't have a project in mind, as it's helped steer our thinking without our even realising it. It's easy to feel helpless, but small changes gather momentum.

ROCKET GLIDER

The swing-arm rocket glider that your writer made last issue was brilliant, and gave a perfect example of how the judicious application of 3D-printed parts to a more traditional build can be so much better than trying to 3D-print the whole thing. Dare I ask if we can hope to see more rocketry projects in the near future?

David
Germany

Ben says: As luck would have it, our Rocketry Correspondent, Jo Hinchliffe, is right now working on a flat-pack rocket as a tutorial for the magazine — readers are invited to think of a fake Swedish name for it, were it ever to arrive on the shelves of IKEA.

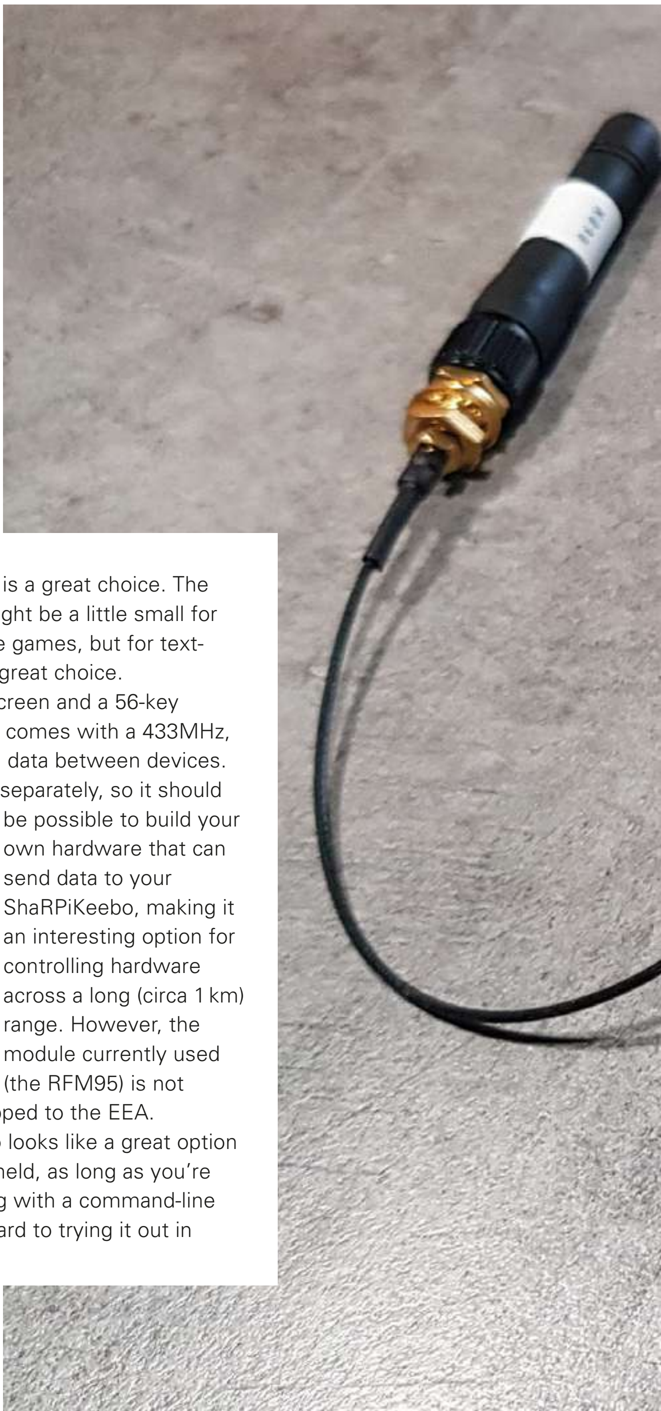
PICO W

My MagPi subscription just dropped through my door – and there's a microcontroller attached to it! Now I just need someone to come up with a load of projects for me to copy. Can anyone out there help me?

Lucy
Bedfordshire

Ben says: It was our pleasure to give away the Raspberry Pi Pico to our subscribers when it launched in January 2021; now there's another version of this affordable microcontroller board, but with added internet connectivity. And yes — as you'll have discerned by now, there's a heap of ideas for you to make with your new Pico W, starting on page 32. Have fun!

CROWDFUNDING NOW



ShaRPiKeebo

The little computer with big battery life

From $150 | **hsmag.cc/ShaRP** | **Delivery:** November 2022

Building a portable computer is easier than it's ever been. With tiny single-board computers such as Raspberry Pi, you just need to find a small keyboard and screen in the form factor you want, strap on a battery, and you're ready to go.

However, there's a big difference between building a computer and building a good computer, and that comes down to understanding your needs and picking exactly the right components for the job.

The ShaRPiKeebo is small and packs a long battery life. The former is thanks to the Raspberry Pi Zero W (or Zero 2 W), and the latter is thanks to the SHARP Memory Display. This display is almost like a cross between e-paper and an LCD screen. It takes very little battery, and is just a single colour (like e-paper), but has much faster update times (like an LCD screen). For on-the-go computing – particularly if it's mostly text-based – this is a great choice. The 400×240 pixel resolution might be a little small for graphical use beyond simple games, but for text-based things, it looks like a great choice.

"**The ShaRPiKeebo looks like a great** option for a battery-powered handheld"

As well as a low-power screen and a 56-key keyboard, the ShaRPiKeebo comes with a 433MHz, long-range radio for sending data between devices. You can get these modules separately, so it should be possible to build your own hardware that can send data to your ShaRPiKeebo, making it an interesting option for controlling hardware across a long (circa 1 km) range. However, the module currently used (the RFM95) is not CE-marked, so can't be shipped to the EEA.

Overall, the ShaRPiKeebo looks like a great option for a battery-powered handheld, as long as you're mostly interested in working with a command-line interface, and we look forward to trying it out in real life. □

BUYER BEWARE

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

Left
The SharRPiKeebo is small enough to fit in one hand

enviro goes wireless

Flexible Power	Alkaline NiMH Micro-USB LiPo
Pico W Aboard	Wireless, standalone boards
Long Battery Life	Run for weeks or months on batteries
Human Friendly	Working examples and custom libraries
Real Time Clock	Efficient sleep and scheduled readings

https://pimoroni.com/envirow

Automation 2040 W

RELAY AND IO CONTROL UP TO 40V WIRELESSLY FOR HOME, OFFICE AND INDUSTRIAL AUTOMATION

PICO W ABOARD

https://pimoroni.com/auto2040w

INKY FRAME 5.7

Your seven colour E Ink pal who's fun to be with

https://pimoroni.com/inkyframe

PICO W ABOARD

PIMORONI

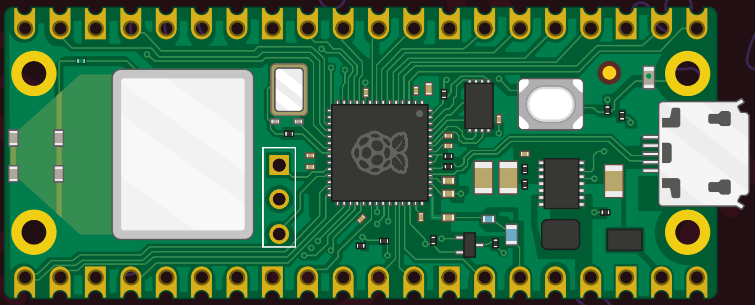
HackSpace
TECHNOLOGY IN YOUR HANDS

LENS

HACK | MAKE | BUILD | CREATE

Uncover the technology that's powering the future

PICO W PROJECTS



The Pico W is, rather predictably, a Raspberry Pi Pico but with added wireless networking. For just $6, you get a dual-core Arm Cortex M0+ microcontroller running at up to 133MHz with 264kB of RAM and 2MB of flash storage, and on top of this is an Infineon CYW43439 which adds wireless networking (the chip also supports Bluetooth Classic and Low Energy, but at present there's no firmware support for this).

All the things we've come to know and love from the original Pico are still there. There's still a huge bundle of input and output options, including Programmable I/O. In fact, the only slight change is that the on-board LED is now attached to the wireless chip rather than the main microcontroller (see box on page 35).

Not only does it have all the old features, but Pico W is in the same form factor as Pico, so adding networking to a project is usually just a case of swapping a Pico for a Pico W.

In this article, we're going to take a look at some great projects to help you get started and get the most out of Pico W. →

FEATURE

JOIN THE ONLINE COLOUR-CHANGING LED COMMUNITY

Add some light to your electronics

While Pico W is very similar to Pico, you will need to get the special wireless version of the MicroPython firmware. You can download this from hsmag.cc/DocMicroPyth.

Make sure that you select the version for Pico W, as the version for the regular Pico won't work. You should have a UF2 file, and this can be loaded onto Pico W.

First, make sure that Pico W is unplugged from USB, then hold down the BOOTSEL button and plug Pico W into your computer using a micro USB to USB cable, then release the button. You should notice a new USB drive appear on your computer. You can drag and drop the UF2 firmware file onto this drive. Once it's copied over, the drive will disappear and Pico W will reboot into MicroPython.

We recommend Thonny for working with MicroPython (though you can connect over serial as well). If you haven't already got this, you can download it from **thonny.org** – it works on Windows, Mac, and Linux (including Raspberry Pi). Open up Thonny and it should automatically detect and connect to MicroPython running on Pico W. In the bottom box, you should see a line something like this confirming it:

```
MicroPython v1.19.1-88-g99c258977 on 2022-06-30;
Raspberry Pi Pico W with RP2040
```

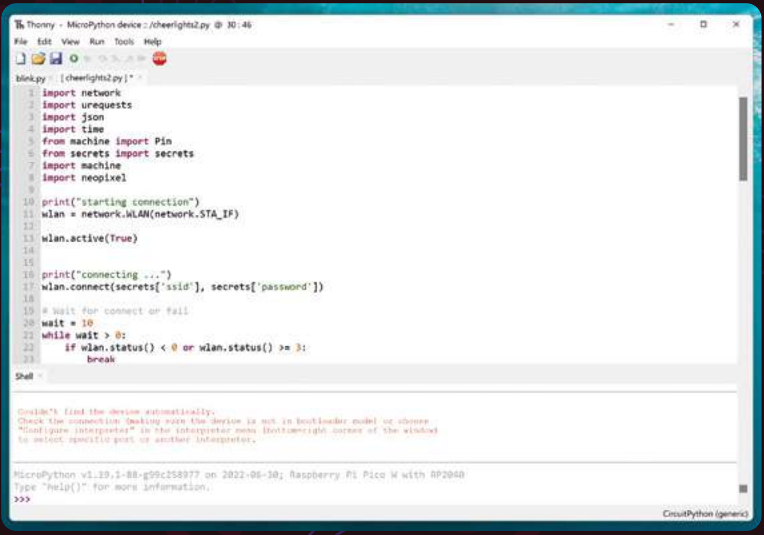
If you see that, then you're all set up and ready to go.

For our first internet-connected project, we're going to create a CheerLight. The idea of this is to have a colour-changing LED linked into a global system of setting colours. You can set the colour by tweeting **@cheerlights** and including a colour name in the tweet. This will change LEDs across the world to the colour you name. In an increasingly divided world, having an LED change colour is just a little reminder of how we're all connected. This can sit in the corner of your office and remind you, in a small way, that you're not alone. You're part of a worldwide community of makers.

We'll be using WS2812B LEDs (also known as NeoPixels) as our light. These can be programmed to display a wide range of colours. These LEDs come in a few different forms, but most commonly strips. You can use any form you like.

