NEW! YOUR OFFICIAL RASPBERRY PI MAGAZINE

The MagPi

Win! 5 ROBOT KITS WORTH £250

100 pages of hacking & making

Issue 38 October 2015

raspberrypi.org/magpi

BUILD A RASPBERRY PI ROBOT FOR £50*

The ultimate in affordable weekend projects

*Including the Pi

BUILD A WEB-POWERED PLANT WATERER
Give your Pi green fingers!

MAKE MUSIC WITH THE PIANO HAT
Step-by-step tutorial with code samples

THE SENSE HAT
Master the Pi's amazing space-age add-on

THE OFFICIAL PI TOUCH SCREEN
Read the first review inside

Also inside:

- LEARN CLASSES IN PYTHON
- MANCHESTER MAKEFEST REPORT
- PI WINS BIG AT FESTIVAL OF CODE
- MORE OF YOUR AMAZING PROJECTS

SPOOKY HALLOWEEN PROJECTS
Scare your friends with these Pi-powered ideas

Issue 38 • Oct 2015 • £5.99

10

9 772051 998001

THE ONLY MAGAZINE WRITTEN BY THE COMMUNITY, FOR THE COMMUNITY

Raspberry Pi SWAG STORE

swag.raspberrypi.org

WELCOME TO THE OFFICIAL PI MAGAZINE!

There's something quite magical about completing your first Raspberry Pi project. Harnessing the power of code to create something that seems greater than the sum of its parts is a genuine rush. Nowhere is this more true than when you're making something like a Raspberry Pi-powered robot. Seeing your machine move, react to its environment, or follow your instructions is a simply staggering achievement. Once you've got the basics of obstacle avoidance down, it's almost trivial to start layering additional behaviours until you have what appears to be your own thinking, intelligent machine. World domination has never seemed so tangible! That is, of course, if you have the cash to do it.

Unfortunately, it's not the cheapest project, often costing north of £100 / $150 to get started. That's not the case any more, though. We've managed to magic up a Model A+, chassis-free build that costs just £50 / $75 including the Raspberry Pi. Want to build it? Join our new Features Editor, Rob Zwetsloot, on page 18 to get started today.

There are lots of other articles you shouldn't miss this month, too, some of which I've highlighted to the right. Mike Cook's amazing Mulder skull (starting on page 58) deserves a special mention, though. It's easily our favourite Halloween project this month!

Russell Barnes

PROUD SUPPORTERS OF: NATIONAL STEM CENTRE

SUBSCRIBE & SAVE 25%
SEE PAGE 26 FOR DETAILS

THIS MONTH:

FIND US ONLINE raspberrypi.org/magpi **GET IN TOUCH** magpi@raspberrypi.org

The MagPi

Available on the App Store
Get it on Google play

EDITORIAL
Managing Editor: **Russell Barnes**
russell@raspberrypi.org +44 (0)7904 766523
Features Editor: **Rob Zwetsloot**
Technical Editor: **David Whale**
Sub Editors: **Laura Clay, Phil King, Lorna Lynch**

DESIGN
Critical Media: **criticalmedia.co.uk**
Head of Design: **Dougal Matthews**
Designers: **Lee Allen, Mike Kay**
Illustrator: **Sam Alder**

PUBLISHING
For advertising & licensing:
russell@raspberrypi.org +44 (0)7904 766523
Publisher: **Liz Upton**
CEO: **Eben Upton**

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

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

CONTRIBUTORS
Sam Aaron, Tim Anderson, Mike Cook, Gareth Halfacree, Lucy Hattersley, Richard Hayler, Jasper Hayler-Goodall, Phil King, Simon Monk, Matt Richardson, Richard Smedley & Sean Tracey.

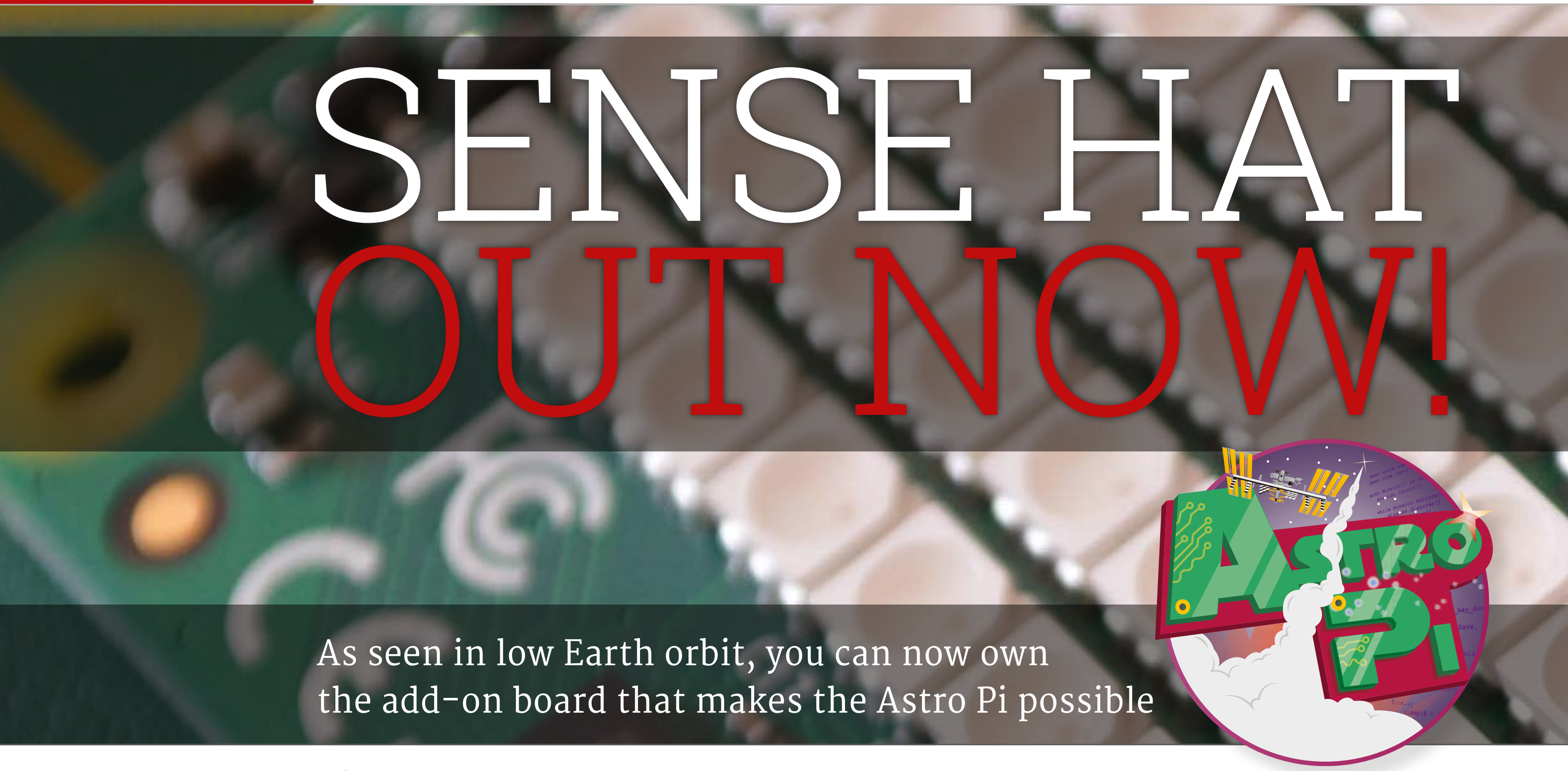
recycle When you have finished with this magazine, please recycle it.

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

The MagPi magazine is published by Raspberry Pi (Trading) Ltd., Mount Pleasant House, Cambridge, CB3 0RN. 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 in the magazine. 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: 2051-9982.

Contents

Issue 38 October 2015 raspberrypi.org/magpi



SENSE HAT OUT NOW!

As seen in low Earth orbit, you can now own the add-on board that makes the Astro Pi possible

By the time you read this, the Sense HAT will have been out for a little while. Hopefully you've managed to get your hands on one, but if you're still unsure as to what it even is, you may at the very least be familiar with the project it's attached to: Astro Pi.

The Sense HAT is the add-on board holding the key sensors that enables Astro Pi to exist in the first place, and will be attached to a selection of Raspberry Pis as they head up to the International Space Station at the end of 2015. The Sense HAT had slightly more humble beginnings, though, at least according to project lead Jonathan Bell:

"I kinda hijacked a pet project of James [Adams's] and turned it into a space-faring board," he reveals. James Adams is the director of hardware at Raspberry Pi, and he and Jonathan were the main forces behind the Sense HAT.

"Effectively we wanted to produce a HAT that would be a neat, fun example of how to design a HAT," Jonathan continues, telling us the simple origins of the Sense HAT. "It was an exercise in how to design a HAT that can be put into mass-production: how would somebody go about doing that so hundreds of thousands of HATS could be produced, and how would we design the board to deal with that."

Halfway through development, what was once a relatively basic HAT received a whole host of ST microsensors, the same kind used on mobile phones. "Eventually we said,

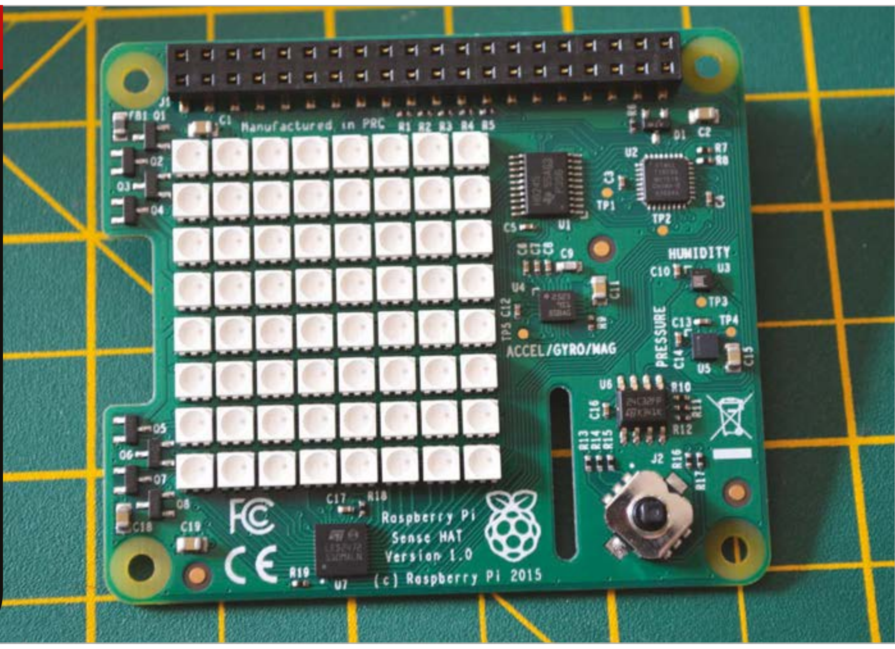
Below The Sense HAT is quite small, but packs a large number of sensors and features

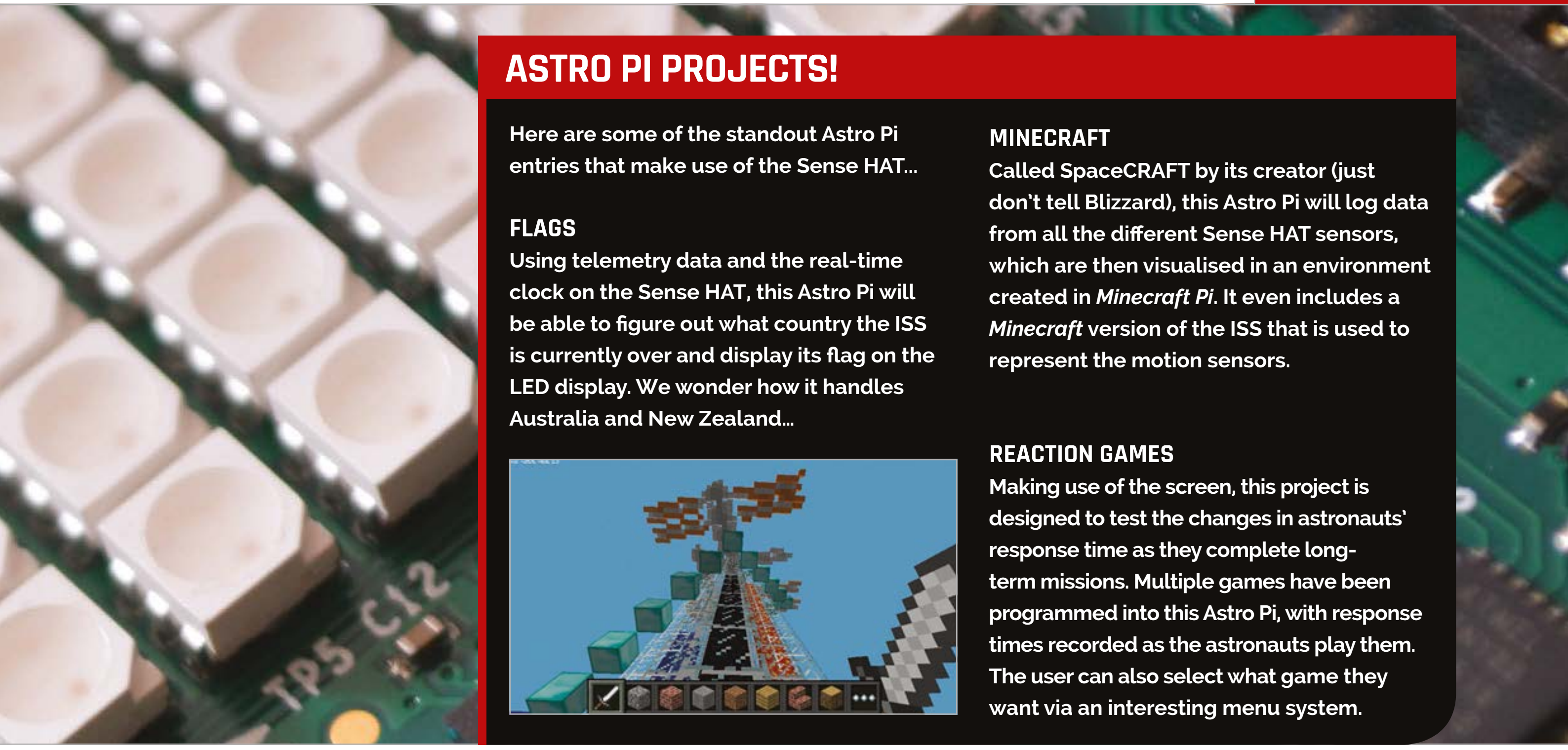
HACKING THE SENSE HAT

While the Sense HAT is perfectly usable with a Raspberry Pi as it is, with a bit of tinkering you can make it stand alone using the Atmel microcontroller on the board. Jonathan reveals how:

"The Atmel has two methods of communication with the Raspberry Pi: we have the standard method which, with the stock firmware, is using I²C and appears as a straightforward framebuffer device to Linux. So if you were to rewrite that, you would reprogram the unit, the Atmel, through the SPI interface. But then, once you've done the reprogramming, there's nothing to stop you making the Atmel communicate via SPI and communicating with the sensors over I²C."

With enough reprogramming, you can have the tiny chip connect to the sensors and the joystick, while also driving the LED matrix. Get to it!



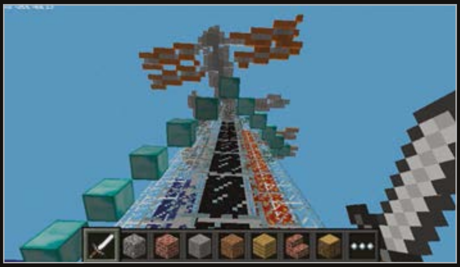


ASTRO PI PROJECTS!

Here are some of the standout Astro Pi entries that make use of the Sense HAT...

FLAGS

Using telemetry data and the real-time clock on the Sense HAT, this Astro Pi will be able to figure out what country the ISS is currently over and display its flag on the LED display. We wonder how it handles Australia and New Zealand...



MINECRAFT

Called SpaceCRAFT by its creator (just don't tell Blizzard), this Astro Pi will log data from all the different Sense HAT sensors, which are then visualised in an environment created in *Minecraft Pi*. It even includes a *Minecraft* version of the ISS that is used to represent the motion sensors.

REACTION GAMES

Making use of the screen, this project is designed to test the changes in astronauts' response time as they complete long-term missions. Multiple games have been programmed into this Astro Pi, with response times recorded as the astronauts play them. The user can also select what game they want via an interesting menu system.

hang on a minute: what happens if we put loads of sensors on this thing and turn it into a kind of a cool toy!"

The Sense HAT was eventually completed with three key sensor chips: separate pressure and humidity chips that can also both measure temperature at different levels of sensitivity, and a positional chip that contains an accelerometer, a gyroscope, and a magnetometer. There's also an 8×8 LED display and a tiny little joystick that you can use in conjunction with the sensors or completely on their own.

All of this is accessible on the Raspberry Pi by just popping the HAT on the GPIO ports and getting the special Python library for it, which is what the space-bound Pis will be using when they get sent off-world.

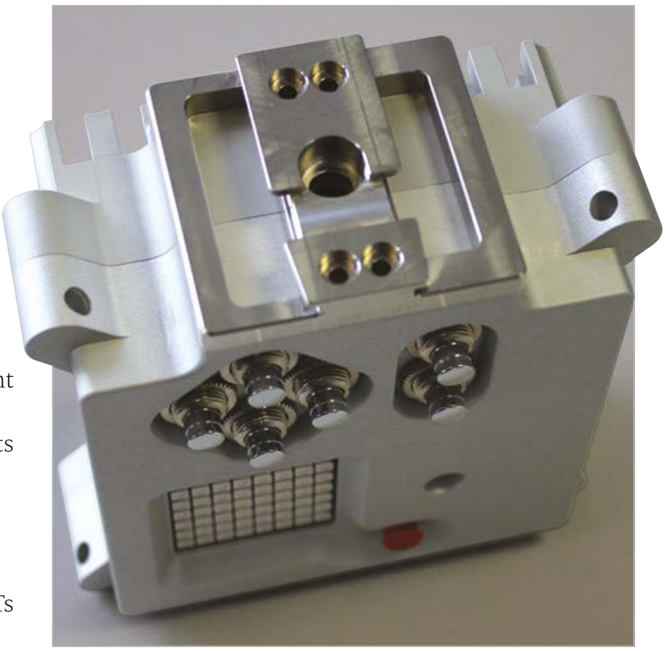
"The Astro Pi experiments make good use of the HAT itself," Jonathan tells us, "some of them in quite unusual ways. We have a few Easter eggs up there, which you'll have to find out about, but there have been some ingenious uses of the sensors. One of the experiments that caught our eye in terms of sensing was one that attempted to detect an astronaut. The astronaut detector sits there, monitoring the humidity, and if there is a certain percentage change in humidity in the module, it thinks there's an astronaut present. It flashes a message on the LED matrix saying 'are you there?'.

"Can we detect a human presence in a ventilated module? And the answer is, yes you can; humidity is monitored on the ISS already and hopefully we'll be surprised that the experiment has a good result. Anyway, once it says 'are you there?', it will then ask you to take a picture. Press a button, go around the other side of the Astro Pi, and smile for the camera. So, through that experiment, they're trying to track if they can get a variation of humidity, and if that variation is also correlated with the presence of an astronaut in the module. That's two good experiment objectives right there."

There are many more experiments where that came from, all of which we'll start seeing reports from in early 2016. As for now, though, if you want to do your own experiments Earth-side, Sense HATs are available from the Swag Store (**bit.ly/1PWWZsU**). You can read our review of the Sense HAT starting on page 78 of the magazine, and we also have a quick Getting Started guide beginning on page 48. Send us your best projects and we'll feature them in *The MagPi* #39!

> "It was an exercise in how to design a HAT that can be put into mass-production"

Below The case from the Astro Pi project isn't necessary to use a Sense HAT, nor is it on sale yet, but it looks really cool, so here's a picture of it



BUILDING IQAUDIO

It started at a hacking event in Edinburgh, now IQaudIO is bringing high-end audio within the reach of any Pi owner. Founder **Gordon Garrity** explains how it all happened

IQaudIO (**iqaudio.co.uk**) is a small company which makes Raspberry Pi audio add-ons, including the Pi-DAC+ and Pi-DigiAMP+ cards which convert a Pi into a high-quality audio source. But how did it all start?

"My background is software development and I did computer science at university," says founder Gordon Garrity. "I came across a job posting for software development director at Simple Audio, a hi-fi company in Glasgow. It put two of my passions together, software development and hi-fi.

"Eben Upton [Pi founder] was at the Edinburgh Music Hack as a sponsor. The Pi had just been launched and Eben was providing five or six Raspberry Pis to the hacking community. I was sitting next to him in one of the lectures, said that I worked for Simple Audio and that it would be great to get some music onto Raspberry Pi. He said yes, cool, carry on and let us know what happens. He has been very supportive because we are bringing a real use to the Pi.

"Simple Audio was sold and I decided that I wanted to work on the Pi. My daughter was going to university and she wanted to have some high-quality music playback. I didn't want to spend £700 on a hi-fi system for her.

"There were two or three of us sitting around a dining room table having curry and chilli for a few nights, coming up with ideas, and we put some prototypes together.

"From initial concept to having a prototype was relatively quick; the problem was the software. At the time, there was no solid and reliable digital audio driver for the Raspberry Pi. There was a student in Germany, Florian Meier, who was doing his thesis on digital audio playback. He wrote the I^{2}s drivers for the Raspberry Pi, and that was the real enabler."

The I^{2}s interface enables audio to be passed between devices without using USB. "The disadvantage of using USB on the Pi is that the USB and the Ethernet share the same bus," says Gordon. "Transferring files to the Pi, such as connecting to a NAS, takes bandwidth away from the processor. This was especially important when the Pi was a single-core processor. Now with a quad-core processor, it is a bit less of a problem, but still important.

"We worked with Florian, we sent him some prototypes; we then worked with some of the other community members. It was all done with a community spirit. We shared our hardware, and that was how we got device drivers and support into software packages such as RuneAudio, Volumio, Pi MusicBox, and Moode Audio.

Working weekends

"It took six to twelve months of working evenings and weekends because we didn't have the money to throw at this at the start. Other individuals and companies were doing similar things and as to who was first, it was too close to call. But the market is big enough, we have our own niches."

IQaudIO may be a limited company, but it is still a part-time affair. "We all have full-time jobs," reveals Gordon. "It is very much a family event. My wife handles all the sales and the shipping, and whenever the kids are on holiday, I have them putting packets together.

"IQaudIO is not just one person. There are three or four of us involved. I have the support of a manufacturer, a couple of hardware designers, a software engineer, and my family too.

CAN THE PI DISRUPT THE AUDIO INDUSTRY?

A Raspberry Pi with a high-quality audio card can sound as good as much more expensive specialist audio products. Is this having a disruptive effect on the industry? Gordon Garrity does not think so.

"I think the Pi itself is pretty insignificant at the moment in the hi-fi industry," he says. That said, the Pi with its highly active community has some advantages over the big names. When the BBC changed its streaming services, abandoning most of its SHOUTcast streams, some were caught out. "You see some well-known companies still not being able to deliver a firmware update that allows people to listen to BBC radio stations. Whereas on the Pi the community had got it sorted within a couple of weeks. People that were using Pi-based music were then in advance of those who had purchased high-end commercial solutions."

The Pi does have a problem when it comes to supporting proprietary systems like Spotify Connect or Apple Music. "The problem is getting the support of those streaming companies when they are already working with consumer electronics manufacturers who bring them many more customers. Their focus will not necessarily be aligned with the community focus on that," says Gordon.

> "From initial concept to having a prototype was relatively quick; the problem was the software"

What's next? "We have a roadmap. We are looking at maybe turning the Raspberry Pi into more of an audio system. It would be great to be able to offer a kit that could be put together as part of a hobby, and that looks good as well."

What about the industry drive towards high-resolution audio? "This can turn into a religious war," Gordon tells us. "What is important is that people enjoy the music. The whole end-to-end chain needs to be good. If you have got good-quality audio, then we are not going to detract from that with the DAC and the DigiAMP. We try to keep the component count and signal lengths as low as possible.

"There will always be specmanship. Rolls-Royce always said that the performance of their engines was 'adequate'. Going up to 32-bit and 384MHz is fine, but we are approaching specmanship at that point. The hardware we use already supports that bitrate and frequency, but the underlying operating system and drivers are not necessarily there, and the chances of customers having such source material is diminished. We need to hit the sweet spot, and that is 44.1MHz and above.

"We are continuing to monitor the hardware manufacturers. I am ex-Texas Instruments, and one of the reasons we brought the DigiAMP+ to market was that it uses a new TI chip.

"We are interested in what is happening in the music industry as well. We have got interest from some new players in the market – I can't mention any names."

SKYCADEMY: MISSION ACCOMPLISHED

24 teachers took to the outdoors to launch some high-altitude balloons in a brand-new Picademy

While Raspberry Pis might be going up in a rocket to the International Space Station for Astro Pi, you don't need to hitch a ride on a Soyuz to get your own Pi into the stratosphere. That's what the recent Skycademy course was all about: a Picademy designed to show teachers and educators how to get started with high-altitude balloons (HABs).

"I'd been thinking about the possibility for a while," says Dave Akerman, who's well known in the Raspberry Pi community for making Pi-powered HABs, even holding the record for the highest altitude an amateur has streamed images from. "I approached Lance Howarth at Raspberry Pi back in January. It was really when James Robinson joined the Foundation that things started to move, with the planning starting in April."

James Robinson is a member of the Raspberry Pi Foundation's education team, and apparently the Skycademy concept was one of the first event ideas mentioned to him when he joined:

"Having done a launch myself, I'd seen that there's not a huge amount involved in each launch, and, although it's not overly technical, there's lots of different disciplines. My role in organising Skycademy was to try and bring some educators together to organise all the logistics behind it and sort of hold their hands throughout their first launch to show them that it is possible, to bring them together to work as a team, and really to give them that support during their first launch. Then hopefully they're going to go away and teach others while doing further launches."

So after juggling the places and managing a waiting list, on 24 August four teams of six teachers met from around the UK to start planning a launch over the next two days.

"The teams were loosely geographical," James tells us. "Our Southern team was Exeter to Kent, so it was very loose. We gave them all an overview of high-altitude ballooning; we talked about the different disciplines and things that were involved. We talked about the maths and gave them an overview of the flight and flight profile and how it works. Then we went through a series of workshops showing them how to set up their payload, build a container around it, and how to track and receive any telemetry coming from those payloads. We tried to give them as much preparation on that first day as possible."

As the teachers went off for the night, James and Dave's work wasn't done, as Dave reveals:

"James and I then continued late into the evening, getting a receiving station set up at Pi Towers, then grabbed a few hours' sleep before the launch day."

"Speaking of sleep, we'd both lost plenty of that already, worrying about the wind conditions and landing predictions, both of which were rather poor. At one point I was emailing James at 4am to tell him that we would probably have to cancel the flights!"

In the end, conditions improved and with a change in launch site, all that was left was the launches.

"I was really blown away by how quickly they took on board what we'd shown them, as were the two experts that we had. As soon as Dave had demonstrated his launch, the teams just disappeared

SPLASHDOWN

While the launch went very well for Dave and three of the teams, one of them had a bit of trouble, as James relays to us:

"We had one team that had a slight issue which did result in the loss of their payload, unfortunately. They underfilled their balloon slightly, so the balloon took a very shallow trajectory into the atmosphere, which meant it flew for longer and ended up in the North Sea… From an objective point of view, it was quite nice that a team lost a payload because it does show that actually, it's not easy and it's not always going to be a success, and you have to plan and take your time. I felt bad for them on the day, though."

and were off setting up and launching their own balloons."

After four successful launches, the teams tracked their balloons and went to retrieve them when they fell back down to Earth following a lofty flight to 35km above sea-level. The final day saw the teams evaluating what they had done and figuring out how they could perform their own launches for students. Skycademy wasn't just those three days, though, and the attendees will be reporting back over the next several months about launches they've been doing.

"The event was just the beginning of a project," James explains. "This marks the start of a year-long project beyond these two-and-a-half days: we're funding each of those teachers to do a launch themselves with their students. Because that's really important to us that, actually: it's not just a bit of fun for a few days, there's lots of learning and great opportunities in this activity to show that science, technology, engineering, and maths are all really interrelated disciplines. They all come together in this kind of thing so we can launch a space mission, or pseudo-space mission, for no money really. We're going to support each of the attendees financially and with support logistics over the next year to do their own launches."

If you regret not getting yourself signed up to Skycademy, don't worry: this is not the one and only time the course will be run. If the results of this course go well, James and Dave hope they can continue it as an annual event.

Check back over the next year, in which we'll hopefully be able to show off some of the successful launches from attendees of the Skycademy.

> "They're going to go away and teach others while doing further launches"

Above Left The balloons are inflated and then sent off into the sky, hopefully to be retrieved later

Top High above the Earth, our intrepid weather balloons take shots of the world below

THE ATTENDEES

24 PEOPLE ATTENDED THE INAUGURAL SKYCADEMY AND WENT AWAY WITH THE SKILLS TO START TEACHING OTHERS ABOUT HAB. WHAT DO THEY PLAN TO DO?

SARAH

CONSULTANT / SCOUT LEADER AND CODERDOJO MENTOR

I plan to run an ongoing project within our local CoderDojo, empowering our youth mentors by giving them set roles and responsibilities, and then rolling it out across CoderDojo in the North West. Within Scouts, we will run a HAB flight as part of a tech camp with various groups in our area involved.

ANDY

COMPUTER MANAGER / SCOUT LEADER

Launches with Scouts around Cambridge and then, hopefully, Cambridgeshire and further? A programme building up to the launches involving understanding the process and tracking other balloon launches.

SUE

TEACHER

I will bring in tracking first: I'll need to give pupils some idea of how GPS etc works. I will liaise with Science to see if they can bring in the Physics and Geography to bring in the tracking/map-reading and weather aspects. Lots of conversations to be had with other departments to get them involved so I'm not doing all the work.

NICHOLAS

HEAD OF COMPUTING

I plan to look at working within school to create a cross-curricular project with Science, Maths, Geography, Computing, Design, and English. Each of these areas can contribute directly to the flight, or use data from the flight within lessons.

GOBOX

MONTHLY MISSIONS FOR ROBOTS

Build up your robot, and knowledge, month-by-month with **Dexter Industries'** innovative new kit

GoBox – the world's first subscription-based robot kit – is set to launch in December, having already achieved its main Kickstarter goal within days (**kck.st/1UDs4rQ**). The idea originated from Dexter Industries' COO Taryn Sullivan, who saw how excited her nephews got when their Tinker Crates (containing science experiments) arrived in the mail every month.

"We've run quite a few workshops around the GoPiGo [robot], and we noticed that in the second and third workshop, people were overcoming big hurdles quicker and learning so much more," reveals Dexter founder John Cole. "This leads us to believe that consistency and spreading out learning over time is a key to success... The biggest reason, though, is to continue to engage and excite people to keep learning. Some programs do it with badges or levels, but we thought creative monthly missions were the right way for robotics."

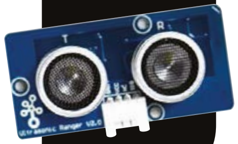
Designed for newcomers to robotics and programming, GoBox assumes no prior knowledge, and takes you step-by-step through a year-long course. In the first month, subscribers will receive a GoPiGo 2 kit: this improved version of Dexter's Pi-powered robot car is more durable and easier to build. In each subsequent month, they'll be sent a project 'mission' to accomplish, complete with detailed instructions, links to helpful videos, and the relevant extra sensor or part to attach to their robot. An example mission sees students using their GoPiGo with a light sensor to mimic an animal such as a bat.

While the GoPiGo can be programmed in many languages, the default missions are all based on Scratch, "because it's so great for beginners." However, if a Kickstarter stretch goal is reached, Dexter will create parallel materials for each lesson in Python. There are also plans to

UPGRADE YOUR ROBOT

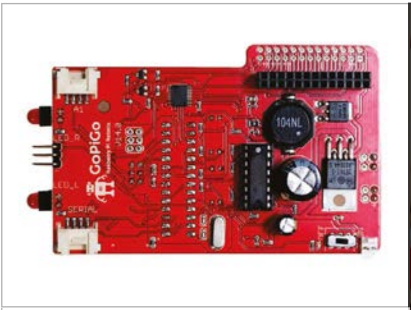
Sent monthly for you to enhance your GoPiGo, sensors and other components include the following:

- Light sensor
- Sound sensor
- Line follower
- Ultrasonic sensor
- Buzzer
- LED
- Motion sensor
- Button
- Infrared sensor
- Camera (advanced only)
- Servo kit (advanced only)



GOPIGO

Launched last year via Kickstarter, the GoPiGo kit (**dexterindustries.com/gopigo/**) turns the Raspberry Pi into a two-wheeled robot. The GoBox programme includes a new version of the GoPiGo that's more durable and easier to assemble. Some parts have been moved around for easier access to the ports, switches, LEDs, and batteries. It's also easier to connect sensors. However, all the software/firmware updates are backwards compatible with the original GoPiGo, as is the GoBox programme.



Missions involve using Scratch

Left The GoPiGo control board

Bottom left An assembled GoPiGo robot car

Below GoBox missions were a hit with testers

> There's a lot to learn about robotics… trying to do it all at once can be overwhelming

develop a 'bridge' at the end of the course to help students advance to Python. "We want to support this GoBox community as they grow," enthuses John, "so you can count on us to be there with the next step to keep learning!"

Go test

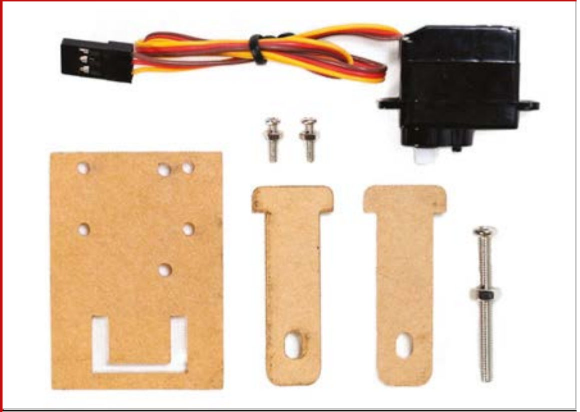
To fine-tune the concept, GoBox underwent rigorous testing with parent-child teams at various skill levels. "There was universal feedback that the programme was really fun," says John, "and we have taken all the good critical advice for improvements to make it easy for beginners… We're also including videos that will show everything in action, if people are more visual/auditory learners."

The modular nature of GoBox also proved popular and effective. "There's a lot to learn about robotics and programming," explains John, "and trying to do it all at once can be overwhelming, boring, and frustrating for many beginners… Breaking it out over time gives people achievable goals, and when they feel successful, they are more likely to come back for more."

Asked whether some students might become impatient for the next monthly mission to arrive, John says this shouldn't be a problem. "What's great about these missions is that they are open-ended enough that more advanced students can take it far beyond where we lead them. That's the whole reason we've designed the missions in this way; the open-endedness slightly nudges new learners to look around, take it in, and think about the concepts and missions on a higher level rather than [ripping] straight through."

While John tells us the current missions are aimed more at one-to-one learning, he reveals that Dexter plans to develop a curriculum specifically for the classroom setting, as the GoPiGo robot has already attracted great interest from schools. "But [in their present form], GoBox missions are great for STEM centres, libraries, after-school programs, and camps!"

GOBOX OPTIONS



Kickstarter reward levels for GoBox include six- and twelve-month programmes, with or without the GoPiGo robot included (in case you already have one). An advanced option adds a Pi Camera Module and servo kit. There's also an international version to reduce postal costs, with shipments delivered quarterly rather than monthly (each containing three missions).

FESTIVAL OF CODE 2015

The world's biggest hacking event saw over a thousand youngsters showcasing their projects, including some remarkable Raspberry Pi creations...

Left The teams made project presentations in the ICC auditorium

Mighty oaks from little acorns grow – and this applies to both the ethos and rapid growth of Young Rewired State's annual Festival of Code. What began in 2009 as a weekend gathering of 50 self-taught young coders, aimed at introducing them to the benefits of open data, has since expanded to become the world's largest annual hack event for young people. Some 1,200 under-18s attended this year's finale at Birmingham's ICC (International Convention Centre), showcasing the projects their teams had created over the previous week at one of 66 technology centres, most in the UK, plus a few overseas. So great was the attendance – along with parents, mentors and centre leads – that it took six hours to process them all through ICC registration on the Friday afternoon.

"This year it felt like we had finally come of age," says YRS founder Emma Mulqueeny. "Every year we learn hard lessons! And every year we try to make sure we address those. This year it really seemed like we had managed to work out how to address the issues of scale... The feedback has been incredible and really wonderful to see. The standard of the hacks produced by the teams was also higher than ever."

Infectious enthusiasm

The sheer level of enthusiasm is something noted by most attendees, including Raspberry Pi Foundation CEO Philip Colligan, who was on the judging panel for the heats and semi-finals. "The kids are awesome," he tells us. "Walking around, you can see that they love every minute of it." He's also amazed at what the entrants managed to achieve in just a week: "They're learning new programming languages, how to access and use open data, they're doing physical hacks and physical computing projects." Most of all, however, he stresses the importance of the kids solving problems that they care about, which means they're totally driven to learn what need to create and showcase their project.

Fellow judge and Raspberry Pi Foundation colleague Marc Scott was equally impressed. Describing the scene on the Friday evening, as entrants continued to work on their projects, he says: "It's quite an amazing sight to see that many kids all hunched over laptops, tablets, phones, and Raspberry Pis, with screens full of nothing more than text."

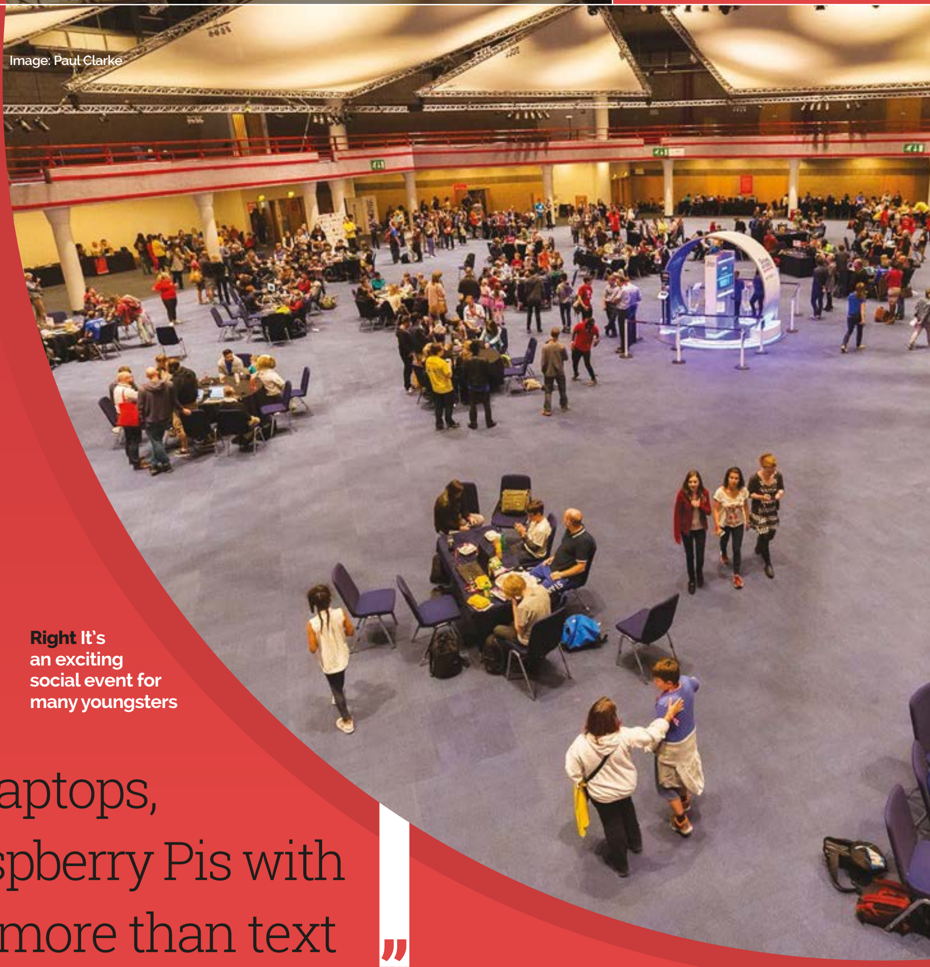


Image: Paul Clarke

Right It's an exciting social event for many youngsters

" Kids all hunched over laptops, tablets, phones, and Raspberry Pis with screens full of nothing more than text "

"The atmosphere on the Friday is always a slightly anxious one," adds Emma. "Everyone is nervous ahead of presentations the next day. However, for the ones coming back, it is a huge reunion – and the whopping amount of pizza always helps! On Saturday it is much more buzzy; everyone is showcasing what they have built, they have all connected again with each other or made new friends: sometimes these are lifetime friendships."

Social experience

The benefits for the Festival's young participants are manifold, according to Marc. "For many attendees it's their first real experience of interacting with the 'hacker community' face-to-face. They'll have people they know through social media that share their interests, and maybe one or two friends at school, but nothing on the scale of the Festival of Code. It's nice to get together with a couple of friends to hack a project together, but being able to stand in a room filled with nearly a thousand people and know that each and everyone of them is as passionate about coding as you are is very empowering."

Emma concurs: "Many of these young developers have taught themselves, and before the Festival existed, these youngsters were pretty isolated people,

REGISTER FOR 2016

If you want to get involved in next year's Festival of Code – as a participant, mentor or centre – you can register your interest at **festival.yrs.io**. The Festival is completely free to participate in and open to anyone under the age of 18, although you need to have at least some basic coding experience. Successful applicants are sent to a nearby technology centre, where mentors will help them to design their project during the Festival week, before it's showcased at the weekend finale.

Image: Paul Clarke

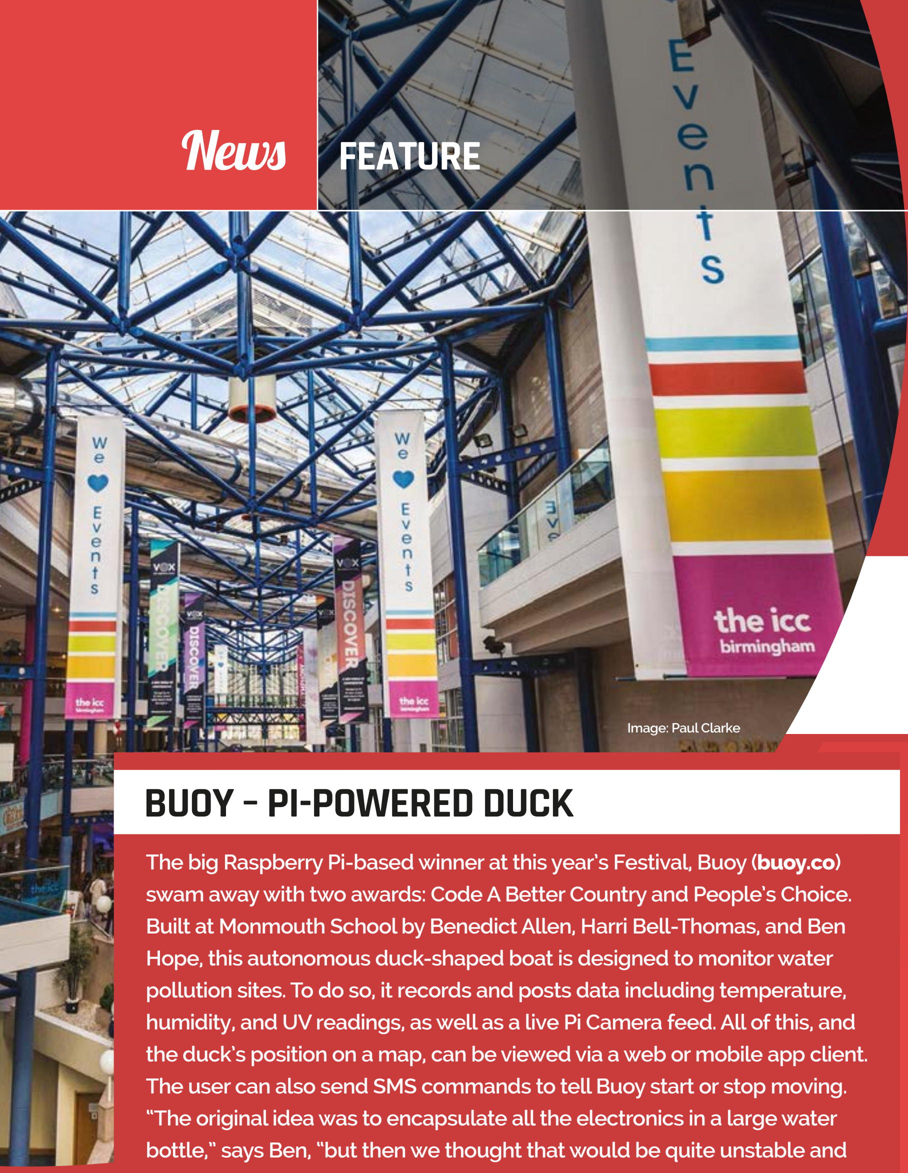


Image: Paul Clarke

PI PROJECTS

OTHER RASPBERRY PI-BASED PROJECTS AT THIS YEAR'S FESTIVAL INCLUDED...

ALLAMA

This smart alarm clock works out the best time to wake you up, depending on current traffic conditions, which the Pi monitors via Google Maps. Just enter your route, the time it takes you to get ready and arrival time, and aLlama (**allama.co.uk**) does the rest.

BUOY – PI-POWERED DUCK

The big Raspberry Pi-based winner at this year's Festival, Buoy (**buoy.co**) swam away with two awards: Code A Better Country and People's Choice. Built at Monmouth School by Benedict Allen, Harri Bell-Thomas, and Ben Hope, this autonomous duck-shaped boat is designed to monitor water pollution sites. To do so, it records and posts data including temperature, humidity, and UV readings, as well as a live Pi Camera feed. All of this, and the duck's position on a map, can be viewed via a web or mobile app client. The user can also send SMS commands to tell Buoy start or stop moving. "The original idea was to encapsulate all the electronics in a large water bottle," says Ben, "but then we thought that would be quite unstable and not protective enough of everything on the inside." The use of a giant rubber duck was inspired by the thousands of rubber ducks which were accidentally released by a tanker in 1992 and which have subsequently been used for measuring ocean currents.

While the Buoy prototype is powered by two Pis – one to collect data from sensors and the other to control the motor via a Gertboard – the team are confident they can get it running off just one Pi. They also plan to make Buoy more customisable for different situations, and hope to eventually make it available to the general public: "It would have plenty of uses in pools, ponds, and just any body of water that needs constant monitoring, whether that be through the sensors or the Pi Cam stream coming from it."

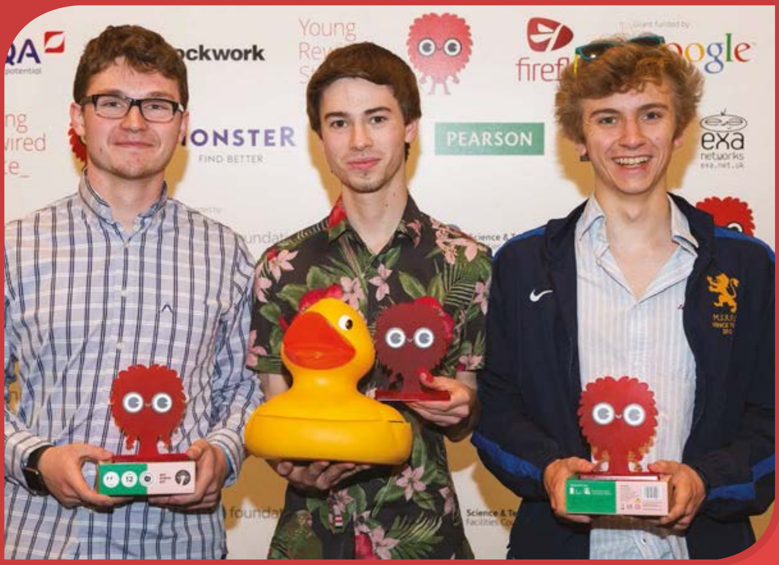


Image: Paul Clarke

both at school and in their learning. Now they are growing up with hundreds/thousands of people exactly like them. They love teaching each other, they love learning from each other, and they love challenging themselves. What I see every year is a child spotting something that they want to learn by the time they come back, and then doing so; so the impact on the actual skill they are picking up is huge."

Such is the lure of the annual Festival that many previous participants who pass the maximum age threshold of 18 ('graduates', as Emma calls them)

"They love teaching each other, they love learning"

return to mentor youngsters or act as judges. Most go on to university, while others have gone to work in startup companies or have formed their own with the aid of the US-based Y Combinator seeding fund.

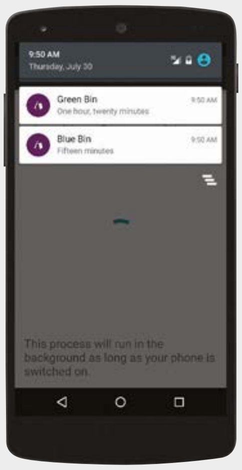
Hardware hacks

The wide range of projects this year (**bit.ly/1KANCNv**) was impressive, particularly given the short development time. "I'm always blown away by what is produced for the Festival of Code," says Marc. "There are apps, websites, and hardware hacks that range from the purely zany, to those that could genuinely do some social good, or even make some money. The standard is always high, and you can see that the attendees have really given it their all, no matter what the level of technical expertise they possess."

While the majority of projects this year were software-based, hardware hacks seemed to find favour with the judges, including several Raspberry Pi-based projects. Most notable of these was Buoy

WHEEL YOUR BIN

If you forget to put your bin out on the correct day, the Wheel Your Bin mobile app will alert you. It works using ultrasonic detectors in a wall-mounted, Pi-powered box, so it knows if your bin is in the usual place or out for collection.



Green Bin 9:50 AM
One hour, twenty minutes

Blue Bin 9:50 AM
Fifteen minutes

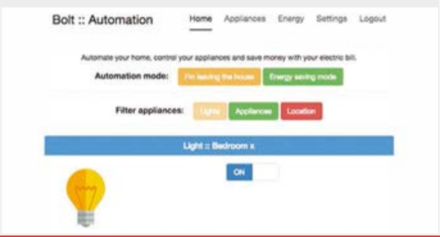
This process will run in the background as long as your phone is switched on.

BARTRACKER

Designed to help consumers track the route of ethically produced goods, this Pi-based client-server system uses a standard retail barcode scanner. Just scan a product's barcode and it'll tell you where and when it was previously scanned.

BOLT

This home automation system makes use of multiple Pis – to log electrical current data and manage relays for remotely controlled household appliances. One of the Bolt team, Humza Bobat, has already wired up a working prototype in his house!



Automate your home, control your appliances and save money with your electric bill.

Automation mode: I'm leaving the house | Energy saving mode

Filter appliances: Lights | Appliances | Location

Light :: Bedroom x

ON

PIPIC

This Pi-powered photo booth automatically takes a picture when its motion sensor is activated and then tweets it, along with metadata such as the time and weather conditions.

(see boxout on left), a water pollution-detecting Pi-powered duck, which won the 'Code A Better Country' and 'People's Choice' awards. It was a favourite of Philip's: "They'd actually managed to get it to the stage in just a week where this thing was navigating itself around the pool, which was just amazing." And while not all of the hardware projects made use of the Pi, Philip notes: "Most of the kids there that I spoke to owned one or more Raspberry Pis and they're using them to learn on, which is exactly what we want."

Another standout project was the Intelligent Elephant Alarm Clock designed by a trio of girls aged seven and eight, which was a finalist in the Should Exist category. Created using ScratchX and a PicoBoard, the clock makes use of local weather and traffic data to determine when the user is woken by one of five animal alarms. "They had one of the most complicated but effective Scratch programs I've ever seen," says an astonished Philip. He was also impressed by 'Best Example of Design' winner ArduDuck, a USB dongle that instantly adapts any PC to the user's special requirements so they can start using it straight away.

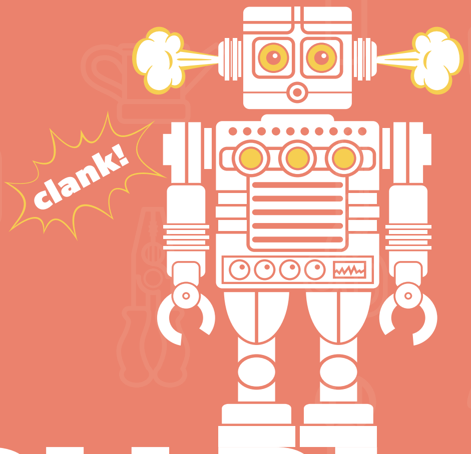
Considering the high standard of entries, Philip says it was really hard to judge them: "There was a fair bit of argument amongst the judges," he reveals. Projects were assessed using four criteria: the quality of code, novelty factor, whether it addressed the problem that was being solved, and use of open data.

Following heats and semi-finals for four main categories, and final project presentations on the main stage, the winners were revealed on the Sunday afternoon. "When we announce the semi-finalists and the finalists, it is a bit like X Factor for geeks!" laughs Emma. "The finale is *always* emotional. Tears from parents, mentors and centre leads, nerves from the finalists, and huge sadness and grief when it is all over: no one wants it to end and to have to wait another year."

NEW PI GUYS

Festival of Code judges Philip Colligan and Centre Lead Marc Scott have both recently joined the Raspberry Pi Foundation. Philip is the new CEO, responsible for overseeing all of the organisation's charitable activities, while former teacher Marc (left) has joined the Education Team as Head of Curriculum. We'll bring you full interviews with both of them in a later issue.

Image: Paul Clarke

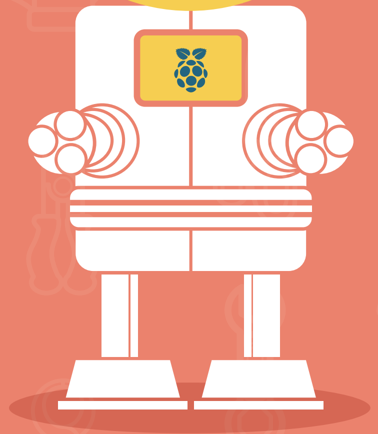


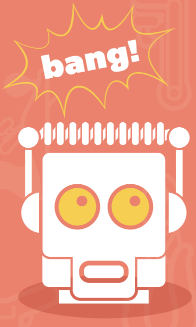
BUILD A RASPBERRY PI ROBOT FOR £50

Need a little project to help start the robot uprising? Build our tiny little automaton and conquer the world. Or your living room…

NAME OUR ROBOT!

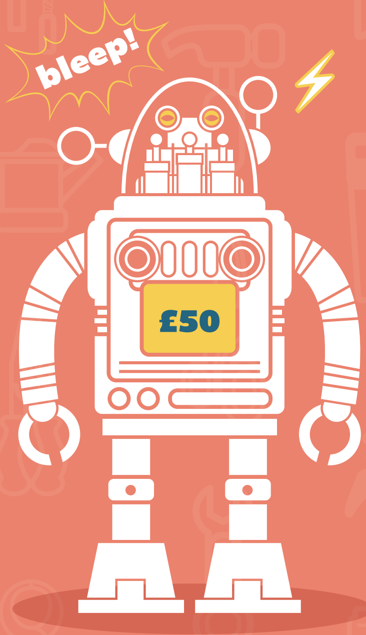
We don't have a name for our Raspberry Pi robot yet. Tweet us your suggestions @TheMagP1 and we'll pick the best one!





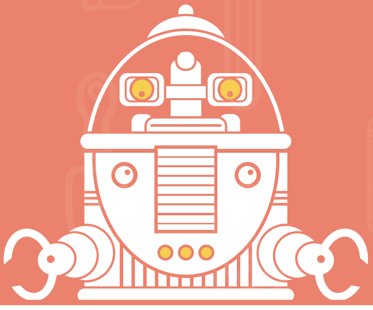
If you've never built a robot, you are truly missing out. The process of assembling one and finally getting it to work is exactly as it seems in the movies. The elation and pride at your final product as it trots along is unequalled.

While there are plenty of great, easy-to-assemble kits on the market, we like the idea of creating a more manual, DIY robot. And we also don't want you to break the bank or spend weeks making it. So we decided to sit down and think up a robot that could be put together in a weekend for £50 ($77), including the Raspberry Pi. So get your clothes, your boots and your motorcycle, and head off to grab everything you'll need to build your own little robot pal.



YOUR SHOPPING LIST

What you need to build your own affordable Raspberry Pi robot



[1] Raspberry Pi A+

While it may be the cheapest and smallest Raspberry Pi, the A+ is certainly powerful enough to process everything you need to make this robot a reality. We're also going to use it as the chassis; it's strong enough for what we have in mind, at least.

£15.50

[2] Micro metal gear motors

For our tiny little robot built from an A+, we need a couple of motors small enough to attach to it, and powerful enough to drive it around. These guys are just that.

£3.40 each

bit.ly/1g7Tva0

[3] Wheels

You need wheels for the motors. That's pretty self-explanatory. These ones are small enough for the job.

£4 for a pair

bit.ly/1g7UaZe

[4] Ball castor wheel

We're opting to use a ball castor on the robot's trailing edge instead of more motors. It keeps the cost down and makes it a bit simpler to control in our experience. While you can get away with one castor, if you can spare the change, buying another one wouldn't hurt.

£2

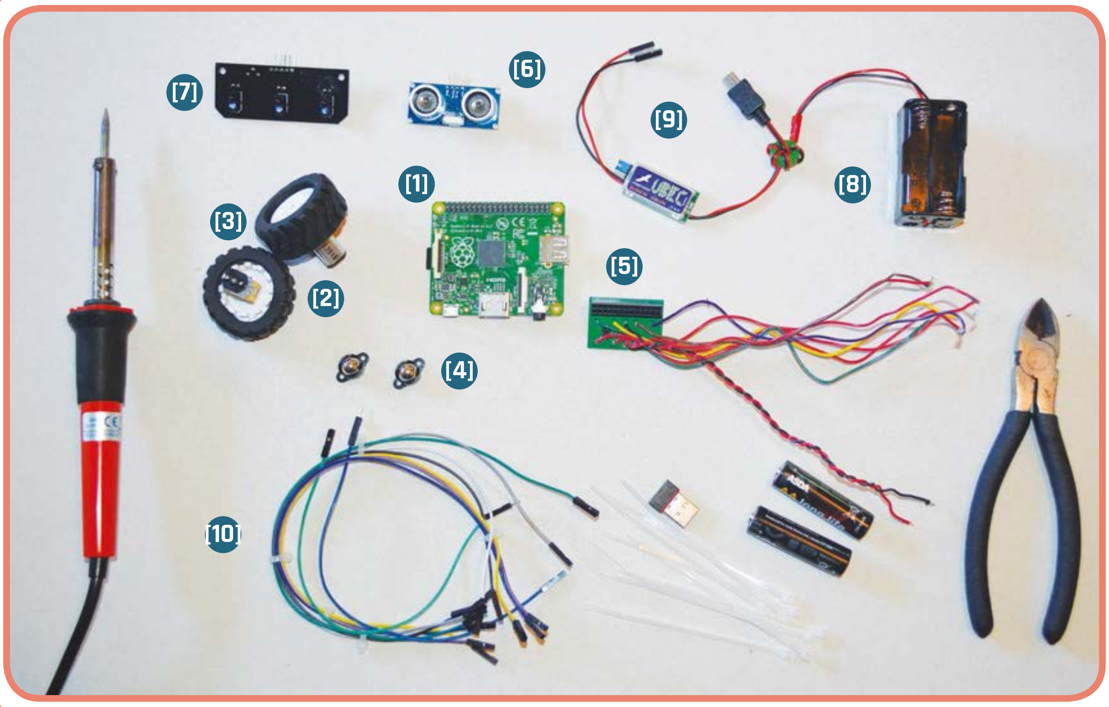
bit.ly/1XHPlHX

[5] PicoBorg motor board

PiBorg's tiny motor board is able to control four separate motors using only a Raspberry Pi and a modest external power source. It's the perfect size for our little robot, as well.

£6.99

piborg.org/picoborg



[6] HC-SR04 ultrasonic sensor

We don't want our robot to race straight into a wall, so we've got an ultrasonic sensor that can let it know where to stop… or who to follow.

£2.49

bit.ly/1JSxo29

[7] Three-way line follower

It almost gives the robot little wings, but the separation of the Ryanteck line sensors is perfect for our purposes in this robot. And it's cheap.

£4.99

bit.ly/1NgC4kS

[8] Battery case

This holds the batteries that power the robot. You can pick up a four-battery case for very little money from just about anywhere.

90p

[9] UBEC battery cable

Unless you fancy having your robot leashed to a power supply, you'll need one of these amazing cables from Dawn Robotics to convert the battery input to something that won't damage the Raspberry Pi.

£5.99

bit.ly/1HLKih7

[10] Bits and bobs

You'll need a bit of wire, some Blu-Tack, batteries, a few resistors, a wireless dongle, and preferably a soldering iron to make sure everything goes together correctly. If you don't have some of these by now, it's a good time to grab them.

TOTAL COST: £49.66

A LITTLE INSPIRATION

Using a Raspberry Pi Model A+ as a chassis? Cable-tied motors? If you pay very close attention to everything on the Raspberry Pi blog, this may sound a little familiar to you as this was also how the Toast-E robot was created by Gert van Loo last year. We've taken a little inspiration from him to make our cut-price robot, which functions similarly to his. Read more about it here: **gertbot.com/toaste.html**

PREPARE YOUR COMPONENTS

Soldering, wiring, and more…

Before we start constructing the robot, we need to begin preparing the components and the Raspberry Pi to be used for this particular project. Let's start with the Raspberry Pi.

Prepare the Raspberry Pi

If you haven't already got one prepared, create a Raspbian SD card for the Model A+. Perform the updates required to get packages and kernel up to date, and have it for the moment log in to desktop. Connect the WiFi dongle if you haven't already; you can control the robot via SSH on the fly while it's connected to the wireless, making for better control.

Connect to the WiFi via the normal graphical method and the Pi will remember the password and the wireless network. By using DHCP, your router will likely give the Pi the same IP address when it connects; to make sure this is always the case, however, we can set a static IP.

Open the terminal and use **`ifconfig`** to determine the IP address DHCP is giving it (inet addr), the Broadcast Range (Bcast) and the Subnet Mask (Mask). Once you've made a note of those, use **`sudo route -n`** to figure out the Destination and Gateway address for the network and router.

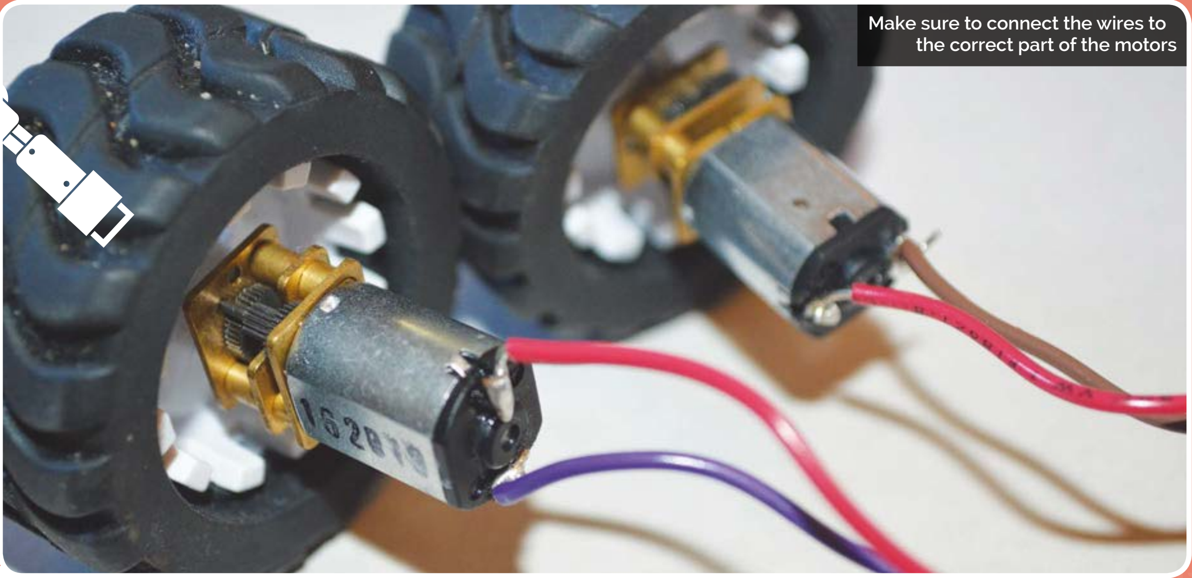
Open up the network interfaces file (in the terminal, use **`sudo nano /etc/network/interfaces`**) and you'll be greeted with the current settings for your network devices. Your wireless settings will look something like:

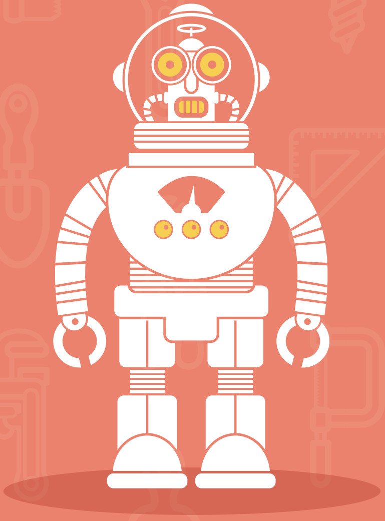
```
allow-hotplug wlan0
iface wlan0 inet manual
wpa_roan /etc/wpa_supplicant/
wpa_supplicant.conf
```

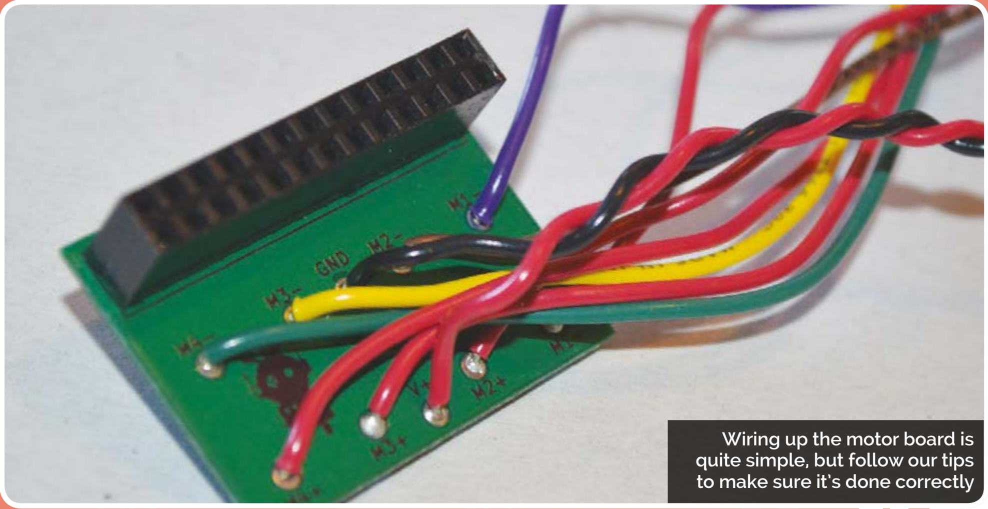
Change **`manual`** to **`static`** and add the following five lines:

```
address [the IP address you
        want to use]
netmask [the netmask]
network [the destination]
broadcast [the broadcast range]
gateway [the gateway]
```

Save, and after a reboot, the IP will stay static. If you want to, you can also go to **`sudo raspi-config`** and have the Pi boot to desktop, before getting on with the rest of the robot.


Make sure to connect the wires to the correct part of the motors




Wiring up the motor board is quite simple, but follow our tips to make sure it's done correctly

Soldering the PicoBorg

In our pictures, you can see that we have a fair few extra wires on our PicoBorg that are unused: this is because we've already put ours together for four motors. For this project, all you need to do is solder up the first and second motor, along with the battery power cables that drive the motors. Make sure you have plenty of cabling for the different input and outputs, and we suggest using a red wire for the positive and a black wire for the negative.

For the PicoBorg, it's advisable to use a good-quality, single-core cabling and when soldering, you need to make sure you don't overheat the PCB. You can read some more tips on how to solder the PicoBorg on the specs page (**piborg.org/picoborg/specs**), or you can spend a little more to get a pre-soldered one.

Once it's connected, you'll also have to solder a couple of wires onto the top side of the board: we need a 5V and 3V3 pin for the ultrasonic sensor and the line follower respectively, and there's none spare on the remaining 14 pins of the GPIO once the PicoBorg is affixed. From the picture, you can see we have attached a cable with a female pin port to pins 1 and 2 of the header: these are the two pins that when affixed onto the Raspberry Pi are closest to the edge where the SD card goes. The pin on the corner is pin 2, which is 5V. The pin next to it is the 3V3 pin.

Soldering the motors

The motors won't have wires attached to them, so you can solder the PicoBorg wires directly to the motors. This is fairly simple: all you need to do is make sure the positive and negative wires on each motor control go to the positive and negative connectors on the motor.

Wire up the battery pack

We need the batteries to connect to both the UBEC and the motor board. What we've done is strip the wire down a bit and soldered the motor-board cables straight to the battery case's wires. The UBEC is connected via its female connectors onto the end of the wires. You should probably cover the whole lot with insulation tape once finished, to avoid any chance of a shock!

Attach the ultrasonic sensor resistors

The ultrasonic sensor will be able to mostly connect directly to the GPIO pins, with the exception of the ECHO pin on the HC-SR04 requiring some resistors to work properly: a 1K and a 2K resistor. The 1K needs to go between the pin and the GPIO pin and the wire going to the ECHO port. The 2K needs to attach the GPIO pin to the ground pin we're using. As you'll see over the page, they'll be very close to each other, so it will be pretty easy to bridge this gap without breadboards.

ALTERNATIVES

Fancy splashing some more cash? Here's what you could go for...

DAGU 2WD CHASSIS

bit.ly/1Njk1dP

The inspiration for the wheel configuration of our robot, this chassis sold by Dawn Robotics is one of the simplest around. It's a fair bit bigger than the A+ as well, so you could always mount a Raspberry Pi B+ or 2 on top of it for more power and possibilities. It's quite cheap and a good next step.

SENSE HAT

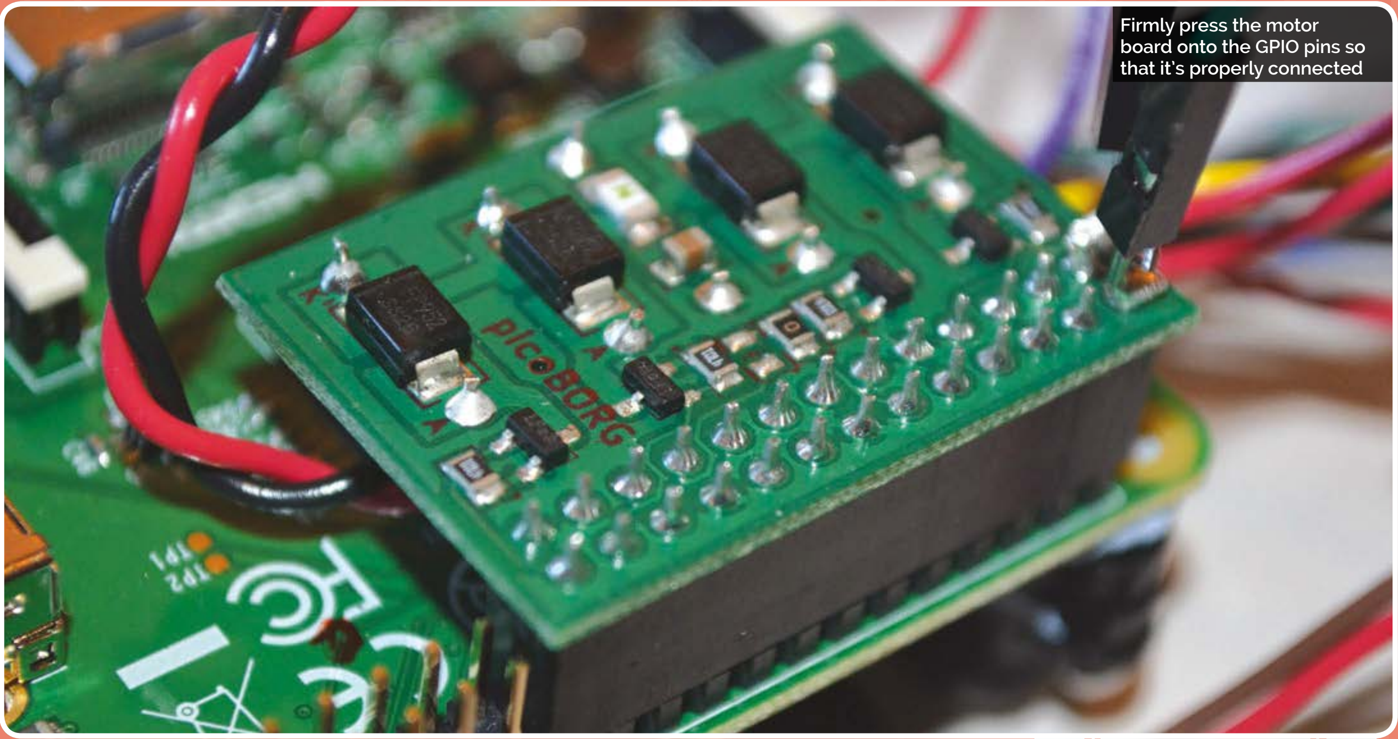
The Sense HAT can sense extra variables, such as if you're going up an incline or how fast you're going. It can also display little images, such as arrows to show where it's going. It can give your robot a little more life and make it a bit more interesting, and it won't take away any GPIO ports.

BUDGET ROBOTICS KIT

bit.ly/1L3Jmbw

Want something a little simpler to make with the same capacity for expansion? The Ryanteck budget robotics kit requires you to already have a Raspberry Pi, but otherwise the rest of the gear needed to build and power it is included.

Dagu 2WD Chassis


Firmly press the motor board onto the GPIO pins so that it's properly connected

CONSTRUCT YOUR ROBOT

Plug it all together. Just don't forget the Blu-Tack

The first thing we're going to connect to the Raspberry Pi is our motor board. Most of the big components are connected to it in some way, so it's best to start with that first so we can wrangle all the wires around it. Look at our images for reference on orientation when placing the PicoBorg onto the GPIO: you want the PCB going into the Raspberry Pi rather than hanging off it; note that it connects with the pins on the farthest side from the USB port.

Put the wheels on the motor and then attach them: we ran a cable tie through the screw mount holes and tightened it very firmly to keep the motors in place on the edge of the PCB. Try to make sure they're tied in the same place and with plenty of room for the wheels to spin without scraping against the side of the Pi.

Attach the castor wheel to the rear of the Pi: we put a single one on the microSD card slot, or you could have two either side. These castors should come with some spacers, which are good for keeping the Pi off the floor. We attached ours with Blu-Tack, but a couple of tight cable ties will hold things in place much better if your Pi manages to get up to speed!

We then used Blu-Tack to attach the line follower and the ultrasonic sensor: the line follower was stuck on below the PCB, and the ultrasonic sensor on the USB port. The ultrasonic sensor won't be very stable, so if you can find a better way to attach it, go ahead.

Now, you'll need to plug everything in. The power cables soldered to the PicoBorg should be attached to the correct components (the ultrasonic sensor takes the 5V, which is pin 2; the other goes to the line follower) and then you should refer to our diagram (on the next page) to see where the other cables should be attached. Luckily, there's plenty of room left on the GPIO pins.

The final thing remaining is to place the battery pack on the Raspberry Pi: there should be some space next to the PicoBorg. You might think the next task would be to plug the UBEC cable in and start the Pi, but we don't recommend doing it that way just yet. We'll program the robot first, so that it can safely use the motors in the way we desire, rather than have the entire robot powered up.

DISASSEMBLE

Done with your robot for a while and need to get some of your parts back? Most of the robot will be easy to take apart – just remove a few wires – but you'll also have to snip those cable ties and probably unsolder the battery pack and extra wires on the motor board to make sure it's all usable elsewhere. If you can come up with more practical, reusable alternatives for some of the ways we've constructed this robot, don't hesitate to use them!

PROGRAM YOUR ROBOT

The final step before unleashing your robot upon the world

>STEP-01

First boot

Connect an external power source to your Raspberry Pi (not the battery pack and UBEC cable for the moment) and let it boot up. Give it about 30 or so seconds to get turned on – we usually go by when the wireless dongle starts to flash – and then connect to the Pi using SSH from another computer. You can do this either in the terminal of OS X or Linux, or by using PuTTY in Windows. Use the IP address you set as static to connect, with the user name **pi** and password **raspberry**.

>STEP-02

Install PicoBorg

The PicoBorg is the main thing you need to install to enable you to make the robot work. To do this, you need to first make a new **picoborg** directory with **`mkdir picoborg`**, move into it using **`cd picoborg`**, and then run the following commands:

```
$wget http://www.piborg.
org/downloads/picoborg/
examples.zip
$unzip examples.zip
$chmod +x install.sh
$./install.sh
```

>STEP-03

Write your code

Use cd to go back to the home folder, with **`cd ~`**, and make a new directory for our robot. Let's call it **robotcode** (**`mkdir robotcode`**) and use cd again to move into it. We can now write the line-following script that will allow the Raspberry Pi to automatically follow a script and be a true robot. Type in:

```
$ nano MagPibot.py
```

…and copy the code in from the code listing over the page, or download it from our website and copy and paste the code into the script. Save it.

>STEP-04

First tests

If the batteries are in the battery case, take them out. This will make sure the motors don't start as we begin the test. To run the script, you'll need to use:

```
$ sudo python MagPibot.py
```

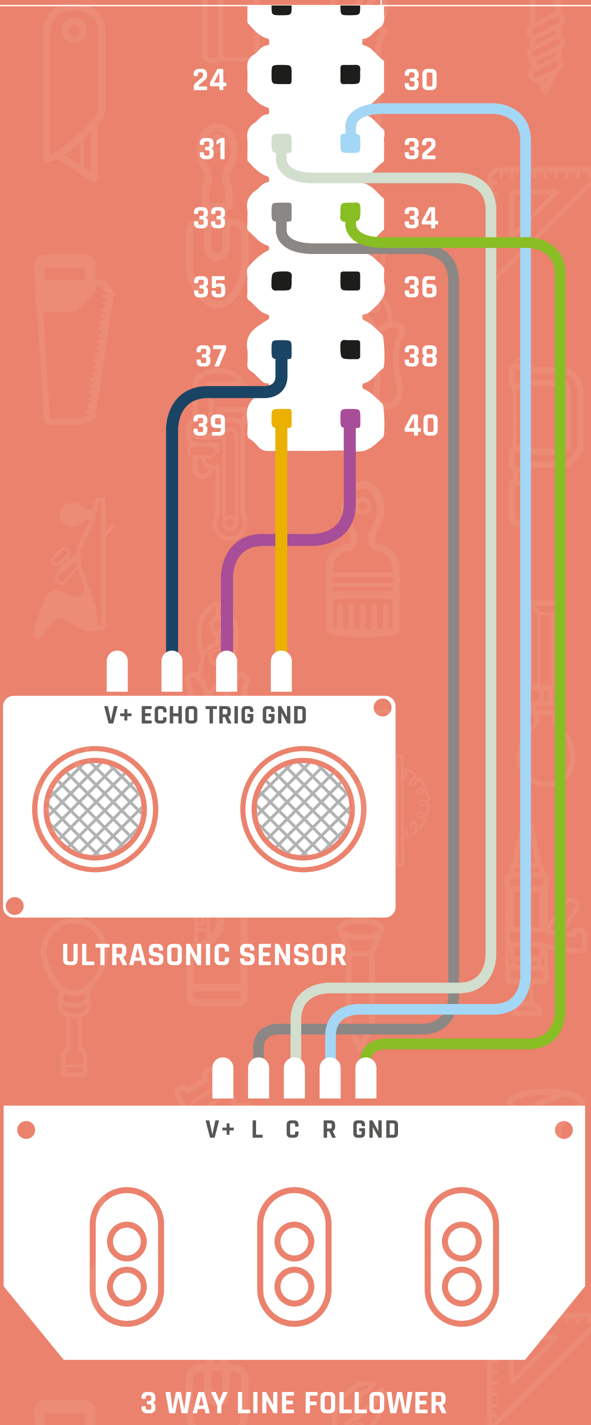
We need to run it as sudo so it can access the GPIO pins properly. It will start up all the GPIO pins and make sure everything is working before asking you to press the **RETURN** key to continue. If the motors were powered, you'd be able to start the full script from here. For now, quit the program with **CTRL+C**.

>STEP-05

First run

Shut the Raspberry Pi down and put the batteries back into the battery case. Remove the external power cable, and plug in the UBEC. The Raspberry Pi should boot up as it did before, and you will be able to connect to it via SSH. If you don't have a course set up for it to run on, no problem: for now, we can have it just run along a surface. Run the Python script like in the previous step, and make sure you're near the Raspberry Pi to grab it in case it starts racing off a little too quickly.

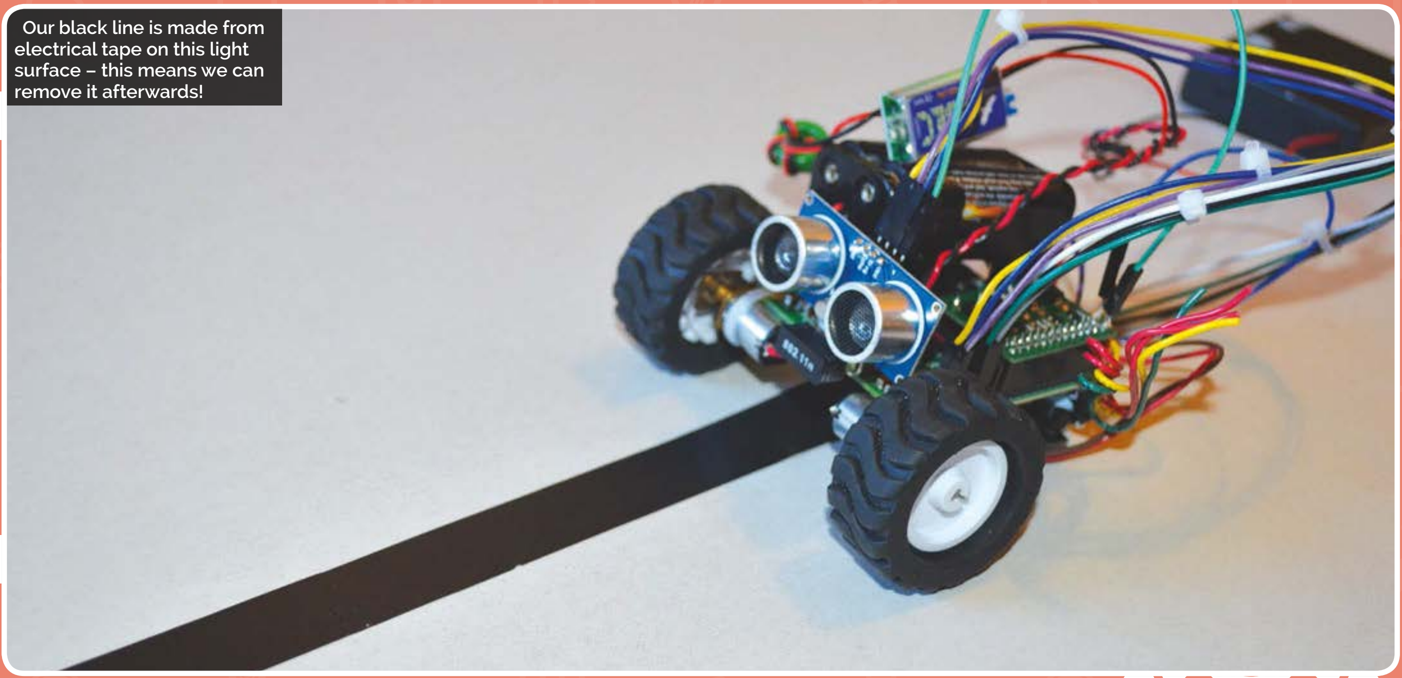
Remember, you can turn it off with **CTRL+C**.



>STEP-06

Tweaks and updates

Our code served our purposes, but it may not be exactly what your robot needs. Over the next page we'll explain the important parts of the code so that you can have a go at changing it yourself; this mainly revolves around how the robot will react to the line it's following or any objects in its path. You can open up the Python script as we did earlier with **`nano MagPibot.py`** and make the changes there.


Our black line is made from electrical tape on this light surface – this means we can remove it afterwards!

A ROBOT'S BASIC FUNCTIONS

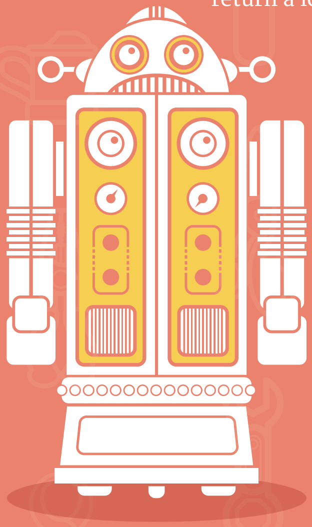
How to control your robot and how it can control itself

Follow a line

The code listing on the next page has a very specific function of allowing the robot we've built to do line-following. This is where it tries to follow a piece of black tape to a goal. The sensors we've attached to the bottom of the Pi can detect where there's a black line, opposed to a white or light surface. They return a low signal when they're over the black line (0), so we can make use of that to tell our robot how to control itself.

As we're solely interested in course corrections, we only make use of the left and right sensors. In our code, we check the input of those sensors in each cycle of the `while` loop and see if one of them has been activated. We then use an `if` statement to see whether the course correction needs to be made or if it can carry on forward. When a sensor is triggered, the motor for that side is kept on, while the opposite motor is briefly switched off so that the robot can turn. It then also moves forward a little before carrying on its way.

You may need to tweak the timings on the turn and the forward motion to make sure the robot stays on your course a little better. The timings are literally done in seconds, so play about with them.



Don't hit a wall

The final part of the code is the use of the ultrasonic sensor. It's used very simply to sense the distance to whatever objects are in the way of the robot as it trundles forward. At the moment, we've set it so that if it's within 5cm of an object, it should stop and not hit it. The `distance > 5` measurement is for that; depending on the braking capabilities, grip, and weight load on your robot, you may want to change this measurement.

When the robot finds an obstacle in its way, it activates one of the two bits of code designed to turn it off. It stops the motors, does the GPIO cleanup (which is important), then lets you know what's happened.

Add more yourself

This code shows off just about every major function of the robot. What can you do with this knowledge and code? With a bit of tweaking, you could make it remote-controlled by sending direction commands to the robot over SSH, or you could give it a preset path to follow that you've programmed into it.

Have fun with your robot, and don't be scared about experimenting and making it the robot you've always dreamed of. You could maybe even think about expanding your robotics expertise with a more complicated project in the future...

Language
>PYTHON 2.7

MagPibot.py

```
#!/usr/bin/env python

# Import libary functions we need
import RPi.GPIO as GPIO
import time
import sys
GPIO.setmode(GPIO.BCM)

# Set which GPIO pins the drive outputs are connected to
DRIVE_1 = 4
DRIVE_2 = 18

# Set the GPIO pins of the ultrasonic sensor
TRIG = 40
ECHO = 37

# Set the GPIO pins of the line sensor
CENTRE = 31
LEFT = 32
RIGHT = 33

# Set all of the drive pins as output pins
GPIO.setup(DRIVE_1, GPIO.OUT)
GPIO.setup(DRIVE_2, GPIO.OUT)

# Function to set both drives off
def RobotStop():
    GPIO.output(DRIVE_1, GPIO.LOW)
    GPIO.output(DRIVE_2, GPIO.LOW)

# Function to set both drives on
def RobotForward():
    GPIO.output(DRIVE_1, GPIO.HIGH)
    GPIO.output(DRIVE_2, GPIO.HIGH)

# Set ultrasonic pins function
GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(ECHO, GPIO.IN)

# Give sensor time to settle
GPIO.output(TRIG, False)
print “Waiting for ultrasonic sensor to settle...”
time.sleep(2)
print “Settled!”

# Set line follower pins as inputs
GPIO.setup(CENTRE, GPIO.IN)
GPIO.setup(LEFT, GPIO.IN)
GPIO.setup(RIGHT, GPIO.IN)

try:
    # Start by turning all drives off
    RobotStop()
    raw_input(“We’re ready to roll, press ENTER to continue”)
    while True:
        # Get distance to object

        GPIO.output(TRIG, True)
        time.sleep(0.00001)
        GPIO.output(TRIG, False)

        while GPIO.input(ECHO)==0:
          pulse_start = time.time()

        while GPIO.input(ECHO)==1:
          pulse_end = time.time()

        pulse_duration = pulse_end - pulse_start

        distance = pulse_duration * 17150

        distance = round(distance, 2)

        # Check line followers
        line_left = GPIO.input(LEFT)
        line_right = GPIO.input(RIGHT)

        if distance > 5:
            if line_left == 0:
                # Drive 1 state
                GPIO.output(DRIVE_1, GPIO.HIGH)
                # Drive 2 state
                GPIO.output(DRIVE_2, GPIO.LOW)
                # Get back on course
                time.sleep(1)
                RobotForward()
                time.sleep(0.5)
            elif line_right == 0:
                # Drive 1 state
                GPIO.output(DRIVE_1, GPIO.LOW)
                # Drive 2 state
                GPIO.output(DRIVE_2, GPIO.HIGH)
                # Get back on course
                time.sleep(1)
                RobotForward()
                time.sleep(0.5)
            else:
                # Move forward
                RobotForward()
        else:
            RobotStop()
        print ‘I have stopped, please turn off my power!’
            GPIO.cleanup()
        sys.exit()

except KeyboardInterrupt:

    # CTRL+C exit, turn off the drives and release GPIO pins
    print ‘Stopped early, please turn off my power!’
    RobotStop()
    GPIO.cleanup()
```

SUBSCRIBE TODAY!

Subscribe to the Official Raspberry Pi mag today for a whole host of benefits

Subscription benefits

- Save up to 25% on the price
- Free delivery to your door
- Never miss a single issue
- Get it first (before stores)

100 PAGES OF RASPBERRY PI

SAVE UP TO 25%

Pricing

Get the first six issues:

£30 (UK)

£45 (EU)

£50 (Rest of World)

Subscribe for a year:

£55 (UK)

£80 (EU)

£90 (Rest of World)

Direct Debit

UK readers can pay **£12.99** by Direct Debit every three months.

Four ways to subscribe:

- Visit **www.bit.ly/MagPiSubs**
- Call +44 (1)1202 586848
- Use the form to the right
- Search 'The MagPi' on app stores

Available on the App Store

Get it on Google play

SUBSCRIPTION FORM

YES! I'd like to subscribe to The MagPi magazine & save money

This subscription is: ☐ For me ☐ A gift for someone*

Mag#38

YOUR DETAILS Mr ☐ Mrs ☐ Miss ☐ Ms ☐

First name .... Surname ....

Address ....

....

Postcode .... Email ....

Daytime phone .... Mobile ....

If giving The MagPi as a gift, please complete both your own details (above) and the recipient's (below).

GIFT RECIPIENT'S DETAILS ONLY Mr ☐ Mrs ☐ Miss ☐ Ms ☐

First name .... Surname ....

Address ....

Postcode .... Email ....

PAYMENT OPTIONS

1 DIRECT DEBIT PAYMENT £12.99 every 3 issues (UK only)

Instruction to your bank or building society to pay by Direct Debit

DIRECT Debit

Please fill in the form and send to:
The MagPi, Select Publisher Services Ltd,
PO Box 6337, Bournemouth BH1 9EH

Service user number 8 3 8 7 7 3

Name and full postal address of your bank or building society:

To: The Manager Bank/building society ....

Address ....

....

.... Postcode ....

Name(s) of account holder(s) ....

Branch sort code ☐☐☐☐☐☐ Account number ☐☐☐☐☐☐☐☐

Reference ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐ (Official use only)

Instruction to your bank or building society
Please pay Select Publisher Services Ltd Direct Debits from the account detailed in this instruction subject to the safeguards assured by the Direct Debit Guarantee. I understand that this instruction may remain with Select Publisher Services Ltd and, if so, details will be passed electronically to my bank/building society.

Signature .... Date ☐☐/☐☐/☐☐

Banks and building societies may not accept Direct Debit instructions for some types of account.

SUBSCRIPTION PRICING WHEN PAYING BY CHEQUE OR CREDIT/DEBIT CARD

6 ISSUES	☐ UK £30	☐ Europe £45	☐ Rest of world £50
12 ISSUES	☐ UK £55	☐ Europe £80	☐ Rest of world £90

2 CHEQUE

I enclose a cheque for .... (made payable to Select Publisher Services Ltd)

3 CREDIT/DEBIT CARD ☐ Visa ☐ MasterCard ☐ Maestro ☐ Switch

Card number ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Expiry date ☐☐ ☐☐ Valid from ☐☐ ☐☐ (if shown)

Issue number ☐☐ (if shown) Security number ☐☐☐
(last 3 digits on the back of the card)

Signature .... Date ☐☐/☐☐/☐☐

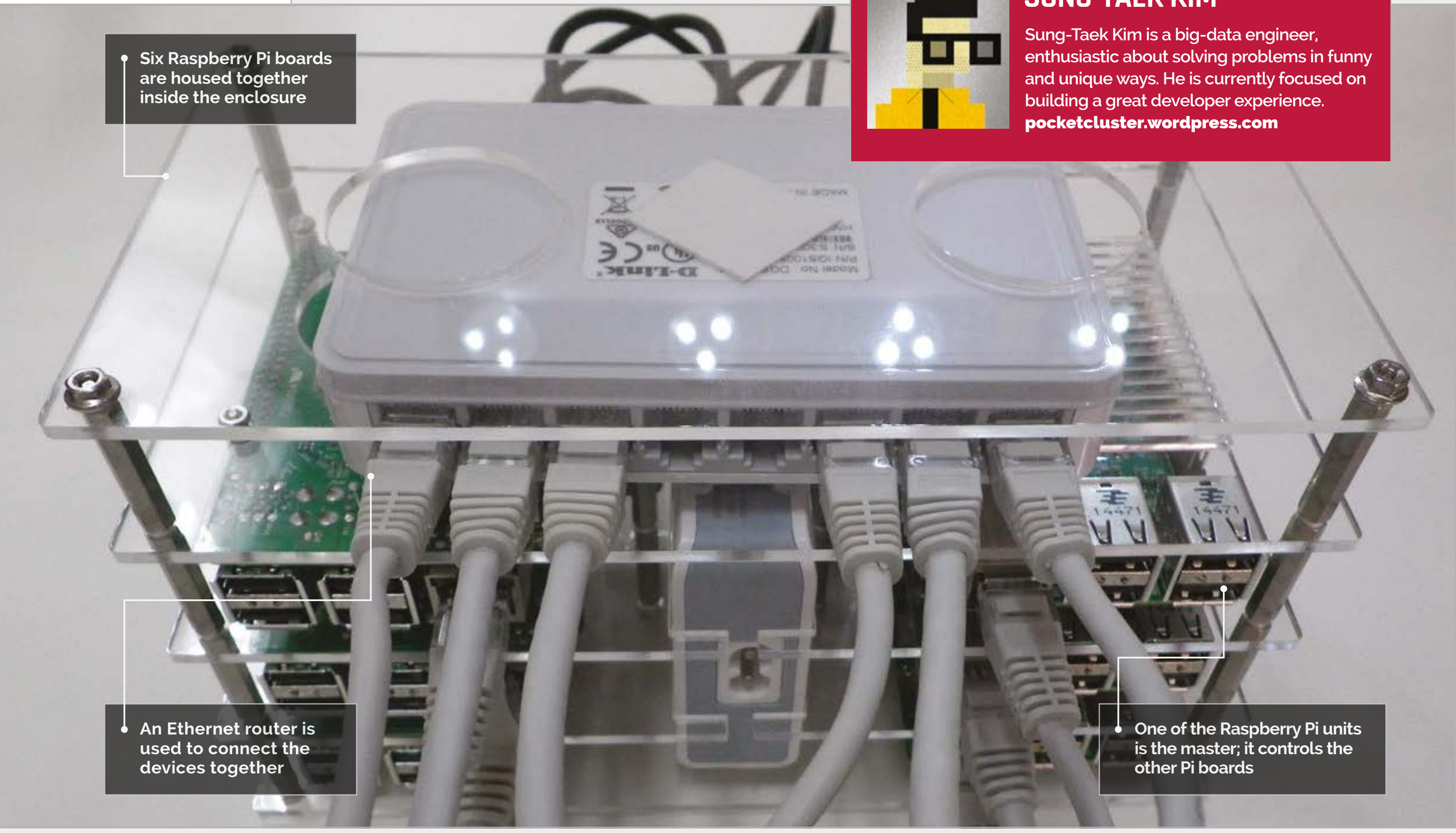
I would like my subscription to begin from issue .... (month + year)

RETURN THIS FORM TO:
MagPi Magazine Subscriptions, Select Publisher Services Ltd, PO Box 6337, Bournemouth BH1 9EH

☐ Please tick this box if you DO NOT want to receive any other information from Select Publisher Services Ltd.