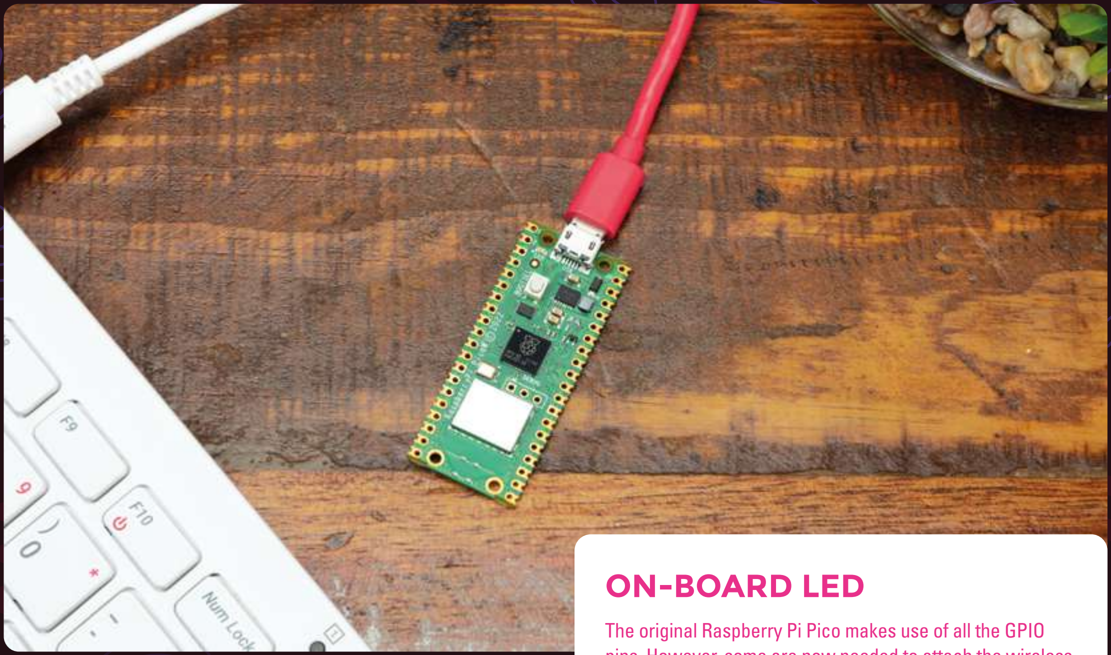
First, let's wire it up. Your LEDs should have three inputs, usually in the order 5V, Data In, and GND. 5V and GND should be connected to VBUS and GND on Pico W,

Below The Thonny MicroPython IDE makes it easy to program and transfer files to your Pico


Thonny - MicroPython device :: /cheerlights2.py @ 30 : 46
File Edit View Run Tools Help
blink.py
[cheerlights2.py] *
Shell
CircuitPython (generic)

```
import network
import urequests
import json
import time
from machine import Pin
from secrets import secrets
import machine
import neopixel

print("starting connection")
wlan = network.WLAN(network.STA_IF)

wlan.active(True)


print("connecting ...")
wlan.connect(secrets['ssid'], secrets['password'])

# Wait for connect or fail
wait = 10
while wait > 0:
    if wlan.status() < 0 or wlan.status() >= 3:
        break
```

```
Couldn't find the device automatically.
Check the connection (making sure the device is not in bootloader mode) or choose
"Configure interpreter" in the interpreter menu (bottom-right corner of the window)
to select specific port or another interpreter.

MicroPython v1.19.1-88-g99c258977 on 2022-06-30; Raspberry Pi Pico W with RP2040
Type "help()" for more information.
>>>
```



Left You can program Pico W from most computers, including a Raspberry Pi 400

while Data In should be on GPIO 18 (you can use another GPIO pin if you like, but you'll need to change it in the code). You can solder these on or connect them with jumpers and crocodile clips.

CODING IT UP

We need to connect Pico W to the wireless network. Obviously, this needs an SSID (network name) and password. You can put these directly in your MicroPython code, but this can be a bit inconvenient because it means you have to be a bit careful about uploading your code to an online code repository, and it means you have to type the details into the code each time you take them from a repository onto your device. Instead, we're going to use a secrets file. In this, we'll put all the things that we don't want to be publicly known. It'll include the WiFi login details, as well as secret keys for things we'll be using in other examples.

```
secrets = {
'ssid': 'XYZ',
'password': 'XYZ',
'aio_key': 'XYZ',
'aio_username': 'XYZ',
'ifttt_key':'XYZ',
}
```

Enter that in Thonny and press Save, select 'MicroPython device', and call it **secrets.py**.

Now, let's connect to the network:

```
import network
import urequests
import json
import time
```

```
from machine import Pin
from secrets import secrets
import machine
import neopixel

ssid = secrets['ssid']
password = secrets['password']

print("starting connection")
wlan = network.WLAN(network.STA_IF)

wlan.active(True)

print("connecting ...")
wlan.connect(ssid, password)

wait = 10
```

→

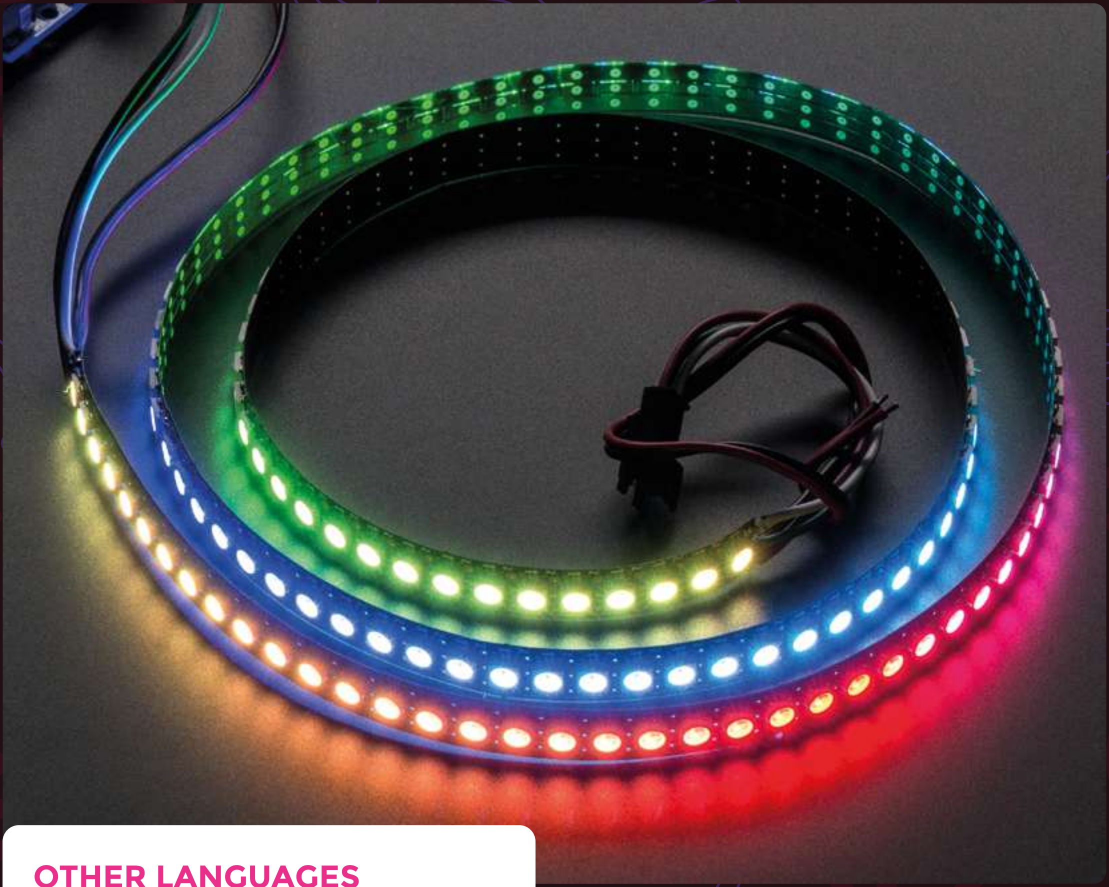
ON-BOARD LED

The original Raspberry Pi Pico makes use of all the GPIO pins. However, some are now needed to attach the wireless module. One solution was to reassign GPIO 25. On Pico, this is attached to the on-board LED, but on Pico W, it's attached to the wireless module.

The LED remains, but now it's controlled by the WiFi module rather than the main microcontroller. Fortunately, much of this is hidden by MicroPython, and there is a pseudo pin called 'LED'. You can blink the LED with:

```
import machine
import time
led = machine.Pin("LED", machine.Pin.OUT)
while True:
         led.toggle()
         time.sleep(0.5)
```

This pin should behave as any plain pin does, but it can't be used with PIO, PWM, or other special functions.



Right WS2812B LEDs come in all sorts of shapes and sizes, but they're all controlled in the same way

OTHER LANGUAGES

We love MicroPython because it lets us get projects up and running really quickly. However, there are times when another language is more appropriate.

C/C++ The official C/C++ SDK is available at: **hsmag.cc/PicoSDK**.

This supports networking through the LWIP networking stack and gives you the ability to work with the networking at a very low level. There are some examples to help you get started at **hsmag.cc/PicoExamp**.

At the time of writing, this is the only other language to support the wireless networking. However, we expect there to be other options very shortly.

Arduino Since the Arduino port of Pico includes the C/C++ SDK, it should be possible to use this with WiFi once the new SDK is brought in. However, at the time of writing, we were not able to test this.

CircuitPython Adafruit's flavour of Python for small devices typically gets new features quickly, so we would expect WiFi on Pico W to work very soon. It's likely to be based on the requests module, which is functionally very similar to urequests, which we use here.

Rust Pico has proved popular with the embedded version of this memory-safe language, and there is an active community working with this.

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

# Handle connection error
if wlan.status() != 3:
    raise RuntimeError('wifi connection failed')
else:
    print('connected')

status = wlan.ifconfig()
print( 'ip = ' + status[0] )
```

We also need a bit to set up our LEDs:

```
pixels = neopixel.NeoPixel(machine.Pin(18), 10)
```

Now that we're online, we need a way of grabbing the colour. There's a back-end system that grabs colour changes from Twitter and posts them to the URL **api.thingspeak.com/channels/1417/field/2/last.json** in JSON format. We just need to download this and extract the 'field2' data. This is hex encoded, so we need to extract the red, green, and blue values from this and set our LED accordingly.

As long as we're connected to the internet, we can use the urequests module to do just this, and the json module will convert a JSON string to a Python dictionary. The following code grabs the latest colour from the URL and returns a colour:

```
def get_colour():
    url = "http://api.thingspeak.com/channels/1417/
field/2/last.json"
    try:
        r = urequests.get(url)
        print(r.status_code)
        if r.status_code > 199 and r.status_code <
300:
            cheerlights = json.loads(r.content.
decode('utf-8'))
            print(cheerlights['field2'])
            red_str = '0x' + cheerlights['field2']
[1:3]
            green_str = '0x' + cheerlights['field2']
[3:5]
            blue_str = '0x' + cheerlights['field2']
[5:7]
            colour = (int(red_str), int(green_str),
int(blue_str))

            r.close()
            return colour
        else:
            return None
    except Exception as e:
        print(e)
        return None
    return colour
```

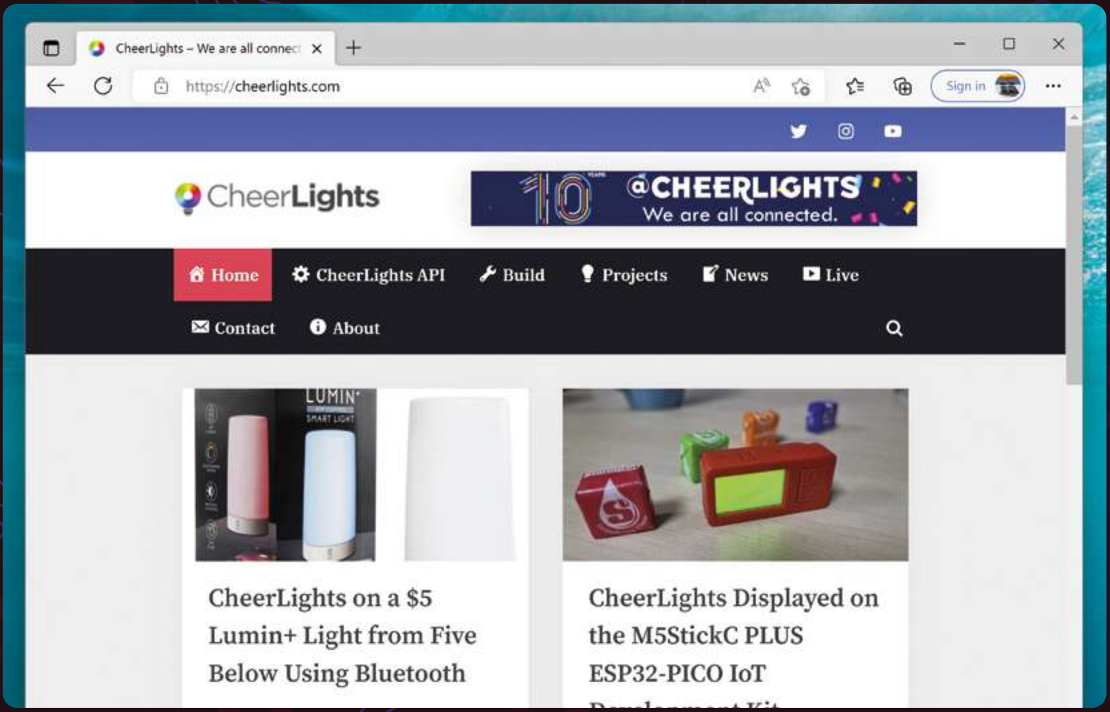
We have to do a bit of string manipulation to get the colours out of the string. First, we use slicing to get the two hexadecimal digits that relate to the colour, then we add '0x' to the start of these two digits. The '0x' lets Python know that the string contains hexadecimal characters, so we can then use `int()` to convert this new string into an integer.

Finally, we just need a loop that checks the colour every minute and updates the LED.