☐ Please tick this box if you DO NOT want to receive any other information from other companies.

☐ Please tick this box if you DO NOT want to subscribe to The MagPi newsletter.



Six Raspberry Pi boards are housed together inside the enclosure

An Ethernet router is used to connect the devices together

One of the Raspberry Pi units is the master; it controls the other Pi boards

SUNG-TAEK KIM

Sung-Taek Kim is a big-data engineer, enthusiastic about solving problems in funny and unique ways. He is currently focused on building a great developer experience.

pocketcluster.wordpress.com

Quick Facts

- Pis aside, the parts only cost around $60
- 106 screws are needed for the enclosure
- A cluster of Raspberry Pis is also known as a 'Bramble'
- GHCQ has the largest Pi Bramble: 66 boards
- China's Tianhe-2 is the biggest supercomputer: 3,120,000 cores

PI SPARK BIG DATA CLUSTER

One data scientist shows us how to string six Raspberry Pis together to build a supercomputer and experiment with big data

The Raspberry Pi is great for learning computer science, but there's one area that's big news but requires big computers, and that's 'big data'.

Big data software typically runs on clusters of networked computers, working together to perform the heavy lifting required. This clustered nature makes learning big data tricky, because you need several computers wired together to practise. Sung-Taek Kim, a software engineer from Korea, decided that the Raspberry Pi would be perfectly suited to the task. "Raspberry Pi is a great education platform to learn how big data software works," he tells us. "It is [comparatively] slow and low-powered, [so] that you would have hands-on experiences when your data manipulation methods execute as planned."

In fact, the light performance of the Raspberry Pi becomes an advantage when learning big data techniques. "Once you miss a small detail," explains Sung-Taek, "you feel the operation processes slow down."

"Sending data across [a] network takes time," he adds. "All the CPUs in your cluster compete for resources such as memory or disk [space], and a node or two could suddenly refuse to work, just like [in] a Google-class data centre cluster." He explains that the relative slowness of a Pi cluster is actually an advantage, enabling you to prepare for such events.

Sung-Taek's cluster is based around six Raspberry Pi 2 boards wired together with Ethernet cables via a D-Link 8-port Gigabit Desktop Switch.

"Theoretically, you would only need one Raspberry Pi," says Sung-Taek, "since Spark exploits the [nature] of a master-slave scheme. Prepare a Raspberry Pi as a slave and your laptop as a master. Connect two Raspberry Pi devices and you have a Spark cluster."

Sung-Taek suggests using between three to eight Raspberry Pi boards for the project. "Once you have more than ten Raspberry Pis," he says, "it's a headache to find a proper power source, to arrange the network and power cord."

The hardest part seems to be building the enclosure. Sung-Taek hosts schematics on GitHub (**github.com/stkim1/pocketcluster**), but accuracy is vital. Even a half millimetre offset in the cutting template could render one of the acrylic tiers useless, he warns.

as well." Java is another must, and the listed software packages include NumPY, Scipy, and Scikit-Learn. On top of that, you'll be learning the MapReduce programming model, which is where you'll encounter Hadoop and Spark.

They are all skills worth learning, though: big data is at the forefront of computer science and is an interesting area to study and work in. "The Raspberry Pi cluster lets you prepare [for] events that could take place in a production environment," says Sung-Taek. "Further, the cluster lets you easily find where to apply optimisation work, since its hardware resource is limited indeed."

" Once you have more than ten Raspberry Pis, it's a headache to find a proper power source "

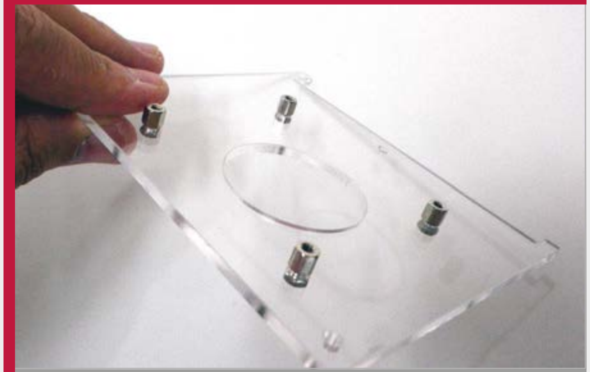
Aside from the Raspberry Pi units, the project isn't expensive. The power supply, network switch, cables, screws, and enclosure only came to around $60. A complete list of materials is available via a *Make:* article (**bit.ly/1J2jpDf**) written by Sung-Taek.

The software requirements are quite intense. "Python is a must," he says, "In order to fully exploit what a Spark cluster provides, it would be a good idea to learn Scala

"You might want to plan carefully and start with a small number of nodes," he advises. "Pay attention to details in each step, and make sure your plan is checked with specifications. Once you've successfully built [a cluster], make a bigger one. One of the best strategies to avoid costly mistakes is to follow and observe what others have done."

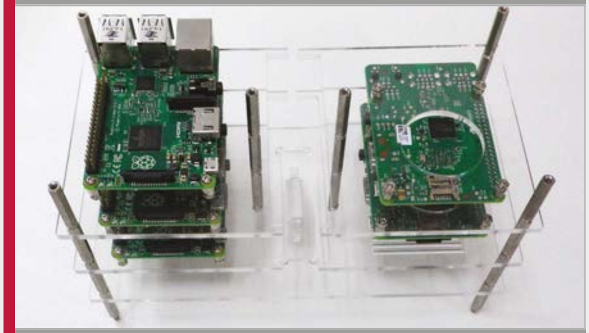
Below Six Raspberry Pi devices are used to build the supercomputer

BUILDING A PI SPARK CLUSTER



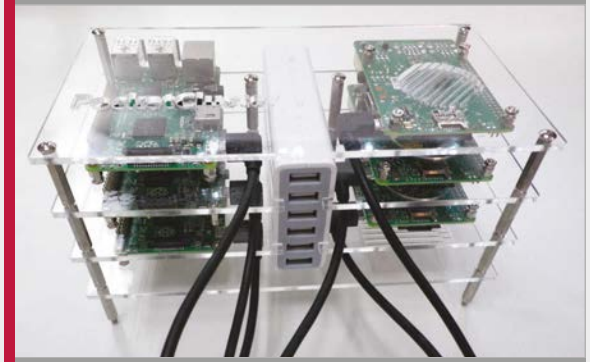
>STEP-01
Building the case

The schematics for the boards can be found on Sung-Taek's website. You can use wood, synthetic, or acrylic panel to cut out the shapes, so long as each is no thicker than 3mm and strong enough to hold a USB charger.



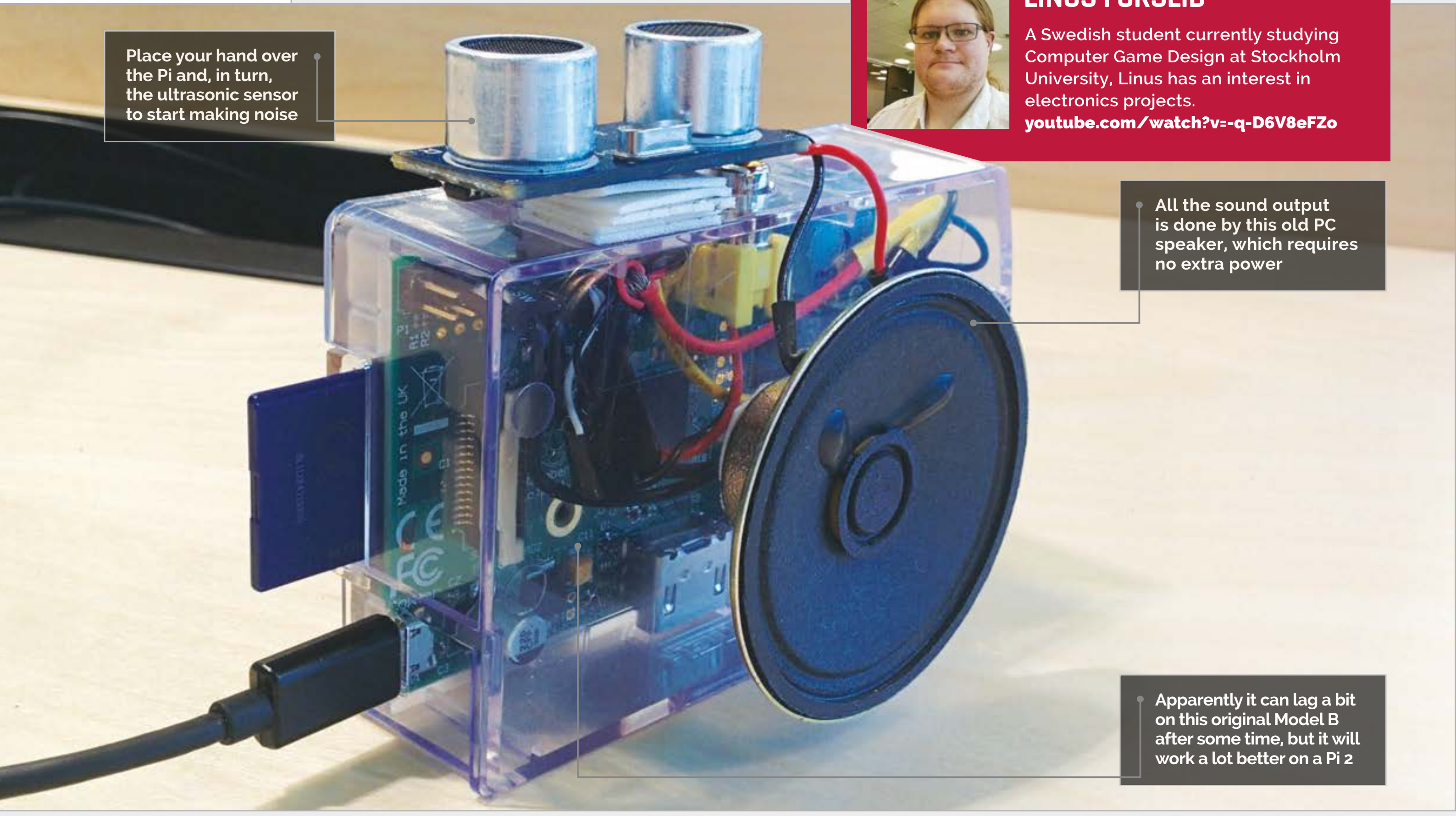
>STEP-02
Cluster assembly

The Raspberry Pis are attached to the boards using M2.5 screws. Then use 5mm pillar screws and hex nuts to screw the boards together.



>STEP-03
Hooking it up

The six-port router is held in the middle of the boards. Each Raspberry Pi connects to the router via Ethernet. Notice that the case leaves enough space for a power socket for each Raspberry Pi.


Place your hand over the Pi and, in turn, the ultrasonic sensor to start making noise
All the sound output is done by this old PC speaker, which requires no extra power
Apparently it can lag a bit on this original Model B after some time, but it will work a lot better on a Pi 2

LINUS FORSLID
A Swedish student currently studying Computer Game Design at Stockholm University, Linus has an interest in electronics projects.
youtube.com/watch?v=-q-D6V8eFZo

ULTRASONIC THEREMIN

An inventive use of the Raspberry Pi that makes use of the kind of components you might have lying around the workshop

Quick Facts

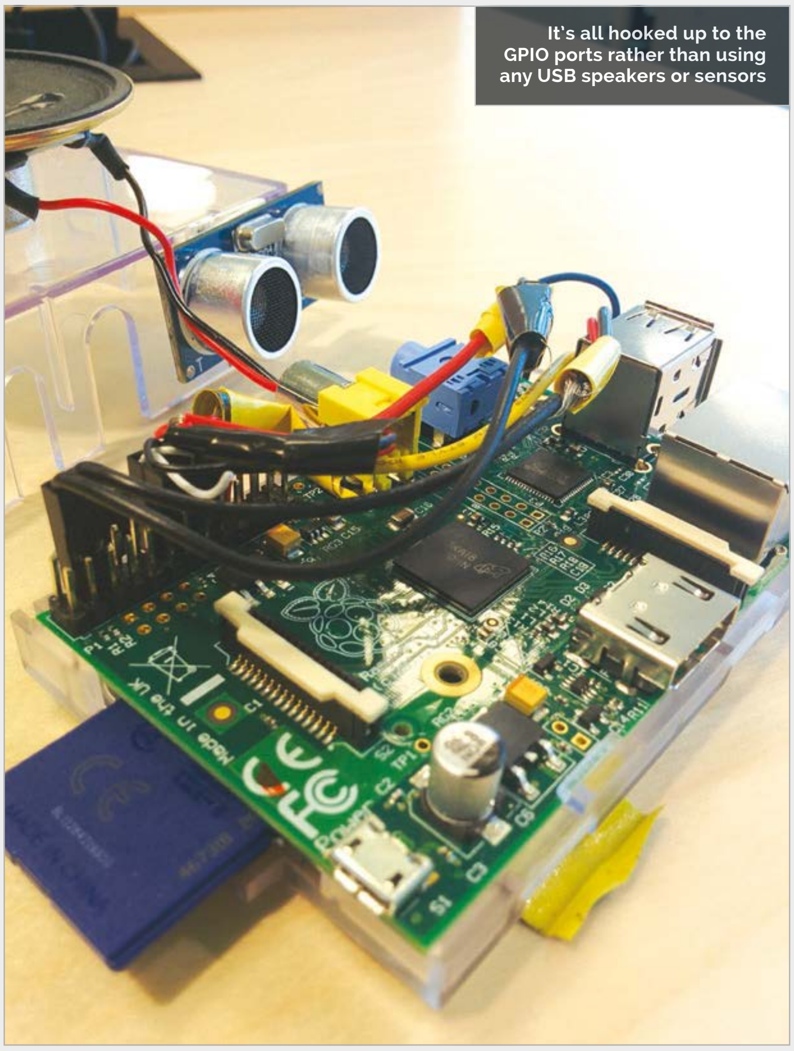
- Linus had been using his Raspberry Pi as a HTPC
- This was his first electronics project of this type
- It only took him an evening to make and two hours to fine-tune
- It's held together with electrical tape, which is not recommended
- Linus hopes to add light guides for better accuracy

Thinking outside the box is what we like here at *The MagPi*. The Raspberry Pi is so versatile and so open that you can do so much with it that perhaps other people wouldn't have dreamt of before. So here's a good question: with a Raspberry Pi, some speakers, and an ultrasonic distance sensor used for robotics, what would you make? Linus Forslid decided to build a type of theremin.

"A theremin is a device that measures distances to objects nearby and turns that into sound, so that when you move your hands around it, you can generate music," Linus explains. It's not exactly like a normal theremin, though. "Unlike a classic theremin, my project uses an ultrasound sensor rather than measuring electrical fields with antennas, and currently my project only has a single sensor, so you can only control the tone it produces. Eventually, I plan to add another so that the beat can also be manually controlled."

Theremins were used to create the eerie, creepy music for classic sci-fi films. Due to this and its unusual method of operation, the theremin has become intriguing to many musicians and techies, including Linus:

"I've always wanted a theremin, from the first day I heard of it. It's just such a cool thing. As it turns out, though, an actual theremin is not cheap or easy to build, but this project was easily doable and I had almost all of the components at home already, so the cost was minimal as well. I decided to use a PC speaker instead of the Pi's own audio jack and an external speaker, mostly because I could, in addition to it not requiring an additional power source."


It's all hooked up to the GPIO ports rather than using any USB speakers or sensors

> I've always wanted a theremin, from the first day I heard of it. It's just such a cool thing

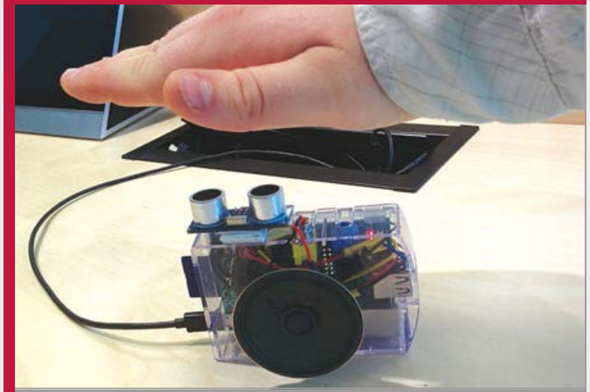
The hardware is one thing, but actually translating it to sound, surely that takes a bit more work on the code to get it operational?

"Not really, I think." Linus adapted some pre-existing code for the project. "The script that controls the theremin is coded in C using wiringPi, and it took maybe an hour to get it working. I've spent a couple of hours beyond that just tweaking the numbers to try out different sounds and find one I like. I wouldn't say it was complex, even if the tuning took a while."

The whole process is deceptively simple then, even though it's hardly your typical Pi-based project. There are a few caveats, however, and Linus's Pi-powered theremin could do with a few improvements, as he explains:

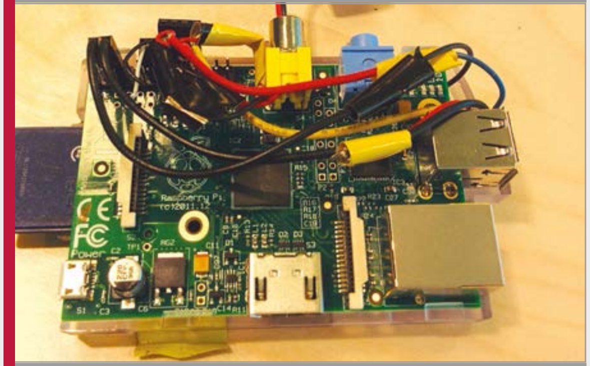
"It's surprisingly hard to operate well, as you need a very steady hand and good aim to get the exact tone you are after… I plan to add another sensor to control the beat of the sound. Right now, it is set entirely by the script, which means it's basically static. By allowing the user to control the beat, my hope is that it will actually be useful as a musical instrument."

CREATE MUSIC



>STEP-01
Wave your hand

The ultrasonic sensor measures the distance to the closest surface in its range quite precisely. Your hand acts as this surface, and moving it up and down creates a difference in measured distance.



>STEP-02
Translate the data

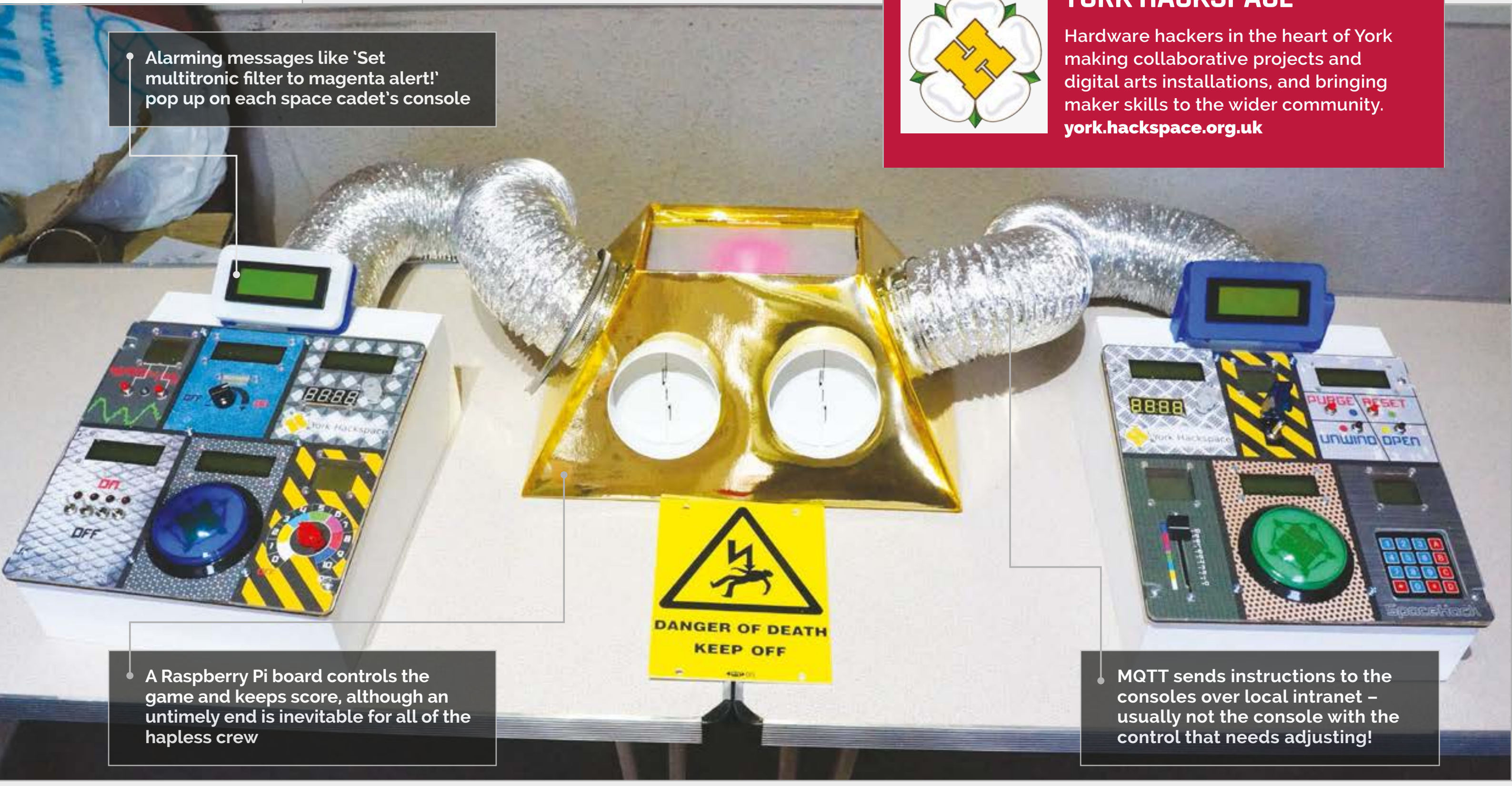
The distance measured is translated to a number, which is supplied to the script. The range is between zero and 120 centimetres, so there's a wide range of sounds the system can make.

>STEP-03
Make a sound

The value is converted into an output signal via the script, which is then played through the speaker. It changes in almost real-time, so moving your hand around changes the tune dramatically.

YORK HACKSPACE

Hardware hackers in the heart of York making collaborative projects and digital arts installations, and bringing maker skills to the wider community.
york.hackspace.org.uk



Alarming messages like 'Set multitronic filter to magenta alert!' pop up on each space cadet's console

A Raspberry Pi board controls the game and keeps score, although an untimely end is inevitable for all of the hapless crew

MQTT sends instructions to the consoles over local intranet – usually not the console with the control that needs adjusting!

Quick Facts

- Console One still features its original spaghetti wiring
- SpaceHack has featured at around 15 maker events
- For Towel Day it became the Infinite Improbability Drive
- The laser cutter used lives in Rob Whitfield's bedroom!
- SpaceHack uses the same communications protocol as Facebook Messenger

SPACEHACK

Join the unfortunate trainee crew on the Starship Guppy, as things begin to go badly wrong, and chaotic fun ensues

What other fast-paced, Pi-based console game lets you shout "Set multitronic filter to magenta alert!" to the general puzzlement of passers-by? York Hackspace have been drawing a lot of attention at maker events over the past year or so, fulfilling their aim of creating something memorable for visitors to their stand.

It was Bob Stone who suggested a physical version of Henry Smith's *Spaceteam*, a 'cooperative shouting game for phones and tablets', but given the definitive Hackspace twist with its retro-futuristic spaceship looks, and homebrew hardware and software, "it would sort of advertise itself within the room, as yelling nonsense with a sense of urgency and panic tends to draw attention in crowds."

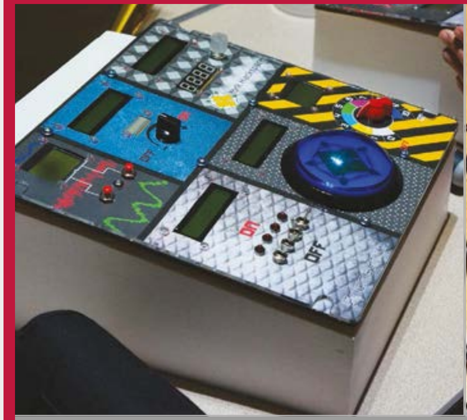
Right SpaceHack's retro-futuristic look seemed a natural fit in the MakeFest setting of Manchester's Museum of Science and Industry

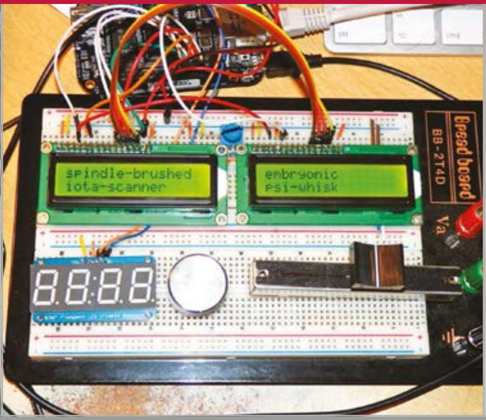
Disaster simulator

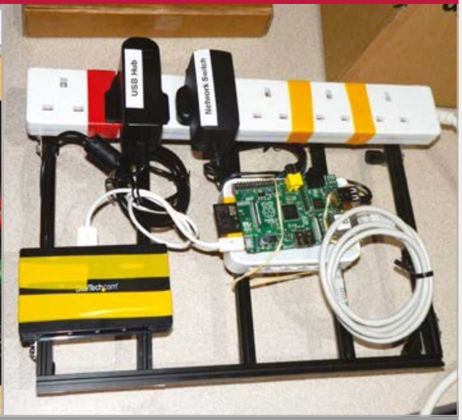
"Welcome aboard the USS Guppy, recently refurbished to the very highest standards of modern space-worthiness by some new lowest-bidding contractors we found on the net. We're proud that this venerable old boat, a veteran of many a heroic space battle, has once again been declared officially 'Good enough for Government work.' "

The hints are all there before you start your "routine pass out by an

MAKING OF THE CONSOLE GAME







>STEP-01

The console case

Heavy-duty woodwork supports a laser-cut ply top, retro-futuristic control graphics designed in Inkscape, and a fairly thick acrylic top to withstand some heavy use by baffled space cadets.

>STEP-02

Inside the console

The BeagleBone Black is a fairly dumb console, picking up its instructions from the controller, displaying instructions sent over MQTT on the LCD, and reporting control values.

>STEP-03

Pi controller

A single B+ Pi board powers the whole game, generating text, as well as sound effects, and hosting the game's MQTT broker. A network switch connects the Pi to the consoles, via spacey-looking ducting.

asteroid belt orbiting an unstable Red Giant star near the edge of the Forbidden Zone, to investigate some unusual radiation signatures."

SpaceHack's control panels have been reconfigured, and as things begin to go wrong on the space mission, emergency instructions issued by the ship's computer – showing on the console's LCD – don't seem to apply to that console's switches, dials, and buttons. The only way to avert disaster is to shout out the instructions so that fellow space cadets at one of the other three consoles can search for the right switch to flip, dial to turn, or button to push.

> "Welcome aboard the recently refurbished USS Guppy, once again declared officially 'Good enough for Government work'"

In a Jam

How long can disaster be staved off? There's no shortage of volunteers to find out, whenever the York team brings SpaceHack to an event. If you follow the maker events online, you'll have seen the favourable comments. Spotting a couple of familiar faces playing SpaceHack at the MOSI MakeFest, we asked them what they'd thought of the experience.

"I thought playing it was great fun and highly engaging. The intentionally confusing instructions add to the madcap antics and to me were reminiscent of TV's *The Generation Game* challenges, where contestants struggled with seemingly simple activities," Raspberry Jam's Alan O'Donohoe told us. Claire Garside invited the York team to Leeds Raspberry Jam, and was enthusiastic about SpaceHack's ability to excite people about STEM in a fun way: "Whether taking the challenge myself, or observing the wide range of contestants it entices, SpaceHack always initiates an enthusiasm and excitement for everyone through gaming with Pi. And I do mean everyone!"

If you'd like the chance to shout "Plug in the centrifugal F-screen!", then look out for SpaceHack at Derby Maker Faire on 26 October – and if you want to build your own, full details are up at **github.com/yorkhackspace** for the hardware (including laser cutting and 3D printing), console software, and Raspberry Pi controller software. "Increase the omnicrontensor shell! Set multitronic filter to magenta alert!"

Below Families are naturally drawn to what's best described as 'a game of collaborative shouting'

MARIO LUKAS

Mario Lukas is a hardware hacker, computer scientist, and technology tinkerer from Aachen, Germany.
mariolukas.de

FABSCAN PI

Mario Lukas builds 3D scanners from the Raspberry Pi and off-the-shelf components. He tells **Lucy Hattersley** how it's done

Quick Facts

- All the parts cost less than $100 in total
- Commercial scanners cost around $3,000
- The FabScan software creates PLY and STL files
- The mesh files can be used to 3D-print objects
- Mario is experimenting with dual and green lasers

Few things in modern life are more fun than 3D-scanning an object, then using a 3D printer to create a copy. But 3D scanners are still very expensive to buy. FabScan is an open-source, DIY 3D laser scanner being developed by René Bohne and Mario Lukas.

Initially, FabScan was connected to an external computer, but Mario and René have recently created a Raspberry Pi version that houses all the components required inside a single box. This Raspberry Pi-powered edition means you can build a complete 3D scanner for under $100 (£65).

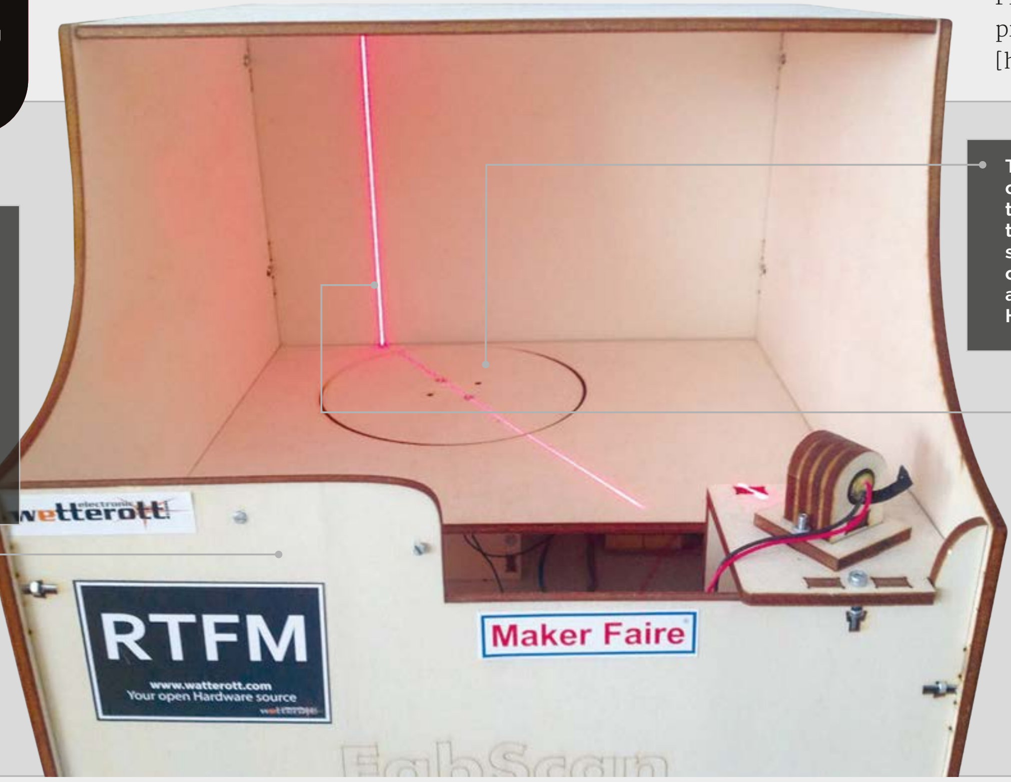
The FabScan project has a long history. It started out as a Bachelor's thesis by Francis Engelman at RWTH University in Aachen (Germany) in 2011. In 2014, development was taken over by Mario and René.

"The FabScan without the Raspberry Pi is an open-source, do-it-yourself 3D laser scanner," says Mario. "After a few months I realised that people had problems [getting] the FabScan software working on all the different operating systems and hardware setups.

"The FabScan uses a laser cut plywood case," continues Mario. "There is a stepper motor connected to a turntable in the middle." Inside the case is a 5mW red laser and Logitech Lc270 webcam. The laser and a webcam are mounted in the front of the case. An Arduino with a FabScan shield controls the stepper motor and laser.

You place an object on the turntable so the laser runs across its surface. As the turntable rotates, the webcam monitors the movement of the laser and creates a point cloud of data. The data is turned into a 3D mesh, and this file can be exported and used in 3D software.

"Since I introduced the Raspberry Pi to the FabScan project, the webcam [has been] replaced by

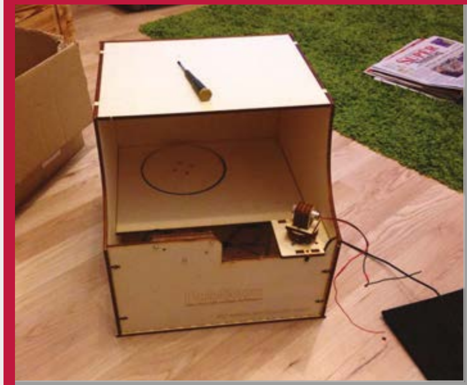


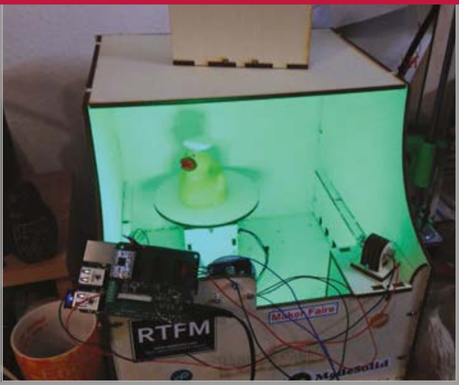
The Raspberry Pi is housed inside the box and runs a web interface. You connect to the latter remotely to begin the scanning process

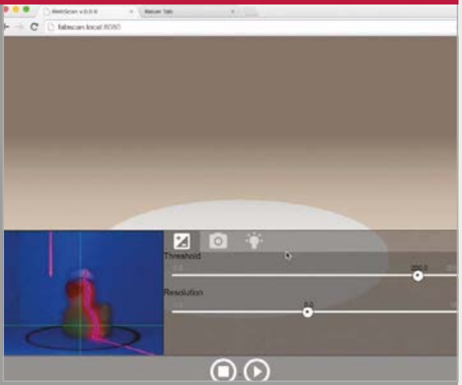
The wooden case contains a motorized turntable that rotates the object being scanned. It is currently controlled by an Arduino, although a custom Pi HAT is being developed

A 5mW red laser is fixed in the corner of the FabScan. It projects a line directly across the turntable. A Pi Camera Module is used to monitor the laser as the object is rotated

HOW DOES IT WORK?







>STEP-01
Box construction

The FabScan box is laser-cut from wooden parts. A stepper motor, controlled by Arduino, is used to control the wooden turntable. The laser is mounted in the corner and emits a line diagonally across the turntable.

>STEP-02
The Raspberry Pi

The Raspberry Pi runs the scanning software, and a Raspberry Pi Camera Module is used instead of a webcam. Using a Pi enables the FabScan to become a self-contained 3D scanner.

>STEP-03
Scanning the duck

An object (in this case a rubber duck) is placed on the turntable. The web software is used to connect to the Pi-powered FabScan. As the duck rotates, the camera monitors the laser and uses it to create the scan.

the Pi Camera Module," Mario tells us. "We are developing a FabScan Pi HAT to replace the Arduino and the FabScan Shield as well."

You can build a FabScan case yourself using sheets of wood, and a complete bill of materials can be found at **hci.rwth-aachen.de/fabscan_hardware**. Alternatively, a FabScan Cube Kit v2 + Electronics bundle can also be ordered from German-based distributor Watterott (**watterott.com/de/FabScan-Cube-Elektronik**) for €105 (£77).

"Anybody who wants to build a FabScan Pi should order the parts and build it," insists Mario. "We are trying to keep the things as simple as possible, and introducing the Raspberry Pi to the FabScan project was a good approach to improve the usability.

"Most of the work was to write new software for the FabScan," continues Mario. "The software provides a JavaScript-based web-enabled user interface for scanning. The user can call the local FabScan in a browser and the user interface will show up.

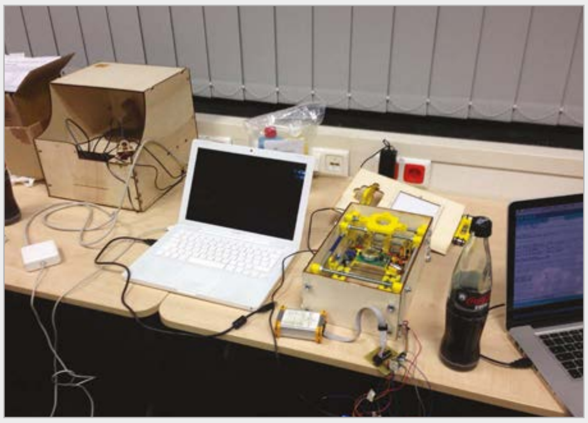
"During the software developing process, I scanned a lot of different rubber ducks, other plastic animals, figures made of gypsum (plaster of Paris), and stuff lying around in my workroom.

> "Introducing the Raspberry Pi to the FabScan project was a good approach to improve the usability"

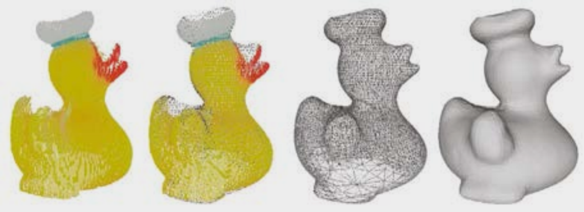
"After a scan, you will get a one-to-one sized three-dimensional model of the scanned object," explains Mario. "The current FabScan Pi software can export the model in [the] STL file format that is mostly used by 3D printers."

Pushing forward with the FabScan project is a challenge, but Mario and René remain unperturbed. "At the moment, the FabScan project is spread all over the web," explains Mario. "In the upcoming weeks, I will try to create a website where all corresponding threads run together.

"Currently I am working on some DEB packages for Raspbian to keep the installation process as simple as possible," reveals Mario. "The next step is releasing a Raspbian image with a pre-installed version of the FabScan Pi software. I also have a huge to-do list with new features mentioned by the FabScan community."



Above The Raspberry Pi 2 has enough power for a FabScan to work without being attached to another computer



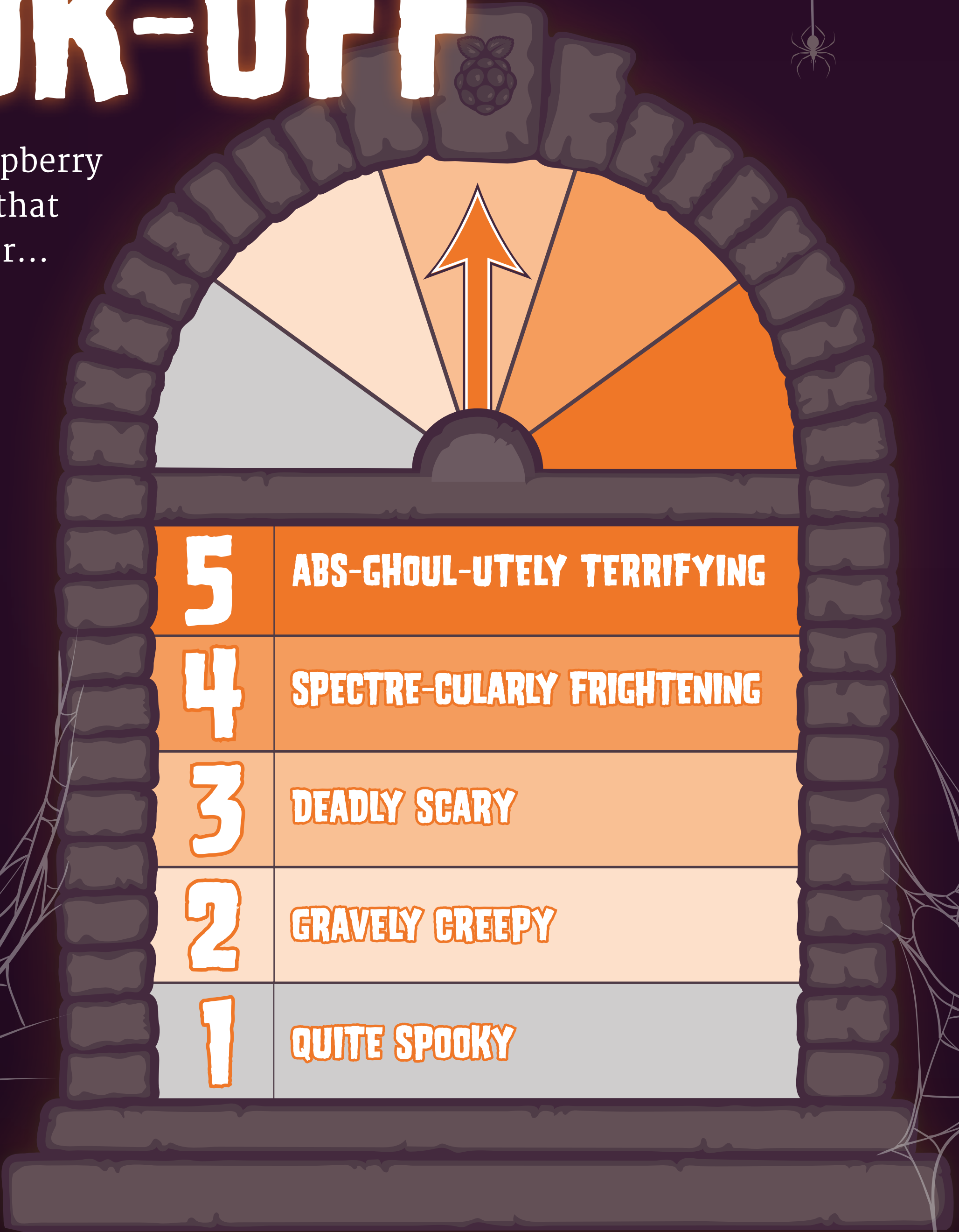
Above The FabScan software is used to create a 3D mesh from the point cloud data detected from the scan

THE GREAT RASPBERRY PI SPOOK-OFF

Get a look at these Raspberry Pi Halloween projects that are absolutely to die for…

Gather round, ghouls and girls, it's that time of the year again when scary rules supreme. Perhaps you're planning on dressing up to go trick-or-treating or marathoning a ghastly amount of horror films, but we've found some people who are deep in their lairs experimenting with a Raspberry Pi to create the spookiest projects the world has ever seen.

We've hunted down the most horrifying and wicked projects for your reading pleasure, but don't worry, the only dark art at work here is the odd bit of C programming. Beware, read any further and you'll be doomed to be inspired by these seven unholy projects… and have to read many more awful puns.



RASPBERRY PI HAUNTED HOUSE

Enter if you dare to the abode with home scare-tomation

STEWART WATKISS

Data centre manager, father, part-time Count Dracula
penguintutor.com/electronics/halloween

Above Surely it can't really be haunted… can it?

It's late. The night grows dark and you're near the end of your trick-or-treat run – but what's this? A house you've never seen before on your road. Eager for more sweets, you make your way to the door. A haunted house sign greets you, but you think it merely decoration. Approaching the door, you press the doorbell - only for glass to break and the light to go out. A door creaks, the sign you had dismissed flashes, and you hear screams as the lights come back on. Startled, you look to your right and realise monsters are partying in the garage next to you, celebrating another victim in their night of ghastly fun.

"I had been trying to think of something fun to do for Halloween and the Raspberry Pi was an obvious choice," the owner of this nightmarish house, Stewart, tells us. "I'd recently built a circuit for home automation using remote-control sockets and had the idea of using that to turn lights on and off automatically. I also spent some time looking around at shops to see what other Halloween props I could add to the project."

The system is deceptively simple, although there's a lot of different components to it. A dedicated doorbell is hooked up to a PiFace board that interacts directly with the Raspberry Pi and some Python code. Stewart chose the PiFace for this, instead of wiring up directly to the Raspberry Pi's GPIO pins, to make sure he didn't send any unwanted signals through the GPIO port – although with the right amount of research and careful wiring on a breadboard, you could do this without the PiFace. The PiFace directly controls the LEDs on a jack-o'-lantern and the haunted house sign, while the porch lights and the monster party lights are controlled by a wireless home-automation remote control that he directly soldered into.

"They are not designed to be used that way," Stewart mentions. "The solder joints were not very reliable and whilst they lasted for Halloween, they eventually came loose. Since I made the project, Energenie have released a Pi-mote Raspberry Pi board for their remote-control sockets, which is easier to use than having to solder onto the remote-control buttons."

Speakers play the appropriate sounds and music as dictated by the code, and the system is waterproofed as, well... Stewart lives in England. The results?

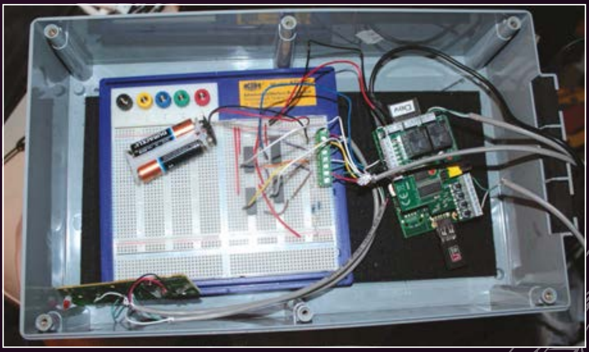
"I just managed to finish it in time for Halloween. We had friends coming over for our children's Halloween party who said they liked it, and the trick-or-treaters were certainly surprised by it. There was one young girl who was a little scared by it, but most thought it was fun rather than scary."