```
while True:
    print("getting colour")
    colour = get_colour()
    if colour is not None:
        pixels.fill(colour)
        pixels.write()
    time.sleep(60)
```

You can get the full code at **hsmag.cc/cheerlight**.

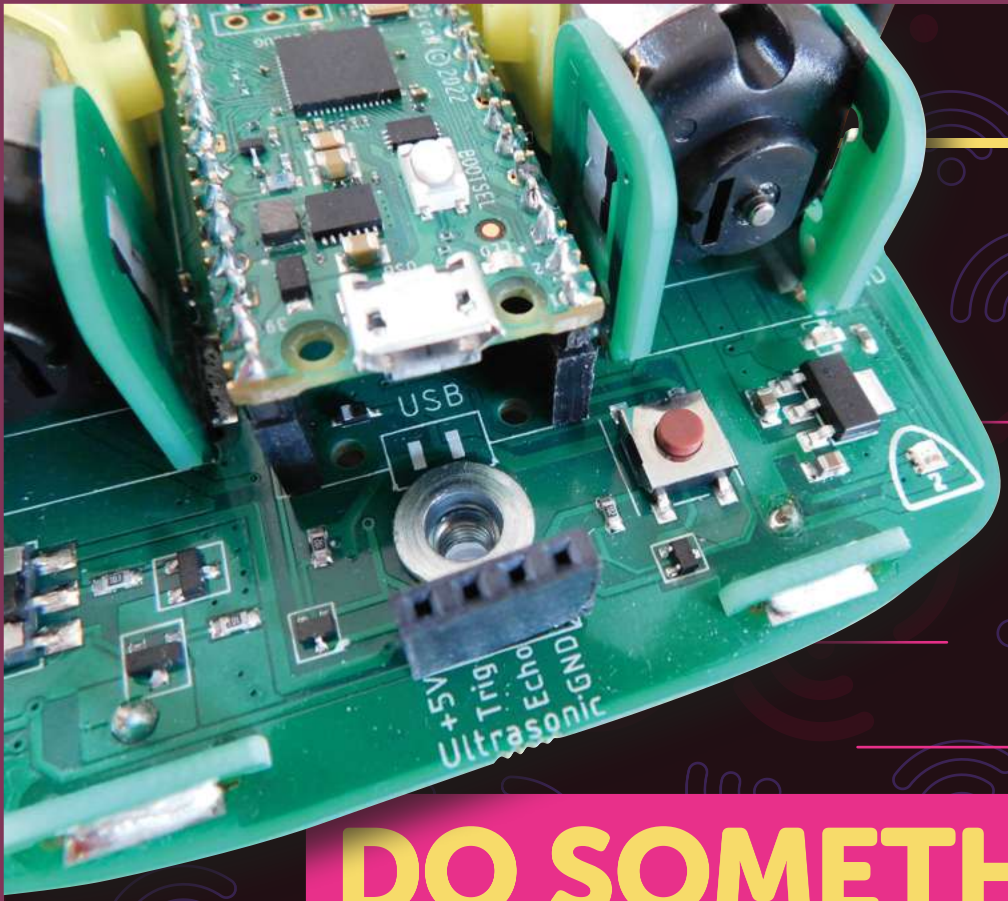
Load it up to your Pico and you should see the LED occasionally change colour. Alternatively, tweet to **@cheerlights** and send a message to lots of devices – including some Pico Ws – to change colour. →



CheerLights on a $5 Lumin+ Light from Five Below Using Bluetooth

CheerLights Displayed on the M5StickC PLUS ESP32-PICO IoT

Left
Find out more about CheerLights at cheerlights.com



DO SOMETHING BUTTON

Use Pico W to control the cloud

Now that we've got data from the internet, let's send some to it and create a button that sends a message to the internet to do something. Do what, you ask? Well, do almost anything! The magic glue in this project is If This Then That (IFTTT), an online service that connects causes to events. This means that you can create a thing that triggers a huge range of possible events. In this example, we'll use it to control our heating using a Hive thermostat. However, IFTTT works with a huge range of different web-connected devices, so you can easily modify this to work with any device that's supported. Take a look here for a full range of supported systems: **ifttt.com/explore/services**.

If This Then That lets you create very simple programs without having to program anything. The programs are known as 'applets' and all need to have the form 'If this event happens, then trigger this action'. There's support for lots of different possible events and actions. The particular event that we'll be using is known as 'Maker Webhooks'. These create a URL that, if you visit it, will trigger an 'If This'.

The first part of this is to create an applet in IFTTT which we will trigger from Pico. Head to **ifttt.com**, create a free account (if you don't already have one), and sign in. You can then press Create to create a new applet. The first part to configure is the If This part. Press Add and search for webhooks. Inside this option, select Receive a Web Request, then give it an event name (you'll need to enter this in the program later on). Press Create Trigger to finish the first part.

Second, we create the Then That part. Again, click on Add. Here, you can configure your applet to do whatever you want. We've configured ours to send us an email, but you might want to browse through the list of options to see if there's some particular service you'd like to trigger.

Now, let's take a look at the Pico W side of things.

The only wiring is to connect a button. One side of this should go to 3V and the other side to GPIO pin 0.

We need to update our secrets file with a couple of bits of information for IFTTT: the event name and key. Technically, the event name doesn't have to be secret, but the secrets file is a good place to put it.

Above
This code also works with the user button on the Kitronik Autonomous Robotics Platform for Pico

Add the following inside the **secrets** dictionary in the secrets file:

```
'ifttt_key': 'XXXX',
'ifttt_eventname': 'web_button',
```

You can find out your key by going to the URL **ifttt.com/maker_webhooks/settings**. There, you should see a section like:

```
URL
https://maker.ifttt.com/use/<your key here>
```

Where we've written <your key here>, you'll find your key and you can copy and paste that into the secrets file.

You can see the full code at **hsmag.cc/picow_ifttt**.

To start, it's exactly the same as the previous example, as we just need to connect to the internet.

Webhooks are triggered by visiting a particular URL. You can also attach up to three bits of data to your webhook. We can do this by adding **"?value1=<data>&value2=<data>&value3=data"** to the end of the URL. We won't actually use this in our example, but you may want it if you use this function in your own code.

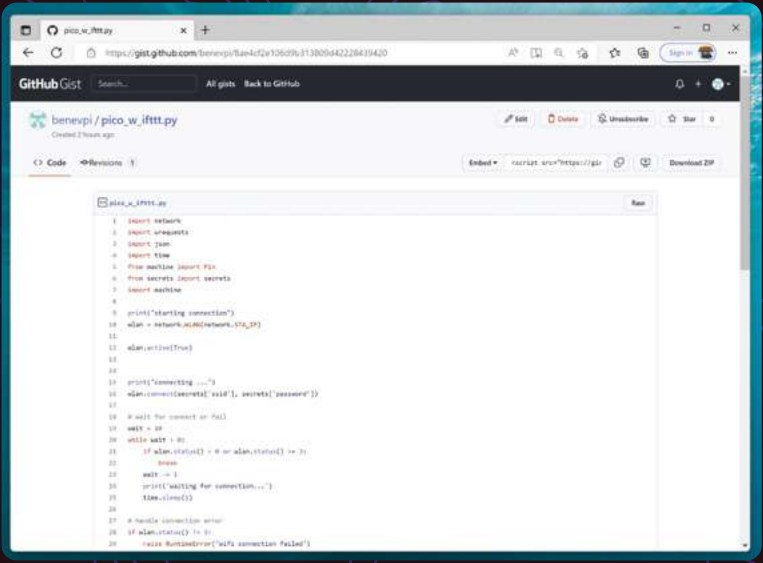
```
def ifttt_webhook(event, key, values=None):
    url="https://maker.ifttt.com/trigger/"+event+"/
with/key/"+key
    if values is not None:
        url = url + '?'
        for counter, value in enumerate(values,
start=1):
            if counter < 4:
                if counter > 1:
                    url=url+'&'
                url=url+"value"+str(counter)+"="+val
ues[counter-1]
    try:
        r = urequests.get(url)
        if r.status_code>199 and r.status_code<300:
         r.close()
            return 0
        else:
         r.close()
            return -1
    except:
        r.close()
        return -2
```

As you can see, this builds up a URL with the relevant values for event and key. You can also submit up to three values in a list, and these will also be passed to the webhook (IFTTT doesn't permit more than three). This uses the same urequests module that we used in the previous project.

Finally, we just need a bit of code to trigger this when a button is pressed.

```
button = machine.Pin(0, machine.Pin.IN, machine.Pin.
PULL_DOWN)

current_state = 0
while True:
    new_state = button.value()
    if new_state != current_state:
        if new_state == 1:
            ifttt_webhook(secrets['ifttt_
eventname'], secrets['ifttt_key'], values=None)
        current_state = new_state
    time.sleep(0.5)
```

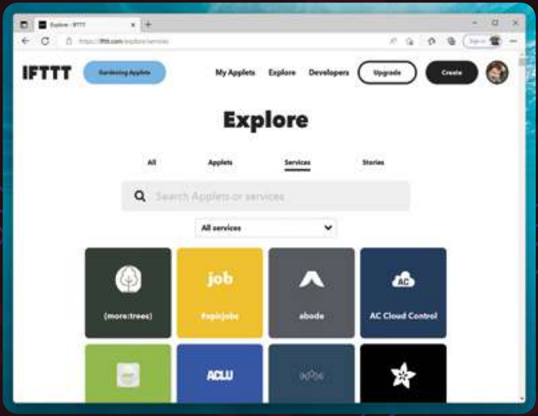
This project is surprisingly powerful because of the range of integrations available from IFTTT. Exactly what you can use it for depends entirely on what online services you use. Potentially, you can use it to log particular events, control your heating, send alerts, even open doors and turn lights on. →



```
import network
import urequests
import json
import time
from machine import Pin
from secrets import secrets
import machine

print("starting connection")
wlan = network.WLAN(network.STA_IF)

wlan.active(True)


print("connecting ...")
wlan.connect(secrets['ssid'], secrets['password'])

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

# handle connection error
if wlan.status() != 3:
    raise RuntimeError('wifi connection failed')
```

Above
You can get the code from hsmag.cc/picow_ifttt

Below
Take a look at the available services on IFTTT to see if there are any that you use already



WEB SERVER ROBOT

Control a buggy from the web

So far, we've looked at getting data from the internet and sending data to the internet, but in both cases, we connected to an external server. However, there are times where we don't want to send data to an external system, but use a wireless connection to send data from one computer to another.

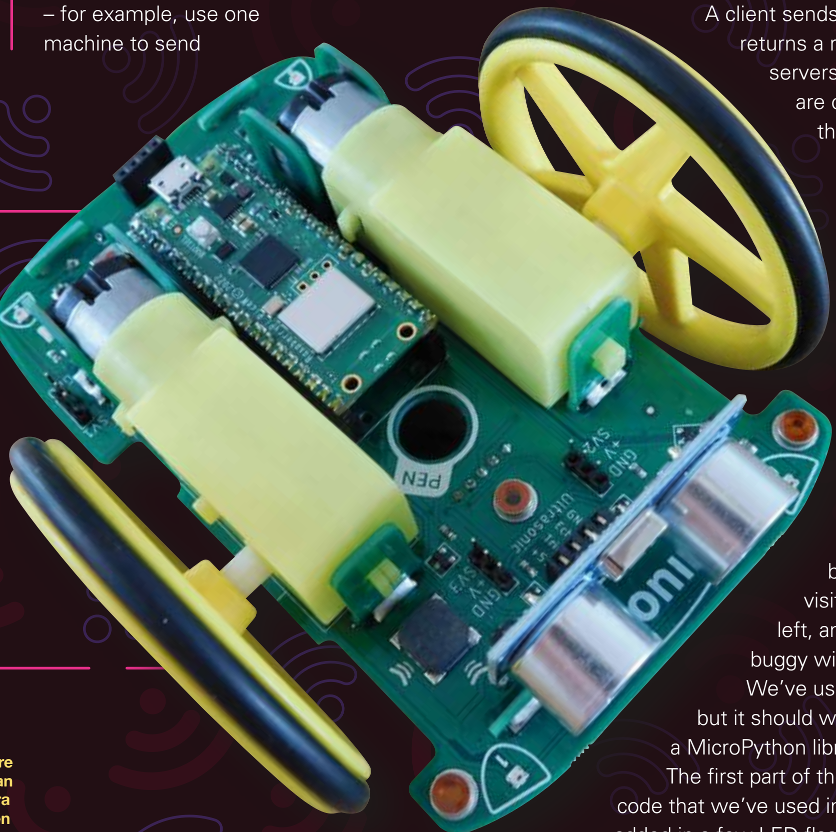
In this case, you can go via the internet – for example, use one machine to send data to Adafruit IO and another to read data from the same feed. Doing this is reliable and you can keep a trail of data, should you need to restart one of the machines. However, it's also slow, and there are times when you might not want to trust your data to the internet.

Fortunately, there is a solution. Hypertext Transport Protocol (HTTP) – the protocol that the web is built on – is actually a very simple text-based protocol. A client sends a request to a server which returns a response. The sort of web servers that power most websites are complex bits of software, but this complexity comes from performance and the features that allow you to automatically place dancing baby GIFs on your web pages and suchlike.

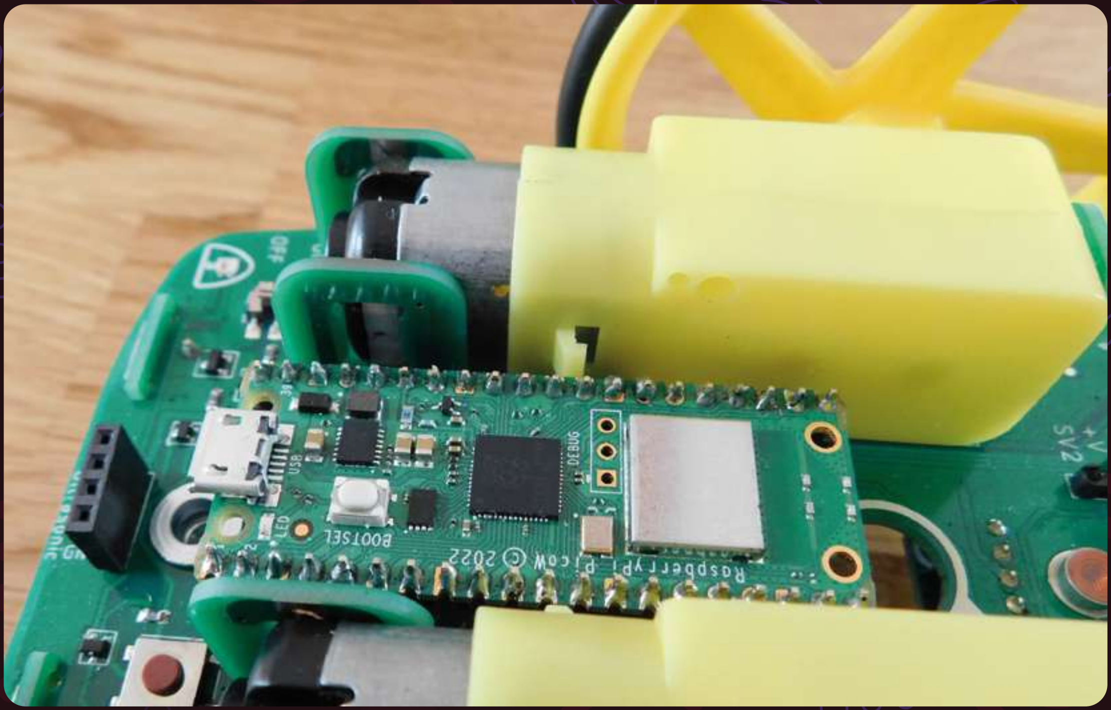
We can do away with all that and have a really simple web server that just controls a buggy. We want it to serve up the same web page whatever address the user puts in (which displays a simple remote control), and if they visit **/buggy/forward**, the buggy will move forward. If they visit **/buggy/left**, the buggy will go left, and if they visit **/buggy/right**, the buggy will go right.

We've used a Kitronik Pico Buggy for this, but it should work with any robot that you have a MicroPython library for.

The first part of the code is the same WiFi connect code that we've used in previous examples, but we've added in a few LED flashes on the buggy's four RGB



Right
There's far more hardware than we've used – you can add extra servos, an extra distance sensor, and even a pen for drawing

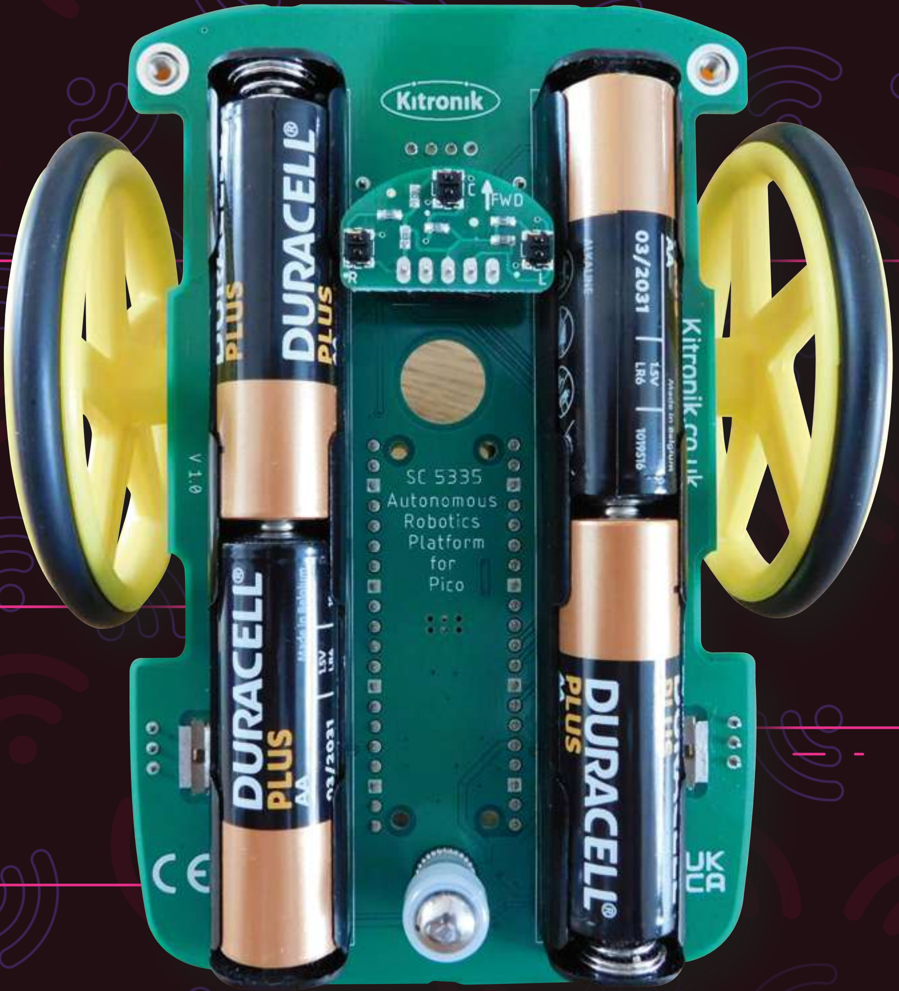


Left This robot was designed for Pico, but since Pico W is pin-compatible, it slots right in and works fine

LEDs so that we can see what's going on when we're not plugged into a computer.

```
import network
import socket
import time
from machine import Pin
from secrets import secrets
from PicoAutonomousRobotics import
KitronikPicoRobotBuggy

print("starting buggy")
buggy = KitronikPicoRobotBuggy()
print("started buggy")

buggy.setLED(0, buggy.YELLOW)
buggy.show()

ssid = secrets['ssid']
password = secrets['password']

print("starting connection")
wlan = network.WLAN(network.STA_IF)

buggy.setLED(1, buggy.YELLOW)
buggy.show()

wlan.active(True)

buggy.setLED(2, buggy.YELLOW)
buggy.show()
print("connecting ...")
wlan.connect(ssid, password)

buggy.setLED(3, buggy.YELLOW)
buggy.show()

# Wait for connect or fail

wait = 10
while wait > 0:
    if wlan.status() < 0 or wlan.status() >= 3:
        break
    wait -= 1
    print(wait)
    if (wait % 2 == 0):
        print("here")
        buggy.setLED(0, buggy.BLUE)
    else:
        buggy.setLED(0, buggy.RED)
    buggy.show()
    print('waiting for connection...')
    time.sleep(1)

# Handle connection error
if wlan.status() != 3:
    raise RuntimeError('wifi connection failed')
else:
    print('connected')
    buggy.setLED(0, buggy.GREEN)
    buggy.setLED(1, buggy.GREEN)
    buggy.setLED(2, buggy.GREEN)
    buggy.setLED(3, buggy.GREEN)
```

→



Right Power comes from four AA batteries, which means this can run without any cables

```
    buggy.show()

status = wlan.ifconfig()
print( 'ip = ' + status[0] )
# Open socket
addr = socket.getaddrinfo('0.0.0.0', 80)[0][-1]
```

Next, we need a variable that holds a string with the HTML response. We've kept this really short and simple, but it could easily be made more aesthetic, or include more information, if you want.

```
response = """<!DOCTYPE html>
<html>
<head> <title>Pico W</title> </head>
<body> <h1>Pico W</h1>
<p><a href="/buggy/forward">forward</a><br>
<a href="/buggy/left">left</a></br>
<a href="/buggy/right">right</a></p>
<h2> The next object in front is at %s</h2>
</body>
</html>
"""
```

We won't go into HTML in detail here as we are focusing on the Pico W, but links are between `<a href=link>` and `</a>`.

You might also notice the final line with `The next object in front is at %s`. Our buggy has an ultrasonic distance sensor that we can use. In Python, we can use `%s` to mark a point in a string where we want to include another string. We'll take a look at how to do that in a little bit.

HTTP runs on top of Transport Control Protocol (TCP). TCP has a range of numbered ports that we can bind to using the socket module.

Once we've created a socket object and bound it to a port and an address (since a device can in theory have multiple IP addresses, sockets are defined by both IP address and port), we can wait for an external device to make a TCP connection.

TCP connections themselves are really just a two-way text stream. You can use them to send and receive all sorts of data in much the same way that you can use the serial port. However, with HTTP, we use it in a very specific way. The client makes a request that is text in a very specific format defining what page they want to access, as well as some details about the client and its capabilities. Again, we won't go into this in depth.

The important thing for us is that we take a look at the request, see if it's for the pages **/buggy/forward**, **/buggy/left**, or **/buggy/right**, and if it is, set the motors appropriately, then whatever page the user requested, returns the same HTML response defined above. The code for all this is:

```
def check_and_move(request, find_str, buggy, left_
motor, right_motor):
    request = str(request)
    if (request.find(find_str) == 6):
        buggy.motorOn("l","f",left_motor)
        buggy.motorOn("r","f",right_motor)
        time.sleep(1)
        buggy.motorOn("l","f",0)
        buggy.motorOn("r","f",0)