COMPONENTS LIST

CONJURE UP YOUR OWN VERSION

- PiFace
- Wi-Fi dongle (used during configuration and testing)
- 2× Reed relays (HE751A0510) – one for the on button, the other for off
- 2× LEDs
- 2× AA batteries with holder
- Doorbell
- Remote control socket with spare remote
- Plastic light-up pumpkin lantern (with test button)
- Plastic haunted house sign
- Waterproof box (or lots of wire to allow the Raspberry Pi to be installed indoors)
- Speakers (with amplifier)

Below A simple box contains the Pi and all the electronics so that it's kept waterproof

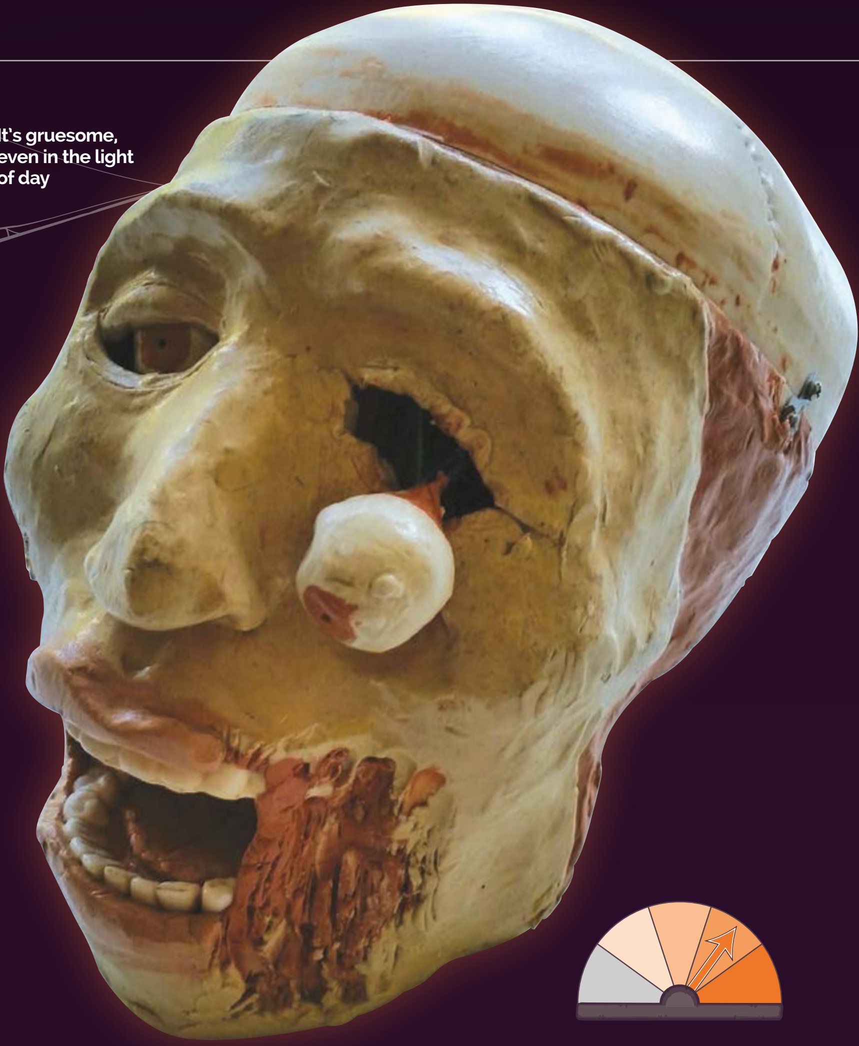


SOCRATIVE ZOMBIE

A philosophical zombie, or a frightening, brain-munching head?

DAN ALDRED

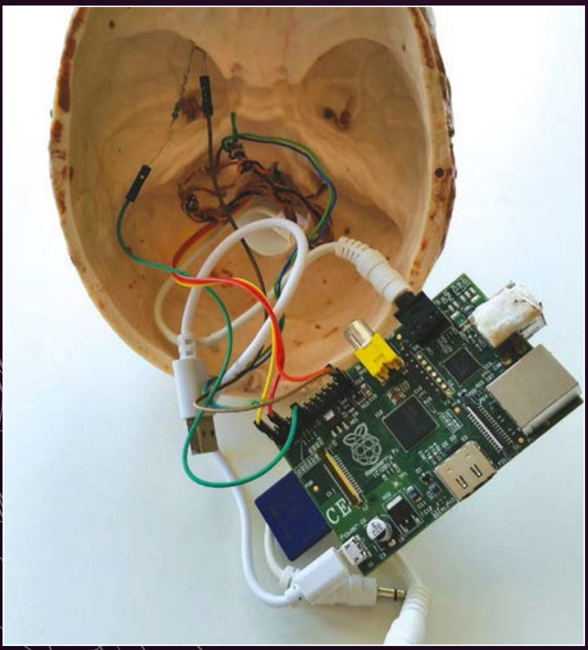
Lead schoolteacher for Computing At School, raiser of the dead
tecoed.co.uk/socrative-zombie.html



It's gruesome, even in the light of day

Out late at night, you should have known better than to enter the strange shed, but it was about to rain and you needed the shelter. A sense of unease comes over you as the door closes behind you. It's probably just because it's a bit of a creepy atmosphere. The winds begin to howl outside, but wait – that's not a howl, it's more of a gurgling groan. And it's coming from inside the shed with you. As you turn to confront the noise, you see an eerie glow from the corner of your eye. You fumble for your phone and turn the torchlight on to see the source of the noise and light… only to find an animated, undead head hungry for your flesh.

Below The inside of the zombie is a less creepy affair, as you can see the parts that make it work



Luckily, it's just a sculpture from a school project, but you get the picture. Its creator Dan Aldred tells us how he came up with such a horrifying idea:

"I wanted to create a Halloween hack that would scare people. Sheds are scary in the dark, but even more with a talking zombie head! I also wanted to create a interesting head that would ask my students questions related to their learning."

We assume they learnt the true meaning of fear alongside their new-found coding and soldering skills. The head is one of the few projects in this feature that makes use of sensors to know when someone is approaching it. In this case, a PIR motion sensor detects changes in temperature that correspond to a warm body entering its field of view. This means it unfortunately would not be able to sense its fellow undead. The sensor is placed in the mouth and controls a Python script which also hooks into LEDs and a speaker to complete the effect.

If you plan to replicate this project, Dan recommends a Raspberry Pi A+ due to its small size, and to create a cron job to start the program once the Raspberry Pi is powered up. How did it fare on the night?

"I had it in the classroom, it got dark about 4pm, and the students were very intrigued and excited," Dan tells us. "Then I moved it to my shed for the 31st and when the trick-or-treaters came around, they went in the shed and were scared. It made them jump; they really liked the zombie head."

COMPONENTS LIST

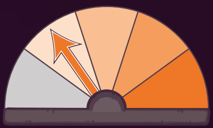
CONJURE UP YOUR OWN VERSION

- Zombie head
- PIR infrared motion sensor
- LEDs
- Speakers

PUMPKIN PI

Usually a delicious dish, can you stomach this version?

DREW FUSTINI
Software developer for element14 community, gourd sorcerer
bit.ly/1ilW2te



Carved pumpkins and jack-o'-lanterns are a big Halloween tradition in many countries. Thanks to the internet, there's been an explosion in people wanting to make their own custom pumpkins to show off, which are a bit more than just a pair of eyes and spooky grin. These days, you can print off pumpkin carving patterns for just about anything, and it's a great way to do something a little creative with some flesh left over to maybe make a delicious soup or dessert.

Carving a pumpkin isn't the only way to express yourself, however. To truly create a 'hack-o'-lantern', we need to add some electronics to its innards and dial up the spook-factor. That's exactly what Drew did with his Pumpkin Pi for the element14 community. Some of the projects he creates are inspired by upcoming holidays, and in this case Halloween was approaching.

The whole system is fairly simple: by activating a preset script from a web interface on his computer, the lights start rhythmically changing colour (check out the YouTube video to properly see it in action: **youtu.be/lIv8H7WPfQw**). The interface is mounted on the Pi and if you use his program for such a project, you can have multiple sounds and light sequences activating within the pumpkin.

The construction of the Pumpkin Pi requires a little more than just pressing a button, though. Taking a pre-carved pumpkin (you could carve one yourself), Drew then lined it with a large plastic food bag to make sure the electronics inside didn't get too damp from the pumpkin's moisture. A carving on the rear was used to pass the power cables through, and the breadboard with all the LEDs and such on was placed on the Raspberry Pi to save space. He also put on chopped-up drinking straws to diffuse the light better throughout the pumpkin. A speaker is attached and the pumpkin is done and ready for testing. If you want to make it a little more permanent, you can make use of a Pi Plate to solder the circuit and then mount it onto the Raspberry Pi.

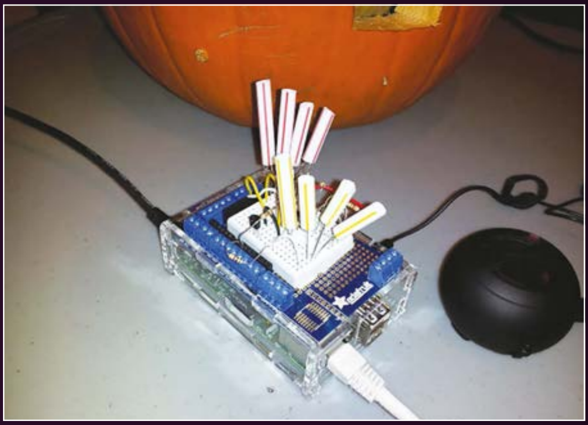
The Pumpkin Pi is simple, but well thought out and put together. If you're going to be inspired by any of these Halloween projects, we'd recommend starting out with this one just to get the feel of creating a creepy contraption for your window or porch.

COMPONENTS LIST

CONJURE UP YOUR OWN VERSION

- A carved pumpkin
- Adafruit Pi Plate
- Adafruit Pi Box
- LEDs
- Resistors and transistors

Above A normal pumpkin, or a lost soul trapped in produce?



Above All you need to hack your pumpkin are some LEDs and some straws, apparently

Above left Significantly less scary with the lights on, nevertheless you can see the motion sensors mounted on crates that make it work

Above right Who's at the door?

SCARY DOOR (AND SCARY PORCH)

Don't get too close to this door if you value your sanity

CABE ATWELL

Electrical engineer, writer, crafter of cursed objects
bit.ly/1p6VQpW

You are entering the vicinity of an area adjacent to a location. The kind of place where there might be a monster, or some kind of weird mirror. These are just examples; it could also be something much better. Prepare to enter... The Scary Door.

Or not in this case, as this is not a real door. Or a *Twilight Zone* parody. While the Haunted House we started off this feature with shows the entrance to a spooky residence, this kind of project is what you might find inside as you wander around. It's a peculiar metal door, with a window you can see through. A knocking sound is coming from behind it; perhaps someone is lost? As you get closer, you see through the window a normal-looking lounge. You can't see anyone, but the knocking is still there and as you reach for the handle, a monstrous face appears.

Cabe's idea was to create a modern haunted house effect on the cheap that was different every time you saw it. As well as the pop-up scares, there's ghosts and monsters flying around in the 'window' (actually an LCD screen), and the scares and sounds you can create are as many as you can imagine. The knocking sound comes from pistons smashing against the door.

As well as the door, Cabe has previously created a Scary Porch, which he describes as "[an] exploration of my idea of consuming someone with light and sound, to the point where they wanted to run away." Unlike our Haunted House project from earlier, which was doorbell-activated, the porch could sense how far away you were from the door. As you started to approach, it would start playing creepy sounds and slowly turn on some bright red lights. As you got closer to the door, the louder the sounds became and the brighter the lights got, until screams and roars began playing when you were in knocking range.

"The Scary Porch was only set up in my shop," Cabe explains to us. "But, people did see it and did what I thought... they wanted to get away from the sound."

Both are quite elaborate projects and if you plan to make anything similar yourself, Cabe advises you exercise extreme caution unless you want to make yourself the centre of your Halloween display. This year he's toning it down a bit: "I plan to... make a wooded path, or [the] front yard a bit more scary."

At Cabe's house, we're sure it'll be more than just a bit scary.

COMPONENTS LIST

CONJURE UP YOUR OWN VERSION

- Too many to list here! Check out the link in the profile to find out what nearly $800 or more can get you in terms of spooky

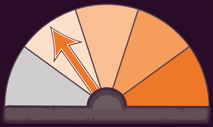
SHOT THROUGH THE STOMACH

Take the scares around with you thanks to this gruesomely interactive costume

LUIS MARTIN

Maker, R&D engineer, major trauma victim

bit.ly/1KNZWot



Scary pumpkins, doors, and houses are one thing, but a good scary costume is a big part of Halloween. While you can't beat the classics of vampires and witches and zombies, more imaginative and horrific costumes are popping up every year. Luis Martin took his costume to a new level by adding a Pi and some clever coding to make it look like he'd been shot right through his stomach, something he'd seen on *The Walking Dead*.

It's actually quite a simple and ingenious design. Using a Raspberry Pi as its main core, Luis took a Pi Camera Module and a 7-inch screen and had a live feed from the camera play onto the screen. By having the Pi and camera on his back, he was able to relay the live feed to the screen that was showing through his shirt, giving the illusion that there was indeed a hole in his stomach. Scary indeed – just don't ask him to turn around to ruin the effect.

It also involves a bit more than electronics, though: a later border was created around the edge of the screen to simulate the gruesome look of flesh to better create the effect through his clothes. With skin-tone paint added along with blood and detailing to get the right effect, he also cut up an old T-shirt and stained it with fake blood. "You have to be eager to stain your hands," Luis tells us. "Especially blood. The more blood in a zombie costume, the more realistic!"

With the shirt ready, Luis took a GoPro harness and had it hold the screen down to his stomach, as well as using it to attach the camera to the correct spot on his back for the illusion to work. This is then simply connected to the screen via the HDMI port on the Raspberry Pi – switch it on and you're ready to go!

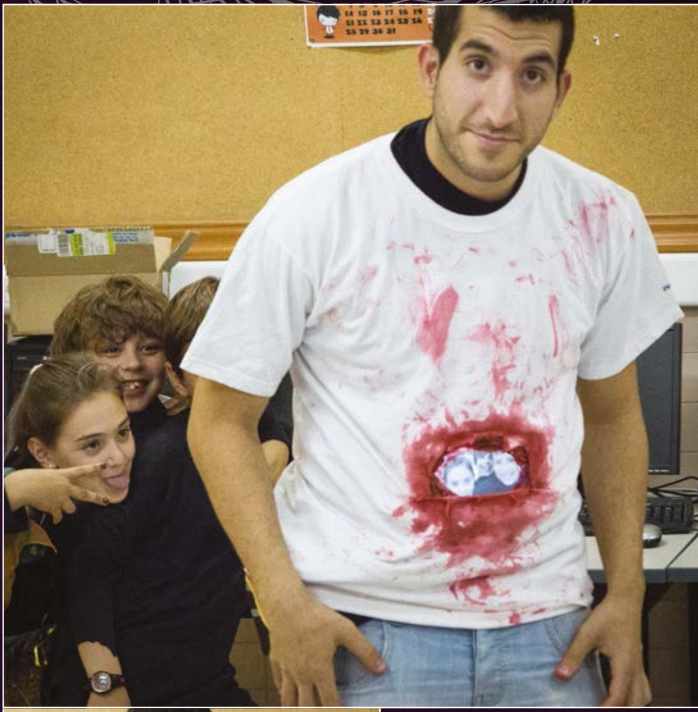
It certainly sounds gruesome, and you'd think seeing a zombie walk around with a hole in his stomach would make everyone scream and run away. "Everyone wanted a picture through the hole, like some kind of blood-coated selfie?"

Kids these days, eh? You put a hole in your stomach and they're not even scared.

COMPONENTS LIST

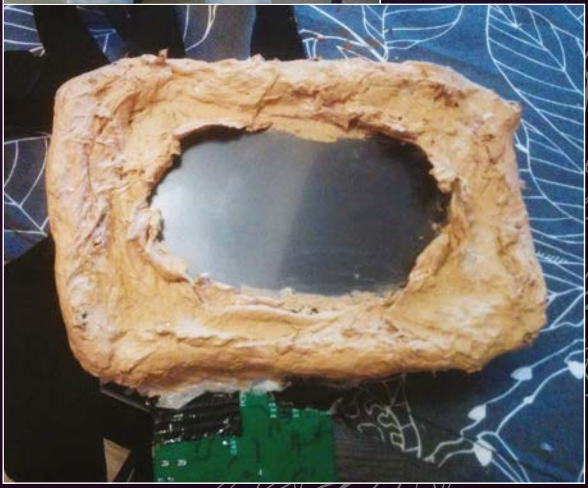
CONJURE UP YOUR OWN VERSION

- Pi Camera Module
- 7-inch touchscreen
- HDMI cable
- GoPro chest mount harness
- Portable battery



Above Is there anything more indicative of modern culture than this image?

Left You don't have to pull a face, but it makes the picture look a bit better



Below The screen's fleshy border adds to the effect

HALLOWEENPI AT 3

A yearly tradition that will rattle your bones and help out the local food bank

CHRIS, PATRICK AND ELISHE

Solution architect, systems specialist, and srtist respectively, purveyors of haunted numbers
on.fb.me/1OuScjv

Scaring people as they come to the door is big business in spooks, it seems. We've seen porches, doors, and monster parties that give at least some warning or try to build a scary atmosphere, but the HalloweenPi is taking the jump-scare approach over building suspense.

To understand HalloweenPi, we need to learn about the history of Halloween at 3. A yearly tradition from the people at 3 Michelle Court, a display of their scariest Halloween-themed ideas is created to wow (and maybe frighten) trick-or-treaters or passers-by. Constantly trying to figure out a way to make each year better by adding to the display, Chris Kyte had a bit of a brainstorm:

"We were considering how to draw in a few more trick-or-treaters to our haunt. We had decided to start collecting canned food donations for our local food bank when I remembered the Raspberry Pi B+ I had received as a birthday present earlier in the year. I started to wonder if I could build a pneumatic prop or two in order to 'up our game'."

Currently, the massive tunnel houses a laboratory of rainbow-coloured mixtures in beakers and test tubes that react to black light, a projected puppet that compliments kids on their costumes, and all the other trappings of a Halloween-themed display. This new addition will have a motion-controller prop attached to it that will spring up and scare people as they arrive for Halloween at 3.

"The HalloweenPi Controlled Pneumatic Prop will be a really spooky piece," says Chris. "It will act as an ambassador to the haunt, reacting to the people who walk by: the prop will spread its wings wide and will scream a mighty scream! The way we've designed the props and pneumatics means that they are modular. They don't have to remain the same each year!"

The frights are for a good cause, though, as Hallowen at 3 takes donations for the local food bank. "The extra excitement generated by the pneumatic props should make a big difference to the number of people that show up. Hopefully, we'll see an increase in food donations as well."

We question how such an endeavour can be truly evil in the Halloween spirit with such a wonderful result, but we'll let it slide this time.

Below top The team are still hard at work on this year's project so they can be ready for the big day

Bottom What exactly happens at 3 Michelle Court?

COMPONENTS LIST

CONJURE UP YOUR OWN VERSION

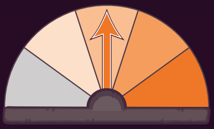
- 4-channel relay board
- 2× PIR sensors
- Wireless key fob to arm and disarm the prop
- A pneumatic cylinder
- 12V pneumatic solenoid
- 12V amplifier

MULDER

Featured in this very magazine, this spooky skull comes alive

MIKE COOK

Writer, bone wizard
vimeo.com/channels/mikespibakery

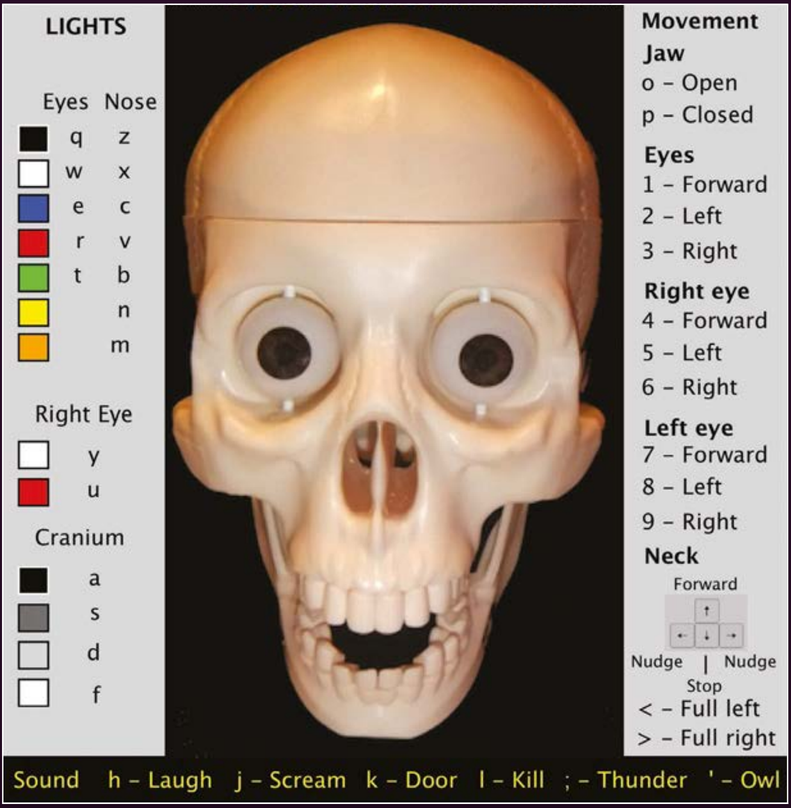


Zombie head? Too much flesh. Meet Mulder the skull, a very interactive head that you can control to scare the living daylights out of anyone who makes the mistake of passing by. With full control over his jaw, neck, and eyes, he can move around to watch anybody. Not only that, but you can light him up in a variety of colours and have him give out deadly screams or creepy owl hoots.

Mulder was made by Mike Cook as part of the Pi Bakery, and featured as a project you can create in this very magazine – head to page 58 to get started.

We won't spoil the surprise, but it involves a lot of soldering and tweaking as you install several servos and motors to articulate the eyes and jaw, along with several lights, a speaker system, and a way to control the neck. It's all controlled from an interface written in Python, the code listing and files from which are available in the tutorial.

We suggest you put a hidden camera near the skull so you can track trick-or-treaters as they approach your door. That's sure to scare them off, leaving more chocolate for you.



LIGHTS

	Eyes	Nose
⬛	q	z
⬜	w	x
🟦	e	c
🟥	r	v
🟩	t	b
🟨		n
🟧		m

Right Eye

⬜	y
🟥	u

Cranium

⬛	a
▨	s
⬜	d
⬜	f

Movement
Jaw
o – Open
p – Closed
Eyes
1 – Forward
2 – Left
3 – Right
Right eye
4 – Forward
5 – Left
6 – Right
Left eye
7 – Forward
8 – Left
9 – Right
Neck
Forward
Nudge | Nudge
Stop
< – Full left
> – Full right

Sound h – Laugh j – Scream k – Door l – Kill ; – Thunder ' – Owl

Left Control Mulder with these simplish keyboard controls

Above In still images, Mulder looks quite tame and funny. In real life, it's a bit more scary

RESOURCES AND INSPIRATION

Not sure what to make with your Raspberry Pi for Halloween? Over the month of October, all the tech news sites will be inundated with Halloween-based stories, so keep an eye on them. Otherwise, you can head over to the Raspberry Pi tag of the Adafruit Learning System (bit.ly/1ir0RHO), where you can get many great ideas for projects, along with the element14 community blogs and the Raspberry Pi Foundation's blog. Be inspired to be spooky!

SHOW US YOUR PROJECTS!

Are you doing something spooky for Halloween? Send us a picture or video to the MagPi Twitter account (@TheMagP1) or email us at magpi@raspberrypi.org and we'll include the best ones in our Community section next month! You should aim to be abs-ghoul-utely terrifying, but even a quite spooky will be good enough.

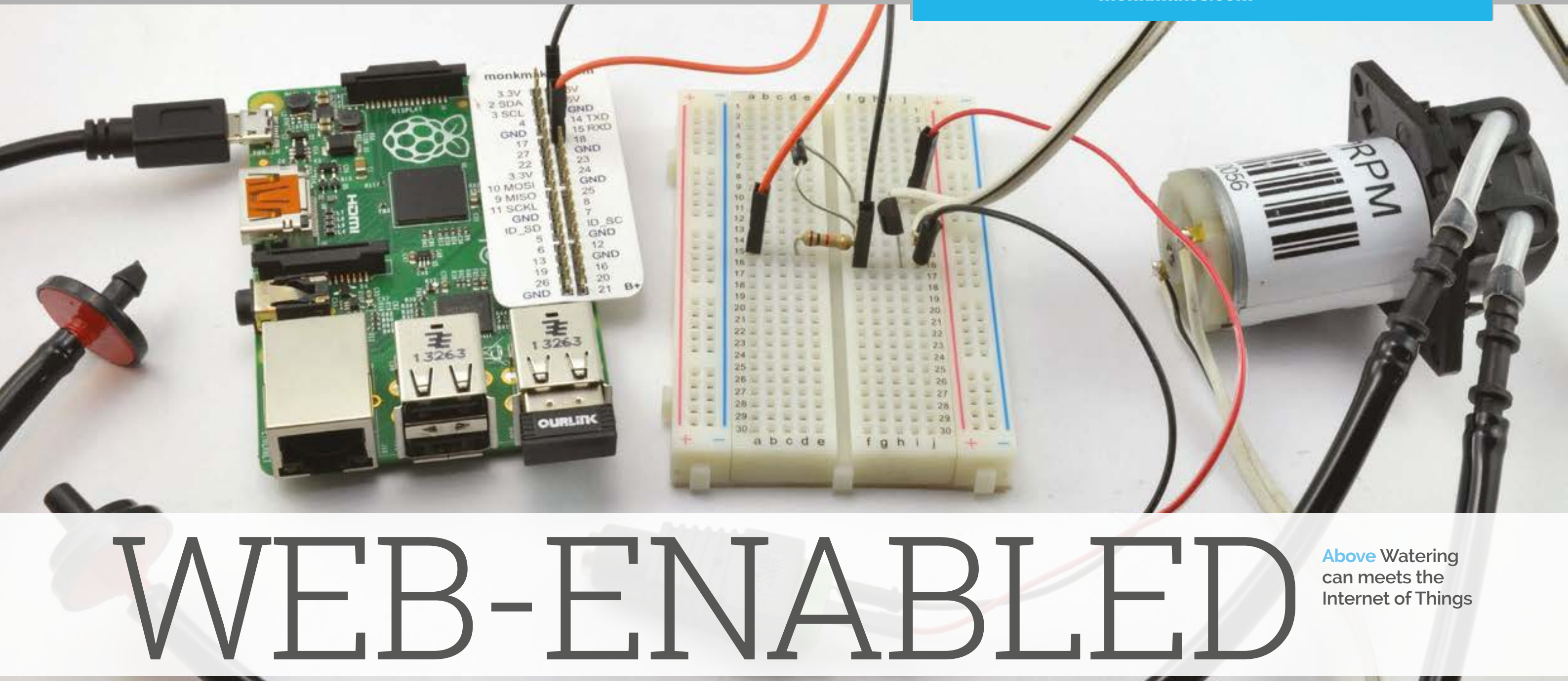
EVERYDAY ENGINEERING PART 8

SIMON MONK

Simon Monk is the author of the *Raspberry Pi Cookbook* and *Programming the Raspberry Pi: Getting Started with Python*, among others.

simonmonk.org

monkmakes.com



WEB-ENABLED PLANT WATERER

Above Watering can meets the Internet of Things

You'll Need

- Half-size breadboard
- 12V peristaltic pump
- 2× male-to-female jumper wires
- 2× male-to-male jumper wires
- 1k ohm resistor
- MPSA14 transistor (CPC: SC10647)
- 1N4001 diode
- DC barrel socket to screw adaptor (CPC: CN19646)
- 12V DC 1A power supply (this just supplies power to the pump)
- Tubing and connectors

Solve real-world electronic and engineering problems with your Raspberry Pi and the help of renowned technology hacker and author, **Simon Monk**

This project automates the watering of your plants. You can use it indoors to take care of house plants while you are away, or with a water butt and outdoor containers.

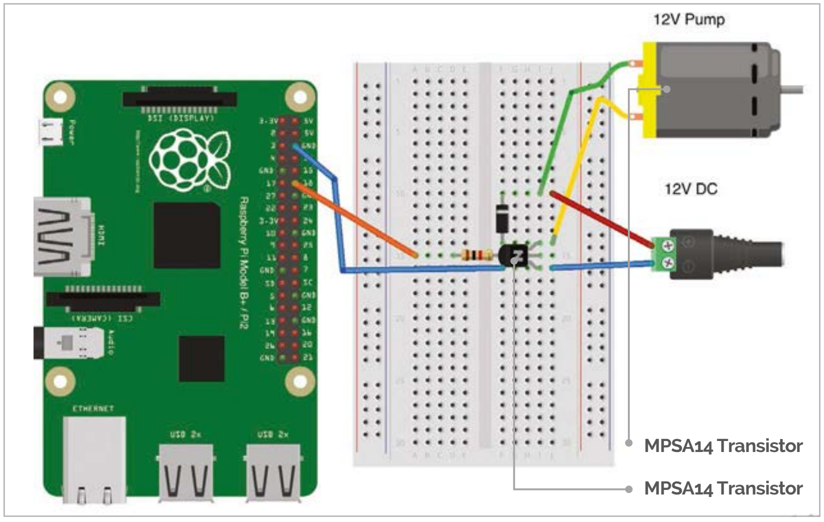
The interface is used to control the duration and start time of your watering, over your local network or even over the internet.

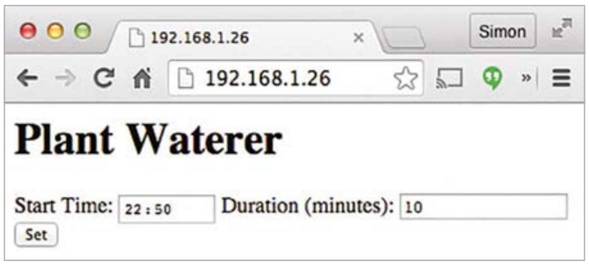
As you'll see from the list of required components on the left, this project uses a small circuit built on breadboard, and a peristaltic pump to take water from your water butt to your containers.

You can find a pump on eBay, where it will cost about £6 from China or twice that from the UK. The pumps are intended for use in an aquarium. They consume about 300mA, a little too much for an old favourite transistor like the 2N3904, so a MPSA14 Darlington transistor, which can cope with 1A, is used instead.

The pumps are usually supplied without leads attached, so you will probably need to solder leads on. If you plan to use the pump outdoors, then solder twin speaker cable or the like to the pump terminals; it must be long enough to allow you to have the pump outside, and your Raspberry Pi and the rest of the electronics indoors.

The pump needs its own 12V DC power supply, which is




192.168.1.26

Simon

Plant Waterer

Start Time: 22:50 Duration (minutes): 10

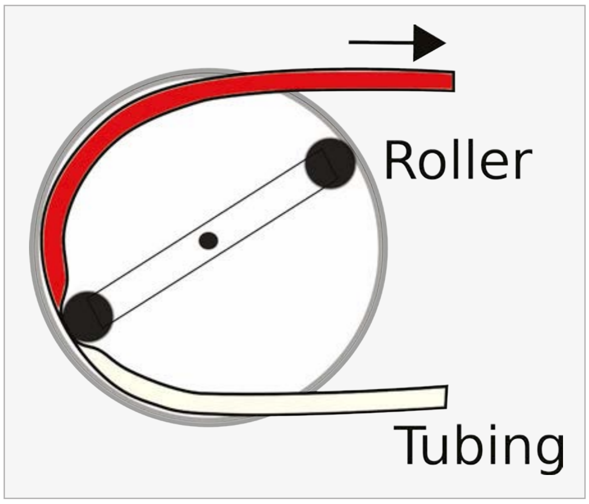
Set

Above Controlling the waterer from a browser

connected to the breadboard using a DC barrel socket to screw adaptor and two male-to-male jumper wires.

The tubing and fittings that we used came from a low-cost supermarket as a 'Micro Irrigation System'. This packet contained a few metres of tubing as well as small posts for anchoring the pipes, and small joining sections that were perfect for connecting the pump to the tubing.

The breadboard, jumper wires, and resistor are probably best bought as an electronics starter kit. The Monk Makes Electronic Starter Kit for Raspberry Pi includes these parts. Most starter kits for the Raspberry Pi will include the breadboard, jumper wires, and some resistors.



Above Peristaltic pump

Peristaltic pumps

Peristaltic pumps use a motor to drive a gearbox that in turn drives a pair of rollers that squeeze the flexible but strong pipe to push the water through the pump.

These pumps have the advantage that they are 'self-priming'. That is, you do not need to get the air out of the intake tube before they start pumping; they will eventually pull the water through. The other advantage they have is that when the motor stops, it effectively seals the pipe, so no water will flow through it even if the water is under a bit of pressure.

Building your waterer

As with all projects, it is a good idea to test the project out and get everything working while the parts are all still in your workspace. Once you know all is well, you can install the project in its final location.

BUILDING THE PROJECT

This is a pretty straightforward project to build. There are just three components on the breadboard. As well as the electronics, you will also need to find a way to attach the pump to your water container and, if it is in the open, keep the rain out of the pump.

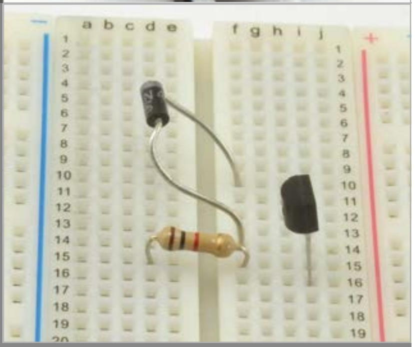
>STEP-01

Solder leads onto the pump

Bell wire or speaker cable is just fine. To make it easier to push the loose ends into the breadboard, tin them with solder so that they don't bunch up when you try to insert them into the breadboard.

>STEP-02

Build the breadboard



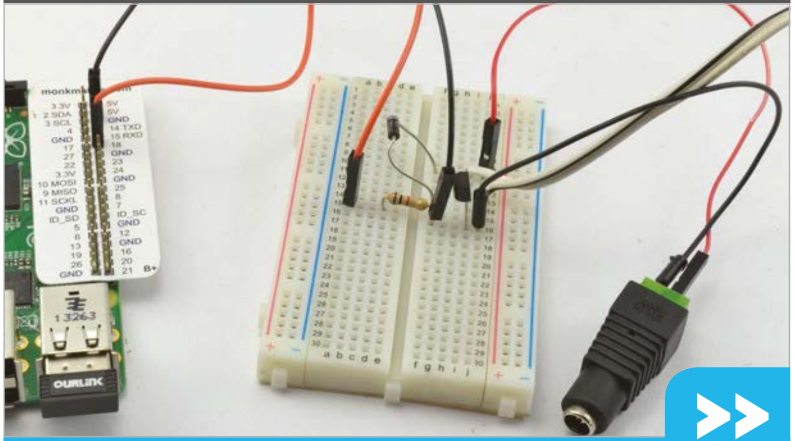
When inserting the components into the breadboard, make sure that you get the transistor and diode the correct way around. The transistor has one curved side that should be facing to the left, and the diode has a stripe on one end that needs to be towards the top of the breadboard. The resistor can be connected either way around. To prevent accidental shorts between the resistor and diode leads, it is a good idea to shorten one or both of the leads.

>STEP-03

Connect the breadboard

Attach male-to-male jumper wires to the screw terminals of the barrel jack adaptor. Use red and black leads, and make sure that you attach the red lead to the screw terminal marked '+'. Fit the other end of the jumper wires into the breadboard.

The wires to the pump motor should also now be fitted to the breadboard, and the female-to-male jumper wires used to connect the breadboard to GND and 18 on the Raspberry Pi GPIO header.



>STEP-04
Testing

Before installing the project, test it out using two short lengths of tube attached to the pump and two glasses, one containing water. Jump ahead to the software section and set the start time for a minute ahead of the current time. Wait for the pump to start, then make sure that it will pump water from one glass to the other. This will also tell you which tube is the inlet and which is the outlet.

Make sure that the tubes are really well connected to the pump. If any air at all can get in, then the pump will not work properly.

>STEP-05
Installation

Our example setup is temporary. The Raspberry Pi lives indoors, with the lead to the pump fed out through a window. The inlet tube to the pump fits under the water butt lid and down into the water. The outlet goes off to the plant container.

Now that the hardware side of the project is complete, we just need to get the software running. The program is written in Python and uses a library called Bottle to provide the web server feature of this project. If you made either the Web Door Lock or the Teenager Alarm Clock from issues 32 and 33 of *The MagPi*, you should already have Bottle installed. If not, to install Bottle, make sure that your Raspberry Pi has an internet connection and then run the commands:

```
sudo apt-get update
sudo apt-get install python-bottle
```

You can download the program for this project from your Raspberry Pi command line, using the command:

```
git clone https://github.com/simonmonk/
pi_magazine.git
```

Waterer_Server.py

```
from bottle import route, run, template, request
import thread, time
import RPi.GPIO as GPIO
import datetime

# Change these for your setup.
PUMP_PIN = 18

# Configure the GPIO pin
GPIO.setmode(GPIO.BCM)
GPIO.setup(PUMP_PIN, GPIO.OUT)
GPIO.output(PUMP_PIN, 0)

# Glogal variables
start_time = "22:00"
duration = "30"

# Handler for the home page
@route('/')
def index():
```

To run the program, change to the directory where the code for this project lives and then run the program using the commands below:

```
cd /home/pi/pi_magazine/08_waterer
sudo python waterer_server.py
```

This will start a web server running on your Raspberry Pi. You can check this by opening a browser window on another computer or your phone and entering the IP address of your Pi. Open the terminal and type **ifconfig** to find yours. If all is well, a webpage will be displayed, inviting you to enter a start time and duration for the watering.

How the code works

The Python code for this program is pretty heavily commented. You will probably find it handy to have the code up in an editor while we go through it.

The program starts by importing the **Bottle**, **thread**, **time**, **RPi.GPIO** and **datetime** libraries that it needs. The constant **PUMP_PIN** is used to specify which GPIO pin is to be used to control the pump. Change this value if you want to use a different pin.

Two global variables are used: **start_time** and **duration**. The variable **start_time** contains the time for the next watering to take place; this is in the format of a string containing two numbers for the hour (24-hour clock) followed by a colon, followed by two digits for the minute. The **duration** is the number of minutes that the watering should last. Both of these variables will be set using the web interface for the project.

Language
>PYTHON

```
    global start_time, duration
    # Change the start_time and duration

    local_start_time = request.GET.get('start_time', '')
    if local_start_time != "":
        start_time = local_start_time
    local_duration = request.GET.get('duration', '')
    if local_duration != "":
        duration = local_duration
    return template('home.tpl', start_time=start_time,
duration=duration)


def update(thread_name):
    global start_time, duration
    watering = False
    end_timestamp = 0
    while True:
        # Get the time formatted as a string for comparison
        # with the alarm time
        current_time = time.strftime("%H:%M")
        # If its time to start watering, do so, but not if
        # you are already watering
        if current_time == start_time \
and not watering:
            print("Starting Watering")
            watering = True
            GPIO.output(PUMP_PIN, 1)
            end_timestamp = time.time() + int(duration) * 60
        # Check to see if its time to stop watering
        if watering and time.time() > end_timestamp:
            print("End Watering")
            watering = False
            GPIO.output(PUMP_PIN, 0)

        time.sleep(1) # allow other threads to run

# start a separate thread to check the time
thread.start_new_thread(update, ("update_thread",))

# Start the webserver running on port 80
try:
    run(host="0.0.0.0", port=80)
finally:
    print('Cleaning up GPIO')
    GPIO.cleanup()
```

The next section of the code supplies the web interface. Bottle uses the **`@route`** directive to indicate that the function that immediately follows it is a handler for web requests. So the default route page of '/' is handled by the function called **`index`**. The **`index`** function uses Bottle's templating mechanism to return the HTML contained in the template file called **home.tpl**. You will find the HTML template used in the project contained in the code directory for it (**bit.ly/1NezVHT**).

The last few lines of the program start up the web server on port 80. The **`try`** / **`finally`** clause is used to set the GPIO pins back to inputs when the program is quit using **CTRL+C**.

Using your waterer

You can use a web browser on a computer, or on a smartphone if it is connected to your Wi-Fi network. For this reason, the Raspberry Pi needs to have a network connection, which also allows the Pi to synchronise with an internet time server. If you find that your Raspberry Pi is an hour out, then you may need to use the **raspi-config** tool to set the time zone.

If you enjoy a bit of web design, the first thing you will probably want to do is to add some styling to the webpages. The place to do this is in the template file **home.tpl**. If you are network-savvy, then you could also set up port forwarding to allow access to the waterer from the internet at large.

“ Since this Python program is mostly acting as a web server, the code to check the start time happens in a separate thread of execution ”

When you enter values into the **Start Time** and **Duration** fields on the webpage, they will be passed as parameters when the form is submitted. These are extracted in the **`index`** function and as long as they are not blank (empty-string), they are used to set new values of **`start_time`** and **`duration`**.

Since this Python program is mostly acting as a web server, the code to check the start time happens in a separate thread of execution. This thread is started by the command **`start_new_thread`**, which causes the **`update`** function to effectively run independently of the rest of the program. The **`update`** function checks the current time against the start time and if they are the same, it starts the watering process by turning on the GPIO pin that is connected to the pump via the transistor.

NEXT MONTH

In the next project in this series, we will build a 'pingometer', an analog meter that tells you how much lag there is in your internet connection. It's something no serious online gamer should be without!

ROB ZWETSLOOT
An avid coder and Raspberry Pi enthusiast with a history of building many things with a Raspberry Pi
twitter.com/RobThez

GET STARTED WITH SENSE HAT

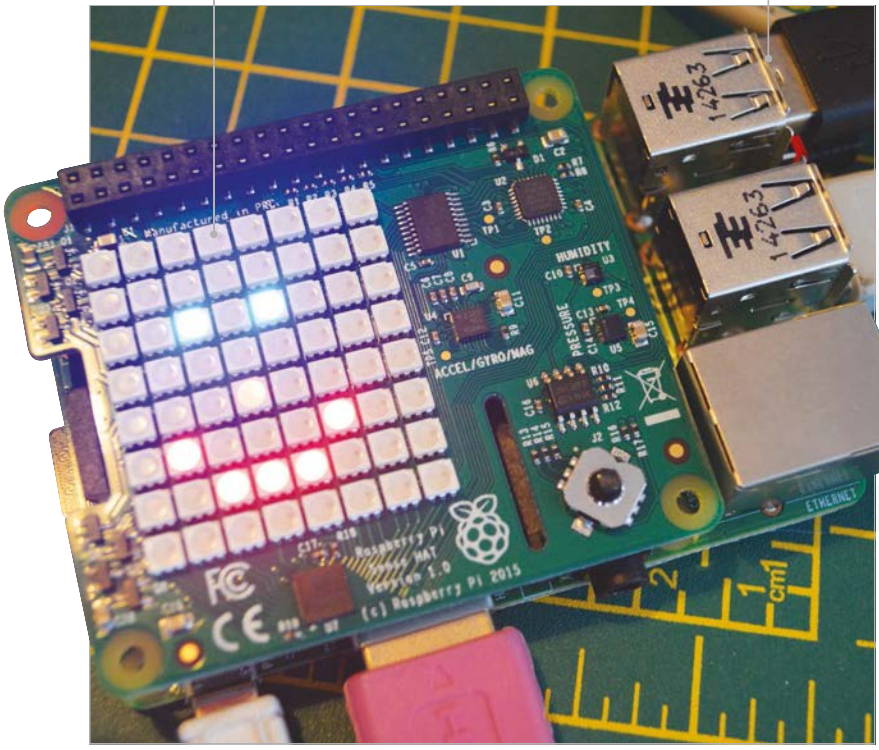
You'll Need

- Sense HAT - swag.raspberrypi.org/collections/frontpage/products/raspberry-pi-sense-hat
- Sense HAT library
- Raspbian - raspberrypi.org/downloads

The exciting new sensor add-on for the Raspberry Pi is very easy to use once you know how. We'll show you how, then.

This issue we've had a bit of a Sense HAT blow out, with our interview with the creators early on in the mag, and a review of it later on. It's an amazing piece of kit and hopefully by now you've been able to snag yourself one.

While it's definitely easy to use, there's a few things you need to learn first to get it going as it's reliant on programming it via Python scripts at the moment. The commands are very simple though, so we can show you in only a couple of pages how to start getting the most out of your new toy!



Learn how to use the features of the Sense HAT, and figure out how to make this smiley face

All you need is a Raspberry Pi to use the Sense HAT - no other gadgets!

>STEP-01

Installing the Sense HAT

Getting the Sense HAT ready to use is quite simple. Turn off your Raspberry Pi and make sure your Raspbian SD card is inserted if it isn't already. Place the Sense HAT on the GPIO pins, carefully aligning them before pressing down firmly so that it properly attaches to the Raspberry Pi. Once that's done, turn the Raspberry Pi back on. If it's attached properly, the LEDs on the SenseHAT will light up in a rainbow pattern during boot time. When it gets to the desktop, the pattern might turn off but that's normal.

>STEP-02

Sense HAT library

While the Sense HAT is now accessible, it's not fully functional as it is. You'll need to download a custom script that will install the necessary libraries for Python to access it, and update some of the core kernel code so that it runs properly. You can download and install the update by opening the terminal and running these two commands:

```
$ sudo apt-get install sense-hat
$ sudo pip-3.2 install pillow
```

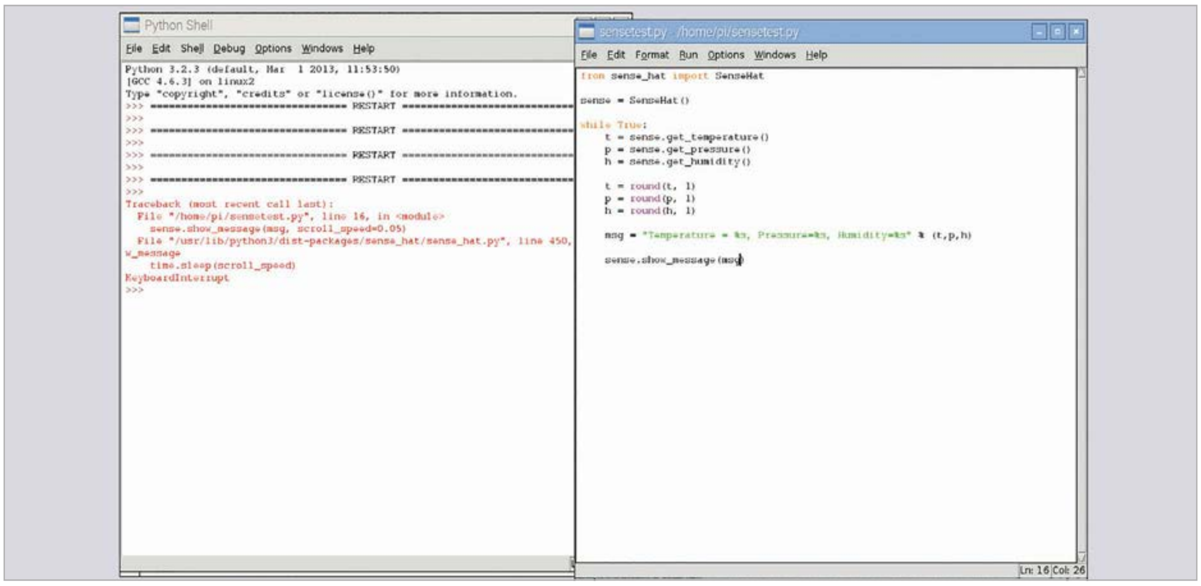
After both commands have run their course (they might take a few minutes each to complete) you'll need to reboot the Raspberry Pi. You might as well do this from the command line with:

```
$ sudo reboot
```

>STEP-03

Get started with Python

We can start controlling the Sense HAT by writing code in Python. As the Sense HAT is using the GPIO pins, we need to have administrative privileges to execute the scripts. This means opening IDLE using


Python Shell
File Edit Shell Debug Options Windows Help
sensetest.py - /home/pi/sensetest.py
File Edit Format Run Options Windows Help
Ln: 16 Col: 26

```
Python 3.2.3 (default, Mar  1 2013, 11:53:50)
[GCC 4.6.3] on linux2
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
>>> ================================ RESTART ================================
>>>
>>> ================================ RESTART ================================
>>>
>>> ================================ RESTART ================================
>>>
Traceback (most recent call last):
  File "/home/pi/sensetest.py", line 16, in <module>
    sense.show_message(msg, scroll_speed=0.05)
  File "/usr/lib/python3/dist-packages/sense_hat/sense_hat.py", line 450,
w_message
    time.sleep(scroll_speed)
KeyboardInterrupt
>>>
```

```
from sense_hat import SenseHat

sense = SenseHat()

while True:
    t = sense.get_temperature()
    p = sense.get_pressure()
    h = sense.get_humidity()

    t = round(t, 1)
    p = round(p, 1)
    h = round(h, 1)

    msg = "Temperature = %s, Pressure=%s, Humidity=%s" % (t,p,h)

    sense.show_message(msg)
```

EASIER PAINTING

Set colours to specific letters using lists (e.g., red could be r=[255,0,0]). You can go even further and create a matrix of letters that can display an image

Left You can set sensor outputs as variables to then be printed out on the LED, combining all the functions of the HAT

sudo in the command line with:

```
$ sudo idle3 &
```

Open up a new window to start writing a Python script. Each Sense HAT script needs to begin with two lines to import the relevant Python module, and then a line to turn it into a variable that makes writing the code easier.

```
from sense_hat import SenseHat

sense = SenseHat()
```

>STEP-04

Text and pictures

The easiest thing to start off with is to get text scrolling along the LED display. The library for the Sense HAT will automatically turn a string of words into a scrolling banner across the matrix. A line like the following will create it:

```
sense.show_message(“Hello World!”)
```

You can control the scroll speed and colours as well. You can also manipulate the individual LEDs by telling the script which pixel to change, and give it an RGB colour variable with something like:

```
sense.set_pixel(0, 0, [0, 0, 255])
```

Put this in a variable set the top left corner pixel as blue

>STEP-05

Environmental sensors

Using two of Sense HAT's sensors, you can measure temperature (in degrees C), pressure (in millbars) and relative humidity (as a percentage). You can get the three individual measurements with:

```
sense.get_temperature()
sense.get_pressure()
sense.get_humidity()
```

You'll have to attach them to a variable thought (e.g., t = sense.get_temperature()) so that you can print out the reading or display it on the LED matrix.