# Listen for connections
while True:
    try:
        cl, addr = s.accept()
        print('client connected from', addr)
        request = cl.recv(1024)
        print(request)

        request = str(request)
        check_and_move(request, '/buggy/forward',
buggy, 100,100)
        check_and_move(request, '/buggy/left',
buggy, 100,0)
        check_and_move(request, '/buggy/right',
buggy, 0,100)
        cl.send('HTTP/1.0 200 OK\r\nContent-type:
text/html\r\n\r\n')
        cl.send(response % str(buggy.
getDistance("f")))
        cl.close()

    except OSError as e:
        cl.close()
        print('connection closed')
```

STARTING AUTOMATICALLY

You probably want your buggy to automatically start the program when you turn it on – it'd be a bit of a faff if, every time, you had to plug it into your computer and start Thonny just to launch the control program. Fortunately, it's really easy to do this in MicroPython – just save your code to the MicroPython device and call it **main.py**. If you do that, when you turn it on, it'll automatically start the code.

You still need to know what IP address the machine's on. Generally, routers will assign the same IP address to a device when it reattaches to the network, so you can run it once tethered and then try this again. If this fails, you'll need to log into your router to see what IP address is assigned – take a look at your router's documentation for details.

As you can see, we first have to accept a TCP connection, then we can receive data. Our **check_and_move** function takes the stringified version of the request as well as the page location we want to check and details of the motor settings. If it finds the **find_str** at position 6 in the request, then it knows that this is the page that the user requested. It turns the motors by the specified amount, waits for one second, then stops the motors.

OUR BUGGY HAS AN ULTRASONIC DISTANCE SENSOR THAT WE CAN USE

After this has happened, the main loop then sends the HTML to the client. Notice that it first sends a header line that has a bit of information about the data. This is required by HTTP. We can insert the distance into our HTML with the line:

```
cl.send(response % str(buggy.getDistance("f")))
```

The `%` command is used to substitute the `%s` in the original string with a new string – in this case a readout from the distance sensor.

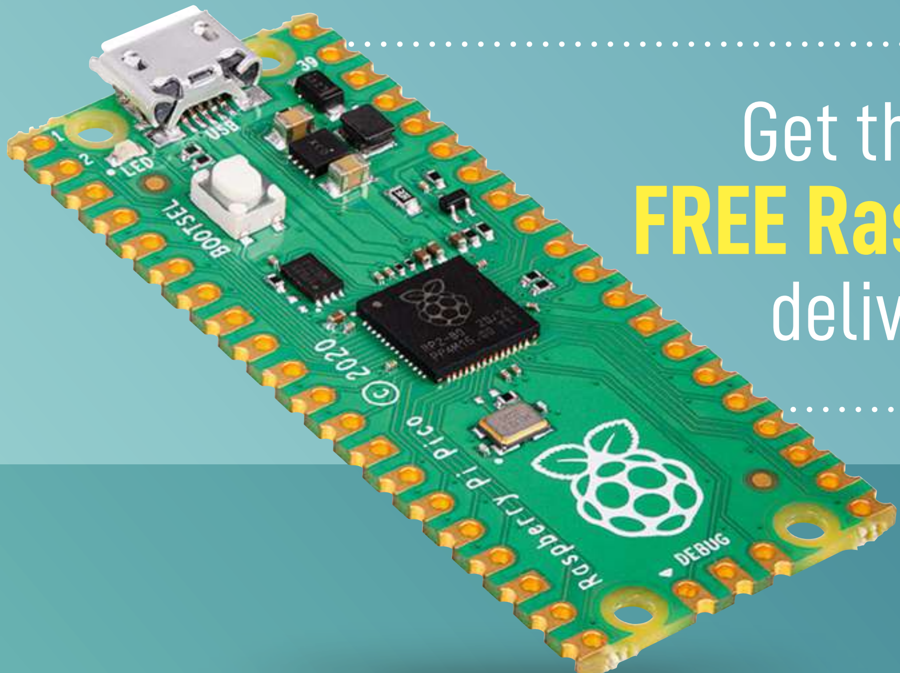
The only thing left to do is close the connection. You have to do this in order to free up memory – you'll quickly run out of memory if you don't.

Once this is loaded on your Pico and running, you can point your web browser to the IP address given in the console when you load the file (typically it'll stay the same between reboots, but it's not guaranteed to). Then you can click on forward, left, and right to move the buggy around.

Our little robot web app is obviously very simple, but you can take this technique and expand it in different ways to let you connect to and control all manner of different Pico W-powered devices. □

SUBSCRIBE TODAY

FOR JUST £10



Get three issues plus a **FREE Raspberry Pi Pico** delivered to your door

hsmag.cc/FreePico

UK offer only. Not in the UK? Save money and get your issue delivered straight to your door at **hsmag.cc/subscribe**. See page 66 for details.

Subscription will continue quarterly unless cancelled

MAKE | BUILD | HACK | CREATE
HackSpace
TECHNOLOGY IN YOUR HANDS
hsmag.cc | August 2022 | Issue #57
RASPBERRY PI
PICO W
PROJECTS
Make internet-connected builds with the NEW $6 microcontroller board
Beer BBQ
Turn a keg into an outdoor grill
Time-lapse Photography
Watch life on fast forward
BEAM MACHINE
ROBOTICS POWERED BY THE SUN'S RAYS
LASER CUTTING KNITTING WELDING DOGS
PLUS
Rocket Glider
CircuitPython PIO
3D PRINTING POLYHEDRA
Jason Coon
The Evil Genius of Fibonacci LEDs
3D PRINTING / ARDUINO

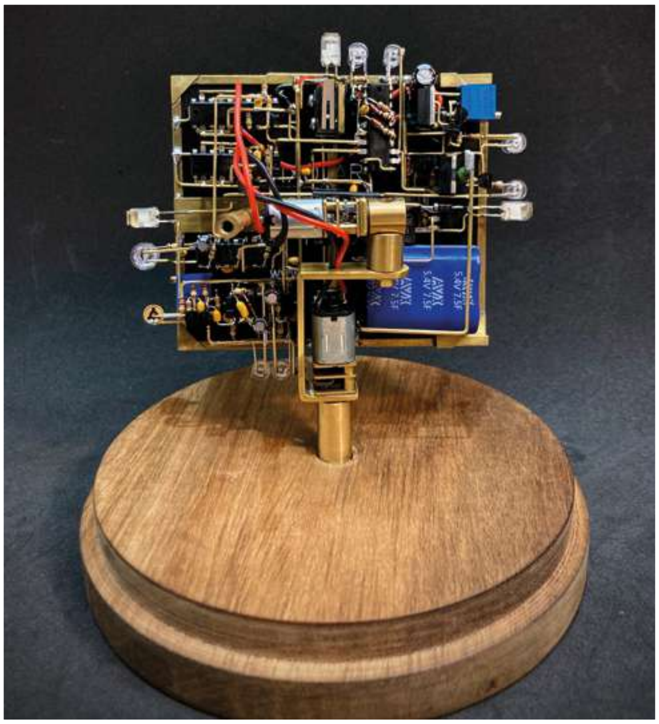
HOW I MADE A SOLAR-POWERED ROBOT FROM OLD FASHIONED PARTS

A design inspired by biology

By Jack Spiggle

The term 'BEAM robotics' is one that would have been familiar to many makers in the early 2000s, but has faded into obscurity in recent years with many of the old BEAM blogs and forums lost to time. 'BEAM' does not necessarily describe a specific form or function of any robot, but rather an ethos which is described in part by the name. The most widely recognised meaning of the acronym BEAM is Biology, Electronics, Aesthetics, and Mechanics. To me, two further elements define BEAM electronic design: the use of recycled components (which I personally no longer do these days, as new components are so cheap and readily available), and the simplicity of the circuits. These simple circuits are the most striking difference between BEAM and modern robots as BEAM robots lack any sort of microcontroller. That's right, we're talking analogue robots!

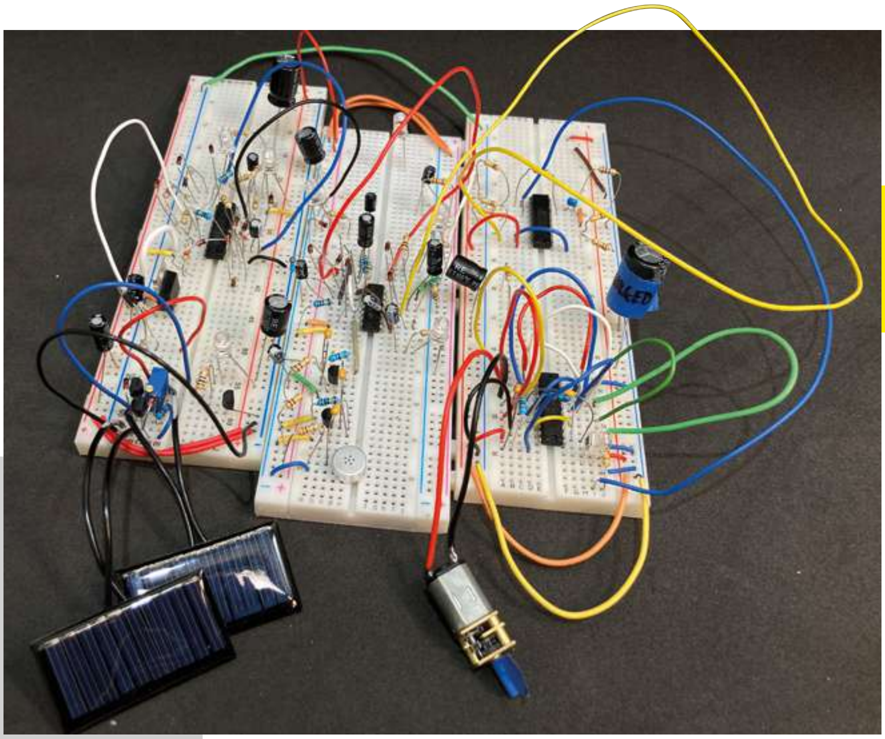
BEAM robots utilise a lot of passive components and other fundamental components like CMOS inverters, but lack any kind of microcontroller. The most complex BEAM robots are made up of blocks that mimic neurons. That is, once a certain stimulus threshold is met on the input of each block, an action occurs.



Valentino Braitenberg's early thought experiments into artificial intelligence (and I mean 1980s early) showed how complex behaviours can arise from a robot created with just a few of these artificial neurons. Such neurons can easily be simulated on a microcontroller, yes, but they can be made even more simply with just a CMOS inverter. This also means that the input and threshold voltages remain entirely analogue. Perhaps this is why I still find BEAM circuits so endearing today: the entire system is built from the ground up, there is no abstraction of signals with 1s and 0s, and no abstracting of logic with high-level code. It really feels like you are manipulating electrons at the lowest level in order to create an intelligent, almost life-like system. I think that all of this is secondary to my robot's main function, which is to look pretty. I definitely consider my BEAM circuits to be more sculpture than robot, and few BEAM roboticists (bar Mark Tilden, the father of BEAM) ever created such complicated robots.

THE ROBOT

Enough with the philosophy, let's talk about the robot. What does it actually do? It sits in a glass bell-jar from IKEA, charging via the large solar panel and turning to face →



“ I ABSOLUTELY ADORE WORKING WITH BRASS ”

the brightest light source. It spends most of the day sitting rather stationary, only to move about quickly, like an excited dog, when in direct sunlight or when you pass your hand over it. Sitting on my desk, it generally wakes up around 8–10:00 in the morning and stops anywhere from 5–10:00 in the afternoon. To make it more interactive, I also added a circuit that makes the robot do a little dance when you give it three whistles.

MECHANICS

The mechanical assembly of this robot is centred around the motors and the solar panel. The motors are common N20 gear motors which were chosen for two reasons: 1) if the robot is still operating decades from now and a motor breaks, there will likely be replacements available and 2) a wide variety of gear ratios were available to experiment with. In hindsight, I wish I had chosen a higher-quality motor as the N20 motors are noisy and have backlash that made the robot very hard to tune. The motor mounts and other brass components were all made by hand using tools like a hacksaw, drill, soldering torch, and my favourite two tools, a jeweller’s saw and needle files. I absolutely adore working with brass: it is easy to cut/bend, readily available in all shapes and sizes, solders well, and looks beautiful!

The solar panel is housed in a frame made from square brass tubing (most of the brass is from K&S Precision Metals). This brass frame was very important as everything builds off it both mechanically and electrically. Because I didn’t want to damage the solar panel while building, I made the frame such that one side can be removed for the panel to slide out. Wherever possible, I tried to make the robot dismantlable in favour of permanent connections, in the hopes that the robot will still be serviceable decades from now.

ELECTRONICS

Onto the electronics, and the circuitry can be broken down into three major sections: the solar charging, the motor control, and the whistle/dance circuit. A lot of the robot was based on BEAM circuits originally designed by Wilf Rigter in the early 2000s with a lot of modifications and adjustments. The full circuit is much too big and nuanced to fully cover here, but there is an Instructable and a half-hour YouTube video overviewing the circuit in more detail.

The circuit is entirely free-formed, meaning that there is no PCB. Connections between components are created in three-dimensional space using bent brass wire, which I think looks stunning and creates a kind of electronic, steampunk vibe (BEAMpunk?). The process of free-forming feels very similar to routing a PCB, and I find it quite therapeutic as long as nothing goes wrong. Indeed, lots of testing was done throughout assembly to ensure everything worked and a little confidence soldering goes a long way.

Interestingly, this robot actually took me longer to prototype and breadboard than

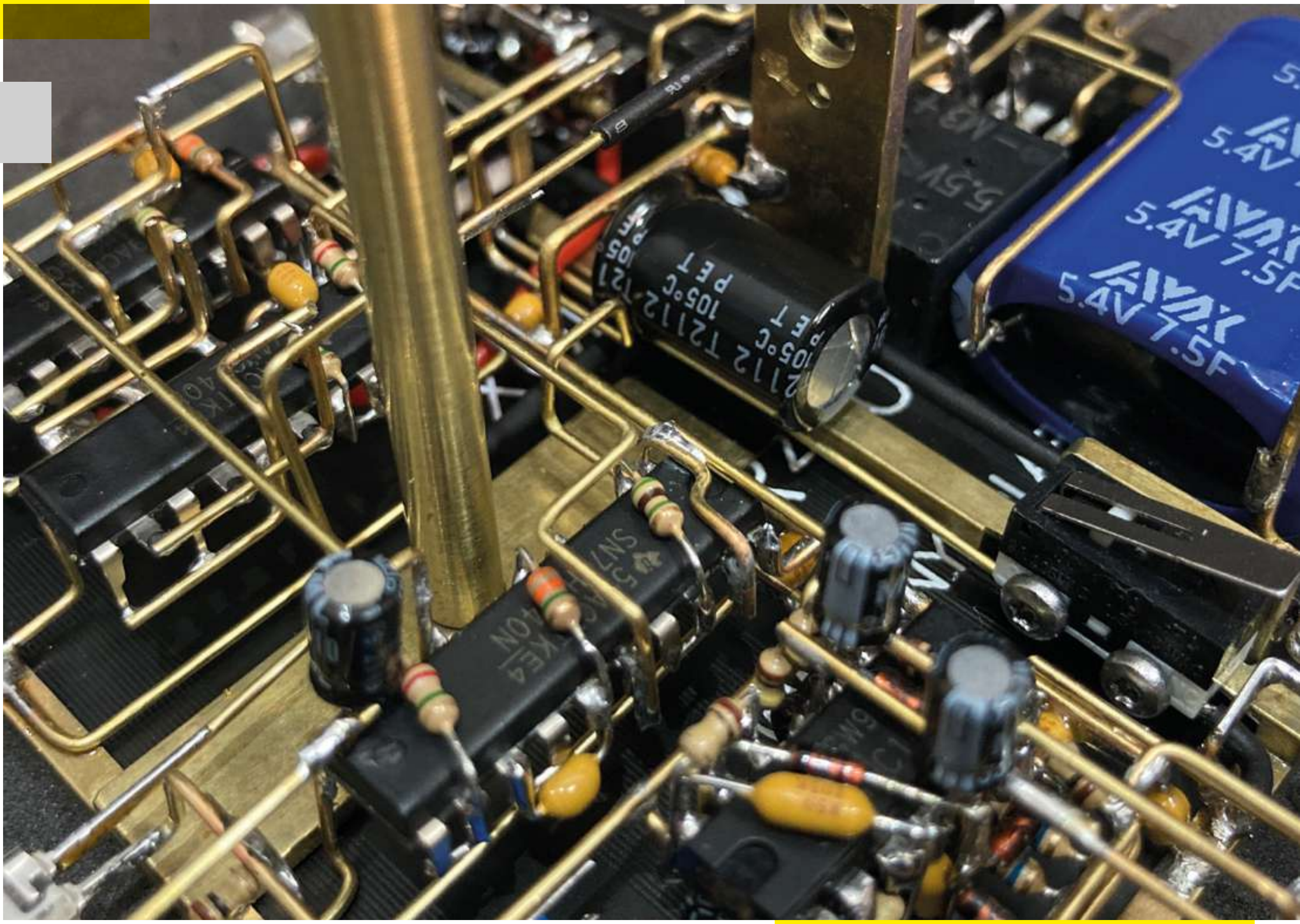
it did for me to build/free-form! Without code, any changes to the robot's behaviour must be made by physically altering the circuit, which is significantly easier to do on a breadboard than on the eventual free-formed robot.

SOLAR-CHARGING CIRCUITRY

Starting the circuit explanation with power, the broad goal is to be able to run the robot solely off the solar panel. However, if you have ever played around with solar panels before, you might have noticed that they are immensely less efficient when out of direct sunlight and I wanted this robot to live on my desk all year round. The solar panels generally found on indoor devices like calculators are amorphous panels. These are different to the polycrystalline and monocrystalline panels used outdoors, and output lower power for their area. The ANYSOLAR panel used in this project is a high-efficiency monocrystalline panel that boasts up to 25% efficiency and, more importantly, responds to a wide enough range of light to be effective indoors. Even still, the datasheet says to expect anywhere from 100–500 times less power when used indoors. Evidently, power is a big hurdle, but luckily we have plenty of BEAM tricks we can use in order to squeeze everything we can from this panel.

Indoors, the panel outputs <100 μA on a cloudy day and 5–10 mA on a sunny day, but the motors draw around 20 mA each! Clearly, the panel alone would never be able to power the robot, which is a problem that many solar-powered BEAM projects face. This problem is solved with a ubiquitous BEAM circuit known as a 'solar engine'. The goal of a solar engine is to allow the panel to charge a storage capacitor until there is enough energy to do useful work, at which point the solar engine turns on the rest of the circuit. Simple solar engines are robust and easy to implement, but are not always the most efficient. Hence, I have used a clever Wilf circuit, dubbed the Dual Slope Sampling Solar Engine (DSSSE). This DSSSE samples the storage capacitor voltage for only a short moment every five seconds or so and only draws significant power during that →

Above ↑
The frame before free-forming



sample time. The solar engine on this robot turns on at around 3.5 V and turns off again at around 2.5 V. When off, only the charging is connected, and when on, the motor controller and microphone circuit turn on alongside a blue standby LED.

The DSSSE actually charges two capacitors: one is a smaller 330 mF supercapacitor and the other is a larger 7.5 F supercapacitor. The circuit starts by charging the 330 mF capacitor so that, in the morning, it takes much less time for the robot to initially turn on and face the sun. Once locked onto the sun, the panel begins charging the 7.5 F capacitor until it matches the 330 mF capacitor voltage, and the two capacitors act as one.

MOTOR CONTROL CIRCUITRY

The motor control circuitry is really elegant and was also (surprise, surprise) originally designed by Wilf. He called this circuit the Power Smart Head (PSH), and one PSH is used for each of the robot's axes. The PSH uses a modified CMOS oscillator to create what I would describe as a PWM-based, proportional controller, with a large dead-band. The oscillator has two resistor/capacitor (RC) pairs. The first RC pair dictates the oscillator frequency, and the duty cycle of this oscillator is altered by a pair of photodiodes that act as the robot's 'eyes'. This creates the PWM. The second RC pair creates the dead-band so that the robot does not just constantly move and

waste power. The interesting thing here is that from controller input (the voltage between the eyes) to the motor output (the PWM duty cycle), the signals remain entirely analogue!

WHISTLE/DANCE CIRCUITRY

The microphone circuit is based on yet another Wilf design that he called 'BEAM Sonics'. It consists of preamplification, followed by some BEAM-style logic, again mostly made of inverters and RC pairs. This logic ensures that the whistles are of the correct length, and that there is a distinct pause between each whistle. Once it counts three suitable whistles, it triggers a circuit to make the robot dance and another to lock the whistle circuit for 20 minutes so that the robot is not dancing too often and wasting power. The microphone circuit has three LEDs: one green that flashes for every whistle, one amber that flashes when the circuit activates, and one red that remains on while locked.

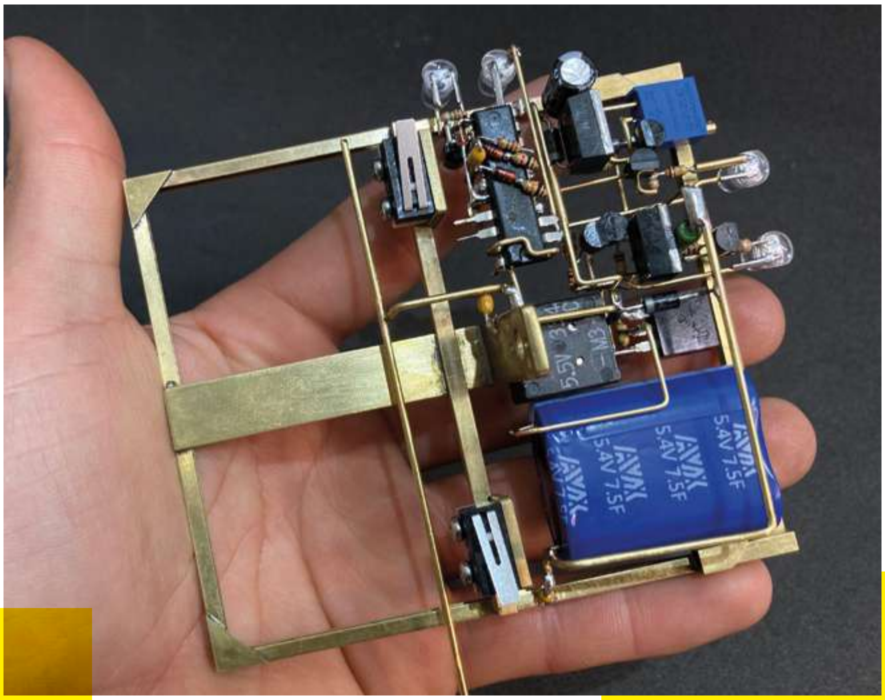
The dance circuit is simply three oscillators, two of which force the motors to oscillate and the other to make a pink, hexagonal LED flash.

" THE SECOND RC PAIR CREATES THE DEAD-BAND"

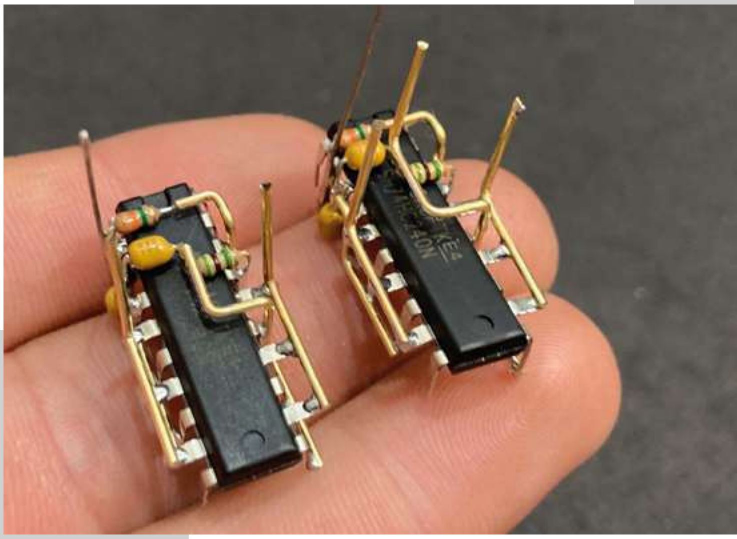
CLOSING THOUGHTS

While BEAM circuits may take a little more effort to build and are nowhere near as powerful as a microcontroller, I would encourage anyone interested to do some research, and give these old BEAM circuits a look (the Wayback Machine might be your friend here). If you are keen to build any, I think that the best circuits to start with are 'pummers' that charge during the day and slowly flash at night, and 'photovores/photopoppers' that hop about using two motors to locomote toward light. Although the obvious solution is often to slap a microcontroller onto projects, there may be times where simpler analogue circuits make more sense. In fact, in recent years, analogue computers are once again being considered for use in artificial intelligence!

Even without building a BEAM solar engine, the principle of storing charge from a solar panel in a capacitor until there is enough energy to do something useful can be applied to many solar-powered projects. I hope this article might have inspired you to consider solar power for future projects, or perhaps to try free-forming, which can be a fun way to bring a little more artistry to any project. □



Above ↑ The completed solar logic



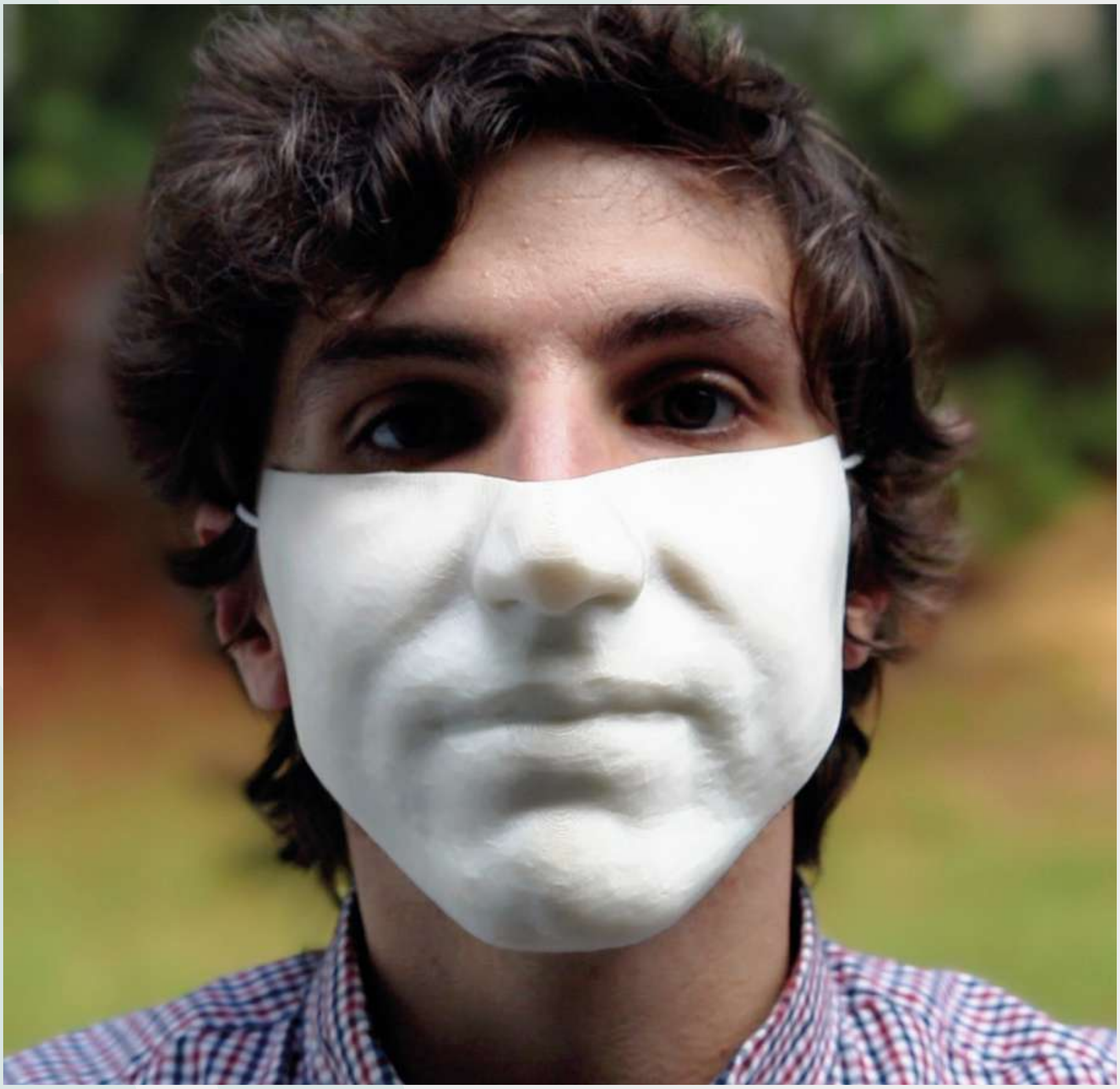
Right → The two Power Smart Heads

HackSpace magazine meets...

Andrew Sink

3D printing professional, evangelist, and GoldenEye fan

Andrew Sink is a shining evangelist for 3D printing. He was there during the 3D printing boom of 2014, and kept the faith while the buzzwords died down and people refocused on what the potential of the technology was. He's written software to make design for 3D printing more accessible to new users, and his work is on the shelves of big retailers, without you even realising it's 3D-printed. He's on a mission: to bring 3D technology to the world in a way that people can understand intuitively. We're very lucky to have got a few minutes' worth of his insights to share with you here. →



Above
Andrew SInk, wearing a mask of his own face. Thankfully this is the version without makeup

HackSpace: You're obviously not just a guy who uses 3D printing from time to time: you're an evangelist. What excites you about 3D printing?