>STEP-06

Motion sensing

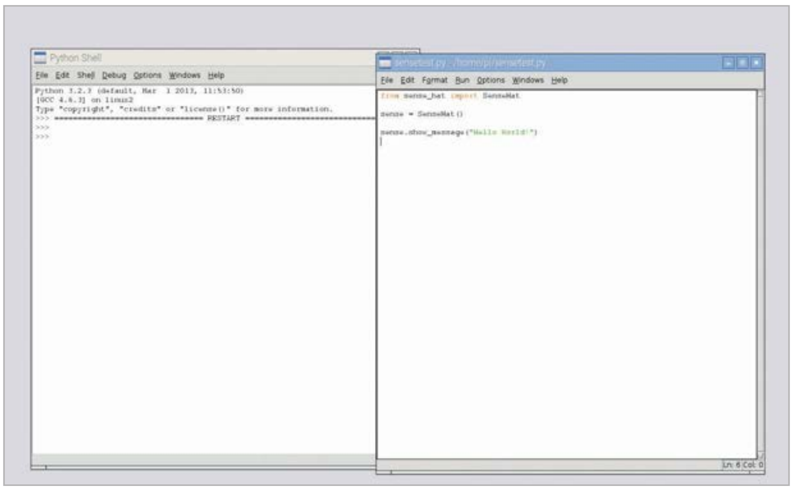
The gyroscope, accelerometer and magnetometer can be used to detect motion in real time. This is a little trickier to use than the other parts, as it involves understanding how the three-dimensional values represent the orientation of the Raspberry Pi, and then how the accelerometer can be used to figure out the movement within space. Both the gyroscope and accelerometer functions (get_orientation().values() and get.accelerometer_raw().values() respectively) will return three values from which you can use however you please.

This handy guide will explain all you need to know on the motion sensors: **http://bit.ly/1UBdEmA**

SLOW DOWN THE TEXT

You can edit your scrolling text's speed by adding the scroll_speed option. Use something like: **`sense.show_message(“Hello World!”, scroll_speed=0.5)`**

Below The most basic script you can write, but it's very useful to get to grips with the basic coding of the Sense HAT


Python Shell
File Edit Shell Debug Options Windows Help
sensetest.py - /home/pi/sensetest.py
File Edit Format Run Options Windows Help
Ln: 6 Col: 0

```
Python 3.2.3 (default, Mar  1 2013, 11:53:50)
[GCC 4.6.3] on linux2
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
>>>
```

```
from sense_hat import SenseHat

sense = SenseHat()

sense.show_message("Hello World!")
```

RICHARD HAYLER

Richard is a mentor at CoderDojo Ham, and his school Code Club was one of the winning teams in the Primary Astro Pi competition.

richardhayler.blogspot.co.uk
www.coderdojoham.org | @rdhayler

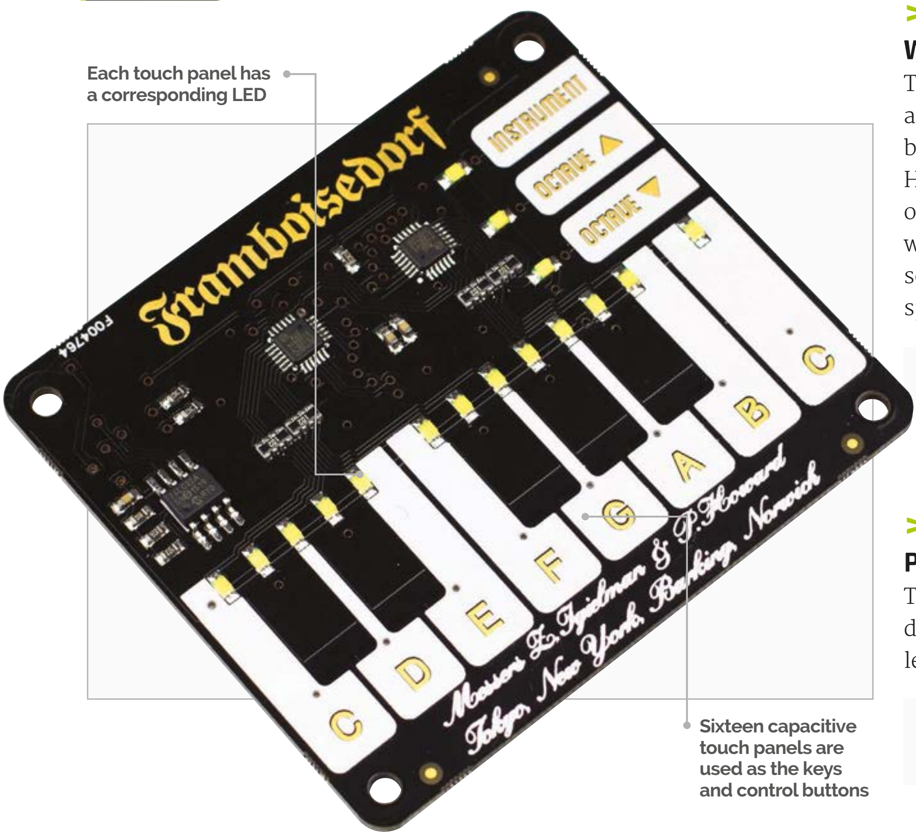
PLAYING IT BY EAR WITH PIANO HAT

The latest HAT from Pimoroni is a great way to unleash your ivory-tinkling tendencies. Let's use it to build a relative pitch tester

You'll Need

- A Piano HAT **shop.pimoroni.com/products/piano-hat**
- The Piano-HAT library **github.com/pimoroni/Piano-HAT**
- Headphones or an external speaker

Relative pitch is the ability to identify a given musical note by comparing it to a reference note. Unlike perfect pitch, relative pitch can be improved with training. The Piano HAT is a versatile piece of hardware that we can use to create a fun game that tests people's skill in recognising different notes. It was inspired by Zachary Igielman's legendary PiPiano and it turns your Pi into a functional musical keyboard. Each of the 16 capacitive keys also has an LED so you can create you own 'learn to play' tutorials or just give your performances a visual appeal.


INSTRUMENT
OCTAVE
OCTAVE
Framboisedorf
C
D
E
F
G
A
B
C
Messrs Z. Igielman & P. Howard
Tokyo, New York, Barking, Norwich
FOO4764

Each touch panel has a corresponding LED

Sixteen capacitive touch panels are used as the keys and control buttons

>STEP-01

Getting started with Piano HAT

Like most HATs, this one is straightforward to use. Simply plug it carefully onto the GPIO pins of your Pi. Then install the Piano-HAT Python library. This requires the I²C bus on the Pi to be enabled, and there are plenty of instructions for this online. But to make life super-easy, those Pirates at Pimoroni provide a handy script that takes care of everything:

```
curl -sSL get.pimoroni.com/pianohat | bash
```

>STEP-02

Wired for sound

There are two options for getting audio output from a Pi. If you are using a HDMI monitor or a TV that has built-in speakers, the audio can be played over the HDMI cable. If not, you can switch to use headphones or a speaker plugged into the headphone jack. The Pi will normally auto-detect the available outputs, but sometimes it gets this wrong. To force audio to use a specific output, you can use this command:

```
amixer cset numid=3 2
```

The second number determines the output: HDMI = 2, jack = 1, auto = 0.

>STEP-03

Play it again

The Piano-HAT library has a nice collection of demonstration Python scripts. A good one to start with lets you use the Piano HAT as… a piano!

```
sudo python Pimoroni/pianohat/
simple-piano.py
```

Relative_Pitch.py

Language
>PYTHON 2.7 & 3

DOWNLOAD:
github.com/topshed/RelativePitch

```
import pianohat # import libraries we need
import pygame
import time, random

pygame.mixer.pre_init(44100, -16, 1, 512) #Configure pygame sound
pygame.mixer.init() #Initialise pygame mixer
pygame.mixer.set_num_channels(16)
pianohat.auto_leds(True) # LEDs light when keys pressed
# A dictionary mapping sounds and notes onto keys
NOTES = {'0':['C','./sounds/piano/39172__jobro__piano-ff-025.
wav'], # C
  '1': ['C Sharp', './sounds/piano/39173__jobro__piano-ff-026.
wav'], # C sharp
  '2':['D','./sounds/piano/39174__jobro__piano-ff-027.wav'], # D
  '3':['D Sharp','./sounds/piano/39175__jobro__piano-ff-028.wav'],
# D sharp
  '4':['E','./sounds/piano/39176__jobro__piano-ff-029.wav'], # E
  '5':['F','./sounds/piano/39177__jobro__piano-ff-030.wav'], # F
  '6':['F Sharp','./sounds/piano/39178__jobro__piano-ff-031.wav'],
# F sharp
  '7':['G','./sounds/piano/39179__jobro__piano-ff-032.wav'], # G
  '8':['G Sharp','./sounds/piano/39180__jobro__piano-ff-033.wav'],
# G sharp
  '9':['A','./sounds/piano/39181__jobro__piano-ff-034.wav'], # A
  '10':['A Sharp','./sounds/piano/39182__jobro__piano-ff-035.wav'],
# A sharp
  '11':['B','./sounds/piano/39183__jobro__piano-ff-036.wav'], # B
  '12':['C','./sounds/piano/39184__jobro__piano-ff-037.wav'] # C
}

def handle_note(channel, pressed): # handler for key presses
    global note
    global correct
    if channel < 13 and pressed: # Only for note keys
        if str(channel) == note: # Did the player get it right?
          print('correct, it was a ' + str(NOTES[note][0]) )
          pianohat.auto_leds(False)
          for x in range(16): #
Flash all the lights to celebrate
            pianohat.set_led(x,
True)
          time.sleep(0.05)
          for x in range(16): # Them turn them off
            pianohat.set_led(x,False)
          pianohat.auto_leds(True)
          correct = True
        else:
          print('wrong, try again')

def play(note): # Play a note from the dictionary
        pygame.mixer.Sound(NOTES[note][1]).play(loops=0)
        time.sleep(1)

pianohat.on_note(handle_note) # Set keys to use our handler

while True:
  print('Here comes a C')
  time.sleep(1)
  play('0') # Play a C
  time.sleep(2)
  correct = False
  print('Now identify this note')
  time.sleep(2)
  note = random.choice(list(NOTES)) # Pick a random note
  play(note)
  print('press the key for the note you heard')
  count = 6 # Set countdown timer
  while count > 0 and correct == False:
    time.sleep(1)
    print(str(count) + ' Seconds remaining')
    count -=1
  if not correct: # If they didn't get it right, tell them the
answer
    print("time's up, it was a " + str(NOTES[note][0]))
```

If you press the Instrument key, you'll notice that the sounds change from pianos to percussion. The Piano-HAT library itself does not map any of the keys to a particular sound; that is all done using Python. The sounds themselves are WAV files, which are played using the Pygame library.

Let's map our own sound to one of the Piano HAT's keys. Find a short WAV file online (or create your own using Sonic Pi) and save it as **mysound.wav**. Then type in the code from **Listing_l1.py** (on page 52) and run it: your sound should play when the D key is pressed.

>STEP-04
A little light music

You'll notice that the relevant LED lights up when any key is pressed. This is the default behaviour, but can be disabled using:

```
pianohat.auto_leds(False)
```

Add this line immediately before the **handle_note** function in **Listing_l1.py**. Now we can add code so that only the D key's LED will work. Insert

```
pianohat.set_led(2,True)
```

...before the **pygame.mixer.Sound** line and

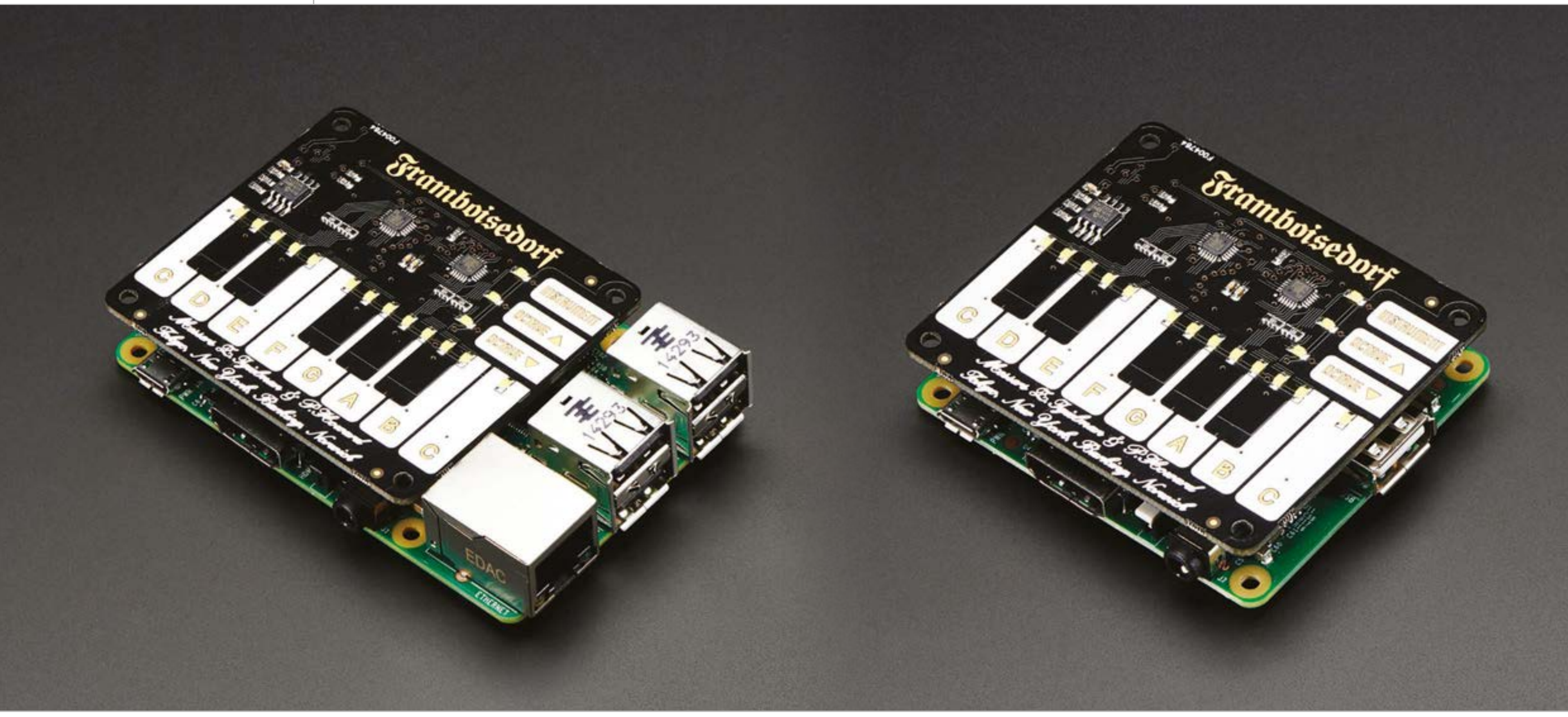
```
pianohat.set_led(2,False)
```

...after it.

Now rerun the program to verify that only the D lights up when tapped.

>STEP-05
Use a dictionary

Mapping keys to sounds using a bunch of **if…** statements is easy but rather long-winded. For our relative pitch test, we want to associate the key (an integer) with the note (a text string) and the sound



to be played (also a string). A simple way is to use a Python construct called a dictionary. A real-world dictionary has an index of words, and each word has definitions. In a Python dictionary, the word is called the 'key', and the definitions the 'values'.

Listing_12.py uses a simple two-item dictionary to map sounds and names (the values) onto the C and D keys (the keys). Give it a try.

"In a Python dictionary, the word is called the 'key', and the definitions the 'values'"

>STEP-06

Putting it all together

We've now explored everything needed for our relative pitch tester: we'll use the piano sounds that come with the Piano-HAT library for our notes.

Type up the code from the **Relative_Pitch.py** listing (on page 51) and run it with **sudo**. A reference note (a C) is played, then, after a pause, the note to be identified. The player then has 6 seconds to press the correct key for the note they just heard. If they get it right, all the LEDs will flash in celebration, otherwise they're asked to try again.

As an extension, how about using different, selectable instruments?

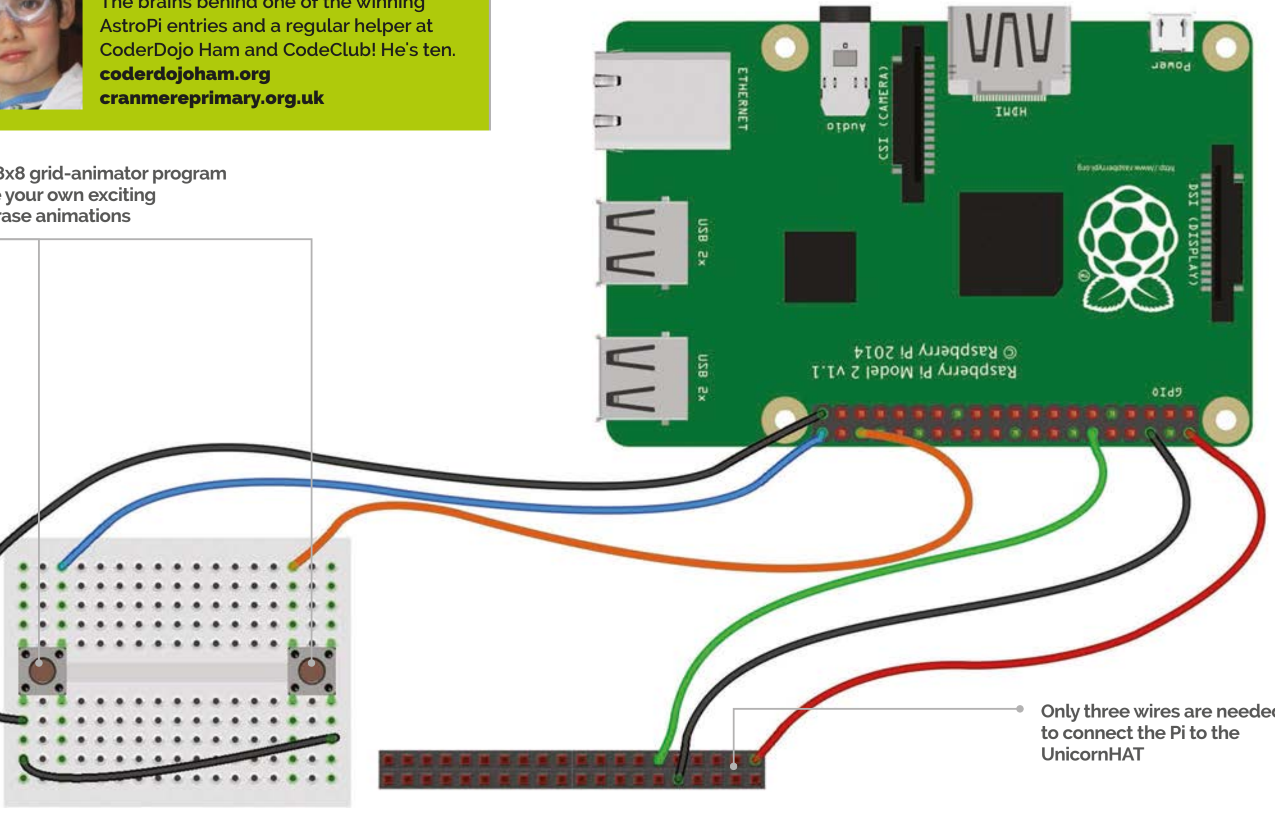
Listing_11.py

```
import pianohat
import pygame
import signal

pygame.mixer.pre_init(44100, -16, 1, 512)
pygame.mixer.init()
pygame.mixer.set_num_channels(16)

def handle_note(channel, pressed):
        if channel == 2:
                pygame.mixer.Sound('./mysound.
wav').play(loops=0)

pianohat.on_note(handle_note)
signal.pause()
```

Listing_12.py

```
import pianohat
import pygame
import signal

pygame.mixer.pre_init(44100, -16, 1, 512)
pygame.mixer.init()
pygame.mixer.set_num_channels(16)
NOTES = {0:['Sound1','./mysound.wav'],
                2: ['Sound2', './mysound2.wav']}

def handle_note(channel, pressed):
        if channel == 0 or channel == 2:
                pygame.mixer.Sound(NOTES[channel][1]).play(loops=0)

pianohat.on_note(handle_note)
signal.pause()
```

JASPER HAYLER-GOODALL
The brains behind one of the winning AstroPi entries and a regular helper at CoderDojo Ham and CodeClub! He's ten.
coderdojoham.org
cranmereprimary.org.uk



Use the 8x8 grid-animator program to create your own exciting catchphrase animations

Only three wires are needed to connect the Pi to the UnicornHAT

You'll Need

- A Unicorn or Sense HAT **shop.pimoroni.com**
- A tall 40-pin header (for Sense HAT) **shop.pimoroni.com**
- Some jumper wires, two buttons, and a breadboard
- The 8x8 grid animator program **github.com/topshed/RPi_8x8GridDraw**
- The Fuzzy Wuzzy library **github.com/seatgeek/fuzzywuzzy**

PIPHRASE!

Make your own version of the popular TV show, Catchphrase. All you need are a few components and a LED matrix, in this case the Unicorn HAT...

Catchphrase is a TV game show where contestants try to guess the well-known phrase or saying being visualised on a big screen. In this project, you will be making a similar game, using either a Sense HAT or a Unicorn HAT to display the animation. We have mainly written this project for a Unicorn HAT, but with a little modification it can work with the Sense HAT. To make the game two-player, we have added some buttons to see who is the fastest at guessing the catchphrase.

>STEP-01
Put your hat on

Later on, we will attach a breadboard to use some buttons, so we'll need access to the Pi's GPIO pins. Therefore you shouldn't just place the Unicorn HAT directly onto the GPIO pins; instead, connect it using jumper wires. Surprisingly, the Unicorn HAT only uses three pins (5V, GND, and GPIO 18), so these are the only ones we need to worry about.

If you have a Sense HAT, remove the header already attached and replace it with a tall one instead. This should leave some pins sticking out of the top of the Sense HAT, which will be enough to attach the wires.

>STEP-02
Download the animator program and get creating

When you download the 8×8 grid animator program, make sure you use the appropriate one for your HAT. If you have a Unicorn HAT, use the command:

```
sudo python 8x8grid-unicorn.py
```

To create an animation, simply select a colour from the palette on the right, and click on a circle on the grid to turn that colour on. Add another frame by pressing the >> button; this will duplicate the current frame. If you need to clear the frame, press the 'clear frame' button. You can also play your animation on

Maingame-Uni.py

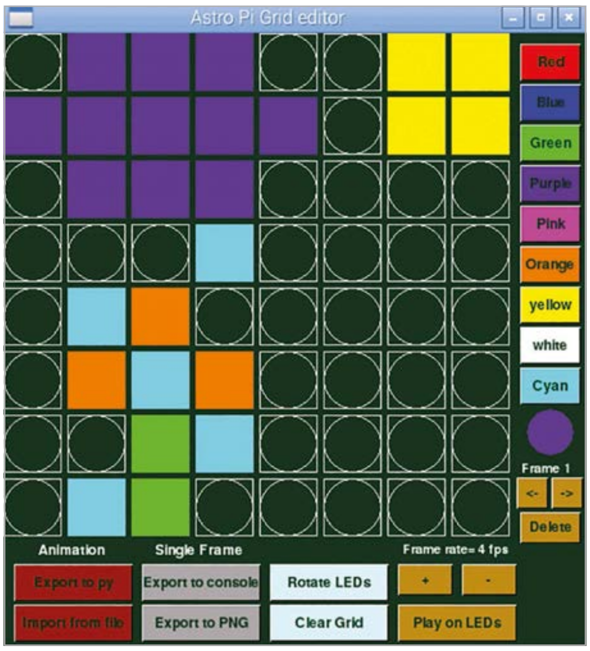
```
import unicornhat as uh #import the necessary modules
import time, random,signal
import catchphrases_uni as cp # the file with our
animations
import RPi.GPIO as GPIO
from fuzzywuzzy import fuzz

def halfthing(num,colour): #turns on half the LEDS to
indicate who pressed button first
        uh.clear()
        for x in range (num,num+4):
                for y in range(0,8):
                        uh.set_pixel(x,y,colour[0],col
our[1],colour[2])
        uh.show()

def  pressed(num): # Function to be run when buttons
pressed
        global running
        running = False # stops animaton from playing
        halfthing(num,(255,0,0))
        get_answer(3) # call with goes set to 3

def b21pressed(channel): # Run when button on GPIO 21
pressed
        pressed(0)

def b16pressed(channel): # Run when button on GPIO 16
pressed
        pressed(4)

def get_answer(goes): # Asks the player for their
answer
    answer = raw_input('Name that catchphrase? ')
    match = fuzz.ratio(answer, picked) # compare
answer to catchphrase title
    if  match >=85:
        print 'Correct!'
    elif match >= 60 and match < 85: # if it nearly
matches
        goes=goes-1
        if goes==0:
            print('run out of goes')
```

the LEDs and delete frames. When you are happy with your animation, click 'export to py' and it will be saved as **animation8x8.py**.

>STEP-03
Tinkering with the animations

You will then need to rename the file (to your own choice of name) by using the command line:

```
mv animation8x8.py yourchoice.py
```



Right Use the 8x8 grid-animator program to create your own exciting catchphrase animations

You should then repeat step-02 until you have at least three animations. Make sure you rename them after each export.

Now we need to copy the animations into the catchphrase code. First of all, open up **yourchoice.py** (or whatever you called it) in IDLE, then find the line which has **frames=[** on it. Highlight this all the way to the final **]**, then copy and paste it into a new file called **catchphrases.py**. Do the same with the rest of the animations and paste them into the same file. At the end, you'll need to use a Python dictionary and a list to link the animations to their name; an example **catchphrases.py** is included in the GitHub repository for this project.

>STEP-04
Connect and test buttons

To connect the buttons to the Raspberry Pi, you need three jumper wires, one breadboard, and two buttons. To make the circuit, follow the image on page 55; to test the buttons, use the **listing1.py** code. If using a Sense HAT, make sure you use the same GPIO pins (16 and 21).

>STEP-05
Fuzzywuzzy matching

Fuzzy matching is about finding out how close one string is to another. Open up a Python shell (either using the command line or IDLE) and then import the **fuzzywuzzy** library.

```
>>>from fuzzywuzzy import fuzz
```

Language
>PYTHON 2.7

DOWNLOAD:
github.com/snake48/PIPHRASE

```
        else:
            print 'Close.. Try again (' +( str(goes) +
' guesses remaining)')
            get_answer(goes)
    else:
        print "That's not right"
    print 'Press ^C to exit'

def show_catchphrase():  # Display an animation on the
LED matrix
    picked = random.choice(cp.ANSWERS.keys()) # Pick
random animation from file
    for i in range(3):
        for x in cp.ANSWERS[picked]:
            if running:
                uh.set_pixels(x) # display a frame from
animation
                uh.show()
                time.sleep(0.25) #pause before next
frame
        time.sleep(1) # pause before we play animation
again
    return picked

GPIO.setmode(GPIO.BCM) #set GPIO
numbering to BCM
GPIO.setup(21,GPIO.IN, pull_up_
down=GPIO.PUD_UP) #Set these pins as
inputs
GPIO.setup(16,GPIO.IN, pull_up_down=GPIO.PUD_UP)
#set callback functions that are run when button
pressed
GPIO.add_event_detect(21,GPIO.FALLING,callback =
b21pressed,bouncetime=300)
GPIO.add_event_detect(16,GPIO.FALLING,callback =
b16pressed,bouncetime=300)
try:
        running = True # use this to stop animation
when button pressed
        picked = 'blank'
        picked = show_catchphrase()
        signal.pause() #stops the code from exiting
except KeyboardInterrupt: # catch ^c
        exit() #exit the code
```

Fuzzywuzzy gives you a score out of 100 for how close two phrases are.

```
>>>fuzz.ratio('hello world','catchphrase')
18
>>>fuzz.ratio('piphrase','pyphase')
80
```

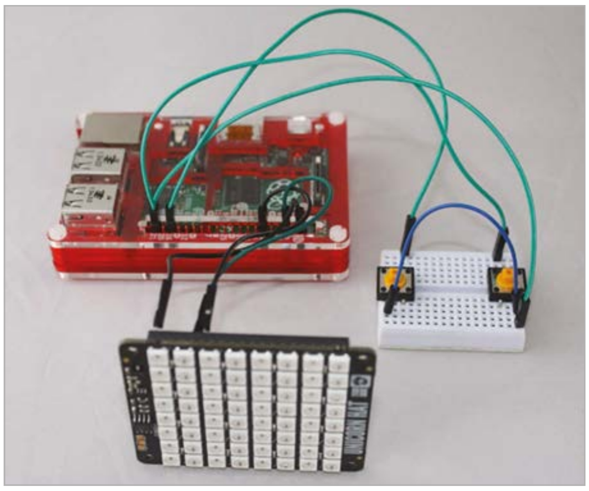
We can use this to check how close the player's answer is to our catchphrase (because we don't want to be too mean!).

>STEP-06
The final code

Now we need to put everything we've learned together to create the catchphrase game. Type the code from the main listing (above) into a file called **piphrase.py**, then run it using:

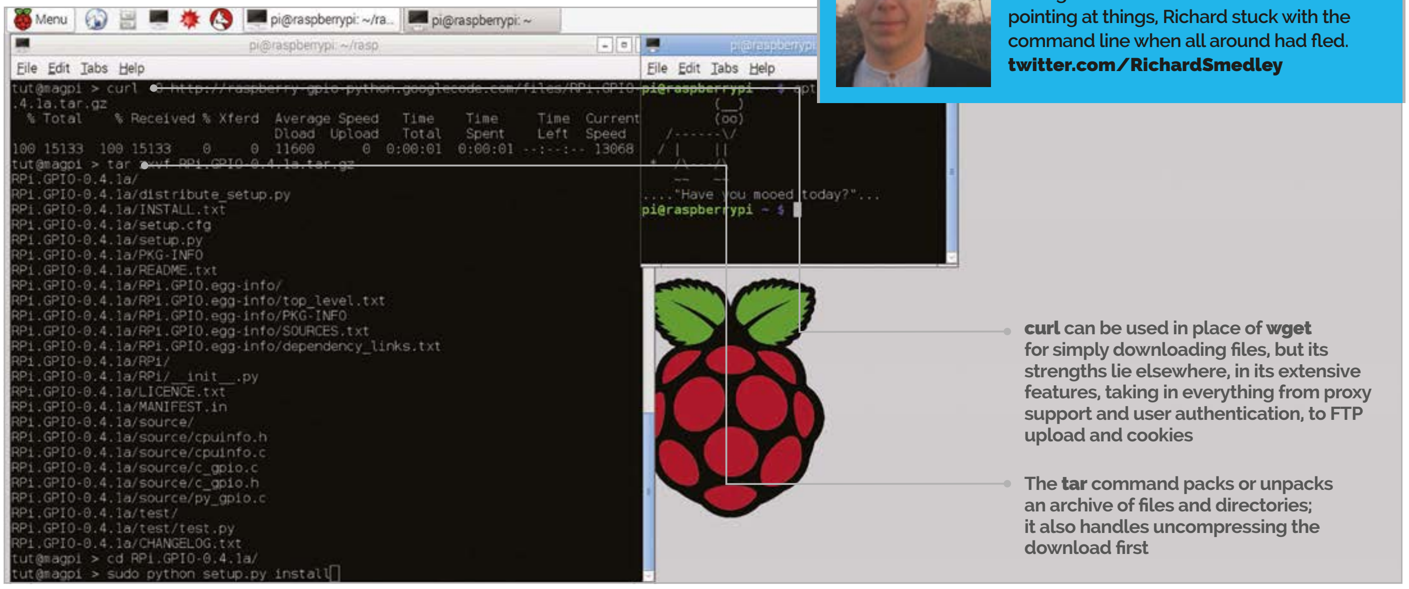
```
sudo python piphrase.py
```

The program should make an animation play three times on the Unicorn HAT while waiting for someone to press either button. When one of the buttons is pressed by a player, their side of the Unicorn HAT lights up. They then need to type in the catchphrase they think is being shown. The program will the tell them if their answer is correct or incorrect. The player might also be shown the message 'try again' if their answer falls somewhere between 60 and 80 percent correct according to the fuzzy matching.


Left This project will work with a Sense HAT or a Unicorn HAT. If you're using a Unicorn HAT, it should be connected like this

Listing1.py

```
import RPi.GPIO as GPIO #import the necessary modules
import time

GPIO.setmode(GPIO.BCM)  #set GPIO numbering to BCM
GPIO.setup(16,GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(21,GPIO.IN, pull_up_down=GPIO.PUD_UP)

while True:
    print('b1= '+str(GPIO.input(16))+ ' b2= '+str(
GPIO.input(21)))
    time.sleep(0.1)
```



```
tut@magpi > curl -O http://raspberry-gpio-python.googlecode.com/files/RPi.GPIO-0
.4.1a.tar.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 15133  100 15133    0     0  11600      0  0:00:01  0:00:01 --:--:-- 13068
tut@magpi > tar zxvf RPi.GPIO-0.4.1a.tar.gz
RPi.GPIO-0.4.1a/
RPi.GPIO-0.4.1a/distribute_setup.py
RPi.GPIO-0.4.1a/INSTALL.txt
RPi.GPIO-0.4.1a/setup.cfg
RPi.GPIO-0.4.1a/setup.py
RPi.GPIO-0.4.1a/PKG-INFO
RPi.GPIO-0.4.1a/README.txt
RPi.GPIO-0.4.1a/RPi.GPIO.egg-info/
RPi.GPIO-0.4.1a/RPi.GPIO.egg-info/top_level.txt
RPi.GPIO-0.4.1a/RPi.GPIO.egg-info/PKG-INFO
RPi.GPIO-0.4.1a/RPi.GPIO.egg-info/SOURCES.txt
RPi.GPIO-0.4.1a/RPi.GPIO.egg-info/dependency_links.txt
RPi.GPIO-0.4.1a/RPi/
RPi.GPIO-0.4.1a/RPi/__init__.py
RPi.GPIO-0.4.1a/LICENCE.txt
RPi.GPIO-0.4.1a/MANIFEST.in
RPi.GPIO-0.4.1a/source/
RPi.GPIO-0.4.1a/source/cpuinfo.h
RPi.GPIO-0.4.1a/source/cpuinfo.c
RPi.GPIO-0.4.1a/source/c_gpio.c
RPi.GPIO-0.4.1a/source/c_gpio.h
RPi.GPIO-0.4.1a/source/py_gpio.c
RPi.GPIO-0.4.1a/test/
RPi.GPIO-0.4.1a/test/test.py
RPi.GPIO-0.4.1a/CHANGELOG.txt
tut@magpi > cd RPi.GPIO-0.4.1a/
tut@magpi > sudo python setup.py install
```

```
pi@raspberrypi ~ $ apt
                 (__)
                 (oo)
           /------\/
          / |    ||
         *  /\---/\
            ~~   ~~
...."Have you mooed today?"...
pi@raspberrypi ~ $
```

curl can be used in place of **wget** for simply downloading files, but its strengths lie elsewhere, in its extensive features, taking in everything from proxy support and user authentication, to FTP upload and cookies

The **tar** command packs or unpacks an archive of files and directories; it also handles uncompressing the download first

RICHARD SMEDLEY

Having often found words better than pointing at things, Richard stuck with the command line when all around had fled.
twitter.com/RichardSmedley

COMMAND LINE PI PART 8: DOWNLOADING AND INSTALLING

Richard Smedley presents your cut-out-and-keep guide to using the command line on the Raspberry Pi. In part 8, he covers downloading and unpacking software, and creating new Raspbian SD cards

You'll Need

- Raspbian **raspberrypi.org/downloads/** – though most of the tutorial series will work with the command line running the Linux default Bash shell on any GNU/Linux PC.

Running an **apt** command (see part 3) allows access to a huge collection of software – several thousands of packages in the main Raspbian repository – but sometimes we need to add software from outside the main repository.

If we are lucky, we find that someone has packaged up the software in the .deb format used by Raspbian, or even created a whole repository to take care of the dependencies. We'll look briefly both at adding repositories, and dealing with other kinds of downloads, along the way trying the venerable vi editor.

Information about repositories is kept in the file **/etc/apt/sources.list** – which on a new install just has the Raspbian repository (there are actually two: one for the binaries you use, and one to get the source code, which enables you to learn from or modify any Raspbian software). To add a new repository, edit the file and add it in the same format:

```
deb http://apt.adafruit.com/raspbian/ wheezy main
```

Wheezy is a Debian release name – they've been named after characters in the *Toy Story* series of films since 1996 (former Debian project leader Bruce Perens was involved in the early development of Debian while working at Pixar). Jessie followed Wheezy in April 2015.

Most software is in the main repository; other components, like non-free, allow repositories to contain software you may not be free to pass on, keeping it separate from Raspbian's FOSS repository; main can be freely copied or mirrored anywhere.

vi editor

If you tried **robots** for a while after part 3, you'll be used to the **hjkl** keys for directional movement. If you're feeling brave, now is the chance to try them in a serious task – editing the **sources.list** file. It's not compulsory – use nano if you wish – but sooner or later you could come across a Linux or BSD computer without nano; vi is *always* the default on such machines. It's there in Raspbian in the form of *vim.tiny*, which you can call as **vi**.

VI IMPROVED

If you really want to get to grips with vim, you'll need to **apt-get install vim** – the vim.tiny package already in Raspbian is very limited.

```
vi /etc/apt/sources.list
```

vi is a modal editor – you start in command mode, moving to the line you wish to modify (**hjkl** in place of arrow keys); to edit, hit **i** for insert mode, and you can now edit the line under the cursor. The **ESC** key gets you back to command mode. If editing has gone okay, then **ESC** followed by **:wq** will get you out of vi; if not, **:q!** will leave without saving.

There's a lot more to vi – in the form of the more powerful vim (vi improved), it is a popular editor in the Ruby community. For now, be happy that when faced with a lack of nano, you'll get by – there's an old joke of someone running vi for years, not because they liked it, but because they couldn't figure out how to exit it!

wget & curl

Having added our repository to **sources.list**, we need to get the key and use **apt-key** to install it. Packages authenticated using keys added by **apt-key** will be considered trusted.

```
wget -O - -q https://apt.adafruit.com/apt.
adafruit.com.gpg.key | apt-key add -
```

Wget downloads from the URL given. The **-O -** directs the download to *stdout*, from where it is piped to apt-key (the trailing dash there tells apt-key to read its input from the stdin stream, which is where it receives the output from wget). After any change to the sources.list file, you must run:

```
sudo apt-get update
```

...to update Raspbian's knowledge of what's available to install from Adafruit's packages – see **apt.adafruit.com** for details; e.g. **apt-get install wiringpi**.

Wget is a simple but robust download tool, with a powerful recursive feature that helps fetch entire websites – but it does have mild security risks, so be careful using it to fetch scripts. curl is a file transfer tool, working with many protocols, that can be used for simple downloads. It dumps to *stdout* by default; to save as a file with the same name as the resource in the URL, use the **-O** switch:

```
curl -O http://raspberry-gpio-python.
googlecode.com/files/RPi.GPIO-0.4.1a.tar.gz
```

Unzip

The Python GPIO library downloaded above (and in the screenshot) is compressed with gzip, which losslessly reduces the size of files, and can be decompressed with **gunzip**. The contents here are files rolled into a tar archive (instead of **.tar.gz**, you'll sometimes find similar archives ending **.tgz**), and the **tar** command can do the decompression and untarring in one:

```
tar zxvf Rpi.GPIO-0.4.1a.tar.gz
```

Note that the dash is not needed for single letter options in tar. The first switch, **z**, calls gzip to decompress the archive, then **x** extracts the contents. **v** is verbose, informing you of the process as it happens, and **f** tells tar to work with the named file(s), rather than stdin. Miss out the **z** and tar should automatically detect the necessary compression operation.

The result in this case is a folder containing *inter alia* a setup script to run the installation (read the **INSTALL.txt** file first):

```
cd RPi.GPIO-0.4.1a
sudo python setup.py install
```

While gzip is more efficient than zip (and yet more efficient options like bzip2 are available), sometimes you'll get a plain old zip file, in which case **unzip** is the command you want.

```
unzip 2014-12-24-wheezy-raspbian.zip
```

Disk image

Having downloaded and unzipped an image for Raspbian, you cannot copy it across to a microSD card with regular **cp**, which would simply put a copy of it as a file on the card. We need something to replace the SD card's file system with the file system and contents that exist inside the Raspbian disk image, byte-for-byte, using a handy little built-in utility called **dd**.

dd converts and copies files – any files, even special devices like /dev/zero or /dev/random (you can make a file full of zeroes or random noise) – precisely copying blocks of bytes. To copy our Raspbian image, we need to unmount the SD card we've plugged in. Use **sudo fdisk -l** both before and after plugging in the SD card (you can also use **df** to see what's mounted). If, say, a /dev/sdb appears, with the size equal to the SD card, then unmount with **umount /dev/sdb1**. Now copy the disk image with:

```
 sudo dd if=~/Downloads/2015-05-05-wheezy-
raspbian.img of=/dev/sdb bs=1M
```

Development of Raspbian's great grandparent, Unix, started in 1969, so we've covered a few utilities with a long heritage in this series, but that **if=...** in place of the usual dashes for command-line options indicates a lineage stretching back to the early 1960s, and IBM's DD (data definition) statement from the OS/360 Job Control Language (JCL).

Be very careful that the destination matches the correct disk, or you will lose the contents of another storage device! The **bs=1M** is a block size default – **4M** would be another safe option. Now put the card in another Pi and go and have fun!

EASTER EGG

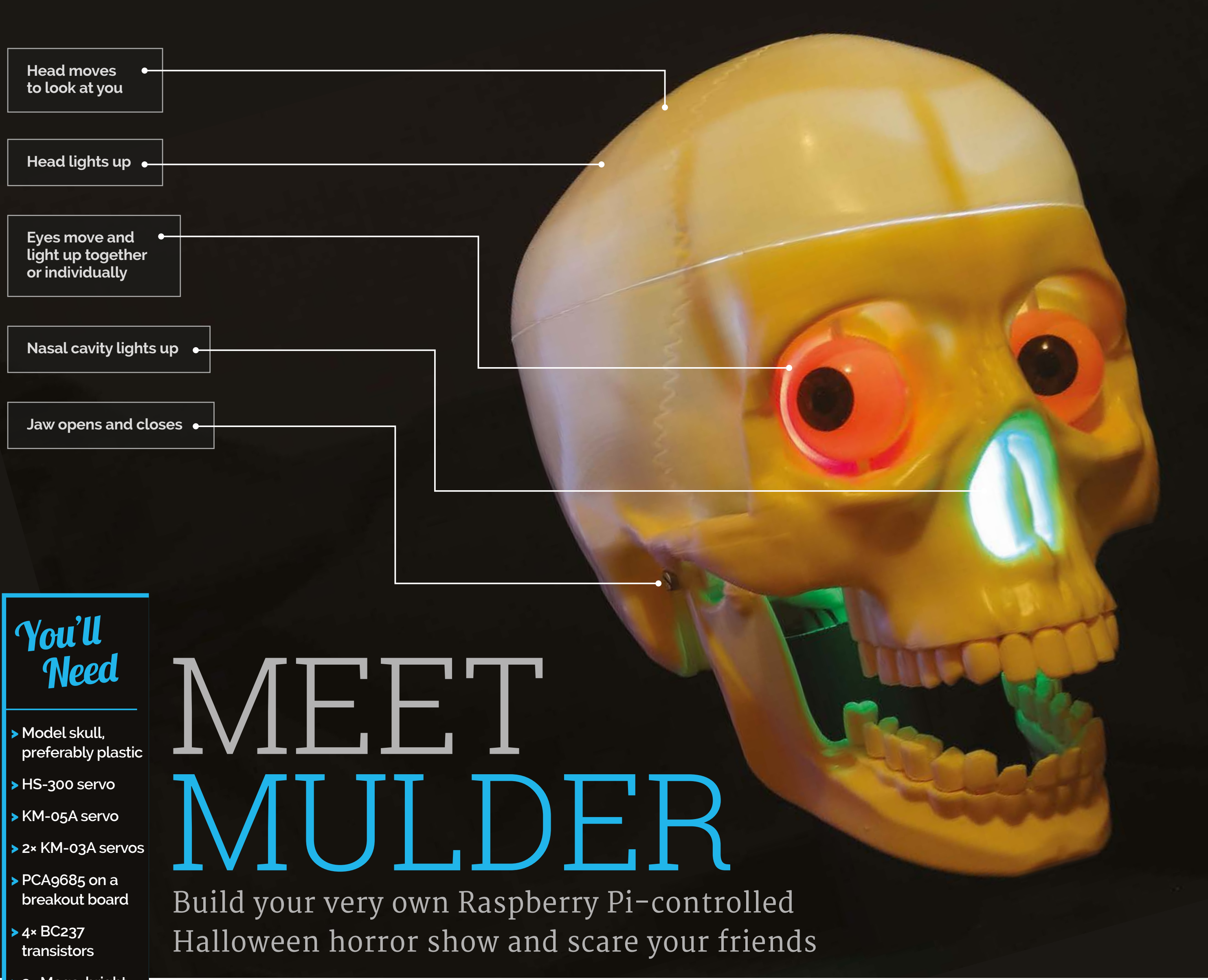
Read **man apt**, and you may see: "This APT has Super Cow Powers." If it's there, you can type **apt-get moo** and see what happens.

MIKE'S PI BAKERY

MIKE COOK

Veteran magazine author from the old days and writer of the Body Build series. Co-author of *Raspberry Pi for Dummies*, *Raspberry Pi Projects*, and *Raspberry Pi Projects for Dummies*.

bit.ly/1aQqu15


Head moves to look at you
Head lights up
Eyes move and light up together or individually
Nasal cavity lights up
Jaw opens and closes

MEET MULDER

Build your very own Raspberry Pi-controlled Halloween horror show and scare your friends

You'll Need

- Model skull, preferably plastic
- HS-300 servo
- KM-05A servo
- 2× KM-03A servos
- PCA9685 on a breakout board
- 4× BC237 transistors
- 3× Mega-bright white LEDs
- 3× 5050 RGB LEDs
- Various capacitors, resistors, and an inductor
- Styrene tube and sheet
- Length of 10mm by 24mm angle aluminium
- Nuts, bolts, and pillars M3 and M2
- Pack of Sugru

While it is Halloween at the end of this month, we had to start this project way back in July. In fact, we had it in mind long before that. One of the things we do here at Mike's Pi Bakery is to keep an eye on children's comics and part-works. One caught our eye back in 2012: it was the first two parts of a build-your-own skeleton, *How Your Body Works*. It was snapped up at the bargain price that the first parts of such a publication often go for. You can still get those issues today on eBay (we checked) although they are a touch more expensive than they were originally. You can also apply the same techniques as we will show you here to any other model head you may be able to get hold of, providing of course that the owner does not object.