Andrew Sink: So, first of all, in my day job, I am a senior applications engineer for Carbon (**carbon3d.com**). Carbon is a 3D printing technology company based outside San Francisco. So my job is primarily interfacing with customers and designing parts for additive manufacturing specific to the Carbon process. I work on a team that's primarily involved with lattice structures, so I spend a lot of time thinking how to design something that used to be a uniform block of material, as a conformal lattice, rather than a solid shape.

I was just at an expo in Detroit, Michigan, and I actually brought a bicycle with us to our booth. And it's got a 3D-printed seat, which was designed by people on the team that I work on. And it's designed for mechanical response; it's designed for comfort, it's designed for printability. Thousands of pages of thought have been dialled into this lightweight bike seat, and then you look at it and go 'Oh, that's pretty neat'.

But what's really exciting about working at Carbon for me is I went into a bike shop to pick up that seat fairly recently. And it occurred to me that is the first 3D-printed part I've seen on a store shelf, that was a retail product. And when I was in Detroit with my wife, for the expo, we stopped in at a Dick's Sporting Goods [a big sports shop in the USA]. And they had the Adidas 3D-printed midsoles, so they had shoes that have a 3D-printed sole. And so I bought my wife a pair. It's cool to be able to walk into a store and buy something that was 3D-printed.

HS: What got you into 3D printing in the first place?

AS: I've been involved in additive manufacturing for about a decade now. I first used a 3D printer when I was in college; we had one in the engineering lab, and I saw it and instantly felt that this is the thing that I'm going to do for the rest of my life – it was such a very clear and amazing technology.

The applications were just immediately apparent. I was taking a class on SOLIDWORKS as a 3D CAD program at the same time; taking a model out of SOLIDWORKS, and then sending it to the printer and then holding it in my hands that same day was an absolute revelation. It absolutely just tied together all the work that I was doing in the digital world to the physical world. And it was immediately apparent to me that it was just such an untapped technology.

"I saw it and instantly felt that this is the thing that I'm going to do for the rest of my life"

HS: And do you still have that same sense of wonder now that you use it day in, day out?

AS: I can't walk by a 3D printer. I'll stop and watch them for a couple of minutes. My house is filled with printers. It's just fascinating to see the industry evolve so rapidly in ten years – materials, software, hardware. And it's just, it's a really cool experience.

HS: You've also been experimenting with photogrammetry, which goes one step further – taking an object from the real world, putting it into the computer, then taking it back out again. What are the applications for that?

AS: The application of photogrammetry is hard. You could be sitting in a room looking at the first camera and saying, 'Well, what are we gonna do with it?' It's such a limitless technology. And it's also very much in its infancy, and we're still figuring out what it's good for.

One of my favourite books is *Timeline* by Michael Crichton. It doesn't discuss 3D printing directly, but it talks about people who accumulate these things called transcription errors, where they're essentially turned into digital data streamed into a different area, and then re-materialised as physical objects. And over time, veins don't line up all the way, bones are slightly misaligned... you accumulate these errors over time. When I started getting involved in 3D scanning, I just immediately thought of that book, because it really breaks down a lot of these core concepts really well. With photogrammetry, the thing you're scanning is going to be the thing you've scanned, but it's going to be different. It can be its own, you know, tangible item. The questions you get asked all the time with 3D scanning are, 'how do I scan a boat?' And that's a very different workflow from 'how do I scan a screw', where you're talking about precision on the magnitude of microns versus metres. 3D scanning, just like 3D printing, means a lot of different things to a lot of different people. So, where I spend a lot of my time is on the accessibility and early entrance, so I think a lot about somebody who's never heard of 3D scanning before. They're going to go to Google and type in '3D scanning'; what's the first thing that they're going to see? And how is that going to be applicable to them? So I've spent a lot of time making videos showing the process of photogrammetry. How do you stitch photos into a model? How do you use a LiDAR scanner to use sensors to detect how a model exists in space, and really try and break it down in a way where somebody with no practical experience can solve their particular problem?

HS: Photogrammetry sounds like something that's really expensive to get into. Do you need to have access to a lot of specialist gear? →

Above Incredibly, this isn't a drawn model, it's a scan of the dragon shown on page 58

AS: Historically, it's been very expensive. But that's because it's been computationally expensive. The hard part is the software. That's where the industry has really lagged behind. A big part of that is conflicting messages from users, you know: I want to scan really small parts, very detailed, and I also want to fly a drone around my house and make a 3D model. Those two use cases are going to accumulate different amounts of data, and those are going to be processed differently.

Right now, there are apps on the App Store that are under $10, that you can use on pretty much any iPhone, a lot of Android phones, and create a 3D model within minutes. There's this really great picture – my wife and I were in Chinatown in Boston. And it was a perfect day for 3D scanning – it was overcast, so the light was very diffused – typically, you get really harsh directional light from the sun, which makes it hard to scan stuff without adjusting the aperture settings on your camera. And it's just, it's kind of a mess.

We got this picture taken with us. I saw the statue right next to us, and I was like, 'Oh, that would be a great 3D scan'. And it is, without a doubt, one of the best scans I've ever made; you can make out individual teeth on the sculpture. This technology is so accessible now that it took me maybe ten minutes to walk around and get the photo set, and then maybe another five minutes after it finished rendering to clean up the base. So, photogrammetry has absolutely gotten to a point where it's affordable, it's usable, and you can get up and running very quickly.

For photogrammetry, it's really hard to get pictures of shiny stuff: bronze, for example, is terrible. Because it goes between dark and very bright, there are almost no gradients in bronze. You're either looking at, like, gold, or you're looking at black. And so a marble statue on an overcast day is just perfect. It's worked out great. 3D scanning shiny objects is sort of like taking a picture of a mirror: you don't really get a picture of the mirror, you just get a picture of everything in front of it.

HS: Did you use an iPhone when you made the mask of your own face?

AS: That was actually done using a fairly high-end industrial 3D scanner. A friend of mine did it while we were at a trade show and we had a bit of extra time. I sat on it for a while – what are you gonna do with a scan of your face? And early in the pandemic, a company launched a bounty programme on a facial recognition spoof. It was something like $10,000 if you could beat their biometric verification, and I thought, well, I've got a really nice-looking scan of myself, it probably wouldn't be a stretch to try and beat this thing. That became my Covid quarantine project. I was at home. I had some time on my hands. And so I thought, what would a normal person do? I will make a hyper-realistic mask of my face.

"This technology is so accessible now that it took me maybe ten minutes to walk around and get the photo set"

I was going back and forth to make-up stores to get all kinds of different products to try and bring out highlights in the model. I was putting paint on the lips and stuff. It was creepy. It was absolutely awful. It worked in the sense that Google Photos will recognise the mask as a picture of me. So it will automatically tag me, which is cool. I'm able to set up an iPhone with the fake mask, but I'm not able to unlock it, so, whatever Apple uses for face ID does recognise it as a face, but it won't recognise it as the same face twice. Google recognising it as a face though, that was a huge win.

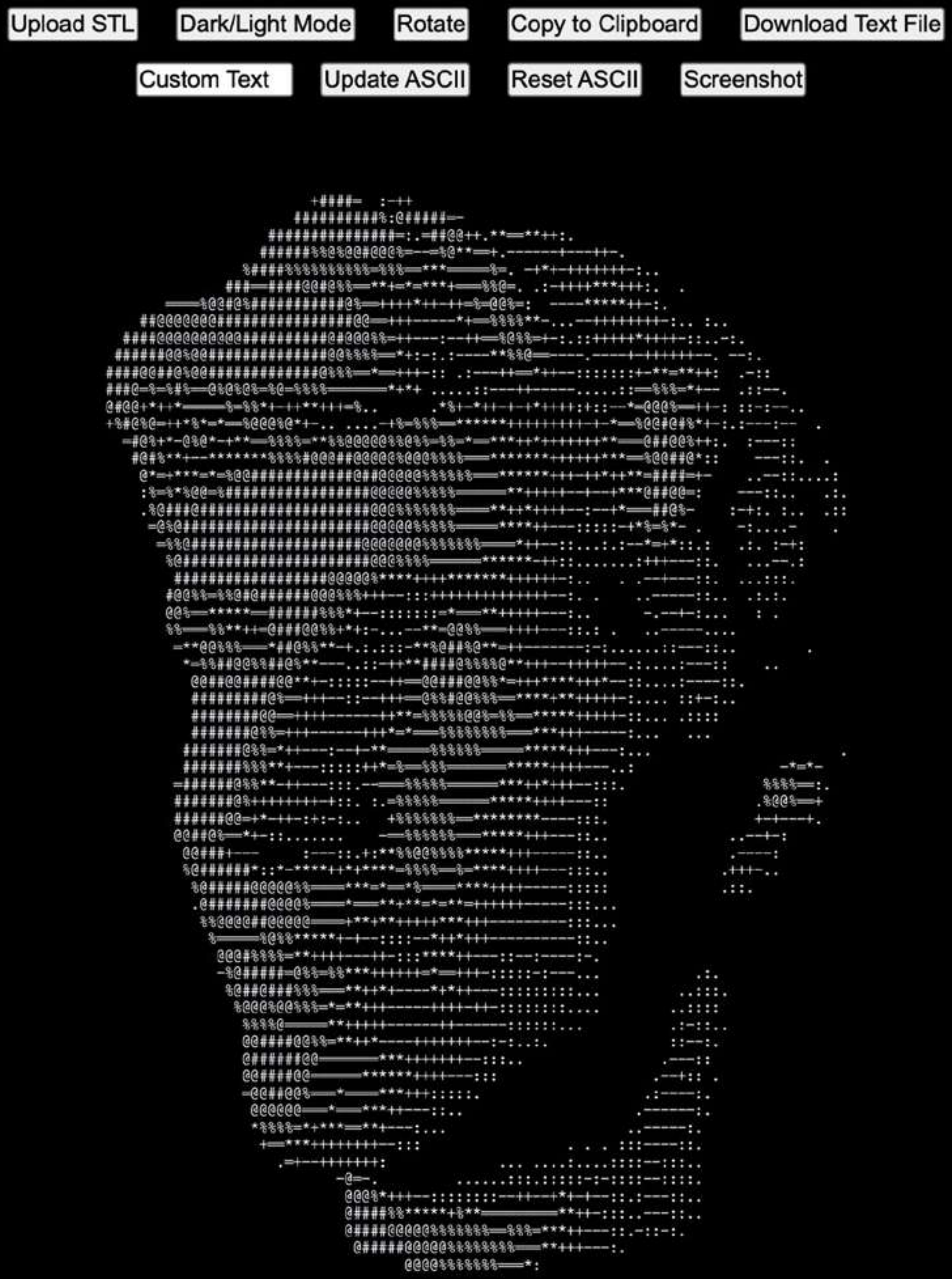
HS: You have a YouTube channel where you talk about 3D technology; you've also written some software to make it easier to create 3D models, haven't you?

AS: Yeah, so I'm a big fan of low poly art. I grew up with GoldenEye, on the Nintendo 64. So, I have a very deep appreciation for doing more with less. You look at the textures of these models, and it's like a guy, and that guy has a, you know, a texture on him. So if you see a face, it's the eyes and the nose. And then he moves his head, you're like, oh, it's kind of a rectangle shape, you know?

One of the things I found was that there weren't a lot of really intuitive workflows for creating low poly art without using programs like Blender, which is very powerful, but not very intuitive, so it's very difficult for beginners. I set out with a goal to design a web-based app to make low poly models. So the URL is **lowpoly3d.xyz** – the whole thing is written in JavaScript. And you can basically upload a 3D model and then you can select the destination amount and remove a certain number of edges, and it collapses the model into a low poly model, which you can then download.

One of the reasons that I wanted to work on the low poly site was just because at this point, hardware has pretty well outstripped software in 3D printing. We have very, very well-made machines that are made very inexpensively across a wide range of technologies. But, on the design side, you kind of have two options: professional parametric CAD, and then more sculpting programmes, like ZBrush, or Maya. Those both have pretty steep learning curves. So I wanted to create a tool that was easy to use, where somebody could say, 'Hey, I downloaded a model, and I'm going to reduce the poly counts. And now I'm going to print it out, and then it's done.' That was the goal. And I'm always excited to see what people are making with it.

And then there's also the STL to ASCII generator. Again, if you grew up →



Above Upload any 3D image to Andrew's andrewsink.github.io/STL-to-ASCII-Generator to generate ASCII art around the time that GoldenEye was out, you've probably seen ASCII art as well. You can upload a 3D model, and it'll basically apply a filter on top of that model that shows up as ASCII art. It's pretty cool. They're both designed to be as approachable as possible: you just drag a model and start hitting buttons, you really can't mess up. And I think that's very encouraging for beginners; maybe they've got a 3D printer for Christmas and they want to learn more about the modelling side, but they don't have access to expensive or difficult-to-learn tools.

HS: What do you see as the future, or possible futures, in the 3D printing multiverse?

AS: Let's split this into two separate parts. On the hobbyist side, I think hardware is commoditised at this point. You can buy a printer that's mechanically sound for about $200. That part's really easy, I think. The improvements will come in the form of ease of use and quality of life upgrades. So things like: can the printer tell you when it's out of filament? Can the printer detect when it's not actually printing? Those features are available right now on more expensive machines, and I think, as those become more prevalent, printers will become easier to use, and consumers will be more incentivised to try them. There was this big boom in like 2014, where every newspaper had a think-piece like 'is this the dawn of 3D printing'? And it turned out that nobody knew how to design for 3D printing, so nobody was actually making their own stuff. For the last eight years or so, we've seen that hardware just continuously gets improved. A lot of the slicing software has seen incremental improvement, but we're still not really at a place where the design software is matching that speed of improvement. So I think on the consumer side, as those quality of life features start to make their way down, the less expensive printers and the software becomes more intuitive. I think that's going to really help drive adoption.

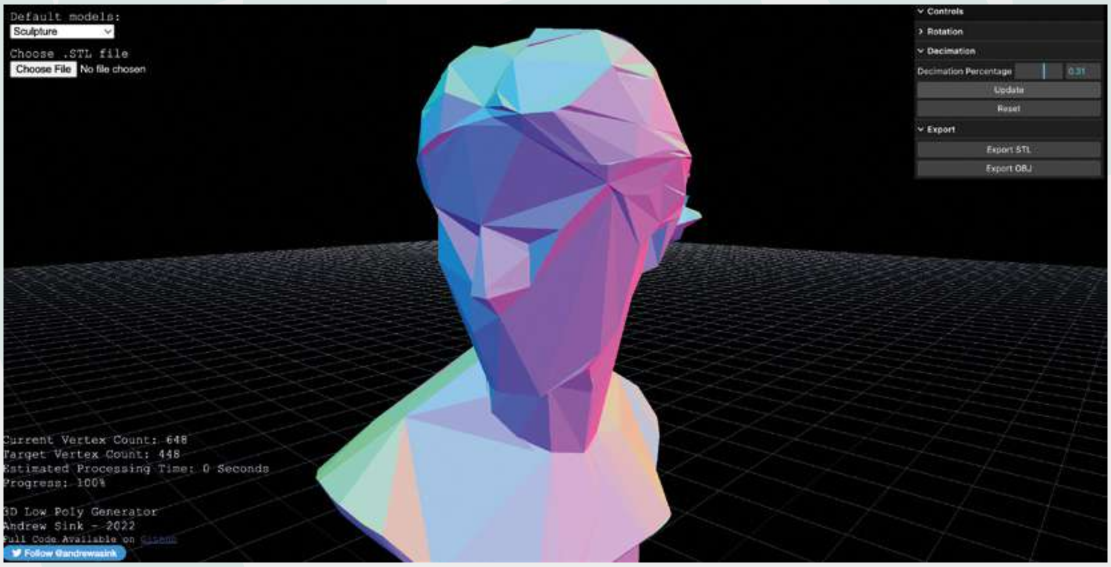
"We're still not really at a place where the design software is matching that speed of improvement"

HS: And how about the industrial side?

AS: As the technology gets faster, the materials become more durable, and have better mechanical properties, you'll get to a point where you're going to buy a car, and your shifter knob is going to be 3D-printed, and you're not going to know it. Let's say there's a factory that makes cars, and they find out that one of their tail-lights – the bracket that they ordered 5000 of – doesn't fit onto the frame, and they need to print a shim. That is where 3D printing will come in and save the day, because you can solve this problem quickly, cheaply, reliably, without having to make tooling to mould this thing. I see a real future for 3D printing in solving problems that are below the surface, not necessarily in people buying things for the novelty of owning things that are 3D-printed.

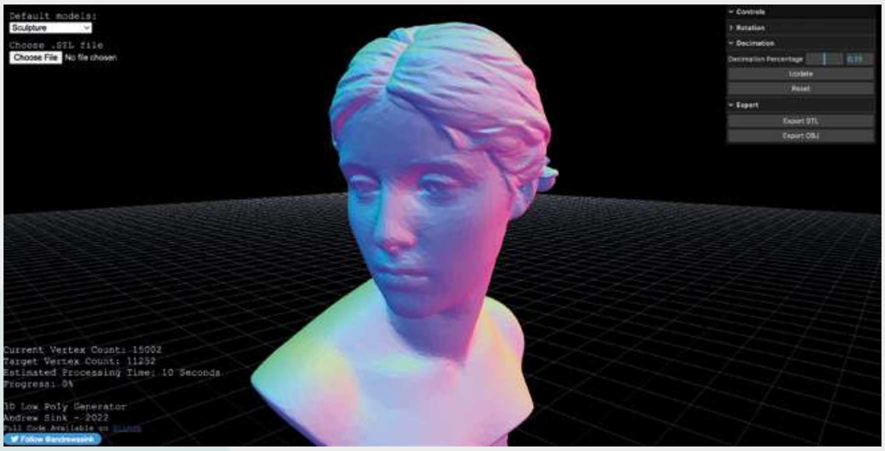
HS: A lot of the time I see 3D printing projects online, someone's made the 1,000,000th Dungeons and Dragons or Warhammer miniature. I look at it and think I've seen this so many times before, and it's just the same as an injection-moulded thing, but less efficient. Do you think 3D printing is still exciting?

Right Compare the dragon shown here with the scanned image on p54; it's unbelievably accurate



```
Current Vertex Count: 648
Target Vertex Count: 448
Estimated Processing Time: 0 Seconds
Progress: 100%

3D Low Poly Generator
Andrew Sink - 2022
Full Code Available on GitHub
```

Left The same model in lowpoly3d.xyz, in high- and low-poly forms



```
Current Vertex Count: 15002
Target Vertex Count: 11252
Estimated Processing Time: 10 Seconds
Progress: 0%

3D Low Poly Generator
Andrew Sink - 2022
Full Code Available on GitHub
```

AS: My Nana, my Italian grandmother, when she first saw a 3D printer, it was an old printer made of wood, and it was printing a fork. And she took one look at it and just immediately she was like, 'Oh, so you just draw something and it pops out'. Zero explanation required. It just immediately clicked in her head. And I thought that was just such a powerful thing. And so I try to make sure that, when I'm working on things like photogrammetry, or 3D scanning or the low poly generator, I'm thinking about what people are going to think when they see it.

Let me take you back to that first camera analogy I used earlier. Just like there are millions of little Baby Yodas and Dungeons and Dragons figures floating around, there's also millions of photos online of flowers, right? But people aren't going to stop taking pictures of flowers because it's already been done. There's something really uniquely personal about making something yourself. When you download something and print it out – it's an accomplishment – you made this. This is something you've brought into the world. I think that's a really special thing. It also leads to all sorts of things – maybe they get curious; maybe they learn CAD and go and create more things. Or, maybe they decided to become a photographer, and take pictures of flowers.

Right This disguise fooled Google, but not Andrew's iPhone