The project

We wanted to get Mulder to move and light up, under the control of the Raspberry Pi, to generate a scary Halloween experience. When it comes to moving things with precise control, there are basically two choices: servos or stepping motors. While stepping motors are easy to control, they can be heavy, and the movement control is only relative to the starting position. That means you need some sort of positional control. Servos, on the other hand, are light and will move to an absolute position; however, in order to hold that position, they need a precisely timed pulse. While the Raspberry Pi is not good with precise timing, there are hardware chips to help, so we chose to use servo motors on this project.

While any Pi can control two servo motors (although many websites claim it can only drive one), two motors would not be enough for all the movement we wanted to incorporate into this project. We wanted the skull to turn around, the jaw to open, and the eyes to move. While you could drive the eyes with one motor, having independent movement on each eye opens up possibilities of it having a glad eye, as well as being cross-eyed. This adds up to needing to control four motors.

A good chip for this is the PCA9685, which will not only control servo motors but will also fade LEDs. It has an I²C interface and 16 PWM (pulse-width modulation) control channels with a variety of driving options. The sole snag is that it is only available as a surface-mount part; therefore before you can use it, it needs to be soldered onto a breakout

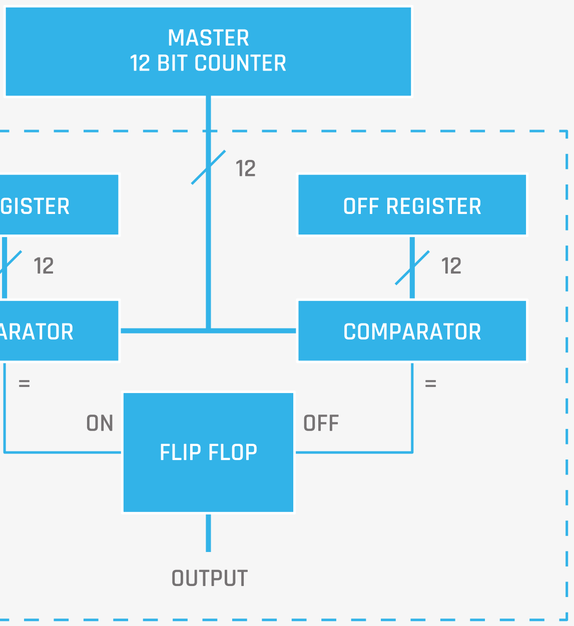
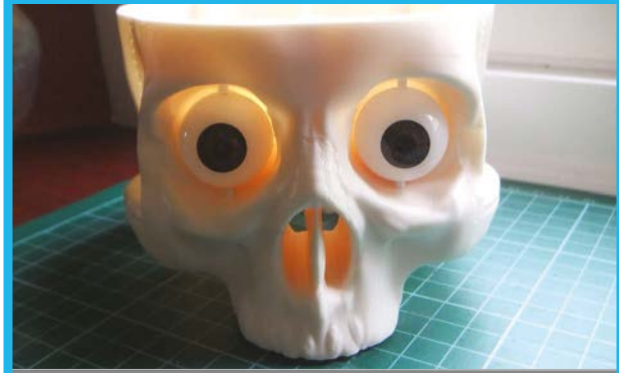


Fig. 1 PCA9685 principle of operation

SQUEAMISH READERS: LOOK AWAY NOW!

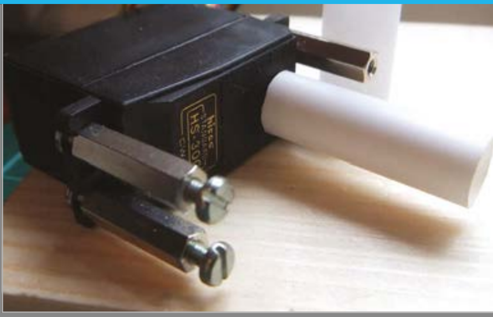


>STEP-01
The eyes

The exact details of the assembly depend on what skull model you have managed to get. The first thing we did was to change the eyes. These were designed to simply push into the eye sockets, allowing only rotation, not left/right movement. We therefore cut the retaining lugs off and drilled each eye so it could rotate vertically. We also drilled inside the skull so that an axle could be inserted vertically into each eye. This axle was glued in place on the eyeball, making sure that it was free to rotate in the eye socket. Finally, a servo horn was glued to the top of the axle. Next, a hole was cut above the nasal cavity; we did this by turning an 8mm drill by hand and, when breakthrough was achieved, we finished it off with a fine router bit in a handheld drill. This is to allow illumination of the nose. In retrospect, we should also have enlarged the holes behind the eyes to 8mm to allow more light through them.

>STEP-02
The neck

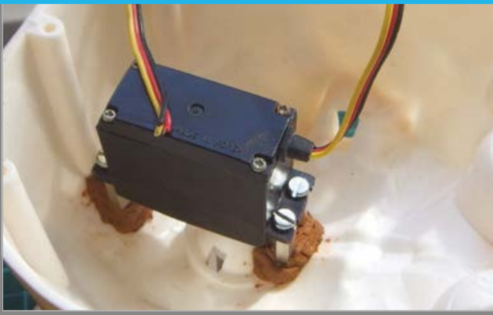
The large servo (type HS-300) was used for turning the head: the first thing to do is to fix a 70mm length of 7/16" (11.1mm) diameter styrene tube to the output shaft. This was done by cutting down one of the plastic horns and gluing it into the end of the tube. For added strength, we drilled two 1.3mm blind holes into the tube and horn, and fixed some 1mm styrene rod into it with aeroplane glue, sanding it flat when it had set. The hole in the base of the skull was enlarged to allow the tube to fit through.



Now the servo needs to be fastened to the skull: there is a difficulty here because there is nothing to fix it to. We wanted to avoid drilling holes in the skull that could be seen from outside, and in any case, there was nowhere flat to fix them.

>STEP-03
Fixing the neck servo

We added hex pillars to the servo's mounting lugs and fixed the base with Sugru. This is a relatively new substance that is great for fixing things, it is a sort of mouldable putty which sets overnight to a hard rubber. Drilling a few 1.3mm blind holes on the inside gives the Sugru something to grip onto. Screws were put into the back two pillars so that the servo was level, touching the inside of the skull, and then the base of the pillars was packed with a sachet of brown Sugru. We made sure the Sugru also protruded through three of



the holes around the neck for added grip. This gave a platform on which to build the rest of the fixtures.

The jaw had a lever on each side that acted as a spring to keep it shut. This was cut off with a sharp knife and a hole drilled in each side to allow it to pivot.

>>

“We use an RGB LED to illuminate each eye and also the nose”

board. However, if soldering is not your strong point, you can get one ready made up on eBay for little cost.

As the PCA9685 can handle 16 motors, and we only need four, the other twelve can be used for LEDs to light up the skull. Unfortunately, while this chip can drive both servo motors and LEDs, you have to resort to a trick to make it drive both at the same time. Essentially, this involves inverting the servo motor drives to allow the rest of the chip to be configured to drive LEDs; this requires a transistor on each output connected to a servo.

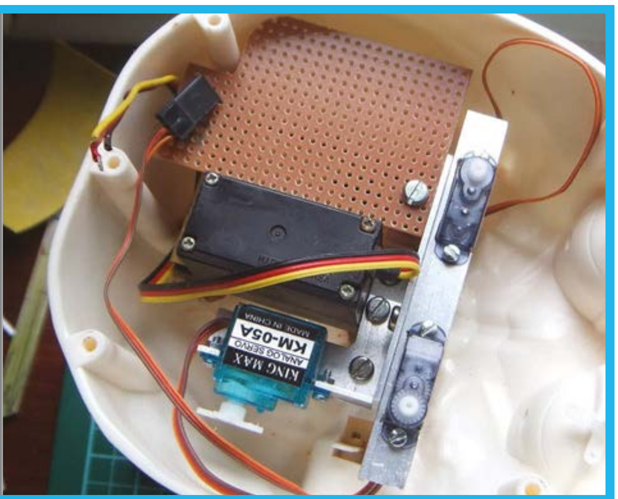
The PCA9685

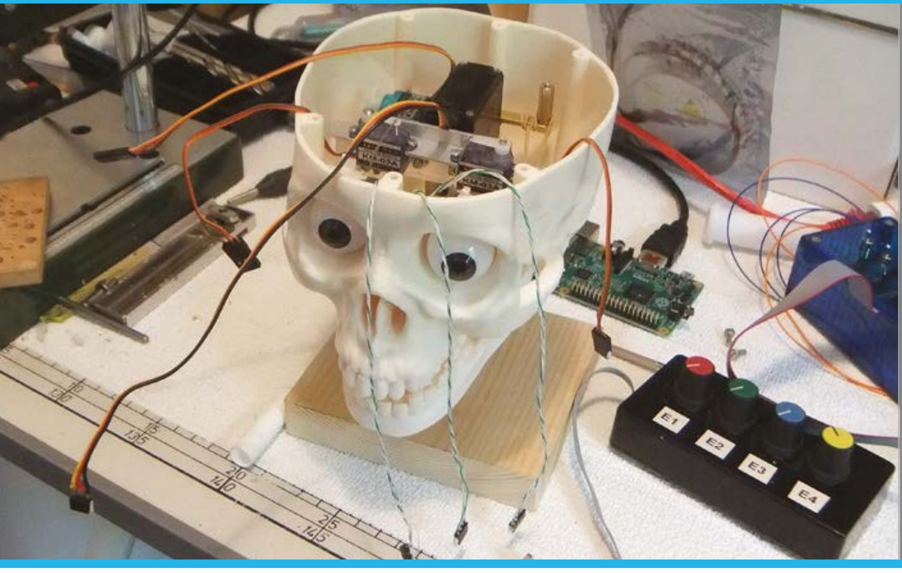
The PCA9685 is in essence a 12-bit counter with two registers for each channel. One pair of these registers determines the count when the channel output turns on, and another pair when it turns off. Hence, the width of the pulse can be altered in 4,095 steps. **Fig. 1** (page 59) shows the block diagram of one channel. A comparator is fed with the master count and the contents of a register; when these two are equal, there is a signal on the = output which toggles a flip-flop. There are 16 channels like this.

The frequency of clocking of the master counter is controlled by a divider register in the chip. In order to get the correct frame rate for a servo, this counter was set to cycle at a frequency of 50Hz. This gives a slight flicker to the LEDs, but we think it adds to the supernatural effect of Mulder. We use an RGB LED to illuminate each eye and also the nose, so in theory they can be set to any colour you want. The remaining three channels were connected to white LEDs to light the skull up from the inside.

>STEP-04
The servo brackets

The jaw was attached after fitting all the teeth, and the pivot hole was picked up and drilled through the mounting point on each side. Then a small styrene tab made from 1mm sheet material was attached with a hole at both ends. Next, a bracket was made to fit the two smaller servo motors (KM-03A) using two pieces of 10 by 24mm angle aluminium. The lower piece attached to the servo fixings and the upper piece was bolted on so that the small servos were level with the eye horns. A small bracket using one of the servo's fixing holes and another hole in the first bracket provided the fixing for the medium servo (KM-05A). An M2 nut and bolt were placed on both the jaw lever and this servo's horn. Finally, a short brass strip was fixed to the rear large servo's mounting holes, and a pillar was attached to provide support for the stripboard used for building the electronics. A second pillar was also fitted to the front brackets, and washers used so the two pillars reached the same height.

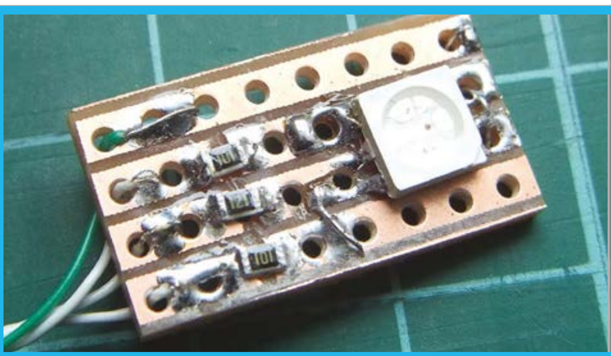




>STEP-05
The actuators

The jaw servo was attached to the jaw lever using four 20mm diameter elastic bands, and the eyes were attached to the eye servo horns with 0.5mm diameter tinned copper wire.

The neck servo's tube was wrapped round with two turns of insulation tape to make it fit snugly into a 20mm plastic pipe adaptor sold for plumbing. This in turn went into another plastic tube to make the total length 90mm. A bolt was drilled through the top two tubes so that they were fixed together. You could use glue, but then you would not be able to dismantle the assembly if you did. The electronics part was built up according to the schematic, with the PCA9685 being soldered onto a PCB adaptor board. We used 0.1" (2.5mm) pitch pin headers for the servo and LED connections.



>STEP-06

The LED board

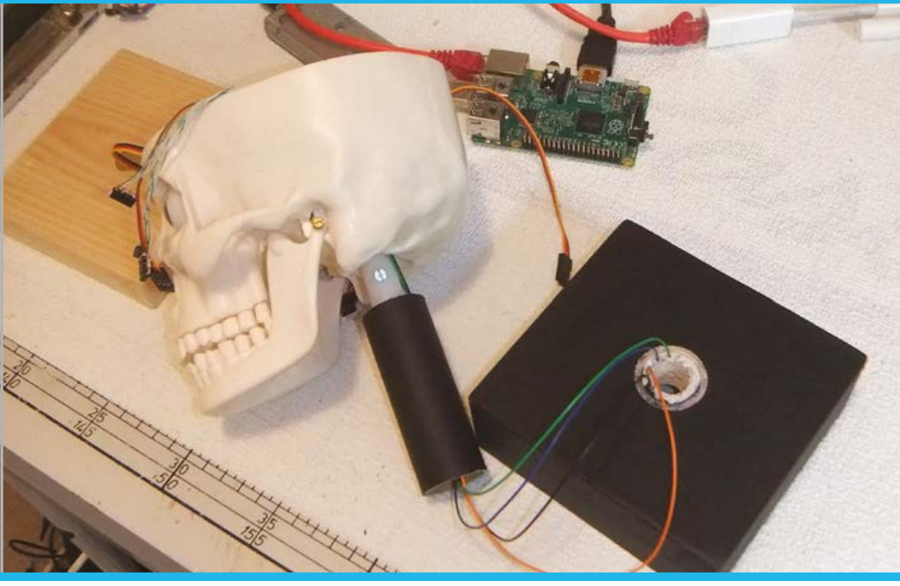
The three small LED boards were constructed using surface-mount components to make them small, and were fixed behind each eye and the nasal cavity with hot melt glue. Small pieces of opaque black paper were placed on top of each of the three LED boards to prevent light from leaking into the cranium. The three white cranium LEDs themselves were mounted on the main stripboard and were pointed upwards. The leads of the LEDs were covered in heat-shrink insulation and kept long at about 20mm, to allow some directional adjustment.

>STEP-07

The stand

A 120mm square base was made using some 19mm thick pine. This had a 28mm diameter 3mm groove cut into it with a saw drill, before the same saw drill bit was used to drill a 16mm hole all the way through to mount the neck tube. Four 1.3mm holes were drilled between the groove and the hole to allow the control wires to be pushed through, and the neck tube was covered by a 28mm cardboard tube taken from an empty roll of cling film. Four 22mm square strip pine pieces completed the base of the stand, and the whole thing was painted with a flat black paint.

Finally, a 0.5mm styrene sheet was used to cut a diffusing sheet used between the electronics and the rest of the cranium, although a sheet of paper worked nearly as well. This prevents spots appearing on the outside of the skull from the narrow-angle LEDs.



The circuit

The full circuit of Mulder is shown in **Fig. 2** (see page 62). Note the inductor filtering the power to the servo motors: this stops them interfering with the PCA9685. Its value is not too critical: basically, get as large a value as you physically have room for; you could even try making the project without one. The BC237 transistors are wired up in a way to cope with the open collector output configuration of the PCA9685 that is needed to drive the LEDs directly. The RGB LEDs were wired up on small pieces of stripboard. The construction was, at times, rather like building a ship in a bottle. We effectively made it so that all the connections to the board were made with plugs and sockets in the style of those fitted normally on servo motors. You will also need an external 5V supply: when we tried to use the Pi's internal supply, the low voltage warning indicator kept coming on with movement or high LED brightness. The final assembly is shown in **Fig. 3**. See the 'how to' boxes for a step-by-step rundown of the construction process.

Pi Control

When it comes to driving this with a Raspberry Pi, all we need to do is to set the registers of the PCA9685 with the right values for the LED's brightness or the servo's position, but there is a little twist. If you just set a servo to point at a specific angle, it will turn to that angle as fast as it can. This is often too fast to look good, so to get round it you have to move it in small increments with a delay in between. If the increments are small enough and the delay is short enough, it looks like the servos are moving slowly. Exactly the same is true of the LEDs when they are required to fade on or off. The code in **Mulder.py** (see page 63) controls Mulder from the keyboard and is written in the Pygame framework.

The code

In effect, the code simply monitors the keyboard for any key press and calls up the appropriate actions. The window is simply a fixed display of the keyboard commands. This is because, in Pygame, a window

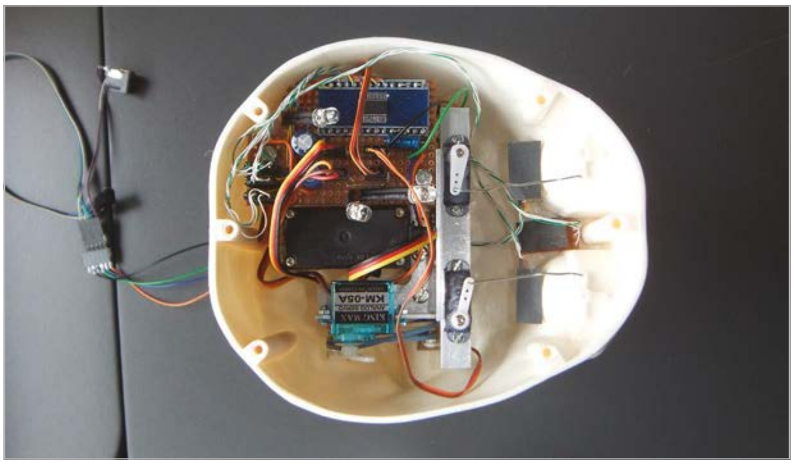


Fig. 3 A top-down view of the finished project

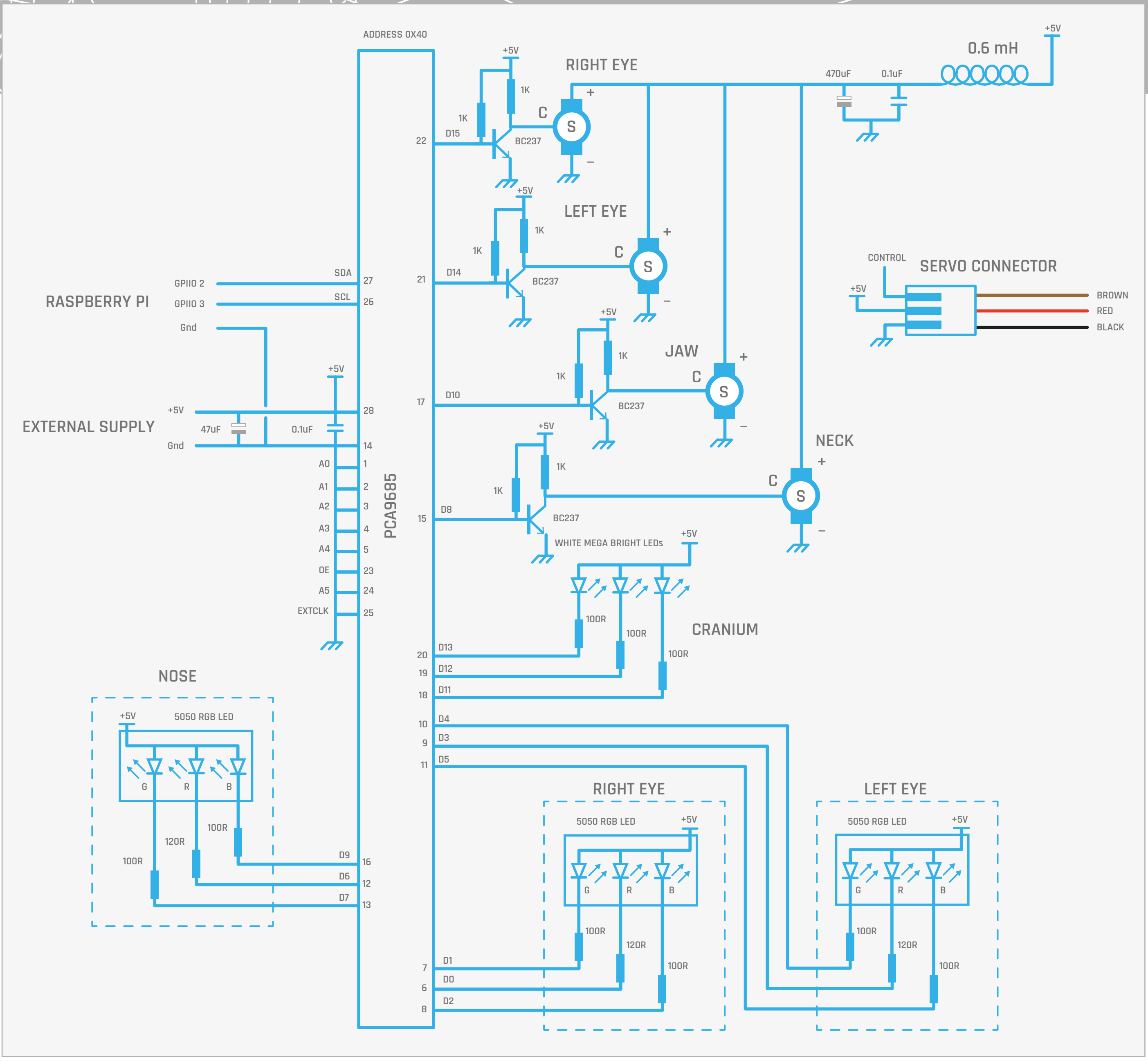


Fig. 2 The schematic of Mulder

is needed to act as a focus for all the key strokes, so you can't get away without any window at all. Each key initiates either a light change, a movement, or a sound; these are all implemented as non-blocking calls. That means you can start to fade the eyes up and, during that fade, make a movement or fade the nose lights. In other words, you don't have to wait for one action to finish before starting another. This is done by having a variable for the target or end state, another one for the present state, and a final one for an increment. Then, every 60ms, a function is called: first it checks to see if a fade has finished, that is if the present state is close to the end state; if not, it then increments the current state variable, and sets the hardware to this state. The only complication is that with the servos, there are four motors to keep track of, and in the LEDs there are four groups of lights, each one having three individual LEDs. This means the actual variables are lists, or lists of lists. Also, although the numbers being sent out to the hardware are all integers, in order to maintain accurate tracking, the increments and current positions need to be in floating point format.

Customisation

Like all Pi Bakery projects, there is ample scope for your own input. The next thing we are going to do is to combine Mulder with Willem Koopman's code from *The MagPi* #32, using OpenCV and face recognition to get Mulder to follow you with his eyes or neck. Another variation would be to put a speaker in the skull so you can make him the source of scary sounds.

Finally, if there are any kids out there wondering why he is called Mulder, just ask your parents.

Mulder.py

```
import pygame, time, random, sys, os
from smbus import SMBus
from copy import deepcopy

pygame.init()
os.environ['SDL_VIDEO_WINDOW_POS'] = 'center'
pygame.display.set_caption("Mulder")
screen = pygame.display.set_mode([424,430],0,32)
mulder = pygame.image.load("images/Mulder.png").
convert_alpha()
screen.blit(mulder,[0,0])
pygame.display.update()
pygame.mixer.quit()
pygame.mixer.init(frequency=22050, size=-16, channels=2,
buffer=512)
pygame.event.set_allowed(None)
pygame.event.set_allowed([pygame.KEYDOWN,pygame.QUIT])
soundNames = [
"laugh","scream","door","kill","thunder","owl"]
mulderSound = [ pygame.mixer.Sound(
"sounds/"+soundNames[effect]+".wav")
                  for effect in range(0,6)]
# command register addresses for the PCA9685 IC
PCA9685 = 0x40 # i2c address of PCA9685
bus = None

# right eye, left eye, jaw, neck
servos = [14, 15, 10, 8]
startPos = [269, 257, 180, 250]
white = [2480.0,4088.0,1608.0]
black = [0.0,0.0,0.0]
green = [0.0,3000.0,0.0]

# PCA9685 registers
lights = [(0,1,2),(3,4,5),(6,7,9),(11,12,13)]
lightPC = [[0.0,0.0,0.0],[0.0,0.0,0.0],[0.0,0.0,0.0],
[0.0,0.0,0.0]]
lightTC =  deepcopy(lightPC) # Light Target colour
lightInc = deepcopy(lightPC) # Light Increment
servoPP = deepcopy(startPos) # servo Present Position
servoInc = [0.0,0.0,0.0,0.0] # servo Increment
servoTP = deepcopy(startPos) # servo Target Position
stepsInFade = 40.0 # change to alter speed
stepsInMove = 40.0 # change to alter speed
fadeSpeed = 0.06 # change to alter speed
fading = False
facing = startPos[3]
maxNeck = 337
minNeck = 177

def main():
   print"Welcome to the world of Mulder"
   busInit()
   print"Press keys to make him work"
   while True: # repeat forever
      checkForEvent()
      # need to move?
      if not checkMove() :
         handleMove()
      if fading :  # need to fade?
         handleFade()
      if fading or not checkMove():
         # delay between steps
         time.sleep(fadeSpeed)

def busInit(): # start up I2C bus
   global bus
   try :
     bus = SMBus(1)
   except :
     print"start IDLE with 'gksudo
idle' from command line"
     sys.exit()

   # initialise the outputs on the PCA9685
   bus.write_byte_data(PCA9685,0,0x31) #sleep mode to 1
   #divider for 50Hz frame
   bus.write_byte_data(PCA9685,0xfe,132)
   # sleep mode set to 0
   bus.write_byte_data(PCA9685,0,0x21)
   bus.write_byte_data(PCA9685,0x01,0x18)
   beginners() # initial position

def beginners(): # initial position
   for r in range(0,4):
      setServo(r, startPos[r]) # move directly
   wipe() # lights out

def wipe(): # clear the lights
    for i in range(0,4):
       setLeds(i,[0,0,0])

def RestoreBeginners(): # restore initial position
   for r in range(0,4):
      setServoTarget(r, startPos[r])
   while not checkMove():
      handleMove()
      time.sleep(fadeSpeed)
   wipe() # lights out

def setServo(servo,pos): # move directly
   reg = (servos[servo] * 4)+6 #register start number
   regValues = [0,0,pos &0xff, pos >> 8]
   bus.write_i2c_block_data(PCA9685,reg,regValues)
def setServoTarget(motor, pos): # set new position
   servoTP[motor] = pos
   servoInc[motor] = (pos - servoPP[motor]) / stepsInMove

def setLeds(led,col):
   for c in range(0,3):
      #register number to start with
```

Language

>PYTHON 2.7

DOWNLOAD:
github.com/Grumpy-Mike/Mikes-Pi-Bakery/tree/master

PROJECT VIDEOS

Check out Mike's Bakery videos at: **vimeo.com/channels/mikespibakery**

```
      reg = (lights[led][c] * 4)+6
      regValues = [0,0,col[c] & 0xff, col[c] >> 8]
      #print reg,regValues
      bus.write_i2c_block_data(PCA9685,reg,regValues)

def setLEDfade(led,target): # set up an LED fade
   global lightPC,lightTC,lightInc, fading
   fading = True
   for i in range(0,3): # work out the increment
       lightTC[led][i] = target[i]
       lightInc[led][i] = (lightTC[led][i] -
lightPC[led][i])/stepsInFade

def handleMove():  # make one increment of all servos
   global servoPP
   for motor in range(0,4):
      if abs(servoTP[motor] - servoPP[motor]) >= 1.0 :
         if abs(servoTP[motor] - servoPP[motor]) >=
abs(servoInc[motor]):
            servoPP[motor] += servoInc[motor]
         else:
            servoPP[motor] = servoTP[motor]
         setServo(motor,int(servoPP[motor]))
      else:
         servoPP[motor] = servoTP[motor]

def handleFade(): # make one increment of all fades
   global lightPC, fading
   fading = False # global of fading in progress
   for led in range(0,4):
      if checkFade(led) == False :
         fading = True
         for i in range(0,3): # work out new colour
            if abs(lightPC[led][i] - lightTC[led][i]) \
>= abs(lightInc[led][i]) :
               # only change if we are not at target
               lightPC[led][i] += lightInc[led][i]
               if lightPC[led][i] < 1.1:
                  lightPC[led][i] = 0.0
            else :
               lightPC[led][i] = lightTC[led][i]
            updateLED(led)

def updateLED(led):
    for c in range(0,3): # write LED values to PCA9685
      #register number to start with
      reg = (lights[led][c] * 4)+6
      regValues = [0,0,int(lightPC[led][c]) & 0xff,
int(lightPC[led][c]) >> 8]
      bus.write_i2c_block_data(PCA9685,reg,regValues)

def checkMove():
   done = True
   for i in range(0,4):
      if servoTP[i] != servoPP[i]:
         done = False
   return done

def checkFade(led):
   done = True
   for i in range(0,3):
      if abs(lightPC[led][i] - lightTC[led][i]) >= 1.0 :
         done = False
   return done

def terminate(): # close down the program
    print "Closing down please wait"
    RestoreBeginners() # move motors back
    pygame.mixer.quit()
    pygame.quit() # close pygame
    os._exit(1)

def checkForEvent(): # keyboard commands
    global facing
    event = pygame.event.poll()
    if event.type == pygame.QUIT :
         terminate()
    if event.type == pygame.KEYDOWN :
       if event.key == pygame.K_ESCAPE :
          terminate()
       # sounds
       if event.key == pygame.K_h :
          mulderSound[0].play()
       if event.key == pygame.K_j :
          mulderSound[1].play()
       if event.key == pygame.K_k :
          mulderSound[2].play()
       if event.key == pygame.K_l :
          mulderSound[3].play()
       if event.key == pygame.K_SEMICOLON :
          mulderSound[4].play()
       if event.key == pygame.K_QUOTE :
          mulderSound[5].play()
       # Lights  - eyes
       if event.key == pygame.K_q : # eyes off
          setLEDfade(0,black)
          setLEDfade(1,black)
       if event.key == pygame.K_w : # eyes white
          setLEDfade(0,white)
          setLEDfade(1,white)
       if event.key == pygame.K_e : # eyes blue
          setLEDfade(0,[0,0,1000])
          setLEDfade(1,[0,0,1000])
       if event.key == pygame.K_r : # eyes red
          setLEDfade(0,[400,0,0])
          setLEDfade(1,[400,0,0])
       if event.key == pygame.K_t : # eyes green
          setLEDfade(0,green)
          setLEDfade(1,green)
       if event.key == pygame.K_y : # right eye white
          setLEDfade(0,white)
          setLEDfade(1,black)
```

```
if event.key == pygame.K_u : # right eye red
   setLEDfade(0,[800,0,0])
# Lights  - nose
if event.key == pygame.K_z : # nose off
   setLEDfade(2,black)
if event.key == pygame.K_x : # nose white
   setLEDfade(2,white)
if event.key == pygame.K_c : # nose blue
   setLEDfade(2,[0,0,1000])
if event.key == pygame.K_v : # nose red
   setLEDfade(2,[1000,0,0])
if event.key == pygame.K_b : # nose green
   setLEDfade(2,green)
if event.key == pygame.K_n : # nose yellow
   setLEDfade(2,[1000,1000,0])
if event.key == pygame.K_m :
   setLEDfade(2,[1000,300,0])# nose orange

# Lights  - cranium
if event.key == pygame.K_a : # cranium off
   setLEDfade(3,black)
if event.key == pygame.K_s : # cranium dim
   setLEDfade(3,[100,100,100])
if event.key == pygame.K_d : # cranium medium
   setLEDfade(3,[600,600,600])
if event.key == pygame.K_f : # cranium bright
   setLEDfade(3,[4000,4000,4000])

# Movement
if event.key == pygame.K_o :  # jaw open
   setServo(2,180)
   servoPP[2]= 180

if event.key == pygame.K_p :  # jaw closed
   setServo(2,336)
   servoPP[2] = 336

if event.key == pygame.K_1 : # eyes forward
   setServoTarget(0,269)
   setServoTarget(1,257)

if event.key == pygame.K_2 : # eyes left
   setServoTarget(0,233)
   setServoTarget(1,192)

if event.key == pygame.K_3 : # eyes right
   setServoTarget(0,322)
   setServoTarget(1,298)

if event.key == pygame.K_4 : # right eye forward
   setServoTarget(1,257)
if event.key == pygame.K_5 : # right eye left
   setServoTarget(1,192)

if event.key == pygame.K_6 : # right eye right
   setServoTarget(1,306)
```

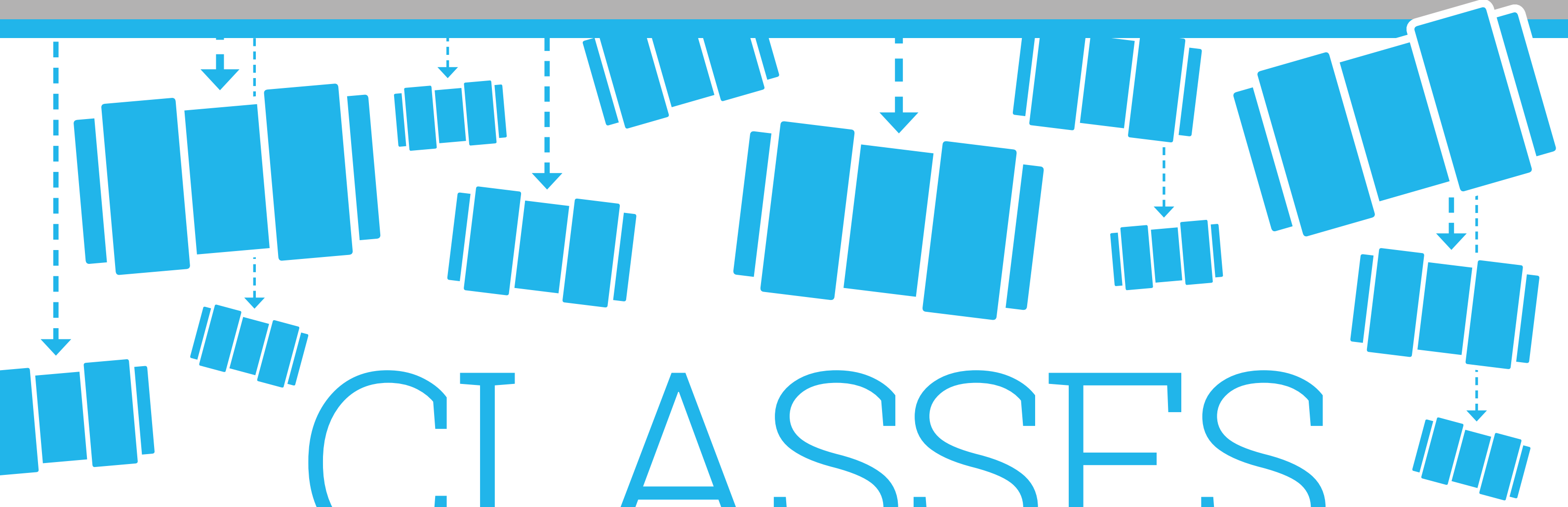
```
      if event.key == pygame.K_7 : # left eye forward
         setServoTarget(0,269)

      if event.key == pygame.K_8 : # left eye left
         setServoTarget(0,233)

      if event.key == pygame.K_9 : # left eye right
         setServoTarget(0,329)

      if event.key == pygame.K_UP : # face the front
         facing = startPos[3]
         setServoTarget(3,facing)

      if event.key == pygame.K_DOWN : # stop movement
         facing = servoPP[3]
         setServoTarget(3,facing)

      if event.key == pygame.K_LEFT :
         facing += 20
         if facing > maxNeck: # limit left move
            facing = maxNeck
         setServoTarget(3,facing)

      if event.key == pygame.K_RIGHT :
         facing -= 20
         if facing < minNeck: # limit left move
            facing = minNeck
         setServoTarget(3,facing)

      if event.key == pygame.K_PERIOD : # max right move
         facing = minNeck
         setServoTarget(3,facing)

      if event.key == pygame.K_COMMA : # max left move
         facing = maxNeck
         setServoTarget(3,facing)

         #debug
      if event.key == pygame.K_0 :
         print"Debug fading",fading
         print "present",lightPC
         print "target",lightTC
         print "increment",lightInc

         print"Debug moving",( not checkMove())
         print "present position",servoPP
         print "target",servoTP
         print "increment",servoInc

# Main program logic:
if __name__ == '__main__':
    main()
```

PART 8

MAKE GAMES WITH PYTHON

SEAN M TRACEY

Sean is a technologist living in the South West of England. He spends most of his time making silly things with technology.
sean.mtracey.org



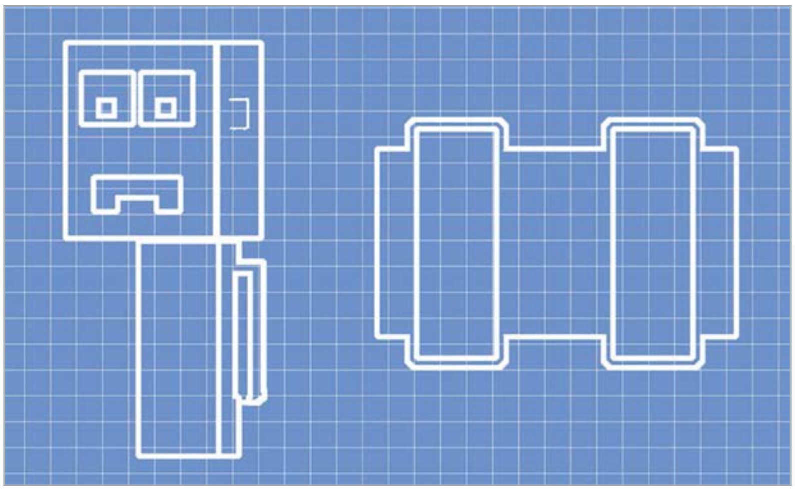
CLASSES

In the eighth part of this series, we make a simple barrel-dodging game using classes

The end is nigh! (And I told you so...)

This is part eight of ten of this series, which you can look at in two ways: we're coming to the end of this series (boo!) or we still have three opportunities to learn and make something great together (yay!), so let's make the best of it. We've put together so much over the last couple of months. We've learned all about how Pygame draws shapes and images, how we can manipulate sounds and control events with our keyboards and mouse, we've made buttons and start screens and life bars and floors that travel too quickly – people, we even built a solar system with gravity and everything! It's been a ride, and it's all been leading up to one thing, a final game, which we'll be making in the last two parts of this series. That gives us one last chance to round up all of the loose ends and learn everything we need to make that game in this tutorial, but what more could we possibly have to learn about? Well, in this piece we're going to cover Python classes. We know that doesn't sound as exciting as 'We're going to make a solar system!' but it's a super-important part of making games – especially in Python. We won't be looking too closely at the code for this piece too much; instead, we'll look at the structures used to make the game work, using the game code to demonstrate the principle. If you just want to work through the code and get it running for fun, though, all of the code and images are on GitHub (**github.com/seanmtracey/Games-with-Pygame**).

Below Think of classes as a blueprint for things that can be made and used many times

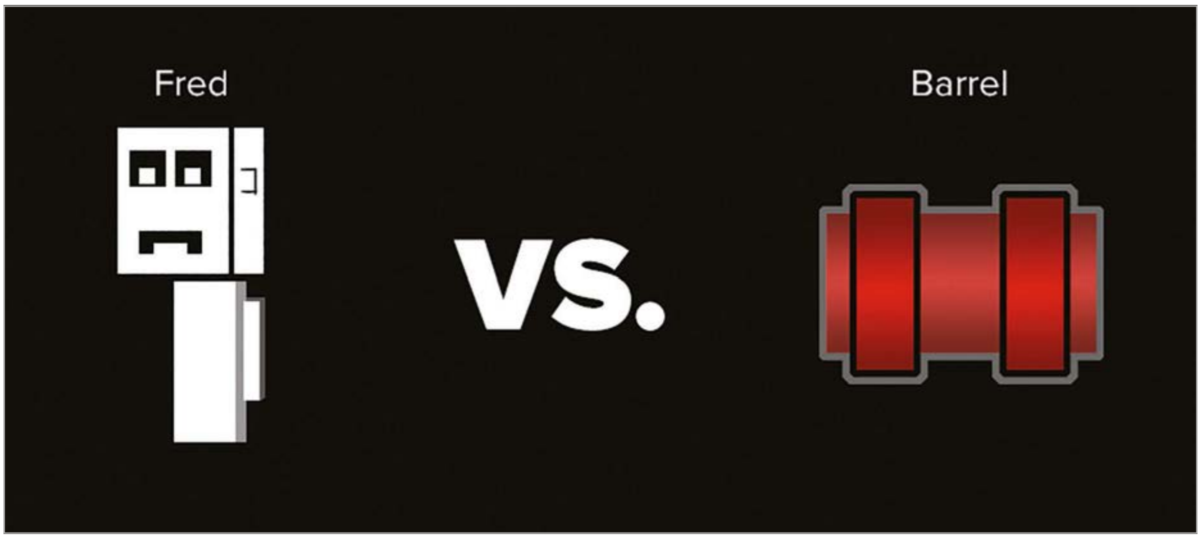


What is a class?

Throughout this series, we've come across a ton of different data types such variables, lists and dictionaries. Each data type exists to help us organise and store data in a way that makes it easy for us to reuse and reference data throughout our games.

A class is another such structure that helps us to organise our code and objects. Classes are designed to be a kind of blueprint for bits of code that might need to be reused again and again. Let's compare and contrast classes with dictionaries. To make a dictionary that, let's say, describes a rectangle, we could write something like this:

```
aRectangle = {
                    "x" : 5,
                    "y" : 10,
                    "width" : 20
                    "height" : 30
          }
```



That would certainly do the job; we have everything we need to make a rectangle: width, height, x, and y. It's great, but what if we want to make another rectangle? Well, that's easy, we can just write it again:

```
rectangleOne = {
      "x" : 5,
      "y" : 10,
      "width" : 20
      "height" : 30
}

rectangleTwo = {
      "x" : 5,
      "y" : 10,
      "width" : 20
      "height" : 30
}
```

But that's a little messy. Instead, we could create a function to make rectangles for us:

```
def rectangleMaker(width, height, x, y):

      return {
              "x" : x,
              "y" : y,
              "width" : width,
              "height" : height
      }

rectangleOne = rectangleMaker(20, 15, 30, 50)
rectangleOne = rectangleMaker(10, 35, 40, 70)
```

That's better, but in order to make rectangles a more convenient thing to create quickly, we've had to write a function which builds one and passes the new 'rectangle' (it's not really a rectangle, it's a dictionary describing something that *could* be a rectangle) back to whatever bit of code wanted it. Great, job done, and that's how we've done things so far, but it's not very semantic. Classes do all that we just looked at and more: they can keep track of themselves and their properties; they can contain functions that can execute code that affects the properties in a clever way, rather than having to trigger things manually. Classes can also be 'extended'; that is, a class can be made up of several other classes to make something with the properties and abilities of all the other classes before it. If we were to make squares with a class, we would first define what a rectangle is with a class:

```
class Rectangle():

  x = 0
  y = 0
  width = 0
  height = 0

  def __init__(self, x, y, width, height):
    self.x = x
    self.y = y
    self.width = width
    self.height = height
```

Our class **Rectangle** has four properties – x, y, width, and height – just like our dictionaries, but it also has a function, **__init__**. This is a special function; when we want to create a new rectangle, we simply call **Rectangle** and pass it the values we want to use into it:

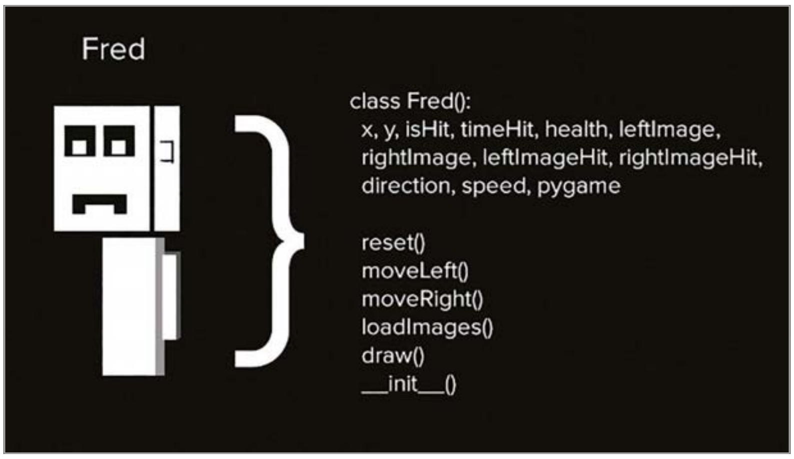
```
rectangleOne = Rectangle(0, 5, 20, 30)
rectangleTwo = Rectangle(0, 5, 20, 30)
```

When we call **Rectangle** like this, we are triggering something called 'instantiation'. In simplest terms, instantiation means we're creating something new from a blueprint (our class) that can operate independently from other objects in our code. When we instantiate a new **Rectangle**, that special function **__init__** will be called and will receive the variables we pass through to it. There's something a little different here, though: our **__init__** function is expecting five arguments to be passed to it, but we only passed four... and Python didn't complain. Why is this? When we instantiate a new **Rectangle**, **self** gets passed through to the **__init__** function by the class itself and it refers to its 'self'. With **self**, we can create as many **Rectangles** as we like and have functions inside of the class reference itself, set values, and run code that only effects itself. It's this useful property that makes classes far more useful than standard dictionaries. We can also reference the properties and functions of each instance of our **Rectangles** using '**.**' instead of having to use **[]**.

This is Fred

So, now we have a grip on what a class is – it's a blueprint full of code and variables that can be used many times across our Python projects. This makes them great for games, because we can create and control objects dynamically as and when we need them, rather than having to specify everything by hand, like clouds or cars or buildings or trees or any other object which has variety. That said, this doesn't mean we can only use classes when we expect to make more than one of a thing.

Below This is Fred. He's our game avatar and he's made up of a class. There's only one Fred



```
class Fred():
  x, y, isHit, timeHit, health, leftImage,
  rightImage, leftImageHit, rightImageHit,
  direction, speed, pygame

  reset()
  moveLeft()
  moveRight()
  loadImages()
  draw()
  __init__()
```

Fred is our game avatar. He works from nine to five in a barrel factory that, frankly, flaunts health and safety regulations regarding the storing of barrels in overhead containers. Fred is a simple fella, so much so that we can describe everything about Fred and all he does in a Python class – but there's only one Fred; nobody else would ever agree to the monotonous labour of the barrel factory. Fred is a one-off.

```
class Fred():

  # Fred's preset variables
  x = 0
  y = 625

  isHit = False
  timeHit = 0
  health = 100

  leftImage = None
  rightImage = None
  leftImageHit = None
  rightImageHit = None

  direction = 1
  speed = 8
  pygame = None

  def reset(self, x):
    # Code for getting Fred ready for
    # another day of dodging barrels

  def moveLeft(self, leftBound):
    # Move Fred to the left

  def moveRight(self, rightBound):
    # Move Fred to the right

  def loadImages(self, pygame):
    # Get the image we need to draw Fred

  def draw(self, surface, time):
    # Draw Fred

  def __init__(self, x, pygame,
surface): # Create Fred and acquaint
          # him with himself
```

'We are all stardust'... except for Fred. He's a class; this is the blueprint for Fred's existence and this is him at his most basic. Like we've said, classes are great when you need to control loads of the same but slightly different things, but classes are great for abstracting (hiding away) bits of code that we use a lot but aren't useful everywhere and are used all of the time. So how do we create Fred? Our **Fred** class lives in the **objects.py** file of our projects, so on line 5 of **freds_bad_day.py**, we import objects. This file contains our **Fred** and our **Barrel** class (which we'll look at in a little bit). Once we've imported the **objects** file, we can create Fred with…

```
Fred = objects.Fred(windowWidth / 2)
```

The argument we passed through will set Fred's x coordinate – we always want Fred to start in the middle of the screen, so that's what we've passed through here. Now 'Fred' has been instantiated, using the **Fred** variable, we can access all of Fred's properties and methods. If we want to know where Fred is now, we can simply call:

```
Fred.x
> 500
```

If we want to move Fred to the left, we can just **+=** the **x** property:

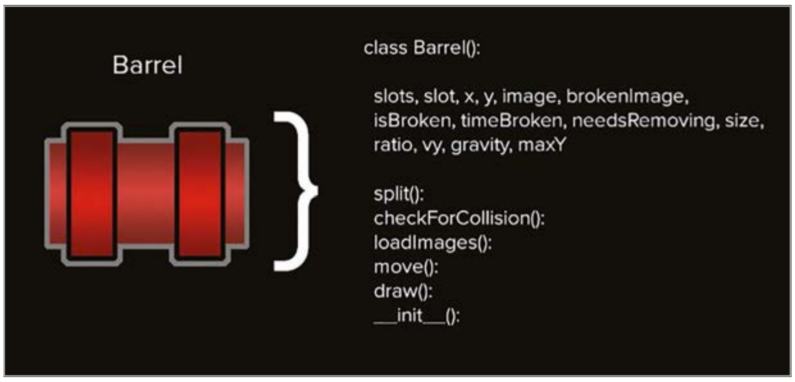
```
Fred.x += 5
```

But wait! Fred has a method called **moveLeft()**; instead of manually setting the x coordinate, we can call the **moveLeft** method instead. This looks tidier, makes it obvious what's happening when the code is being read by somebody else, and lets us do more than one thing when we move Fred to the left:

```
Fred.moveLeft()
```

Instead of adding an arbitrary number, **moveLeft()** checks Fred's **speed** property and adds that to Fred's **x** property. It also checks what direction Fred is moving in and sets the **direction** property accordingly. This means when it comes to drawing Fred, we can draw him facing the correct direction without having to manually check which direction he's facing every time.

```
def moveLeft(self, leftBound):

    if self.direction is not 0:
      self.direction = 0

    if((self.x - self.speed) > leftBound):
      self.x -= self.speed
```



class Barrel():

slots, slot, x, y, image, brokenImage, isBroken, timeBroken, needsRemoving, size, ratio, vy, gravity, maxY

split():
checkForCollision():
loadImages():
move():
draw():
__init__():

Above This is a barrel. There are many like it, but this one is ours

It's simple to use, but it saves us a lot of trouble. In our **freds_bad_day.py** file, the code that actually affects Fred is four lines long:

```
Fred.draw()
Fred.moveLeft()
Fred.moveRight()
Fred.loadImages()
```

That's much simpler than having functions in our code that *could* be accessed by any function but really only need to be used for one object.

This is Fred's nemesis

'Do you know what nemesis means? A righteous infliction of retribution manifested by an appropriate agent.' The common barrel is the blight of Fred's life. He spends his day shift running left to right and back again, cursing whichever middle-manager it was who thought storing a seemingly unlimited supply of barrels 20 feet above the ground would be a peril-free idea. Unlike Fred, there are many barrels, but at their core, they're both the same. **Fred** and **Barrel** are both classes, but **Fred** is only instantiated once, whereas our **Barrel** is instantiated potentially hundreds of times (depending on how bad Fred's day is).

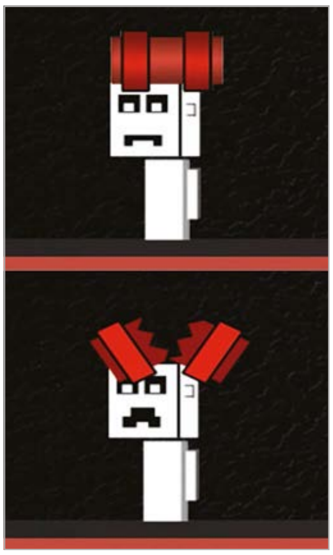
```
class Barrel():

    slots = [(4, 103), (82, 27), (157, 104), (234, 27),
(310, 104), (388, 27), (463, 104), (539, 27), (615, 104),
(691, 27), (768, 104), (845, 27), (920, 104)]
   slot = 0
   x = 0
   y = 0

   image = None
   brokenImage = None

   isBroken = False
```

```
timeBroken = 0

needsRemoving = False

size = [33,22]
ratio = 0.66

vy = 1.5
gravity = 1.05
maxY = 20

def split(self, time):
  self.isBroken = True
  self.timeBroken = time
  self.vy = 5
  self.x -= 10

def checkForCollision(self, fred):
    # Check whether barrel is colliding


def loadImages(self, pygame):
    # Load the images for Fred
```

BASH! CRASH! THUMP!

Much to Fred's dislike, there's more than one barrel in the world, and we need a place to keep track of them. A new barrel is created after a certain amount of time passes (this amount of time gets smaller as the game progresses), and we append that barrel to the **Barrels** list at the top of our **freds_bad_day.py** file. Every time our 'main' loop works through itself, we iterate through the **Barrels** list, move the barrels, and check if they've hit Fred.

Below A barrel splitting when it hits Fred



We do this with each barrel's **checkForCollision()** method. Our barrel doesn't care about what it's hitting unless it's Fred – our (some would say malicious) barrel really wants to know when it's hit Fred, otherwise how will it know to split in half? One of the cool things about **Fred** (and classes as a whole) is that once we've instantiated them, we can pass that reference around our code to other classes and functions as we like. When we call **checkForCollisions()**, we pass **Fred** as an argument. This way, the barrel can check it current position and compare it to Fred's. If there's a collision, the barrel will return **True**, then our main loop can take over and split our barrel in half, reduce Fred's health bar, and tell Fred that he should frown – providing Fred is still standing, otherwise the main loop will end the game.

```
  hasCollided = barrel.
checkForCollision(Fred);

 if hasCollided is True:
   barrel.split(timeTick)
   Fred.isHit = True
   Fred.timeHit = timeTick
   if Fred.health >= 10:
     Fred.health -= 10
   else :
      gameOver = True
      gameFinishedTime = timeTick
```

The cleanup

When our barrel has accomplished its dastardly life deed and hits poor Fred, it splits in two and continues to fall off the screen. When our barrel goes off screen, we should really delete it, because we don't need it any more and it's eating up our Raspberry Pi's resources. What would be ideal is if our barrel could self-destruct, as it were, and remove itself from our game, but our barrel isn't keeping track of itself inside the game state; the game is keeping track of the barrel in the game and the barrel is looking after itself. This means that once our barrel has split in two, we need a way to reference it so we can delete it. If we just delete the barrel at the index it's at in the **Barrels** list, all of our remaining barrels will flash while they shift position in the list. Instead, we add the barrel's index to the **barrelsToRemove** list and once we've finished drawing all of our barrels, we remove all the split barrels before they're drawn again. No mess, just smooth cleanup.

```
  def move(self, windowHeight):
      # Move our barrel

  def draw(self, surface, pygame):
      # Draw our barrel

  def __init__(self, slot):
      # Create a barrel and slot it in the
       # right place
```

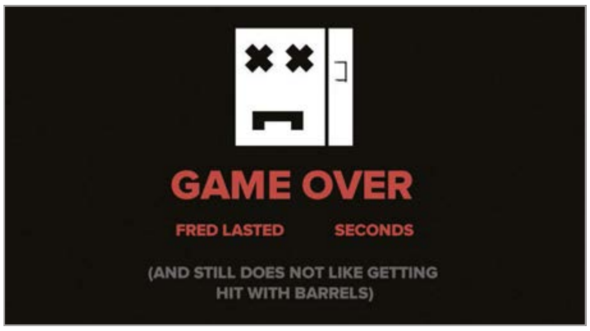
The **Barrel** class is simpler than our **Fred** class; this is because it needs to do less. Our barrel only needs to move down and accelerate. When we instantiate **Fred**, we passed through the x coordinate that we wanted Fred to start off from. We don't want our barrels to be able to appear just anywhere along the x/y axis of our game; instead, we want them to appear in one of the 13 slots at the top of our game. Our **Barrel** class has a **slots** list that contains x and y coordinates for each of the barrel holes at the top of our game window. When we want to create a barrel, we don't pass the x coordinate; instead, we pass a random integer between 0 and 12 (remember, computers count from 0, so 0 is the first item in a list) and the coordinates for that slot will become the location for that barrel. All of this happens in that handy **__init__** function.

```
if barrel.needsRemoving is True:
    barrelsToRemove.append(idx)

for index in barrelsToRemove:
  del Barrels[index]
```

Recap

There's been a lot to take in for this part, so let's have a recap before we start looking forward to the next two parts (we're going to be a building a space shooter, by the way). So, from the top... A class is like a blueprint of code for an object that we want to use. We can use classes once or thousands of times. Using classes helps us keep our code tidy and reusable. When we use a class, we create a new instance of that class which is called 'instantiation'. Each instance of a class can reference itself and so long as it's in the correct scope, any other object can read the properties of our class and can trigger the methods that the class may have. Each method has access to a reference to itself when it's called, even if nothing is passed; this reference will be the first argument passed, but we can pass as many arguments as we want to use, just like any other method.

The next two parts of the series are also our last, but don't despair: we're going to use absolutely everything we've learned so far to make a super-cool space shooter game. There's gonna be lasers and spaceships and gravity and ZAPZAPZAP sounds everywhere. You're gonna love it.



Freds_bad_day.py

```
import pygame, sys, random, math
import pygame.locals as GAME_GLOBALS
import pygame.event as GAME_EVENTS
import pygame.time as GAME_TIME
import objects

windowWidth = 1000
windowHeight = 768

pygame.init()
pygame.font.init()
surface = pygame.display.set_mode((windowWidth,
windowHeight), pygame.FULLSCREEN)

pygame.display.set_caption('Fred\'s Bad Day')
textFont = pygame.font.SysFont("monospace", 50)

gameStarted = False
gameStartedTime = 0
gameFinishedTime = 0
gameOver = False

startScreen = pygame.image.load("assets/startgame.png")
endScreen = pygame.image.load("assets/gameover.png")

background = pygame.image.load("assets/background.png")
Fred = objects.Fred(windowWidth / 2)
Barrels = []
lastBarrel = 0
lastBarrelSlot = 0
barrelInterval = 1500

goLeft = False
goRight = False

def quitGame():
  pygame.quit()
  sys.exit()

def newBarrel():
  global Barrels, lastBarrel, lastBarrelSlot

  slot = random.randint(0, 12)

  while slot == lastBarrelSlot:
      slot = random.randint(0, 12)

  theBarrel = objects.Barrel(slot)
  theBarrel.loadImages(pygame)

  Barrels.append(theBarrel)
  lastBarrel = GAME_TIME.get_ticks()
  lastBarrelSlot = slot

Fred.loadImages(pygame)

# 'main' loop
while True:

```

```
timeTick = GAME_TIME.get_ticks()

if gameStarted is True and gameOver is False:

  surface.blit(background, (0, 0))

  Fred.draw(surface, timeTick)

  barrelsToRemove = []

  for idx, barrel in enumerate(Barrels):
    barrel.move(windowHeight)
    barrel.draw(surface, pygame)

    if barrel.isBroken is False:

      hasCollided = barrel.checkForCollision(Fred)

      if hasCollided is True:
        barrel.split(timeTick)
        Fred.isHit = True
        Fred.timeHit = timeTick
        if Fred.health >= 10:
          Fred.health -= 10
        else :
         gameOver = True
         gameFinishedTime = timeTick

    elif timeTick - barrel.timeBroken > 1000:

      barrelsToRemove.append(idx)
      continue

    if barrel.needsRemoving is True:
      barrelsToRemove.append(idx)
      continue

  pygame.draw.rect(surface, (175,59,59), (0, windowHeight - 10, (windowWidth / 100) * Fred.health, 10))

  for index in barrelsToRemove:
    del Barrels[index]

  if goLeft is True:
    Fred.moveLeft(0)

  if goRight is True:
    Fred.moveRight(windowWidth)

elif gameStarted is False and gameOver is False:
    surface.blit(startScreen, (0, 0))

elif gameStarted is True and gameOver is True:
         surface.blit(endScreen, (0, 0))
         timeLasted = (gameFinishedTime -
```

Objects.py

```
class Fred():

  x = 0
  y = 625

  isHit = False
  timeHit = 0
  health = 100

  leftImage = None
  rightImage = None
  leftImageHit = None
  rightImageHit = None

  direction = 1
  speed = 8
  pygame = None

  def reset(self, x):
    self.x = x
    self.y = 625

    self.isHit = False
    self.timeHit = 0
    self.health = 100

    self.direction = 1
    self.speed = 8
    self.pygame = None

  def moveLeft(self, leftBound):

    if self.direction is not 0:
      self.direction = 0

    if((self.x - self.speed) > leftBound):
      self.x -= self.speed

  def moveRight(self, rightBound):

    if self.direction is not 1:
      self.direction = 1

    if((self.x + self.speed) + 58 < rightBound):
      self.x += self.speed

  def loadImages(self, pygame):
    self.leftImage = pygame.image.load("assets/Fred-Left.png")
    self.rightImage = pygame.image.load("assets/Fred-Right.png")
    self.leftImageHit = pygame.image.load("assets/Fred-Left-Hit.png")
    self.rightImageHit = pygame.image.load("assets/Fred-Right-Hit.png")

  def draw(self, surface, time):

    if time - self.timeHit > 800:
      self.timeHit = 0
      self.isHit = False

    if self.direction is 1:
      if self.isHit is False:
        surface.blit(self.rightImage, (self.x, self.y))
      else :
        surface.blit(self.rightImageHit, (self.x, self.y))
    else :
      if self.isHit is False:
        surface.blit(self.leftImage, (self.x, self.y))
      else :
        surface.blit(self.leftImageHit, (self.x, self.y))
```

Language

>PYTHON

DOWNLOAD:
github.com/seanmtracey/Games-with-Pygame

```
gameStartedTime) / 1000

           if timeLasted < 10:
                timeLasted = "0" + str(timeLasted)
           else :
                timeLasted = str(timeLasted)

           renderedText = textFont.render(timeLasted, 1, (175,59,59))
           surface.blit(renderedText, (495, 430))

  # Handle user and system events
  for event in GAME_EVENTS.get():

    if event.type == pygame.KEYDOWN:

      if event.key == pygame.K_ESCAPE:
        quitGame()
      elif event.key == pygame.K_LEFT:
        goLeft = True
        goRight = False
      elif event.key == pygame.K_RIGHT:
        goLeft = False
        goRight = True
      elif event.key == pygame.K_RETURN:
       if gameStarted is False and gameOver is False:
           gameStarted = True
           gameStartedTime = timeTick
       elif gameStarted is True and gameOver is True:
           Fred.reset(windowWidth / 2)

           Barrels = []
           barrelInterval = 1500

           gameOver = False

  if event.type == pygame.KEYUP:

    if event.key == pygame.K_LEFT:
      goLeft = False
    if event.key == pygame.K_RIGHT:
      goRight = False

    if event.type == GAME_GLOBALS.QUIT:
      quitGame()

  pygame.display.update()

  if GAME_TIME.get_ticks() - lastBarrel > barrelInterval \
and gameStarted is True:
    newBarrel()
    if barrelInterval > 150:
      barrelInterval -= 50

```

```

  def __init__(self, x):
    self.x = x

class Barrel():

  slots = [(4, 103), (82, 27), (157, 104), (234, 27), (310, 104), (388, 27), (463, 104), (539, 27), (615, 104), (691, 27), (768, 104), (845, 27), (920, 104)]
  slot = 0
  x = 0
  y = 0

  image = None
  brokenImage = None

  isBroken = False
  timeBroken = 0
  needsRemoving = False

  size = [33,22]
  ratio = 0.66

  vy = 1.5
  gravity = 1.05
  maxY = 20

  def split(self, time):
    self.isBroken = True
    self.timeBroken = time
    self.vy = 5
    self.x -= 10

  def checkForCollision(self, fred):

    hitX = False
    hitY = False

    if fred.x > self.x and fred.x < self.x + 75:
      hitX = True
    elif fred.x + 57 > self.x and fred.x + 57 < self.x + 75:
      hitX = True
    if fred.y + 120 > self.y and fred.y < self.y:
      hitY = True
    elif fred.y < self.y + 48:
      hitY = True
    if hitX is True and hitY is True:
      return True

  def loadImages(self, pygame):
    self.image = pygame.image.load("assets/Barrel.png")
    self.brokenImage = pygame.image.load("assets/Barrel_break.png")

  def move(self, windowHeight):

    if self.vy < self.maxY:
      self.vy = self.vy * self.gravity
    self.y += self.vy

    if self.y > windowHeight:
      self.needsRemoving = True

  def draw(self, surface, pygame):
    if self.isBroken is True:
      surface.blit(self.brokenImage, (self.x, self.y))
    else :
      surface.blit(self.image, (self.x, self.y))

  def __init__(self, slot):
    self.slot = slot
    self.x = self.slots[slot][0]
    self.y = self.slots[slot][1] + 24

```

π))) Sonic Pi PART 3

SAM AARON
Sam is the creator of Sonic Pi. By day he's a Research Associate at the University of Cambridge Computer Laboratory; by night he writes code for people to dance to.
sonic-pi.net

SYNTH RIFFS

Sam Aaron, creator of Sonic Pi, continues his series by looking at synth riffs...

You'll Need

- Raspberry Pi running Raspbian
- Sonic Pi v2.6+
- Speakers or headphones with a 3.5mm jack
- Update Sonic Pi: `sudo apt-get update && sudo apt-get install sonic-pi`

Whether it's the haunting drift of rumbling oscillators or the detuned punch of saw waves piercing through the mix, the lead synth plays an essential role on any electronic track. In last month's edition of this tutorial series, we covered how to code our beats. In this tutorial we'll cover how to code up the three core components of a synth riff – the timbre, melody, and rhythm.

OK, so power up your Raspberry Pi, crack open Sonic Pi v2.6+, and let's make some noise!

Timbral possibilities

An essential part of any synth riff is changing and playing with the timbre of the sounds. We can control the timbre in Sonic Pi in two ways – choosing different synths for a dramatic change, and setting the various synth opts for more subtle modifications. We can also use FX, but that's for another tutorial...

Let's create a simple live loop where we continually change the current synth:

```
live_loop :timbre do
  use_synth (ring :tb303, :blade, :prophet,
:saw, :beep, :tri).tick
  play :e2, attack: 0, release: 0.5,
cutoff: 100
  sleep 0.5
end
```

Take a look at the code. We're simply ticking through a ring of synth names (this will cycle through each of these in turn, repeating the list over and over). We pass this synth name to the **`use_synth`** fn (function), which will change the **`live_loop`**'s current synth. We also play note **`:e2`** (E at the second octave), with a release time of **`0.5`** beats (half a second at the default BPM of 60) and with the **`cutoff:`** opt set to **`100`**.

Hear how the different synths have very different sounds, even though they're all playing the same note. Now experiment and have a play. Change the release time to bigger and smaller values. For example, change the **`attack:`** and **`release:`** opts to see how different fade in/out times have a huge impact on the sound. Finally, change the **`cutoff:`** opt to see how different cutoff values also massively influence the timbre (values between 60 and 130 are good). See how many different sounds you can create by just changing a few values. Once you've mastered that, just head to the Synths tab in the Help system for a full list of all the synths and all the available opts each individual synth supports, to see just how much power you have under your coding fingertips.

Timbre is just a fancy word describing the sound of a sound. If you play a the same note with different instruments such as a violin, guitar, and piano, the pitch (how high or low it sounds) would be the same, but the sound quality would be different. That sound quality – the thing which allows you to tell the difference between a piano and a guitar – is the timbre.

Melodic composition

Another important aspect to our lead synth is the choice of notes we want to play. If you already have a good idea, then you can simply create a ring with your notes in and tick through them:

```
live_loop :riff do
  use_synth :prophet
  riff = (ring :e3, :e3, :r, :g3, :r, :r,
:r, :a3)
  play riff.tick, release: 0.5, cutoff: 80
  sleep 0.25
end
```

In this example, we have defined our melody with a ring, which includes both notes such as **`:e3`** and rests, represented by **`:r`**. We are then using **`.tick`** to cycle through each note to give us a repeating riff.

Language
>RUBY

Auto melody

It's not always easy to come up with a nice riff from scratch. Instead, it's often easier to ask Sonic Pi for a selection of random riffs and to choose the one you like the best. To do that, we need to combine three things: rings, randomisation, and random seeds. Let's look at an example:

```
live_loop :random_riff do
  use_synth :dsaw
  use_random_seed 3
  notes = (scale :e3, :minor_pentatonic).shuffle
  play notes.tick, release: 0.25, cutoff: 80
  sleep 0.25
end
```

There's a few things going on – let's look at them in turn. First, we specify that we're using random seed 3. What does this mean? Well, The useful thing is that when we set the seed, we can predict what the next random value is going to be – it's the same as it was last time we set the seed to 3 (see 'Pseudo Randomisation' box below). Another useful thing to know is that shuffling a ring of notes works in the same way. In the example above, we're essentially asking for the 'third shuffle' in the standard list of shuffles – which is also the same every time, as we're always setting the random seed to the same value right before the shuffle. Finally, we're just ticking through our shuffled notes to play the riff.

Now, here's where the fun starts. If we change the random seed value to another number, say 3000, we get an entirely different shuffling of the notes. So now it's very easy to explore new melodies. Simply choose the list of notes to shuffle (scales are a great starting point), then the seed to shuffle with. If you don't like the melody, just change one of those two things and try again. Repeat until you like what you hear!

Pseudo randomisation

Sonic Pi's randomisation is not actually random – it's what's called pseudorandom. Imagine if you were to roll a dice 100 times and write down the result of each roll onto a piece of paper. Sonic Pi has the equivalent of this list of results, which it uses when you ask for a random value. Instead of rolling an actual dice, it just picks the next value from the list. Setting the random seed is just jumping to a specific point in that list.

Finding your rhythm

Another important aspect to our riff is the rhythm – when to play a note and when not to. As we saw above, we can use **:r** in our rings to insert rests. Another very powerful way is to use spreads, which we'll cover in a future tutorial. Today we'll use randomisation to help us find our rhythm. Instead of playing every note, we can use a conditional to play a note with a given probability. Let's take a look:

```
live_loop :random_riff do
  use_synth :dsaw
  use_random_seed 30
  notes = (scale :e3, :minor_pentatonic).shuffle
  16.times do
    play notes.tick, release: 0.2, cutoff: 90 if one_in(2)
    sleep 0.125
  end
end
```

A really useful fn to know is **one_in**, which will give us a **true** or **false** value with the specified probability. Here, we're using a value of **2**, so on average, one time every two calls to **one_in** it will return **true**. Using higher values will make it return **false** more often, introducing more space into the riff.

Notice that we've added some iteration in here with **16.times**. This is because we only want to reset our random seed value every 16 notes, so our rhythm repeats every 16 times. This doesn't affect the shuffling, as that is still done immediately after the seed is set. We can use the iteration size to alter the length of the riff. Try changing the 16 to 8 or even 4 or 3 and see how it affects the rhythm of the riff. Finally, experiment with different seed values!

Bringing it all together

OK, so let's combine everything we've learned together into one final example. See you next time!

```
use_debug false
live_loop :random_riff do
  #  uncomment to bring in:
  #  synth :blade, note: :e4, release: 4, cutoff: 100, amp: 1.5
  use_synth :dsaw
  use_random_seed 30030
  notes = (scale :e3, :minor_pentatonic, num_octaves: 2).shuffle.take(8)
  8.times do
    play notes.tick, release: rand(0.5), cutoff: rrand(60, 130) if
one_in(2)
    sleep 0.125
  end
end
live_loop :drums do
  use_random_seed 500
  16.times do
    sample :bd_haus, rate: 2, cutoff: 110 if rand < 0.35
    sleep 0.125
  end
end
live_loop :bd do
  sample :bd_haus, cutoff: 100, amp: 3
  sleep 0.5
end
```

FREQUENTLY ASKED QUESTIONS

Your technical hardware and software problems solved...

NEED A PROBLEM SOLVED?

Email **magpi@raspberrypi.org** or find us on **raspberrypi.org/forums** to feature in a future issue.

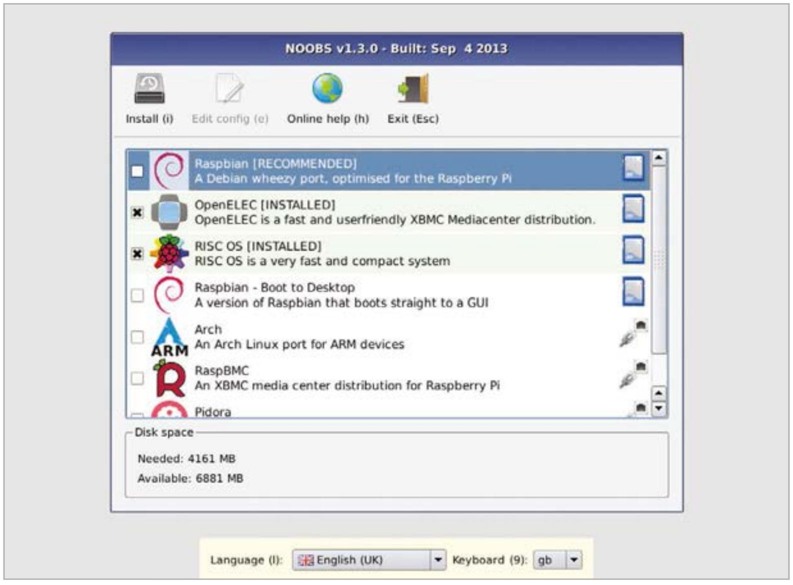
SD CARDS

Do I need to use my SD card with my Raspberry Pi? Can I install an operating system to a USB stick or portable hard drive so I have more space?

You can have the main operating system files on external storage, but the Raspberry Pi requires the boot script on the SD card to actually boot up. This can be advantageous, though, as using fast USB storage to have the operating system on can make certain aspects of the system a little faster. The process is a little long-winded for these FAQ pages, so check out this tutorial on how to set it up: **bit.ly/1id9LrM**

Can I back my SD card up like a hard drive on my PC? I have files on there I'd like to keep!

Yes, you can!... as well as just plugging in some USB storage and copying everything from the Pi folder into it (which will most likely contain most of your documents, pictures and Python programs, etc). The best way is to 'clone' your SD card, though; in Windows, you can use Win32 Disk Imager to create an image of the SD card, which creates an exact copy that you can then use Win32 Disk Imager with to write back to an SD card if you need to restore it. On OS X or Linux, you can use dd in the terminal for the same effect.


NOOBS v1.3.0 - Built: Sep 4 2013

Raspbian [RECOMMENDED]
A Debian wheezy port, optimised for the Raspberry Pi

OpenELEC [INSTALLED]
OpenELEC is a fast and userfriendly XBMC Mediacenter distribution.

RISC OS [INSTALLED]
RISC OS is a very fast and compact system

Raspbian - Boot to Desktop
A version of Raspbian that boots straight to a GUI

Arch
An Arch Linux port for ARM devices

RaspBMC
An XBMC media center distribution for Raspberry Pi

Pidora

Disk space
Needed: 4161 MB
Available: 6881 MB

Language (l): English (UK) Keyboard (9): gb

What's the largest size SD card I can use in my Raspberry Pi? I have an 8GB one already. Can it go larger?

The Raspberry Pi is only limited by the file system – while the Raspberry Pi Foundation has tested up to 32GB cards, if you get an SDXC card that is larger than that, you can get it to work by forcing it to have multiple 32GB partitions. Put NOOBS on the first partition and during installation of Raspbian, it will resize it to be one partition. This might not work with every SD card, though, so it's best to be safe and rely on USB storage if you have a large storage requirement.

Can I have multiple operating systems installed to one SD card?

NOOBS has a limited ability to dual-boot (or as many boots as you want, even) different operating systems to one SD card. This does take up a bit of space, though, as it doesn't have all the files for all its operating systems on the image. To access the NOOBS menu once you've already installed a distro, press **SHIFT** during boot to bring the menu back up and install another distro.

FROM THE RASPBERRY PI FAQ

RASPBERRYPI.ORG/HELP

Does the Raspberry Pi overclock?

The Raspberry Pi models A, A+, B, and B+ operate at 700MHz by default. Most devices will run happily at 800MHz. The Pi 2 Model B operates at 900MHz by default and should run quite happily at 1000MHz. In the latest Raspbian distro, there is an option to change the overclocking options on first boot and at any time afterwards, without voiding your warranty, by running `sudo raspi-config`. You can download the Raspbian image directly or install it via the NOOBs installer, both available via **raspberrypi.org/downloads**. It should be noted, however, that these are experimental settings and that not every board will be able to run stably at the highest setting. If you experience problems, try reducing the overclocking settings until stability is restored.

What are the Raspberry Pi's power requirements?

The device is powered by 5V micro-USB. Exactly how much current (mA) the Raspberry Pi requires is dependent on what you hook up to it. We have found that purchasing a 1.2A (1200mA) power supply from a reputable retailer will provide you with ample power to run your Raspberry Pi for most applications, although you may need a 2.5A (2500mA) if you want to use all four USB ports on the Models B+/2B without using an external powered USB hub.

Why is there no real-time clock (RTC) on the Raspberry Pi?

The expectation is that non-network-connected units will have their clocks updated manually at startup. Adding an RTC is surprisingly expensive, once you have factored in batteries, area, and componentry, and would have pushed the Pi above the Foundation's target price. You can add one yourself using the GPIO pins if you'd like an interesting electronics project.

Can I power the Raspberry Pi from a USB hub?

It depends on the hub. Some hubs comply with the USB 2.0 standard and only provide 500mA per port, which may not be enough to power your Raspberry Pi. Other hubs view the USB standards more like guidelines, and will provide as much power as you want from each port. Please also be aware that some hubs have been known to 'backfeed' the Raspberry Pi. This means that the hubs will power the Raspberry Pi through its USB cable input, without the need for a separate micro-USB power cable, and bypass the voltage protection. If you are using a hub that 'backfeeds' to the Raspberry Pi and the hub experiences a power surge, your Raspberry Pi could potentially be damaged.

THE MAGPI APP

Having trouble with *The MagPi* on the App Store or Google Play? Here are your most common questions answered:

How do I find *The MagPi* on Google Play or the App Store?

All you have to do is go to the search bar and type 'The MagPi' or 'Raspberry Pi' to find us.

I've subscribed to the digital edition and I can't sign in to restore my purchases. Please help!

Since your *The MagPi* purchases are linked to your Google or Apple accounts, there's no need to sign in at all. If you'd like to re-download your purchases on your current device, or make your purchases available on other devices, all you need to do is hit 'Subscribe' on the home screen, then 'Restore Purchases' on the next screen.

How can I search the digital magazine for keywords?

Finding direct references is really easy with *The MagPi* app – all you have to do is tap the screen to get the app's GUI to show, and then press the small magnifying glass icon in the top-right corner of the screen. Now, just type in your search term to find the relevant results.

Get it on Google play

Available on the App Store

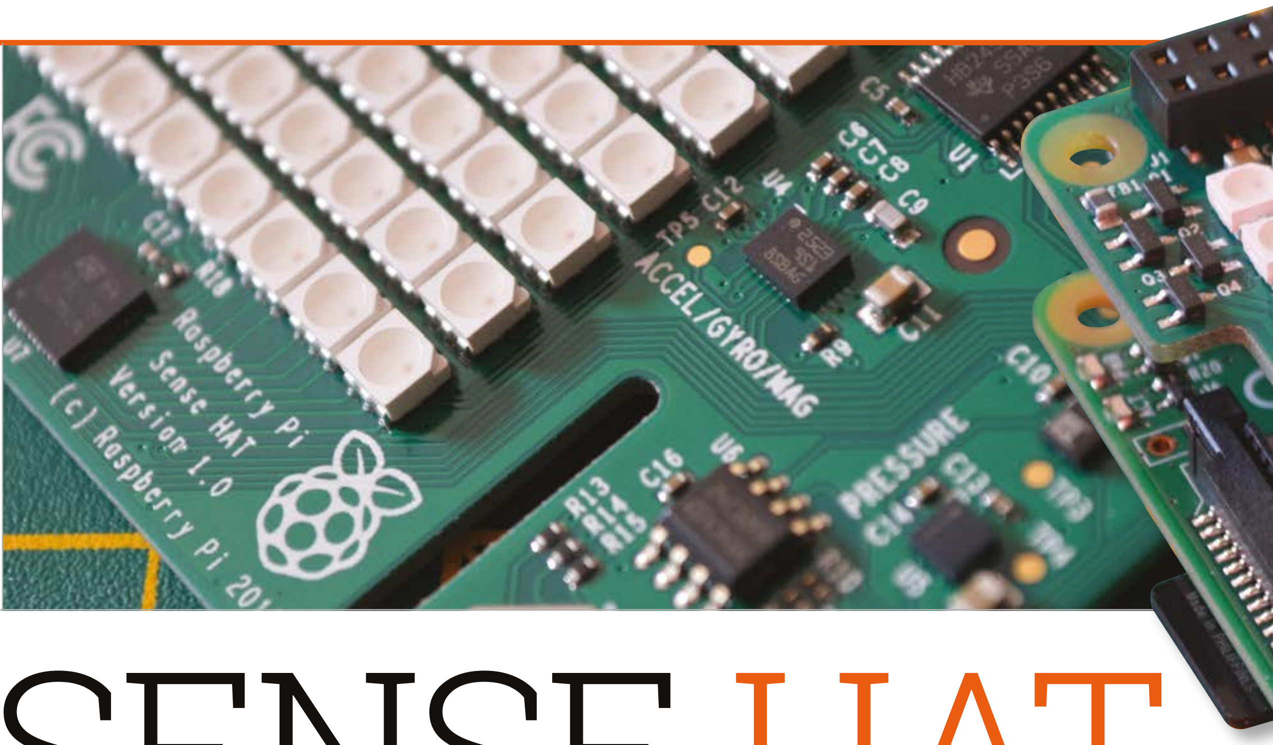
swag.raspberrypi.org

£23/$35

Maker Says

Make your own Astro Pi!

Raspberry Pi Swag Store



SENSE HAT

Space: the final frontier. These are the voyages of the Sense HAT. Its mission: to make sensing the environment really easy for Raspberry Pi users. To boldly go where no Pi has gone before...

If you've been reading this magazine from cover to cover (which you really should, we have some top-quality content in here. Don't worry, we'll wait here for you to catch up. Done? Good!), you'll have seen our release story for the Sense HAT that starts on page 6. This is the little board that makes the Astro Pi possible, hosting a suite of sensors along with a very simple LED display. The idea is that it sits on top of your Pi's GPIO pins and gives you access to much more data from your surroundings through a few extra packages.

Like all the other hardware produced by the Raspberry Pi Foundation, it is fairly cheap. On the other hand, it'll set you back nearly as much as a Raspberry Pi 2 Model B, so could be viewed as being a little expensive for an add-on board. In the grand scheme of things, though, the Sense HAT represents great value for money considering the functions it grants you. The reason it is so inexpensive is because it was designed to be easily constructed, with a few compromises made to keep the price down.

Cheap as silicon chips

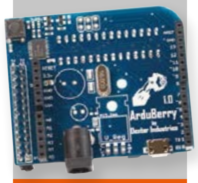
Don't get us wrong, though: the Sense HAT is definitely not a cheap product. It's the same sort of sturdy construction as the Raspberry Pi, and doesn't have any easily broken sticking out capacitors or other components. It's built to be blasted into space, after all, so needs to be pretty robust. We wouldn't start stamping on it to try to test out how tough it is, but it'll survive being man-handled onto and off a Raspberry Pi. It fits very snugly on the GPIO pins (taking up all 40), and can even be used with other GPIO headers or wires to the individual pins. There's no programming issue with this, as long as you're not trying to use the functions of the Sense HAT at exactly the same time, although if you know what pins are used for specific sensors or the LED matrix, you might be able to work around it with a bit of inventive coding.

The specific sensors on the Sense HAT are pretty varied. There's a suite of the usual environmental sensors such as temperature, pressure and humidity, along

Related

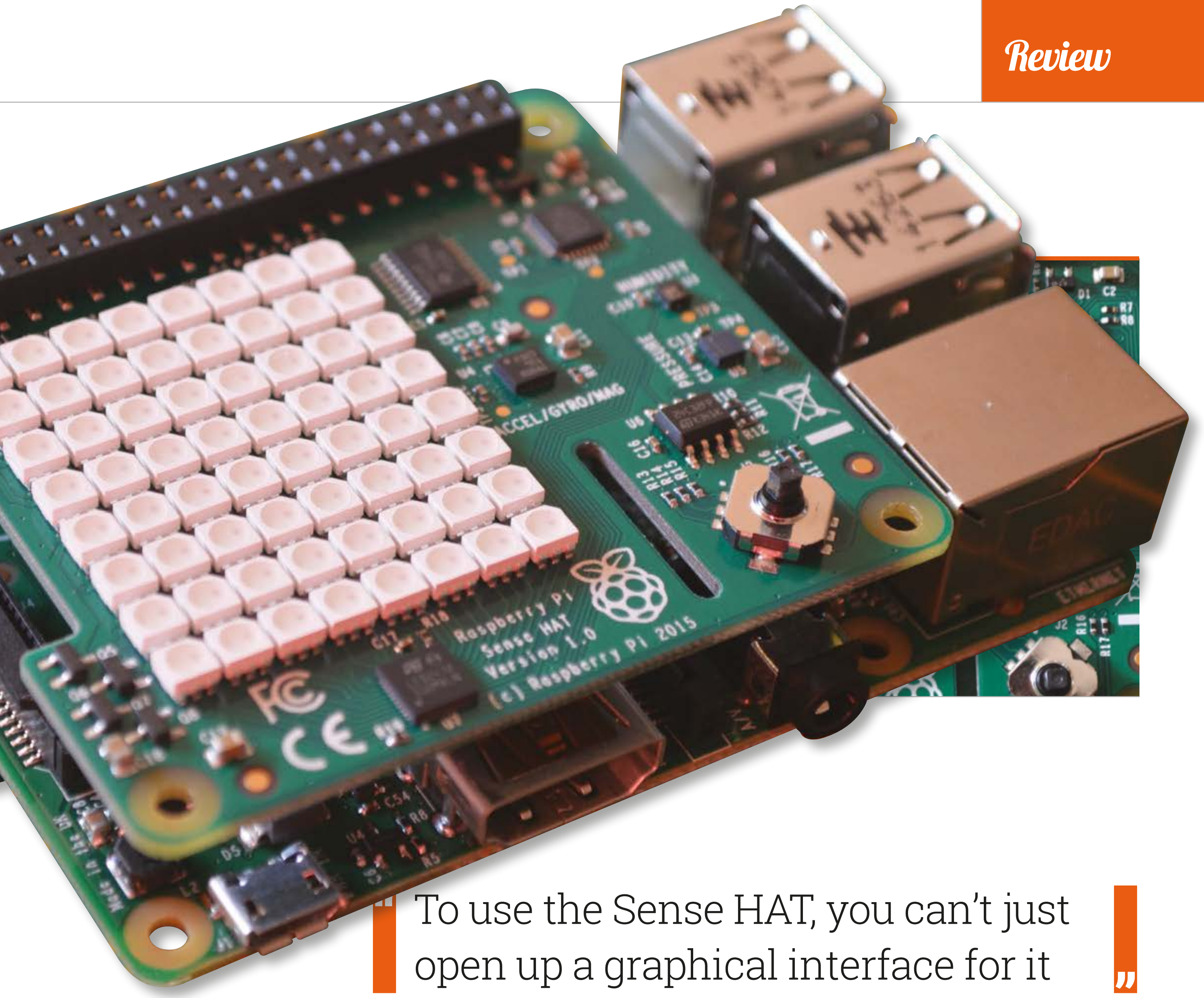
ARDUBERRY

While there's nothing quite like the Sense HAT, the Arduberry allows you to connect Arduino boards to the Pi, enabling you to create your own custom sensor suite, albeit at a greater cost.



$29.99/£19.70

dexterindustries.com/Arduberry



> To use the Sense HAT, you can't just open up a graphical interface for it

with a trio of positional sensors which include an accelerometer, a gyroscope, and a magnetometer. They're all pretty accurate and able to sense a wide range in their respective fields, with the outputs instantly modified to be readable with the Sense HAT library.

Put on your coding hat

To use the Sense HAT, you can't just open up a graphical interface for it... at least just yet. It would actually be quite simple to create a general interface or a custom one for your needs, because frankly the coding side of the Sense HAT is very straightforward, making use of Python's basic syntax and some simple command structures of its own. So you can easily get data from the Sense HAT with only a few lines of code, and very easily print out data or images to the screen. Text output is handled by the code, so you don't need to do any ridiculous coding to change individual pictures for it to scroll across the screen.

There's a massive amount of control over the input and output as well, with the full power of the Sense HAT readily available. There are plenty of online resources as well (through the Astro Pi and Raspberry Pi sites) to help you learn exactly how to use it. We like the smiley-face tutorial example; it made us feel happy.

We sense a winner

All in all then, this is an excellent bit of gear. It's well put together with a huge number of functions and it's very easy to program as well, thanks to some great software available for the board. You can use it on its own or in conjunction with other projects (we want to attach it to a robot in the near future), and if it's good enough for the ISS, it should definitely be enough for any crazy ideas you may have for it.

Last word

The Sense HAT is easy to use and doesn't compromise any of its functions to allow that. It's a fantastic board for the price and really expands the amount you can do with the Pi.

★★★★★

Maker Says

"Turn your Pi into a tablet or infotainment system"
Raspberry Pi

RASPBERRY PI TOUCHSCREEN DISPLAY

The official 7˝ Touchscreen Display for Raspberry Pi is here, but does it live up to expectations?

Related

HDMIPI

The first screen for the Pi was funded by Alex Eames of RasPi.TV on Kickstarter. It was a great community effort.

£75 / $110

hdmipi.com

It's been a long time coming, but the official Raspberry Pi Touchscreen Display is here. It's the first product to make use of the DSI port at the rear of your Raspberry Pi, meaning you don't need to use the HDMI port on your Pi to get video output. Why is that useful? Not only does this mean that you don't need to use a bulky HDMI cable if you're trying to make a small, self-contained touchscreen project, but it also means you can output two different video signals, via HDMI and DSI, for multi-screen output from one the Raspberry Pi for the first time.

This high-quality 800×480 HD-ready display doesn't connect directly to the DSI, though. Instead, it utilises an adaptor board which handles the power and signal conversion. The DSI cable comes from the Pi and connects to this daughter board, which has exactly the same dimensions and mounting holes as a Raspberry Pi A+. There are only two connections needed: DSI and power. The adaptor board takes care of the latter with a great level of flexibility and allows you to power both the screen and Pi in a couple of different ways. It's possible to either connect it to the Pi with 5V and ground jumper wires, or via a micro-USB connector. This means you can power the Pi and screen with a single micro-USB wall wart, though you'll need to make sure you've got a decent 2-amp supply.

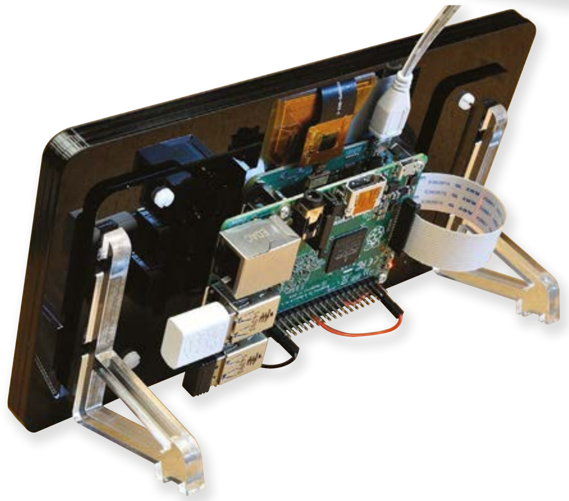
Quality package

You get a lot of technology for £48 / $60. Along with the HD-ready screen and adaptor board, you get all the stand-offs and screws

swag.raspberrypi.org

£48 / $60

Photos by Alex Eames – www.raspi.tv



you need to safely secure it to your Pi, a DSI ribbon cable, and four jumper wires for both power and communications. Pimoroni has also produced an excellent Pibow-esque bezel with integrated stand in different colours. For a small premium, you can add your chosen colour to the shopping basket so you don't have to worry about propping the screen up or 3D-printing your own solution.

Touch friendly

The real star feature of the new display, though, is its touch capabilities. It's actually capable of ten-point capacitive touch, opening the door to some of the most advanced touch capabilities outside Apple's new force-touch technology for the iPhone 6s.

In a recent YouTube video (**youtu.be/Eah3Zq18OyM**), Matt Richardson has demonstrated how Kivy, the popular and easy-to-use user interface creation library for Python, can quickly and easily make good use of the touch elements of the Touchscreen Display, while rigging up simple GPIO projects, too.

In his example project, he has created several on-screen buttons and a slider which interact with his GPIO breadboard. Lights can be turned on by tapping the button, and PWM can be adjusted on the fly with a slider to make it pulse faster or slower in response to your touch inputs.

Of course – and by Matt's own admission – this barely scratches the surface of what can be done with the Touchscreen Display and some simple GPIO programming – we've got a great feeling that Kivy is going to help bring a lot of brilliant ideas projects to the table.

Last word

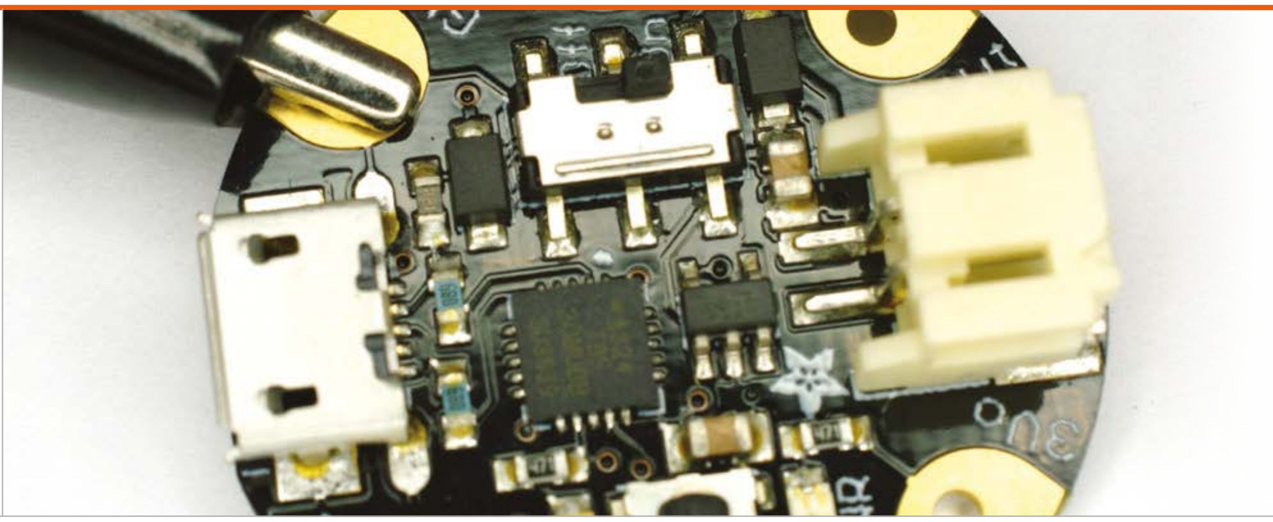
Amazing picture quality, ample connectivity options, and outstanding value for money. Easily the best official Raspberry Pi add-on since the Camera Module.

★★★★★

Maker Says

A tiny wearable platform board with a lot of might

Adafruit



ADAFRUIT GEMMA STARTER PACK

As an introduction to programming wearable electronics, has Adafruit put together the perfect kit?

With wearable computing proving a burgeoning business, it's no surprise to find companies helping makers to get started in the field. The majority of the resulting kits are based around conductive thread which can be sewn into fabric to link components electrically. It's easy, then, to see the LEDs and conductive thread bobbin included in Adafruit's Gemma Starter Pack and dismiss it as just another me-too product. Doing so, though, ignores the titular star of the show: the Gemma.

Designed and built in collaboration with the Arduino company, the Gemma is a smart break-out board for the ultra-compact ATtiny85 microcontroller from Atmel. A cut-down version of the ATmega chips which power the full-size Arduino boards, ATtiny microcontrollers are a great choice when low power draw and a small footprint are more important than the number of pins you can access. This makes them an obvious choice for wearable projects.

The two biggest hurdles to using an ATtiny with conductive thread – the need for special programming equipment and its package type – are solved by the Gemma. The chip is placed on a tiny 28mm diameter disc which features connectors for three programmable I/O pins, a 3.3V supply and ground, and a voltage input. A JST connector allows batteries to be easily connected, while a USB connector coupled with a specially written bootloader means the Gemma can be programmed from any PC using the standard Arduino IDE and the bundled cable.

The Gemma is the star, but there are plenty of other parts to the starter kit. A bundle of colourful crocodile leads makes it easy to prototype your design before sewing it down with the conductive thread, needles for which are also provided. There are four of Adafruit's famous programmable RGB NeoPixel LEDs, in wearable-friendly Flora guise, and a battery holder for the bundled CR2032 batteries with integrated power switch.

The thread itself is of great quality with a generous 23-metre length provided. A thin, two-ply formulation made from stainless steel, it's easier to work with than the thicker thread used in the rival Kitronik Electro Fashion range, but comes with a warning that it is ill-suited for projects in which your components will draw more than around 50mA. A three-ply alternative is available separately, and is good up to 100mA; both are provided on pre-wound bobbins suitable for any heavy-thread sewing machine.

Related

KITRONIK ELECTRO FASHION E-TEXTILES DELUXE KITA

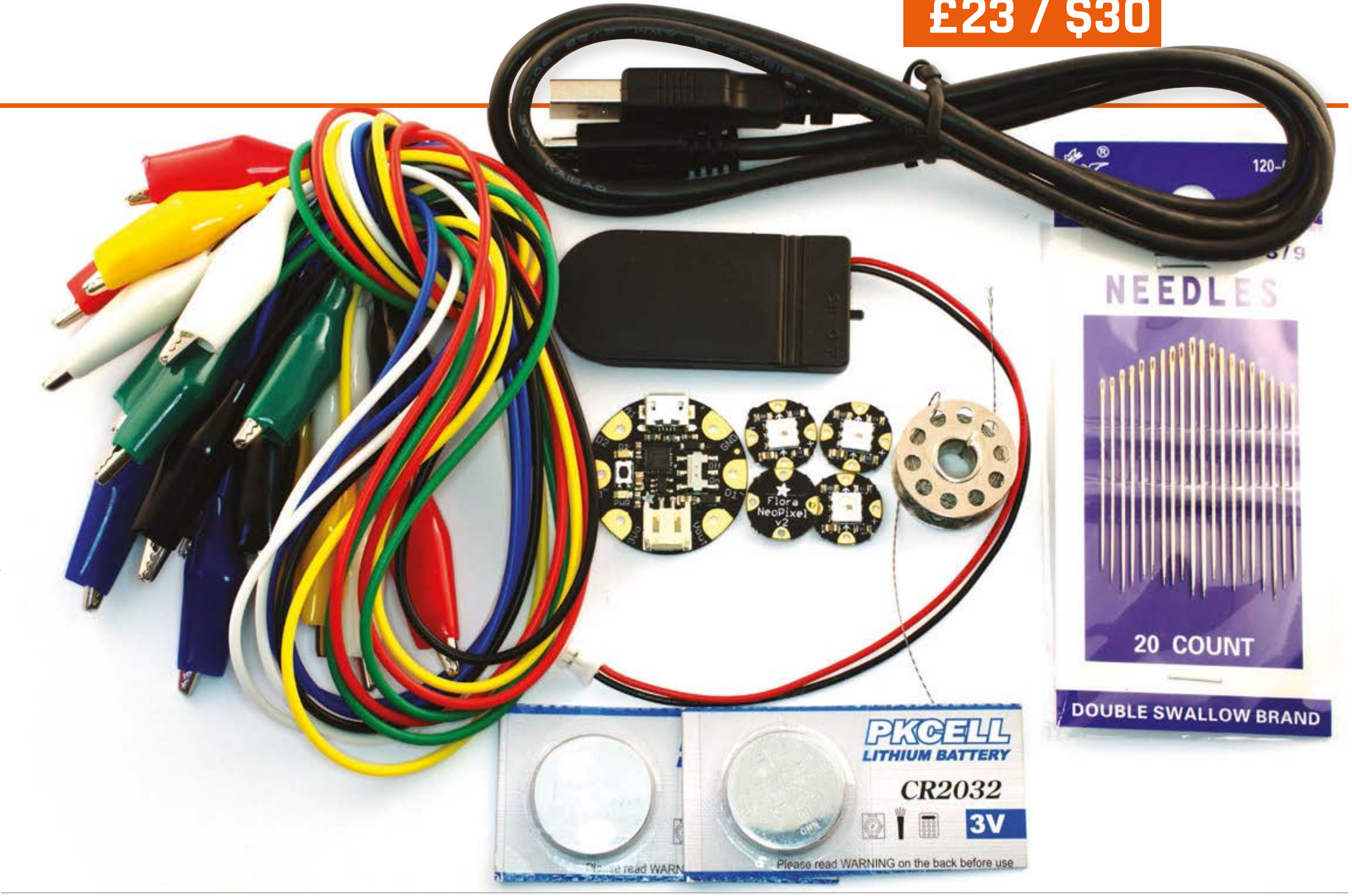
Reviewed in issue 37, the Kitronik kit offers less programmability than Adafruit's, but can be had at a cheaper price in the UK.

£15 / $35

cpc.farnell.com

cpc.farnell.com

£23 / $30



Above The kit has everything you need to get started

> The Gemma is the star, but there are plenty of other parts

As with most electronic textile starter kits, there's an undeniable focus on LED projects. Once you've amused yourself adding RGB glowing eyes to kids' toys, though, you can begin to experiment with the power of the Gemma. Although three input-output pins feels limited, they offer surprising flexibility: two of the three offer pulse-width modulation (PWM) support for dimming LEDs or driving small servos, while one can also be used as an analogue input to read a variety of optional sensors. These, however, must be compatible with the 3.3V logic used by the Gemma, in contrast to the 5V logic more common in the Arduino world.

There are some disappointments, too, for Arduino fans who were hoping that their projects would be directly transferable. Aside from the limited number of pins, the Gemma has no accessible serial port. This means it can't be combined with, for example, a serial-connected Bluetooth or Wi-Fi adaptor to easily make connected clothing.

To focus on what it can't do is to ignore the kit's true goal, however. Adafruit bills the Gemma kit as the perfect way for beginners to get started with programmable wearable and electronic textile projects, and the obvious next step from its non-programmable LED-based kit. The use of RGB LEDs, which can be cycled through your choice of millions of colours, keeps things interesting and the low cost of the kit makes it accessible to those with even a passing curiosity in electronic textiles projects, as well as Arduino users keen to try something a little different in their next build.

The only real negative is the lack of printed documentation. While there's a wealth of supporting information, tutorials, and projects on Adafruit's website, a small booklet like that provided with the rival Kitronik kit would have been welcomed and helped to make this bundle a truly standalone product.

Last word

If you have experience with the Arduino platform and want to get into wearable electronics, the Gemma is the way to go and this kit a perfect way to get started.

★★★★☆

RASPBERRY PI BESTSELLERS

C PROGRAMMING

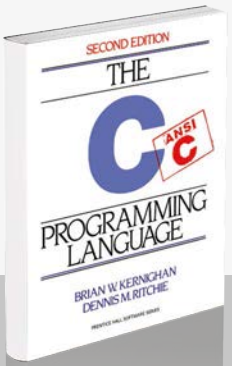
For reliable embedded programming on your Pi, nothing gets you closer to bare metal than the C programming language

HEAD FIRST C

Author: David Griffiths & Dawn Griffiths
Publisher: O'Reilly
Price: £33.50
ISBN: 978-1449399917
oreil.ly/1he0aQR

Approachable intro for coders wanting to learn C, with useful self-study projects.. Not a definitive reference, but a friendly start on the path to understanding C.

THE C PROGRAMMING LANGUAGE



Authors: Brian W Kernighan & Dennis Ritchie
Publisher: Prentice Hall
Price: £41.99
ISBN: 978-0131103627
bit.ly/1F2D4bq

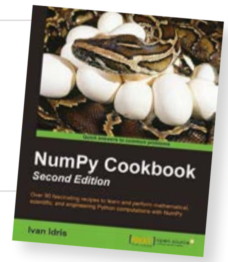
K&R, as it's known, is a work so concise as to almost be a Zen kōan, yet all C is there: it will make you a better programmer, and deepen your understanding of C.

C PROGRAMMING: A MODERN APPROACH

Authors: K N King
Publisher: W W Norton
Price: £52.50
ISBN: 978-0393979503
knking.com/books/c2/

Fast becoming the best-loved book for C learners – comprehensive, very correct in its approach to C standards, and repays hard work with C enlightenment.

NUMPY COOKBOOK



Author: Ivan Idris
Publisher: Packt
Price: £29.99
ISBN: 978-1784390945
bit.ly/1MrED5c

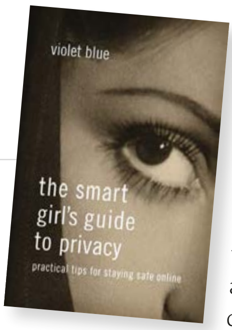
Python's libraries are a terrific asset, and NumPy's fast precompiled functions for numerical routines – complemented by SciPy, Matplotlib, IPython, and Scikits – provide not just MATLAB-like functionality, but a scientific software ecosystem with applications you may not even have considered. Idris surveys all of examples you'd expect for scientists, engineers, programmers, and analysts, as well as unexpected uses like audio and image processing.

"We NumPy users live in exciting times," says Idris, referring to new and future NumPy developments around cloud, concurrency, and online analytical processing (OLAP) for multi-dimensional analytical (MDA) queries. But sticking with the present, NumPy Cookbook introduces the powerful IPython environment and gets you started with NumPy's basic building blocks: arrays and universal functions – where it gets onto some Fancy Indexing, and starts working on WAV files.

Audio, and images, yes, either side of which is more on arrays and functions, as well as speeding up code. All this is done in a broad context of the SciPy ecosystem, and speaking to the wider world – whether MATLAB and Octave, or R and JPype (for JVM languages). Profiling, debugging, testing, and behaviour-driven development follow, making this a useful book for anyone looking at writing practical Python code that has life beyond their own computer. Recommended.

Score ★★★★★

THE SMART GIRL'S GUIDE TO PRIVACY



Author: Violet Blue
Publisher: No Starch
Price: £11.99
ISBN: 978-1-59327-648-5
nostarch.com/smartgirlsguide

The difference in being logged onto a site as a female and as a male has to be experienced to be believed: the author recommends getting a throwaway account on a social media site just to try it. Women are targets online, and privacy is at risk from threats including friends with good intentions, greedy companies, and people from whom we all need to take steps to protect ourselves. The book comes with privacy stickers, to prevent people using your device's camera to record you without your knowledge.

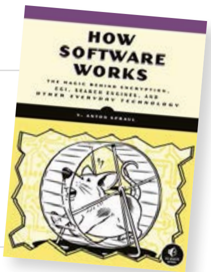
Eight "privacy tips to use right now" provide a ready guide to tipping the balance hugely in your favour, as does taking the privacy check-up. That done, Blue shows what information not to share in order to stay safe. If you think this is paranoia, read here about (male) *Wired* journalist Mat Honan, whose digital devices were wiped and Twitter account offensively defaced – all from the attacker finding Honan's address (a simple whois lookup) and using it to get Amazon account data, then move on to Apple.

Blue provides practical steps to deal with all the horrors and harassment, recover from identity theft, and still be able to pursue romance online. Sadly, a very necessary book.

Score ★★★★☆

HOW SOFTWARE WORKS

Author: V Anton Spraul
Publisher: No Starch
Price: £19.99
ISBN: 978-1593276669
nostarch.com/howsoftwareworks



"We live in an age of magic," writes V Anton Spraul in the introduction, referencing Arthur C Clarke's famous dictum on advanced technology. After all, many of us work with technology but lack a real understanding of what's going on inside the computer. From security topics, like password hashing and encryption, to other underpinnings of modern life like search, and map routes, this book shows what is happening inside programs, what the limitations are, and what may develop in the future.

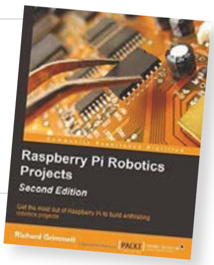
Encryption lies at the heart of so many daily tasks, from shopping to checking your bank balance. Spraul takes us back to pre-computer encryption, with a lucid explanation which will help even non-technical users understand the process. Particularly important is Spraul's discussion of potential flaws and weaknesses in each technology, which will put in perspective for the reader various news stories on security scares.

The insights into game graphics, computer animation, and image compression are fascinating. Explanations of techniques used in search – from quicksort algorithm to hashing – and in concurrency programming, will help beginner programmers on the Pi as they gain understanding to really support the methods they're learning. Map routes is another informative chapter, particularly if you're working on a robotics project.

Score ★★★★☆

RASPBERRY PI ROBOTICS PROJECTS

Author: Richard Grimmett
Publisher: Packt
Price: £29.99
ISBN: 978-1-78528-014-6
bit.ly/1U6oOVD



Right from the set-up chapter, with remote access via SSH and VNC, and nmap to locate your Pi on the network, Dr Grimmett assumes no knowledge, but introduces important topics – the second chapter, on programming with Python, covers some OOP features, using Emacs, and briefly examines C and C++.

The first robotics topic is speech, with Espeak and PocketSphinx to handle input and output – "what self-respecting robot wants to carry around a keyboard?" – and your brief acquaintance with C used to edit and rebuild the controlling program. In the vision chapter, the basics of installation and use of OpenCV are well explained. For robot mobility, the GPIO controls a motor (no soldering necessary), driving wheels, and voice commands control movement. Then, with a choice of ready-assembled legs, and a little Python to interpret voice commands, your robot is ready to tackle uneven ground, or even stairs.

Obstacle-avoidance sensors stop your robot blundering into walls; wireless and GPS find the robot. Finally, Robot Operating System (ROS) manages system control. Then, using model kits that can be controlled by the Pi, take your robot on or under water, or up in the air. Beginner to roboteer in just 276 pages!

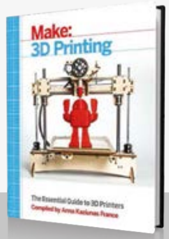
Score ★★★★☆

ESSENTIAL READING: 3D PRINTING

3D printing is within reach of everyone, by building your own, or sharing at a local makerspace

Make: 3D Printing

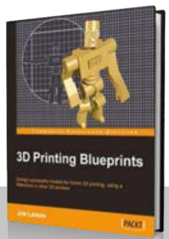
Author: Anna Kaziunas France
Publisher: Maker
Price: £13.50
ISBN: 978-1457182938
oreil.ly/1KliLGq



Arms you with everything you need to know to understand the exciting but often confusing world of 3D printing.

3D Printing Blueprints

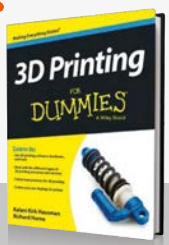
Author: Joe Larson
Publisher: Packt
Price: £30.99
ISBN: 9781849697088
bit.ly/1Ige22Z



Makerbot-focused, project-based guide to preparing designs for 3D printing, using the free and open -source Blender program.

3D Printing For Dummies

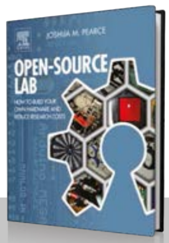
Author: Kalani Kirk Hausman
Publisher: Wiley
Price: £21.99
ISBN: 978-1118660751
bit.ly/1WT7IJD



Puts 3D printing in context, and shows you how to build your own RepRap.

Open-Source Lab

Author: Joshua M. Pearce
Publisher: Elsevier
Price: £33.99
ISBN: 9780124104624
bit.ly/1JyYNas



Academic guide to building a RepRap, then 3D-printing a complete laboratory of scientific instruments.

Eventorbot!

Author: eventorbot
Publisher: Instructables
Price: FREE
bit.ly/1PUix9w

Fewer materials and a stronger structure give a cheaper, easier, and very stable, well-designed 3D printer. 100% open source.

WANT TO GET NOTICED?

ADVERTISE IN THE MAGPI MAGAZINE

REACH THE RIGHT AUDIENCE FAST

- World's #1 Pi magazine
- Block booking discounts
- Free to download & share
- Live links on iOS & Android

The MagPi is the most exciting mag in tech today, boasting one of the biggest & most engaged audiences in the industry.

FOR MORE INFO EMAIL **RUSSELL@RASPBERRYPI.ORG**

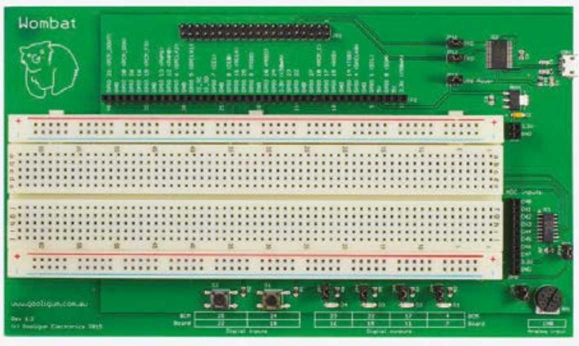
Easy to INSTALL
A joy to WATCH

Made for

lightberry.eu raspberry-at-home.com

The Wombat Prototyping Board

makes it easy to create original Raspberry Pi projects



Provides easy access to all GPIO pins

Adds functionality to your Pi, such as:

- *analog inputs*
- *LEDs and pushbuttons*
- *USB serial console port*

Includes a set of projects to get you started.

SAVE 10%

just use code MAGPI37 at checkout

Code expires 1 Oct 2015

Educational discounts available

www.gooligum.com

EUMAKERS

the world has a new dimension

3D PRINTING

FILAMENT

OVER 60 COLORS AVAILABLE

TRY FOR FREE OUR QUALITY

GET A FREE 3D FILAMENT SAMPLE

VISIT OUR ONLINE STORE AT

WWW.EUMAKERS.COM

Made in Italy

6 RASPBERRY JAM SILICON VALLEY
Mountain View, USA
7 RHÔNE VALLEY RASPBERRY JAM #5
Courthézon, France

RASPBERRY JAM EVENT CALENDAR

Find out what community-organised, Raspberry Pi-themed events are happening near you…

PUT YOUR EVENT ON THE MAP

Want to add your get-together? List it here:

raspberrypi.org/jam/add

1

PRESTON RASPBERRY JAM

When: Monday 5 October
Where: Media Innovation Studio, Media Factory Building, Preston, UK
bit.ly/1NF8vLw
A fun day out with lightning talks, demos and hands-on time with the Raspberry Pi for anyone interested.

2

RASPBERRY JAM LEEDS

When: Wednesday 7 October
Where: Swallow Hill Community College, Leeds, UK
bit.ly/1iiXKBf
Bringing people together to help them discover the exciting potential of the Raspberry Pi for a few hours on a Wednesday evening.

3

TORBAY TECH JAM

When: Saturday 10 October
Where: Paignton Library and Information Centre, Paignton, UK
torbaytechjam.org.uk
More than just Pi, you can learn about robotics, Arduino, and other forms of coding at Tech Jam.

4

9TH EGHAM RASPBERRY JAM

When: Sunday 11 October
Where: Gartner EMEA HQ, Egham, UK
eghamjam9.eventbrite.com
A friendly and fun Raspberry Jam, now with a competition for people showing off their best projects

5

MANCHESTER RASPBERRY JAM

When: Saturday 17 October
Where: The Shed, Manchester, UK
bit.ly/1UWGbU3
A family-friendly event where you can come along with your Pi (or borrow one!) and learn more about it.

6

RASPBERRY JAM SILICON VALLEY

When: Saturday 17 October
Where: Computer History Museum, Mountain View, USA
bit.ly/1IKPPZ1
A monthly Jam in the heart of Silicon Valley, welcoming everyone interested in Raspberry Pi.

2 RASPBERRY JAM LEEDS
Leeds, UK
1 PRESTON RASPBERRY JAM
Preston, UK
5 MANCHESTER RASPBERRY JAM
Manchester, UK
4 9TH EGHAM RASPBERRY JAM
Egham, UK
8 WARWICKSHIRE MINECRAFT RASPBERRY JAM 1
Warwickshire, UK
3 TORBAY TECH JAM
Paignton, UK

7

RHÔNE VALLEY RASPBERRY JAM #5

When: Sunday 18 September
Where: Foyer Laique, Courthézon, France
bit.ly/1UWGJsN
A French Raspberry Jam where you can learn and be inspired to do great things with the Raspberry Pi.

8

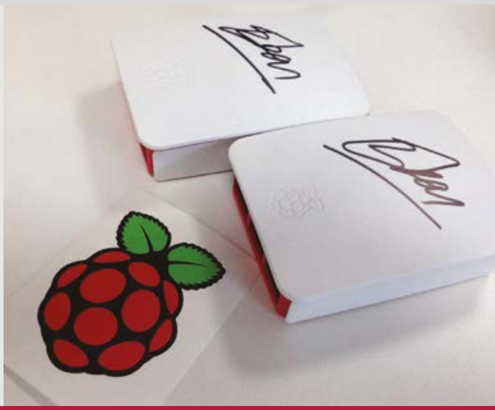
WARWICKSHIRE MINECRAFT RASPBERRY JAM 1

When: Wednesday 28 October
Where: TBC, Warwickshire, UK
bit.ly/1FirdFU
An event for anyone interested in modding and programming in *Minecraft* and Sonic Pi.

DON'T MISS: THE 9TH EGHAM RASPBERRY JAM

When: Sunday 11 October **Where:** Gartner EMEA HQ, Egham, UK

A fun and welcoming event for people interested in the Raspberry Pi. There's the usual show-and-tell and talks, but for the first time at the Egham Raspberry Jam there'll also be a competition. Voted on by the attendees, participants will be judged by quality of their display, any included material on projects, and online resources. There are some amazing prizes on offer, such as signed cases, robot kits, and more. For more details, visit: **eghamjam9.eventbrite.com**



In association with:
raspberrypi.org

5 RASPBERRY PI **ROBOT KITS** WORTH £50 **MUST BE WON!**

WHAT WOULD YOU PROGRAM YOUR **RASPBERRY PI ROBOT TO DO?**

Tell us by 28 October for your chance to win!

How to enter:

All you have to do is email **competition@raspberrypi.org** and let us know what you would do with a Raspberry Pi-powered robot!

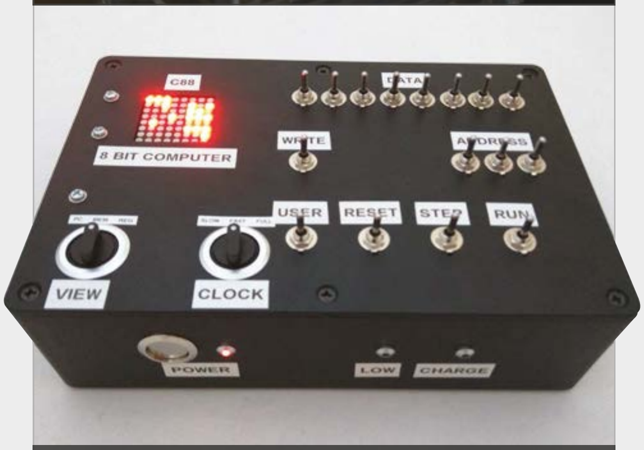
Terms & Conditions

Competition closes 28 October 2015. Prize is offered worldwide to participants aged 18 or over, except employees of the Raspberry Pi Foundation, the prize supplier, their families or friends. Winners will be notified by email after the draw date. By entering the competition, the winner consents to any publicity generated from the competition in print and online. Participants agree to receive occasional newsletters from The MagPi magazine (unless otherwise stated upon entry). We don't like spam. Participants' details will remain strictly confidential and won't be shared with third parties. Prizes are non-negotiable and no cash alternative will be offered.

MANCHESTER MAKEFEST

MakeFests are creative, family fun. MOSI is a museum dedicated to science and industry. Put them together and great things happen

The Nixie tubes give us a warm retro glow



Daniel Bailey's C88 is a homage to early computers

Putting 'slime' into bass speakers is always fun!

This two-day event was about making, science, and fun, as Manchester's Museum of Science and Industry provided an apt backdrop to the modern industrial pioneers of the UK maker community. The first thing to greet the visitor is the wonderful cyberpunk creations of the RayGun Consultancy – household junk lovingly recycled into faux deadly weapons.

Against a backdrop of scientific and industrial museum displays, the education and fun start together. Near a pioneering early aeroplane, Alan Clare of North West-based Red Squirrel Workshop was guiding young people through experimental aeronautics: building basic planes, finding out how they flew, and modifying them till they flew better. Follow their flight path and you were facing the reconstruction of the Small Scale Experimental Machine (SSEM, fondly known as Baby), the world's first stored memory computer – the original was built in Manchester in 1948. Interactive displays let the visitor program Baby, having first been asked to remember some binary commands.

The adjoining tables, under the banner of York Hackspace, contained a homage to Baby in the form of C88, designed in VHDL by Daniel Bailey. An 8-bit computer with only eight bytes of memory, it's a real challenge to program – try programming the simulator at **bit.ly/1JAy7C4**. Also from York Hackspace was the Raspberry Pi-based SpaceHack game – such good fun that we've given it a feature of its own on page 32.

Further Pi boards could be seen in classic projects like the four-foot-wide Etch-a-Sketch, and controlling multiplexed Nixie tubes from Newcastle upon Tyne's Makerspace.

Enjoy the mess

Maker events are always family oriented, and children usually love anything messy – thus the popularity of Musical Bowls, where Rachel Moat of Arduino Manchester's invitation to put your hands in bowls of custard, spaghetti, and water drew in crowds of youngsters. This was all in the pursuit of art and science, as immersion of hands in the gooey substances, via capacitive touch sensors, caused Arduino boards to play back various musical samples.

More mess was found in the workshops and spirited demonstrations of Noisy Toys, a local arts group making sounds and music from recycled hardware (their instruments included repurposed CPU fans and heat sinks), and deep, deep bass from recycled disk drives as wave generators. A lab-coat-attired

B0RKESTRA RASPBERRY PI MEETS ACOUSTIC INSTRUMENTS

Matthew Shotton and Max Leonard created B0rkestra because, Shotton tells *The MagPi*, they both had an interest in "electronic music and we wanted to take a genre which is typically only composed and reproduced via speakers and give it a physical presence and a true acoustic voice.

"It also was a great opportunity to think about instrument design without the constraints of having a human have to physically play them," he continues. "This meant we were free to rethink instruments in a way where they were designed from the ground up for computer control. Sometimes this worked and sometimes this didn't, but we're continually trying to evolve and refine them."

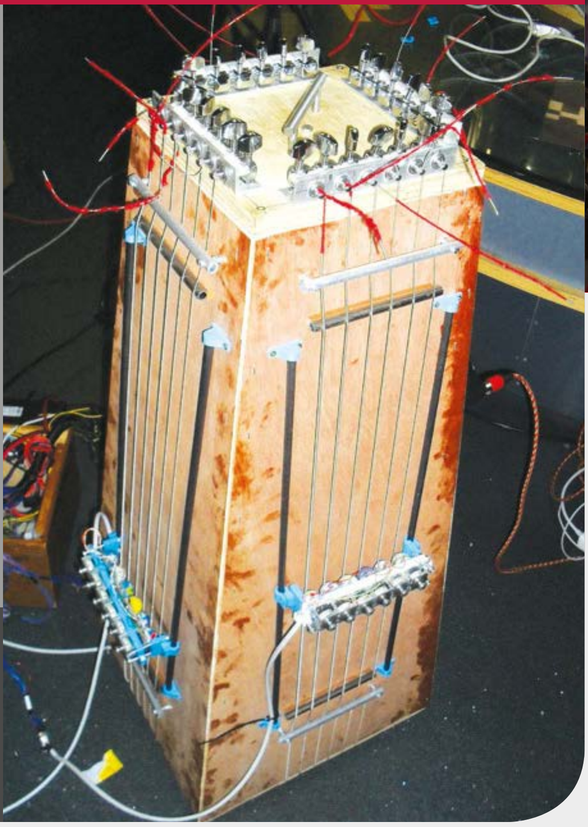
Each of the instruments has a Raspberry Pi running some code which interprets the Open Sound Control (OSC) packets from MIDI controllers, and triggers the physical actuators via the Pi's GPIO pins.

When asked about the two-day MOSI MakeFest, Shotton responds: "We were lucky to be received really well! We had interest ranging from people running experimental music festivals, to [others] running music programs for people with a disability. We also had plenty of kids coming up and making a racket. It was ace!"

More details on the GitHub repository for the project:
github.com/b0rkestra/b0rkestra

Placeholder website:
b0rkestra.com

Hardware designs for the custom MOSFET driver boards are at:
upverter.com/Matt/4962e9158ffbb9c0/b0rkestra-Driver-Board/



Steve Summers and his assistants put cornflour and other substances (alien slime! - actually a 'non-Newtonian fluid') into the cones of 1,000 watt bass speakers to show deep bass as movement and also material behaviour. Later, they had crowds of children linking hands to form an electrical conductor – making music (or at least a buzzing noise) when they made contact.

> Maker events are always family oriented, and children usually love anything messy

Learning loft

Upstairs, the soldering workshops from Manchester's Hacman hackspace saw children and families building bagpipes, a theremin, lie detector, and other fun kits. Given the problems of getting anything perceived as 'dangerous' into a modern education environment, it was great to see kids experiencing the joy of making something just as good as commercial electronic toys.

Nearby, Alan O'Donohoe of Raspberry Jam was busy shepherding families through building multiple Monk Kits, and getting visitors to look beyond surface appearances in some very philosophical discussions, while fellow Raspberry Pi luminaries joined him in introducing more people to the joys of Sonic Pi, *Minecraft*, and all things Pi.

Over the weekend, interactive workshops and crafting sessions covered everything from making jewellery, through creating caterpillars with air-dried clay, to learning chemistry and electronics with a selection of paints – conductive, colour-changing, and even scratch and sniff.

Old friends

Some of our favourites from the Liverpool MakeFest [see *The MagPi* #36] were there, including the DoES Tower and Nerf gun range from Liverpool's DoES Makerspace, and the build-a-bug workshop where kids snapped together laser-cut wooden bug parts around a 3 volt cell and a pair of flashing LED eyes.

Also seen in Liverpool was the RepublicIoT stand, here demonstrating a practical sample application that the team hope to roll out around the UK in the form of pollution monitors for their planned IoT-connected network.

GET INVOLVED

Turn to page 88 to find out about Raspberry Pi meets, MakerFests, and other events near you.

YOUR LETTERS

Code to text

I've been downloading the PDFs of *The MagPi* and following along to some of the tutorials. I'm having trouble at times copying the code over into a text editor like Geany, though - is there a better way of copying over the code? It either grabs multiple columns at once from the PDF or gets the line numbers as well, and it's a pain to try and parse that to make it work in the code. Thanks for any help!
Carl

This mainly depends on the PDF reader you're using. Preview in OS X generally seems to work best in recognising when columns and line breaks are in place, but the Adobe readers can handle the columns well enough and figure when to highlight lines or not. If your PDF viewer won't let you do what you wish, then there's not much you can do about it in terms of copying over the code.

However, a lot of our tutorials have been uploaded to GitHub – usually a link can be found in the intro text for each tutorial. If there isn't one, fear not: we're working on a solution to get them all online and easy to access so you don't have to struggle to copy them over from a PDF. In the meantime, if you're having trouble getting a particular bit of code, drop us an email to **magpi@raspberrypi.org** and we'll try to help.

Missing print

I used to have a complete collection of your magazines in print, but since the beginning of the year, not all of them have been available for me to proudly display on my bookcase. Did you ever make any print copies of issues 30 to 35? I'd really love to have them to keep my collection complete!
Howard Earl

We're actually planning to get this sorted at some point, as we're aware there are people who would like to see them all in print. We haven't got all the specifics down, but it will likely be a compendium of the issues in one book or bookazine - watch this space (and the Raspberry Pi blog) to find out when we actually start selling it!

International issue

Hi, I just wanted to let you know that I love my Raspberry Pi and that I would love to help support you, but I am unable to find *The MagPi* on any news stands in Vancouver Canada, nor am I able to get my credit card to authorise signing up for a six-month subscription. It would be fantastic if there was someone in Western Canada that was distributing Raspberry Pi and accessories. I love the work you're doing and look forward to continuing to support you.

P.S. Thanks so much for keeping *The MagPi* available as a free download – at least I can get my fix that way, although it'd be great to keep a hard copy on my coffee table to convert my non-geeky friends.
Jamie

We're trying to get the magazine onto as many store shelves as possible right now - it's made its way over to the United States so far and hopefully we'll be able to get it elsewhere soon. We're sorry to hear about the subscription not working, but we do sell the individual issues via the Swag Shop (**swag.raspberrypi.org**), which ships internationally, and you can also pay for it via PayPal.

Game HAT

I really like the idea of the Sense HAT and what it can do. I've got one on order and I was wondering what kind of things I could do with it – something different to what it was really meant to. Well, that got me thinking about video games - there's a screen, a joystick, and motion controls like everyone loves right now; I'm wondering if it's feasible to make a game on it. Would you consider making a tutorial?
Steven Gregson

Funnily enough, we saw an amazing little project recently where someone had created '100 famous characters' in pixel art that was only 8×8 – perfect for use with the Sense HAT. You could probably use them to make a game, but you wouldn't be able to make them interact with much. You'd be better off creating a motion- or joystick-controlled game like Snake that's very simple. We'd definitely consider writing a tutorial for something like that.

FROM THE FORUM: PI IN THE PALM OF YOUR HANDS

The Raspberry Pi Forum is a hotbed of conversation and problem-solving for the community – join in via **raspberrypi.org/forums**

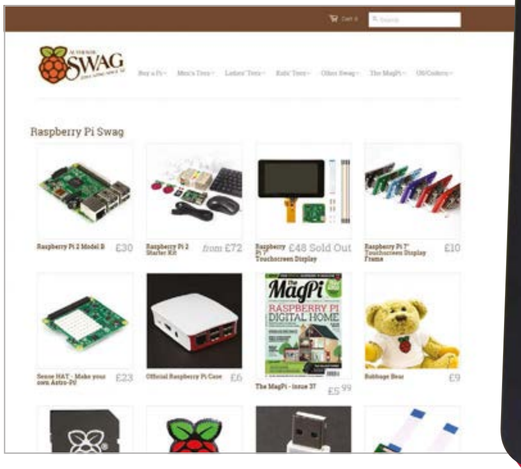
Will *The MagPi* do a feature in a future issue about building a tablet using the new official Raspberry Pi touchscreen? This would be an excellent project for everyone. All you really need is instructions on fitting a good battery and maybe a printable file for a 3D-printed case.
Wool

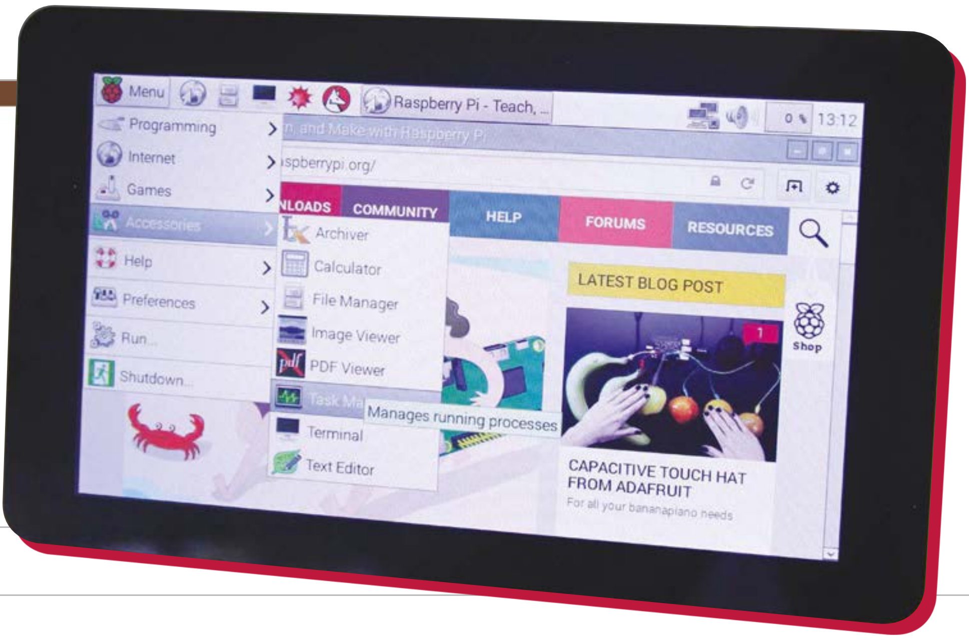
It's definitely not beyond the realm of possibility. It seems like a lot of people (including us!) thought about this sort of use for the touchscreen when they first heard about it, and you can already find projects that have or are doing this exact thing. Doing it ourselves, though, we'd like to do it right; it may not be in the very next issue, but we'll definitely have something.

And not just a tablet either – there are so many things you can do with such a screen and we'll be exploring those, along with cool Sense HAT bits, over the next several months and beyond. As always, you should keep an eye on the Raspberry Pi blog to see if any ridiculous projects using it have been dreamt up, and it's worth looking up the Adafruit Raspberry Pi page to see what they've used it for.

WRITE TO US

Have you got something you'd like to say?
Get in touch via **magpi@raspberrypi.org** or on The MagPi section of the forum at **raspberrypi.org/forums**


SWAG
Raspberry Pi Swag
Raspberry Pi 2 Model B £30
Raspberry Pi 2 Starter Kit from £72
Raspberry Pi 7" Touchscreen Display £48 Sold Out
Raspberry Pi 7" Touchscreen Display Frame £10
Sense HAT - Make your own Astro-Pi £23
Official Raspberry Pi Case £6
The MagPi - issue 37 £5.99
Babbage Bear £9



MATT RICHARDSON

Matt is Raspberry Pi's US-based product evangelist. Before that, he was co-author of *Getting Started with Raspberry Pi* and a contributing editor at *Make:* magazine.

OUR INDISPENSABLE ECOSYSTEM

Businesses can make a profit while enhancing our community, **Matt Richardson** explains

One of the best features of Raspberry Pi is its vibrant community. It's filled with not only enthusiasts like you, but also businesses that thrive by selling Raspberry Pis, designing accessories, assembling kits, hosting workshops, and creating content. The businesses in this ecosystem are a vital part of Raspberry Pi's success.

These businesses contribute to our community of users by offering products, services, and content that reach beyond what the Raspberry Pi Foundation is able to provide. We rely on these businesses to amplify our impact worldwide.

If you look at Raspberry Pi accessories alone, you'll see that there are many market opportunities. Companies in our ecosystem make specialised enclosures, HAT hardware expansion boards, displays, camera accessories, component kits, and mounting solutions. For Pi enthusiasts, these accessories broaden the horizons of what's possible with Raspberry Pi. For instance, Adafruit Industries has a nice selection of tiny TFT touchscreen HATs that attach right on top of the Raspberry Pi. For Adafruit, Raspberry Pi is serious business.

"The Raspberry Pi was, and is, one of the most transformative products at Adafruit," says Limor 'Ladyada' Fried, founder and engineer of Adafruit Industries. "Adafruit was able to build a business and maximise our cause (teaching code and electronics) using the Raspberry Pi as a core component."

Pi accessories can also serve as a bridge from our community to another community of enthusiasts. For instance, the Pi In The Sky project brings wireless communication and GPS technology to Raspberry Pi for high-altitude balloon projects. The amateur near-space exploration community and the Pi community can come together to share knowledge and do more together than they could on their own.

Besides accessories, there are also a lot of kit makers in the ecosystem. Recently, I've noticed more kits aimed at helping people to use their Raspberry Pi in electronics projects. These kits are great when you already have a Raspberry Pi and are looking to take the next step into learning the basics of electronics.

One example is the Monk Makes kit from prolific author Simon Monk. The kit includes a breadboard, jumper wires, and components like resistors, LEDs, a switch, a capacitor, a photocell, and a thermistor. Even better, it includes a set of cards with projects that walk you through how to create some basic circuits.

CamJam kit

Another favourite kit is the CamJam EduKit. It comes in a compact tin and includes a bunch of components for basic electronics projects with Raspberry Pi. The kit was created by the organisers of the Cambridge Raspberry Jam after they sourced the components for the Jam's workshops. I love that this kit is only £5 and the profits go toward funding the Cambridge Raspberry Jam.

I also want to highlight the work of Ethan Saadia, the 12-year-old entrepreneur behind the company 'PCs for Me'. Ethan assembles and sells packs of electronics components that are meant to go along with the official Raspberry Pi learning guides. It's a great idea, and I'm glad to see such a young person turn it into a product.

I wish I had more room in this column to call attention to all the businesses that are a part of the Raspberry Pi ecosystem, but there are simply too many to list. Each company's contribution is indispensable to help enhance the Raspberry Pi as a product, create connections to other communities, and amplify our mission. If you have an idea for a product related to Raspberry Pi, I encourage you to pursue it as a business. With 6 million Raspberry Pis in the wild, you're bound to find more than a few customers.

DEXTER INDUSTRIES

SAVE 15% "MagPi15" discount code

BrickPi

Build a LEGO robot with your Raspberry Pi!

GoPiGo

Everything you need to build a Raspberry Pi robot!

GrovePi

Connect hundreds of sensors to your Raspberry Pi!

www.dexterindustries.com

Wyliodrin

Have you tried Wyliodrin yet? It's free!

Use a browser from any device to program and monitor your Raspberry Pi

Program and monitor your Pi from anywhere in the Internet

Use our graphs to display your sensors' data

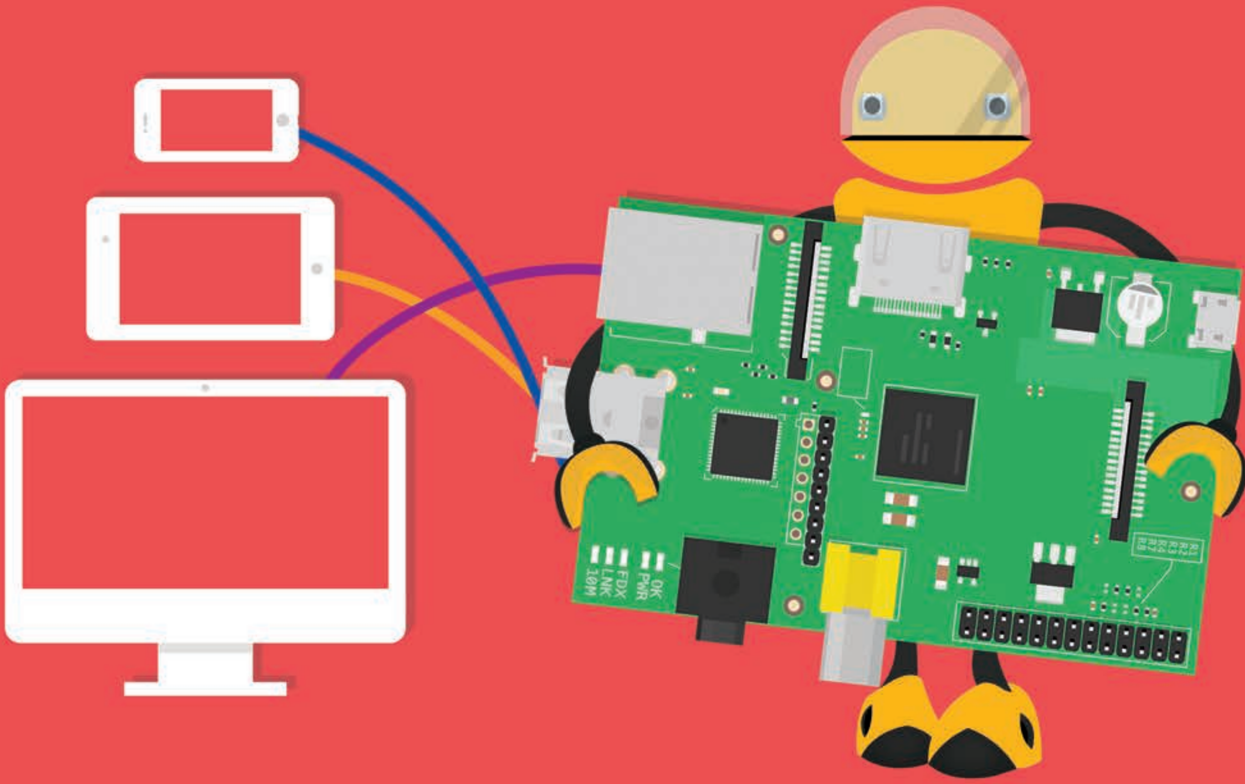
Just drag & drop blocks to create your applications, using Visual Programming

www.wyliodrin.com

python C++ node Perl Java ARDUINO php

VNC® now available for RASPBERRY PI!

RealVNC has released VNC for Raspberry Pi, which means you can now connect to your Pi from any Windows, Mac or Linux computer, or iPhone, iPad or Android device!



Check out our video tutorial to see how you can set up VNC Server on your Raspberry Pi.

Watch the video here: **https://www.youtube.com/user/RealVNCLtd**

REAL VNC

Download VNC: www.realvnc.com/download/
Getting connected: www.realvnc.com/products/vnc/raspberrypi/
For more information contact sales@realvnc.com

Made in the UK

IQaudIO

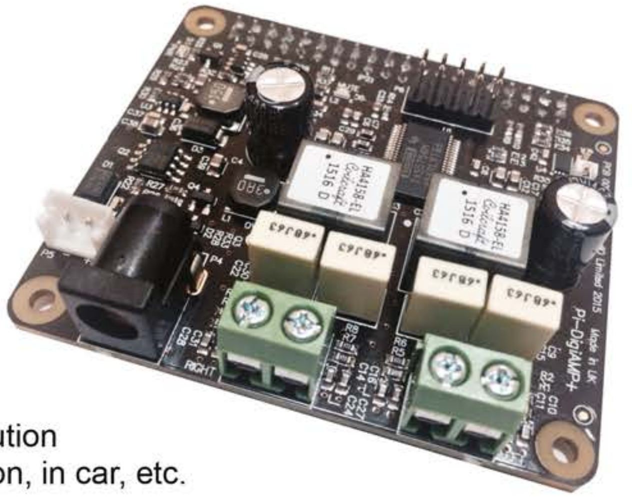
Audiophile accessories for the Raspberry Pi

Pi-DAC+

- Raspberry Pi HAT, no soldering required
- Full-HD Audio (up to 24bit/192kHz)
- Texas Instruments PCM5122
- Line-out through Phono (RCA) sockets
- Headphone Amp with 3.5mm socket
- Out-of-the-box Raspbian support
- Integrated hardware volume control
- Access to Raspberry Pi GPIO
- Connects to your own HiFI's line-in/aux

Pi-DigiAMP+

- Raspberry Pi HAT, no soldering required
- Up to 2x30w of crystal clear stereo amplification
- Full-HD Audio (up to 24bit/192kHz)
- Out-of-the-box Raspbian support
- Integrated hardware volume control
- Provides power to the Raspberry Pi
- Software controlled AMP mute on GPIO22
- I/O available (i2c, 3v, 5v, 0v, GPIO22/23/24/25)
- Add speakers for a complete HiFi streaming solution
- Designed for custom install, retro radio restoration, in car, etc.
- TheMagPi - 4 star review



Buy direct @ www.iqaudio.com

PIMORONI tindie Franken schaum MODMYPI The PiHut LIMETRACE

Supported applications

rune audio · moOde audio player · PiMusicBox · piCorePlayer SQUEEZE AUDIO DEVICE · Archphile · openelec · VOLUMIO

Mike Brady's Shairport-Sync

Pete Manchester's MediaPlayer

AND MANY MORE....

Email: info@iqaudio.com
Twitter: @IQ_audio

WWW.IQAUDIO.COM

IQaudio Limited,
14 Ulysses Road
Swindon SN25 2JR
Company No.: 9461908